

4D v11 SQL

Upgrade
Windows® / Mac OS®



4D®

© 1985 - 2008 4D SAS / 4D, Inc. All Rights Reserved.

4D v11 SQL

Upgrade

*Copyright© 1985 - 2008 4D SAS / 4D, Inc.
All Rights Reserved.*

このソフトウェアとマニュアルは著作権で保護されており、契約に基づくライセンシーの個人的な利用の場合を除き、すべてまたはその一部を複製することはできません。これには電子メディアへのコピー、アーカイブ、ソフトウェア使用許諾契約で許可されていないソフトウェアの使用を含みます。

4D, 4D Draw, 4D Write, 4D View, 4th Dimension®, 4D Server and the 4th Dimension and 4D logos are registered trademarks of 4D SAS.

Windows, Windows NT and Microsoft are registered trademarks of Microsoft Corporation.

Apple, Macintosh, Power Macintosh, QuickTime, and Mac OS are trademarks or registered trademarks of Apple Computer Inc.

Mac2Win Software is a product of Altura Software, Inc.

ICU Copyright © 1995-2007 International Business Machines Corporation and others. All rights reserved.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).
4th Dimension includes cryptographic software written by Eric Young (eay@cryptsoft.com)
4th Dimension includes software written by Tim Hudson (tjh@cryptsoft.com).

Spellchecker © Copyright SYNAPSE Développement, Toulouse, France, 1994-2007.

ACROBAT © Copyright 1987-2007, Secret Commercial Adobe Systems Inc. All rights reserved. ACROBAT is a registered trademark of Adobe Systems Inc.

All other referenced trade names are trademarks, registered trademarks, or copyrights of their respective holders.

1

はじめに

新しい 4D/4D Server の開発環境、4D v11 によるこそ。

4th Dimension は 4D へ

4D の国際的な展開をサポートおよび促進するために、4D v11 SQL 製品ライン名は以下のように統一されました：

- 一般的な "4th Dimension" (フランス語圏では "4^e Dimension") は 4D に置き換えられました。
- 4th Dimension single user アプリケーション (フランス語圏では 4^e Dimension monoposte) は 4D Developer と呼ばれます。

4D v11 SQL 製品ラインは 4D Developer、4D Server そして 4D Client アプリケーションで構成されます。

これらの名前は順次 4D の技術文書やマーケティングドキュメント上でも更新されます。

概要

4D バージョン 11 は、プログラムの内部的なアーキテクチャや、開発者が利用可能なレベルで、多くの新しい機能を提供します。これらの新しい機能には以下のテーマが関連します：

まずバージョン 11 で、4D のデータベースエンジンが部分的に書き直されました。よりパワフルにより効率的になり、あらゆる制限が緩和されました。コードを変更することなく、多くの恩恵を直接アプリケーションが受けられます。

また 4D は、すべてのタイプの SQL リクエストを処理するために利用可能な、完全な SQL エンジンを含みます。SQL 言語が新しい 4D エンジンのコアに統合され、ストラクチャやデータに直接アクセスすることができます。

多くの新しいメンテナンスおよびリファレンスユーティリティ機能が4D v11に追加されました: Maintenance & Security Center、データベースの検索置換機能が4Dに統合され、4D Toolsや4D Insiderを使用する必要はなくなりました。実際これらはv11で提供されなくなります。

4Dの作業環境におけるインターフェースが改善され、より使いやすく、またすべてのプログラム機能にアクセスしやすくなりました。ユーザモードとデザインモードのメニューは1つに統合され、その結果、4Dは(以前のユーザモードの機能を含む)新しいデザインモードと(以前のカスタムモードに対応する)アプリケーションモードから構成されます。

最後に、開発環境がさらに強化されました: 4Dコンポーネントの管理やストラクチャエディタ、階層リスト、メニュー、リストボックスなど、デザインモードと4Dランゲージに多くの機能が追加されました。

このマニュアルについて

このマニュアルには以下の章が含まれます:

- **新バージョンへの移行:** この章では、過去のデータベースを4D v11に変換する際の方法とステップ、および削除された機能を説明します。
- **4D v11 データベースのアーキテクチャ:** この章では、新しいデータベースエンジンやコンポーネントの設定や操作、および4D v11におけるファイル管理の変更について説明します。
- **デザインモードのインターフェースとナビゲーション:** この章では、4D v11で統合された作業環境と、新しいデザインモードからアクセスできる以下の機能を紹介します: 新しいMaintenance & Security Center、ストラクチャエディタの変更点、ストラクチャの検索や置換機能、ピクチャ管理、フォームエディタとメソッドエディタ、そしてコンポーネントの新しい管理方法。
- **ストラクチャエディタ:** この章では、4D v11におけるストラクチャエディタの変更点を詳細に説明します。
- **メソッドエディタ:** この章では、4D v11におけるメソッドエディタの変更点を詳細に説明します。
- **フォームとオブジェクト:** この章では、4Dのフォームとフォームに含まれるオブジェクトに関する新しい機能について説明します。

- メニューエディタ: この章では、4D v11 におけるメニューエディタの変更点を詳細に説明します。
- Maintenance & Security Center: この章では、4D データベースの検証とバックアップ機能を提供する新しいダイアログについて説明します。
- 4D SQLエンジン: この章では、4Dの新しいSQLエンジンについて説明します。
- Web サーバ: 4D v11 の Web サーバ機能に関する変更点を説明します。
- ランゲージ: この章では、4D v11 の新しいコマンドや変更されたコマンドについて説明します。

最低動作環境

4D v11 製品ラインの最低動作環境は以下のとおりです:

	Windows	Mac OS
コンピュータ	Pentium III プロセッサ搭載の PC 互換機	Intel または Power PC Macintosh
OS	Windows Vista, Windows XP	Mac OS version 10.4.5 以降
最低メモリ	512 MB	512 MB
推奨メモリ	1 GB	1 GB
画面解像度	1280x1024 ピクセル	

2

前バージョンからのデータベースの移行

4D バージョン 6.x, 2003.x and 2004.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、バージョン 11 と互換があります。しかしデータベースはバージョン 11 用に変換され、変換後は以前のバージョンで開くことができなくなります。

4D v11ではデータベースエンジンレベルでアーキテクチャ的な変更が行われたため、ストラクチャおよびデータファイルを v11 で使用するためには内部的な変換を行う必要があります。この変換は、後ほど説明する特別なウィザードを使用して行われます。事前に警告した後、ウィザードは元のデータベースのコピーを作成し、そしてデータベースを変換します。これにより以前のバージョンのデータベースも使用することが可能となります。

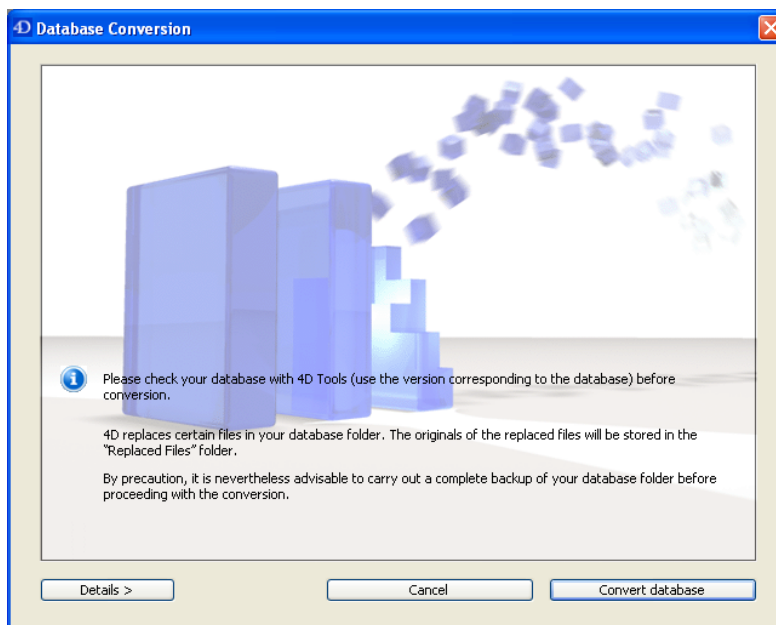
さらに、いくつかの廃止済みまたは旧来のメカニズムが非サポートとなり、変換中に削除されます。(25ページ "保持されないメカニズム"の節を参照)

過去のデータベースの変換：通常の場合

インタプリタストラクチャファイルを変換することができます。ファイルにコンパイルコードが含まれる場合、変換後に再度コンパイルする必要があります。

変換前に、対応するバージョンの 4D Tools を使用して、データベースの整合性を検証することをお勧めします。必要に応じて検証、圧縮、または修復を行います。これを行わず、変換中に異常が検出された場合、処理は中断され、ウィザードは 4D Tools の使用を促す警告を表示します。

以前のバージョンのデータベースを変換するには、4D v11のファイルを開くダイアログボックスで、変換対象のデータベースを選択します。すると変換ウィザードが表示されます。



データベース変換 をクリックして、データベースの変換処理を開始します。また、変換に使用されるデフォルト設定を変更することもできます。これを行うには 詳細> ボタンをクリックします (次の段落を参照)。

変換ウィザードはストラクチャとデータファイルの完全なコピーを作成し、のちに変換処理を開始します。変換に失敗しても、コピーにより、元のデータベースが利用可能です。データベースの複製には追加のディスク空き容量が必要です。もしディスクの空き容量が足りない場合は、警告メッセージが表示され、変換処理は中止されます。

元のファイルは、そのファイルと同階層に作成される Replaced Files (conversion) という名前のフォルダに格納されます。データファイルがストラクチャファイルと同じ場所であれば、Replaced Files (conversion) フォルダには両方の元のファイルが格納されます。

他方、データファイルが別の場所や別のボリュームにある場合、Replaced Files (conversion) はそれぞれの場所に作成されます。

- 注 変換するデータベースのカレントログファイルも Replaced Files (conversion) フォルダにコピーされ、新しい空のログファイルが作成されます。

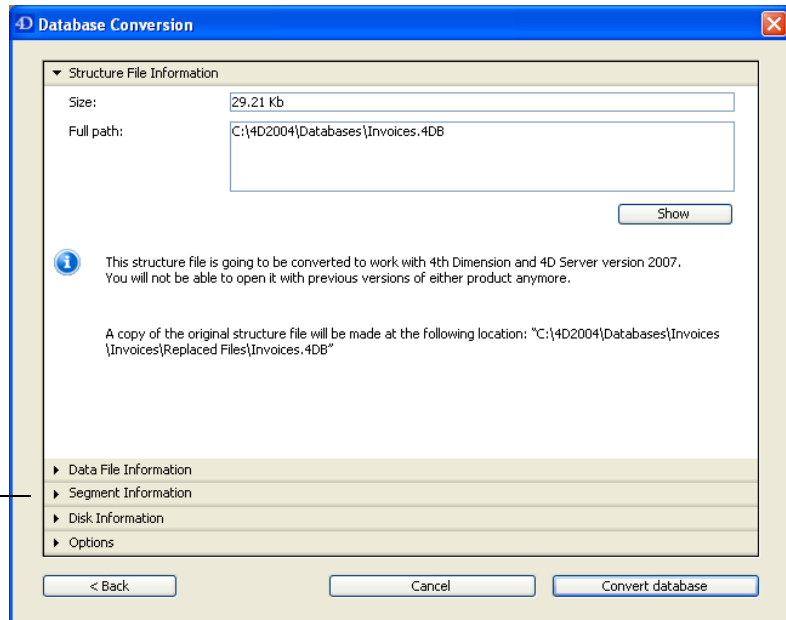
変換中に致命的でないエラーを検出した場合、エラーは変換後のストラクチャファイルと同階層に作成される DataConversion_Log.log という名前のファイルに記録されます。

- 注 デバッガに設定されていたブレークポイントや式などの情報は、変換によって再初期化されます。

変換オプションの表示と変更

変換時に使用される特定のオプションを確認したり、変更したりするには、詳細 > ボタンをクリックします。ダイアログボックスのエリアをクリックすると、対応する情報が表示されます：

情報ページ



- ストラクチャファイル情報: このページは元のデータベースストラクチャファイルのサイズや場所などの情報、およびコピーの作成先などを表示します。

- **データファイル情報**: このページは下のデータベースデータファイルのサイズや場所などの情報、およびコピーの作成先などを表示します。
またこのページでは、変換するデータファイルの指定を行うこともできます。デフォルトで、カレントのデータファイルが選択されます。必要に応じて空のデータファイルを新規に作成したり ([新規データファイル作成 オプション](#))、他のデータファイルを選択... ボタンをクリックして他のデータファイルを変換するよう選択することもできます。
複数のデータファイルの変換に関する詳細は、[23ページ "複数のデータファイルを使用するデータベースの変換"](#) の節を参照してください。
保存先を変更 ... ボタンをクリックして、変換後のデータファイルの格納場所を指定できます。
- **セグメント情報**: このページでは、データベースのデータファイルセグメントの一覧を見ることができます。詳細は [24ページ "複数のセグメントを持つデータベースの変換"](#) の節を参照してください。
- **ディスク情報**: このページは、データベースが置かれているディスクの空き容量を表示します。
- **オプション**: このページには、変換オプションがあります。
 - **データベースを開く際にコードを実行しない**: このオプションが選択されていると、通常データベース開始時に実行されるコードが、変換後に初めてデータベースを起動するときに限り、無効となります。このオプションは、**On Startup** データベースメソッドを使用して呼び出されるコードに適用されます。これにより、変換直後に発生するかもしれない初期化エラーを避けられます。

変換データベースを構成するファイル

バージョン 11 への変換後、4D データベースには複数のファイルが追加されます:

- **データベース名.4DIndy**: ストラクチャのインデックスが含まれます。
- **データベース名.4DIndx (オプション)**: データのインデックスが含まれます。

注 4D v11 では、インデックスはデフォルトでデータファイルの外に格納されます。この件の詳細は、[35ページ "インデックスの管理"](#) の節を参照してください。

- **DataConversion_Log.log**: このファイルには、データベース変換中に発生した異常が記録されます。
- **Replaced Files (conversion) フォルダ**: このフォルダには、変換前のデータベースファイル (ストラクチャ、データおよびログ) が格納されます。

過去のデータベースの変換：特別なケース

複数のデータファイルを使用するデータベースの変換

変換するデータベースが、複数の異なるデータファイルを使用している場合、データベースとそれぞれのデータファイルの変換は別々に行います。追加のデータファイルの変換は **ファイル** メニューの **開く > データファイル...** を使用して行います。

- 1 カレントのデータファイルとデータベースを変換。
これを行う方法は [19 ページ "過去のデータベースの変換：通常のケース" の節](#) で説明されています。
- 2 変換されたストラクチャで、**ファイル**メニューの **開く > データファイル...** を選択する。
標準のファイルを開くダイアログボックスが表示されるので、変換したいデータファイルを指定します。
- 3 変換するデータファイルを選択したら、**開く** をクリックする。
変換ウィザードが表示されます。選択されたデータファイルの情報が表示されます。データファイルは、まず複製が作成され、標準の変換方法と同様に Replaced Files (conversion) フォルダにおかれます。([19 ページ "過去のデータベースの変換：通常のケース" の節](#) を参照)
- 4 ウィザードウィンドウの **データベースを変換** をクリックして、変換を開始する。
変換が終了すると、変換されたデータファイルはデータベースのカレントデータファイルとなります。

コンポーネントを含むデータベースの変換

前のバージョン (バージョン 11 より前) の世代のコンポーネントを含むストラクチャファイルを変換することはできません。ストラクチャファイルを変換する前に、対応するバージョンの 4D Insider を使用して、すべてのコンポーネントをアンインストールする必要があります。

v11の環境でコンポーネントを使用するには、アップグレードしたバージョンのコンポーネントを入手する必要があります。新しい世代のコンポーネントに関する詳細は、 [45 ページ "新しいコンポーネントアーキテクチャ" の節](#) を参照してください。

複数のセグメントを持つデータベースの変換

4D v11 では、データファイルの最大サイズが無制限となります (OS の制限によります)。そのため、セグメントを作成したり使用したりすることはできません。

セグメントを含むデータベースを変換する際には、変換ウィザードがすべてのセグメントの内容を集め、1つの新しいデータファイルを作成します。変換の際には、新しいデータファイルを格納することのできるディスクの空き容量が必要です。

データベースのセグメントは、変換ダイアログボックスの "セグメント情報" ページで見ることができます。 ([21 ページ "変換オプションの表示と変更" の節](#) を参照)

変換中にセグメントを見つけられない場合、変換ウィザードはセグメントを指定するためのダイアログを表示します。セグメントを見つけられなければ、変換を行うことはできません。

キーボードレイアウトを指定したデータベースの変換

データベースを v11 に変換する際、テキストデータは Unicode に変換されます。変換を正しく行うためには、4D が元の文字セットを知っている必要があります。デフォルトでは、カレントのシステムランゲージに対応する文字セットが使用されます。

前のバージョンの 4D では、キーボードレイアウトプロパティを使用してフォームオブジェクトに特定の文字セットを指定することが可能でした。この場合、4D v11 は文字セットを正しく推定することができません。なぜならばそのプロパティはフォームに格納されていますし、データ入力の変換を介して行われているかもしれないからです。

変換前に、開発者は明示的に使用される文字セットを定義しなければなりません。これを行うには、変換するデータファイルと同階層に、テキストファイルを配置します。このファイルは以下の特徴を持ちます：

- 名称: multilang.txt
- エンコード: ANSI または Mac Roman (Unicode は使用できません)
- フォーマット: table_number separator field_number separator subfield_number (optional) separator dialect_code CRLF
 - 区切り文字は ";" (セミコロン)。
 - それぞれの行は改行 (CR または CRLF) で終了します。空行やスペースを置くこともできます。

■ 例:

4つのテーブルを持つデータベースで (TABLE_1 ~ TABLE_4)、
 - TABLE_3 には4つの文字フィールドがあり (field_1 ~ field_4)、
 - TABLE_4 には6つの倍長整数フィールドと、2つの文字フィールドを持つ SUBFIELD サブテーブルフィールドがあります。

- " キーボードレイアウト " プロパティは [TABLE_3]field_3 と [TABLE_4][SUBFIELD]field_1 に Greek を、 [TABLE_3]field_4 と [TABLE_4][SUBFIELD]field_2 に Russian を入力できるように設定されています。

- multilang.txt ファイルは以下のように記述されます:

3;3;1049

3;4;1032

4;7;1;1049

4;7;2;1032

dialect_code は、4D Extensions フォルダにある keyboardmapping.xml ファイルに定義されています。

プラグインの互換性

バージョン 2004 のプラグイン (4D のプラグインやサードパーティーのプラグイン) は4D v11と互換性があり、この環境下で使用することができます。

ただし 4D v11 の新しいエンジンの恩恵を受けるためには、プラグインを更新する必要があります (29 ページ "容量の拡張" の節を参照)。

Mac OS で 4D を Universal Binary で実行させ、パフォーマンスを向上させるためには、プラグインもこのアーキテクチャでなければなりません。そうでなければ、4D とプラグインを "Rosetta" モードで使用する必要があります (34 ページ "ユニバーサルバイナリアーキテクチャ (Mac OS)" の節を参照)。

4D アプリケーションの互換性

保持されないメカニズム

4D バージョン 11 では、いくつかの旧式で効力のなくなったメカニズムがサポートされなくなりました。それらは変換中に無視されるか、置き換えられます。以下のオプションやメカニズムが対象となります:

- "前世代の"コンポーネント: 前世代のコンポーネントは、データベースをバージョン 11 に変換する前に、4D Insider を使用してアンインストールしなければなりません (23 ページ "コンポーネントを含むデータベースの変換" の節を参照)。
- データセグメント: データセグメントは変換時に 1 つのファイルにグループ化されます (24 ページ "複数のセグメントを持つデータベースの変換" の節を参照)。
- サブテーブル: サブテーブルフィールドはサポートされません。これらのフィールドは標準のテーブルに変換され、元のテーブルに自動リレーションが張られます。これを行うために、新しいテーブルにはフィールドが 1 つ追加されます。しかし、サブテーブルを管理する以前のメカニズムは、サブテーブルを基点とするテーブルやフィールド、リレーションに変更が加えられない限り動作します。この件についての詳細は 40 ページ "サブテーブルの変換" の節を参照してください。

注 DUPLICATE RECORD、SEND RECORD そして RECEIVE RECORD コマンドは、サブレコードをサポートしません。

- テーブル操作にユーザグループを割り当てる: ストラクチャエディタでテーブルのデータ操作 (読込、保存、追加、削除) にグループを指定することはできなくなります (100 ページ "ツールバー、インフォメーションバーおよび新規インスペクタ" の節を参照)。同様にテーブルアクセス権にグループを指定することもできなくなります。
これらのコントロールは 4D v11 では無効となり、変換されたデータベースでは無視されます。
4D v11 において、アプリケーションの異なる部分へのアクセスコントロールは、ランゲージコマンドや特定のオプションを通じて、インタフェースレベル (フォーム、メニュー、メソッド) で行うことができます (SQL によるデータアクセスに関する 207 ページ "4D データベースのアクセス管理" の節の例題を参照)。
- ユーザモードアクセス権: 環境設定のアプリケーション/アクセスページのこのポップアップメニューは取り除かれました。ユーザモードは今バージョンより存在しません (70 ページ "新しいデザインモードインタフェース" の節を参照)。デザインモードのアクセス権のみが設定可能です。アプリケーションモードは常にアクセス可能です。
- 4D リソースとシステムリソースへの直接アクセス: "リソース" テーマのコマンドを使用した 4D 内部リソースとシステムリソースへのアクセスはできなくなります。これらのリソースを使用するデータベースの更新作業への影響を緩和するため、代替りのメカニズムが用意されています。詳細な情報は 42 ページ "リソースの管理" の節を参照してください。

- 4D 2003/2004 マクロ: 4D 2003 や 2004 で使用されていたマクロは、4D v11 と互換性がありません。自動変換メカニズムが用意されていますが、XML 標準仕様に適合させるため、開発者による若干の調整が必要になります。詳細は[131ページ "新しいマクロアーキテクチャ"の節](#)を参照してください。
- リソースを使用した ID ウィンドウアイコンのカスタマイズ: ユーザ ID ダイアログボックス (名前の選択とパスワードの入力) はリソースのコピーを使用してカスタマイズできました。このメカニズムは今後使用できず、ファイルのコピーで行うようになります ([68 ページ "ID ウィンドウアイコンのカスタマイズ" の節](#)を参照)。
- バックグラウンドピクチャの "モードの選択": バックグラウンドフォーマットのピクチャと、バックグラウンドの相互関係を設定する "モードの選択" ウィンドウは使用できなくなります。このメカニズムはもともとモノクロ画像のために実装されたものであり、ネイティブモードで管理されるピクチャには適さないものです ([155 ページ "ピクチャフィールドおよび変数の最適化" の節](#)を参照)。
- デザイン, ユーザ, カスタムメニュー標準アクション: 変換されたデータベースや新規に作成されたデータベースでは、これらの標準アクションを選択することはできなくなります。([176 ページ "変換データベースでの標準動作" の節](#)を参照)。

削除された 互換オプション

以下の互換性オプションが4D v11 の環境設定から取り除かれました (アプリケーション / 互換性 ページ):

- v3.x.x のファイルプロシージャ方式を使う
- テキスト描画を 6.8 互換にする

これらのオプションは、廃止されたメカニズムをエミュレートしていました。

- 以前の編集メニューメカニズム (v 6.8 オプション)
メニューエディタの中にあるこのオプションは取り除かれました。このオプションを使用するデータベースを変換した際は、標準の編集メニューを追加する必要があります ([177ページ "編集メニュー\(互換性\)"の節](#)を参照)。
- v3.x.x の Startup プロシージャ方式を使う
v11 では、以前のこの操作は考慮されなくなり、Debut や Startup などの名前を持つプロジェクトメソッドが自動で起動されることはなくなります。開始時のコードは **On Startup** や **On Server Startup** データベースメソッドにおかれなければなりません。

削除されたコマンド

以下のコマンドは、サポートされないテクノロジーに基づくため、4D v11 に存在しません。

- SEARCH BY INDEX
- SAVE OLD RELATED ONE
- SORT BY INDEX

シンタックスの変更

この節では、変換されたアプリケーションでシンタックスエラーを起こす原因となる変更点について説明します。

- アーキテクチャ的な変更により、**QUERY BY FORMULA, QUERY SELECTION BY FORMULA** そして **APPLY TO SELECTION** コマンドで、table 引数が必須となりました。もしこの引数を省略すると、シンタックスエラーが生成されます。
 - 4D v11 のランゲージインタープリタは、以前のバージョンの 4D より厳格になっています。特に暗黙の型変更や配列から変数への変更、またその逆を行うと、以前のバージョンでは許容されていましたが、バージョン 11 ではシンタックスエラーが生成されます。
-
- 注 4D v11 では特定のコマンドの内部的な動作が変更されました。そのため変換されたアプリケーションでは調整が必要となります。これらの変更点は [ページ 245 "ランゲージ" の章](#)、特に [382 ページ "その他の変更" の節](#) で説明しています。
-
- Document システム変数は常にドキュメントの完全なパス名を含みます ([383 ページ "ドキュメントシステム変数" の節](#) を参照)。

ドキュメントフォーマット

SAVE SET/LOAD SET と SEND RECORD/RECEIVE RECORD で処理されるドキュメントの内部フォーマットが変更されました。古いフォーマットのドキュメントを (4D v11 の LOAD SET や RECEIVE RECORD コマンドを使用して) 読むことはできますが、4D v11 で生成されたドキュメントを以前のバージョンで読み込むことはできません。

3

4D v11 データベースのアーキテクチャ

4D v11 のデータベースエンジンは書き直され、よりパワフルに、より堅牢になりました。この新世代エンジンは最新のデータ管理テクノロジーの恩恵を受け、過去のバージョンに存在した制限を緩和しています。

4D v11 データベースアーキテクチャはファイルレベル及び内部アーキテクチャの両方が変更されています。

4D v11 データベースのファイルが格納されるフォルダも新しいアーキテクチャを使用します。

容量の拡張

4D v11. では 4D の過去のバージョンの多くの上限が取り除かれるか緩和されています。

- 次の表は 4D v11 データベースエンジンの新しい上限と 4D の過去のバージョンの上限とを比較しています。:

容量	4D 200x	4D v11
データファイルのサイズ	各 2GB の 127 セグメント	無制限 (セグメンテーションなし)
テーブルの数	255	32767
一テーブルあたりのフィールドの数	511	32767
一テーブルあたりのレコード数	1600 万	10 億
一テーブルあたりのインデックスキー数	1600 万	1280 億
文字フィールドの長さ	80	255
テキストフィールドの長さ	32000 文字	テキスト ¹ 2GB
トランザクションのレベル t	1	無制限 ²

1. 既存の定数 MAXTEXTLEN (値 32000) が MAXTEXTLENBEFOREV11 に名称変更されています。バージョン 11 ではこの定数は意味を持ちません。
2. 互換性のため、変換されたデータベースはデフォルトで 1 トランザクションレベルに制限されています。(下記を参照)

新しいフィールドタイプ

二つの新しいタイプのフィールド、Integer 64 bits と Float は 4D v11 のストラクチャエディタ内で有効です。

- Integer 64 bits: 8 byte の整数は $-2^{63}..(2^{63})-1$ の間に含まれる値操作に利用可能です。

Float: 浮動小数点数。浮動小数点数は正確性を損なうこと無く値を格納することができます。

4D v11 の現在のバージョンでは、これらの種類のフィールドは 4D の SQL エンジンによってのみ利用されます (199 ページ "4D SQL エンジンの使用" の章を参照)。4D 言語内でこれらのフィールドが使用された時、これらの値は内部で本来の数に変換されます。

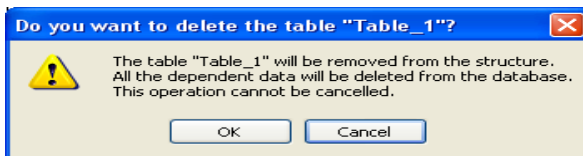
テーブルとフィールドの削除

4D v11 において、テーブルとフィールドの物理削除が可能です。

この操作は 4D の SQL エンジン又はストラクチャエディタを通して実行できます。

4D の SQL ステートメントを使用したより多くの情報に関しては 199 ページ "4D SQL エンジンの使用" の章を参照してください。

ストラクチャエディタでテーブルまたはフィールドを削除するには、削除するテーブル又はフィールドを選択して、編集メニューのクリアコマンドを選択するか、テーブルのコンテキストメニューから削除を選択します。警告ダイアログボックスが表示され、作業を続行してよいか確認されます。:



OK をクリックすると、4D が次の操作を実行します:

- テーブルまたはフィールドがストラクチャから完全に削除されます。テーブルまたはフィールドに関連する全データが、データファイルから完全に削除されます。
- テーブルに関連するトリガメソッドが削除されます。

- そのテーブルのテーブルフォームはプロジェクトフォームに変換され (135 ページ "プロジェクトフォーム" の節を参照)、エクスプローラのゴミ箱に移動されます。

テーブルをゴミ箱に入れる

テーブルを (エクスプローラの) ゴミ箱に入れると、恒久的でないテーブルの削除を行うことができます。テーブルは 4D のエディタに表示されず、データにアクセスすることはできませんが、ゴミ箱が空にされない限り、元に戻すことができます。この機能は以前のバージョンに既に存在しました。

テーブルをゴミ箱に入れるには、エクスプローラで、コンテキストメニューの "ゴミ箱に入れる" コマンドを選択するか、テーブルを選択して Delete や Backspace キーを押します。この場合以下のダイアログボックスが表示されます。



削除されたテーブルとフィールドの番号

テーブルやフィールドが削除された時、データベースの安定性を損なうことを避けるために、データベースの他のテーブル番号やフィールド番号は変更されません。そのため、例えば 2、4、5 と番号付けされた 3 つのテーブルを持つデータベースや、データベースに 1、4、6、8 と番号付けされた 4 つのフィールドをもつテーブルが存在する可能性があります。

これにより、以前の **Count tables** や **Count fields** コマンド使用したテーブルやフィールド数のカウントは無効なものとなりました。なぜならこれらのコマンドは削除によって生じる番号の抜けを考慮していないからです。そのため、このコマンドは名称変更され、4D v11 に新しいコマンドが追加されます。(363 ページ "テーブルとフィールドのカウント" の節を参照)。

新しいテーブルまたはフィールドが作成されると、削除されたテーブルやフィールドの番号は再利用されます。(この原則はレコード番号のそれと同じです。)

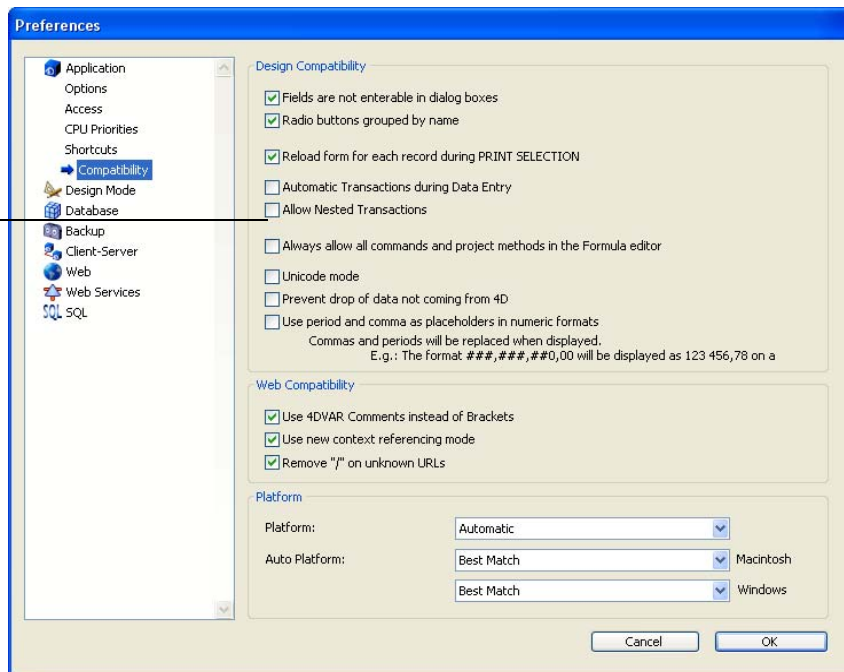
トランザクションの互換性

4D v11 では無制限にネストされたトランザクションを使用できます。

この新しい動作により、過去のバージョンの 4D で作成されたデータベースで動作が変わってしまう可能性があるため、変換されたデータベースではデフォルトでこの動作は無効になっています。つまりトランザクションは 1 レベルに制限されます。

変換されたデータベースで、マルチ階層のトランザクションを利用するには、環境設定のアプリケーション / 互換性ページで、トランザクションのネストを許可するのオプションを有効にしなければなりません。:

変換されたデータベースで、マルチレベルトランザクションを有効にするオプション



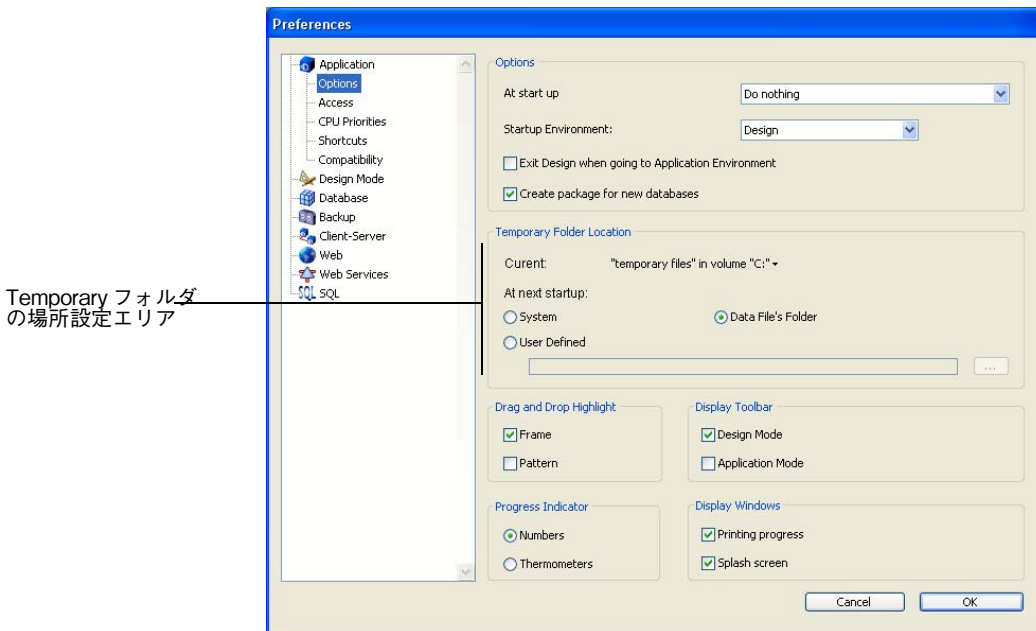
このオプションは変換されたデータベースにのみ表示されます。デフォルトでこのオプションはチェックされません。このオプションは各データベース固有です。

- 注： トランザクションのネストを許可するのオプションは、4D v11 の SQL エンジンで実行されたトランザクションには影響しません。(199ページ "4D SQL エンジンの使用" の章を参照)。SQL トランザクションは常にマルチ階層です。

Temporary フォルダ

4D v11 は "Temporary フォルダ" の新しいオプションを提供します。このフォルダはメモリ上のデータをディスクに格納するために、必要に応じてアプリケーションにより使用されます。

フォルダの場所は環境設定のアプリケーション / オプションページで設定できます：



現在のフォルダの場所はエリアの上部にスクロールダウンリストの形で表示されています。クリックして完全なパスネームを表示したりコピーしたりできます。

3つの位置オプションが提供されます：

- システム：このオプションが選択されると、Windows や Mac OS により指定された場所に置かれたフォルダ内に、4D のテンポラリファイルが作成されます。**Temporary folder** コマンドを使用して、システムにより定義された現在の位置を知ることができます。
ファイルは、データベース名と固有の ID からなる名前のサブフォルダ内に置かれます。
- データファイルフォルダ（デフォルト）：このオプションが選択されると、4D のテンポラリファイルはデータベースのデータファイルと同じ階層に位置する "temporary files" フォルダ内に作成されます。
- ユーザ定義：このオプションは任意の場所を指定するために使用されます。

場所オプションを変更した後は、新しいオプションを有効にするためにデータベースを再始動する必要があります。

4D は選択されたフォルダがデータの書き込みアクセスが可能であるか検査します。条件にあてはまらない場合、アプリケーションは妥当なフォル

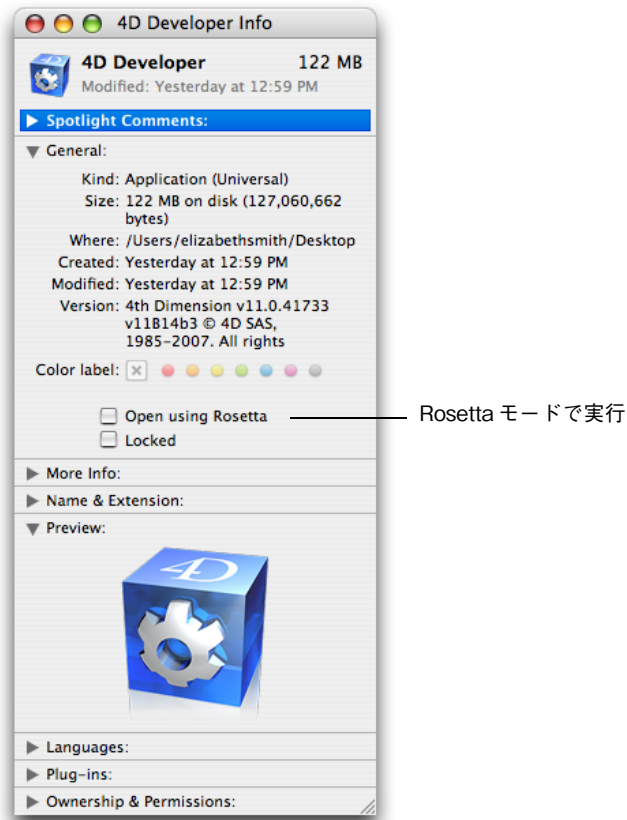
ダが見つかるまで他のオプションを試みます。
このオプションは "extra properties" に格納されます。これはストラクチャ定義が XML で書き出されるときに一緒に書き出されます。([123 ページ "ストラクチャ定義の書き出しおよび読み込み" の節](#) を参照。)

ユニバーサルバイナリアーキテクチャ (Mac OS)

Mac OS 用の 4D v11 はユニバーサルバイナリアーキテクチャであり、4D アプリケーションはインテルプロセッサベースの最近の Apples 社製 PC でネイティブモードで動作します。これはパフォーマンスにおいて相当の利益をもたらします。

デフォルトで、4D v11 アプリケーションはネイティブモードで実行されます。しかし、Rosetta によるエミュレーションモードで実行させることも可能です。これはユニバーサルバイナリでないプラグインを使用する際に必要となります。Rosetta モードで 4D v11 を実行するには、デスクトップ上の 4D アプリケーションを選択し、ファイルメニューの情報を見るコマンドを選択します。

情報を見るウィンドウ内のオプションを使用して、エミュレーションモードでアプリケーションを実行することができます：



インデックスの管理

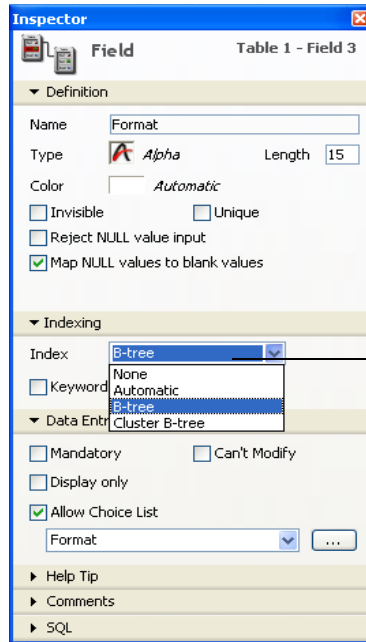
4D v11 ではインデックスの管理が変更されています。まずインデックスの作成モードを選択できるようになりました。また新しい種類のインデックスが利用可能になりました。読み込みダイアログボックス内の新しいオプションを使用してインデックス再構築のタイミングを変更することもできます。

新しいコマンドを使用してインデックスを作成、更新することができます：[363 ページ "ストラクチャアクセス" の節](#)を参照してください。

注：インデックスモードで可能な操作は、すべて非インデックスモードで同じく実行可能です。例えば、リレーションで使用するフィールドにインデックスを作成することはもはや必須ではありません。

標準インデックス アーキテクチャの選 択

4D v11 では作成時に標準のインデックスアーキテクチャを選択できます。(標準インデックスとは一般的なインデックスを指します。この他に、後ほど説明するキーワードインデックスや合成インデックスがあります。) ストラクチャエディタで、ポップアップメニューを使用して標準インデックスを管理できます:



インデックスの選択

3つの選択肢が利用可能です:

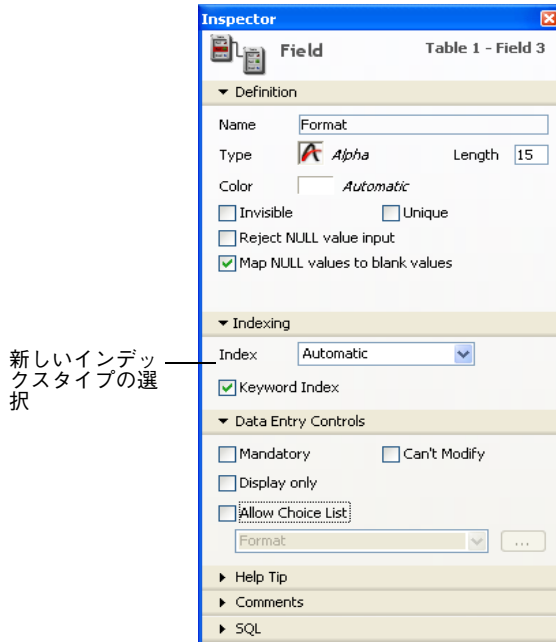
- B-tree: 標準 B-Tree インデックス。以前のバージョンの4Dではこの多目的インデックスが使用されていました。
- Cluster B-tree: クラスタを使用した B-Tree インデックス。このアーキテクチャはインデックスが多くキーを含まない時、例えば同じ値がデータ内で頻繁に繰り返される場合に有効です。
- 自動: 4D がデータに基づき適切なアーキテクチャを選択します。
- なし: インデックスなしまたは既存のインデックスを削除します。

注: 新しいインデックス作成ダイアログボックスを使用して、標準インデックスを作成することもできます。([116 ページ "インデックスプロパティダイアログボックス" の節](#) を参照)。

キーワードインデ ックス

文字フィールドとテキストフィールドに対し、新しいタイプのインデックスであるキーワードインデックスを指定することができます。フィール

ドにこのタイプのインデックスを指定するには、ストラクチャエディタのインスペクタでキーワードをインデックスオプションにチェックします。



キーワードインデックスを選択すると、テキストが単語ごとにインデックスされます。1文字や2文字であっても全単語がインデックス化されます。この新しいタイプのインデックスは検索速度を劇的に向上させます。(キーワード検索についての詳細は、[261 ページ "キーワード検索"](#) の節を参照)。キーワード検索を行うフィールドではこのオプションを有効にします。

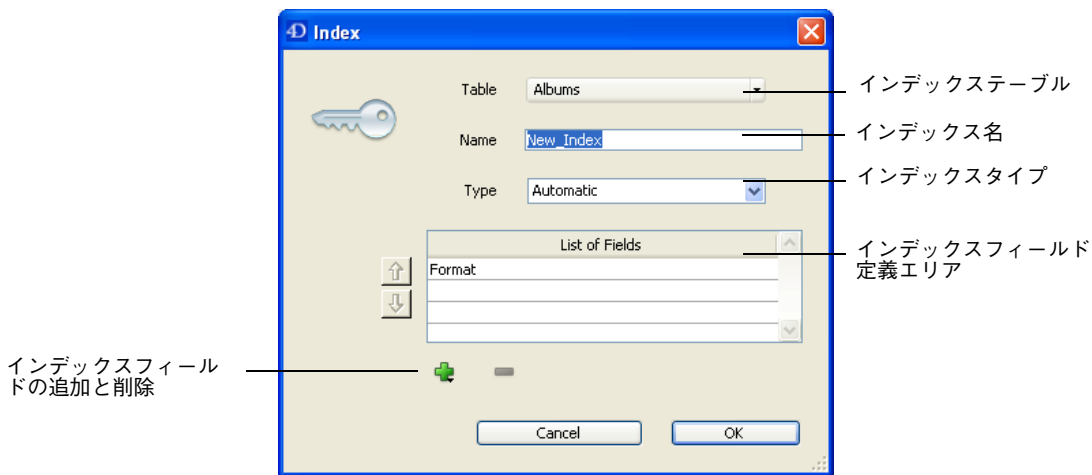
- 注：
- ・ テキストフィールド（レコード内に格納される場合）や文字フィールドに旧来のインデックスを作成することも可能です。実際両タイプのインデックスを同じフィールドで選択することが可能です。4D はコンテキストにより適当なインデックスを使用します。
 - ・ 新しいインデックス作成ダイアログボックスを使用したキーワードインデックスの作成が可能です。([116 ページ "インデックスプロパティダイアログボックス"](#) の節を参照。)
 - ・ 現在のバージョンでは、日本語の文章を単語に分解することはできません。このようなデータが格納されるフィールドにはキーワードインデックスを設定しないでください。ただし、日本語による単語を半角スペースで区切っている場合には、このタイプのインデックスを使用することができます。

複合インデックス

複合インデックスを作成することができます。複合インデックスとは、それぞれの入力ごとに、2つ以上のフィールドの値を結合した値を格納するインデックスです。苗字 + 名前で構成されたインデックスが複合インデックスの一例です。複合インデックスがあれば "Peter Smith" を探す際に標準的な検索 (Smith を検索してから、Peter を検索) と比較して最適化されます。

4D はクエリや並び替えの際、複合インデックスを自動的に利用します。例えば複合インデックス "City+ZipCode" が存在するならば、"lastname=carter & city=new york & zipcode=102@" というクエリを実行する際それが使用されます。

複合インデックスは、4D のストラクチャエディタからアクセス可能な新規インデックス作成ダイアログボックスの使用してのみ作成できます。:



このダイアログボックスの詳細は、[116 ページ "インデックスプロパティダイアログボックス"](#) の節を参照してください。

注：インデックス作成ダイアログボックスは標準インデックスやキーワードインデックスを作成することもできます。

フォーミュラコマンド

QUERY BY FORMULA, QUERY SELECTION BY FORMULA, ORDER BY FORMULA コマンドは、可能であれば、インデックスを使用するようになりました。

クライアント / サーバー環境で、これらのコマンドはサーバ側で実行されるようになります。そのため、ローカル変数を使用することはできなくなります。

互換性のために、変換されたデータベースにおいて、4D の前のバージョンの動作が保持されます (シーケンシャルな操作)。環境ダイアログボッ

クスのチェックボックスを使用して、新しい動作を有効にすることができます。

インデックスをファイルに格納

4D v11 のデータベースで作成されたインデックスは別ファイルとして格納されます。これらのファイルは自動的にストラクチャファイルと同階層に置かれます。ファイルを動かしたり名称変更することはできません。それを行った場合、4D はインデックスファイルをサイド作成します。

2つのインデックスファイルが作成されます：

- *DatabaseName.4DIndx*: データインデックスを含みます。
- *DatabaseName.4DIndy*: ストラクチャのインデックスを含みます (ストラクチャの検索時に使用されます)

主要な利点のひとつは、インデックスが壊れてしまった場合、4D を起動する前にファイルを物理的に削除することが可能であることです。そうすることで自動的に改めて作成されます。

読み込み後のインデックスの再構築

新しいオプション、"読み込み後にインデックスを再構築" がデータ読み込みダイアログボックスで利用可能です。このオプションにチェックがついている時 (デフォルト)、インデックスはデータの読み込みが完了した後に再構築されます。このメカニズムは大量のデータの読み込み速度を改善するのに有効です。

フィールドに既にあるフィールドのデータに比べ少量である場合、読み込み前にこのオプションのチェックを外すとよいかもしれません。この場合、インデックスは徐々に更新され、完全な再構築はされません。

BLOB、ピクチャ、テキストフィールドの保存

4D v11 では、(BLOB, ピクチャ, テキスト型¹ など) 大量のデータを格納することのできるフィールドのデータを、レコードそれ自身の外に格納します。これにより、特にクエリなどで、データベースの実行速度を向上させることができます。4D がレコードにアクセスする際も、これらのフィールドに格納された大量のデータが、自動でメモリにロードされることはなくなります。このデータは実際必要な時のみロードされます。

この新しい動作は自動的であり、既存のコードの変更を必要としません。

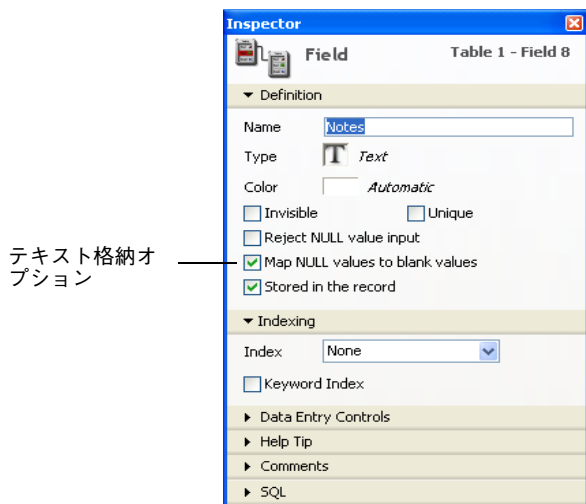
事実、最適化のため、ほとんどの 4D 開発者は一対一でリレートされた特別なテーブルを使用して BLOB、ピクチャ、テキストフィールドを管理しています。このトリックは v11 でも完全に動作しますが、もはや必要ではありません。

1. テキストフィールドのデータをレコード内に格納するよう強制することもできます。次の段落を参照してください。

テキストフィールドの保存オプション

テキストフィールドプロパティのオプションを使用して、テキストデータをレコード内に強制的に保存させることができます。(以前のバージョンの4Dの動作)

この操作は、テキストフィールドに標準インデックスを使用する際に必要となります。(テキストフィールドインデックスに関する詳細は、[36 ページ "キーワードインデックス" の節](#)を参照してください。) 標準インデックスを、テキストフィールド自身の外側に格納されたデータに対して作成することはできません。この場合、選択したフィールドのインスペクタパレット内にある "定義" エリアでレコードに格納オプションにチェックしなければなりません。:



互換性のために、変換されたデータベースではテキストフィールドのこのオプションは、デフォルトでチェックされます。

バージョン 11 で作成されたテキストフィールドでは、デフォルトでチェックはされません。

サブテーブルの変換

バージョン 11 から、サブテーブルは 4D でサポートされなくなります。新規データベースでは、サブテーブルの作成はできません。

変換されたデータベースでは、互換性のために、既存のサブテーブルは今までと同様に動作します。変換時に特別な処理が適用され、サブテーブルは特別な自動リレーションが設定された通常のテーブルに変換されます。この段落では、変換がどのように機能するか、また 4D v11 でのサブテーブルの動作を説明します。

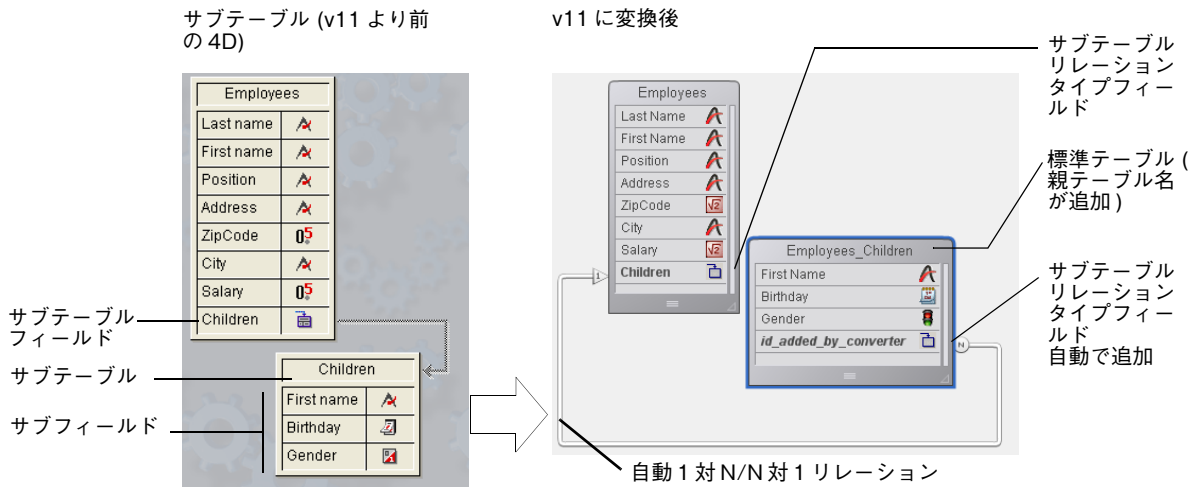
注：変換ウィザードは1レベルを超えるサブテーブルをサポートしません。元のデータベースが1レベルを超えるサブテーブルを含む場合、それらのサブテーブルは無視され、エラー1012が *DataConversion_Log.log* ログファイルに記録されます。

変換メカニズム

データベースがバージョン 11 に変換されると、すべてのサブテーブルが自動的に、親テーブルに自動リレートが設定された標準のテーブルに変換されます。サブテーブルは N テーブルとなり、親テーブルが 1 テーブルになります。

親テーブルでは、サブテーブル型のフィールドが特別な "サブテーブルリレーション" タイプフィールドに変換されます。標準のテーブルとなったサブテーブルは、"Table_Field" に名称変更されます (Table は親テーブル名、Field は親テーブルのサブテーブルフィールド名)。この名称は 31 文字を超えると、越えた部分が削除されます。

変換されたサブテーブルのそれぞれに、"サブテーブルリレーション" タイプの特別なフィールドが追加されます。このフィールドは親テーブルとの自動リレートのために使用され、名前は "<Table>_Relation" のようになります (Table は親テーブル名)。



注： "サブテーブルリレーション" タイプのフィールドを手動で設定することはできません；これはサブテーブルを持つデータベースが変換される時のみ生成されます。

新規に作成されたりレーションは自動で "Auto_Field_Table" と "Auto_Table_Field" という名前がつけられます。

注： リレーション名に関する詳細は、[120 ページ "リレーションプロパティ" の節](#)を参照してください。

変換後、サブテーブル特有の機能（自動機能、ランゲージコマンド、その他）は完全に機能します。しかしながら自動リレーションを使用した他のリレート機能も、変換したサブテーブルで利用可能なことを覚えておいてください。将来のために、変換されたサブテーブルでも標準のメカニズムを使用されることをお勧めします。

変換されたサブフォームを標準のリレートされたテーブルに変更する

変換されたサブテーブルを、標準のリレーションが設定されたテーブルに変更できます。これを行うには、特別な自動リレーションを削除します。

このリレーションを削除すると、サブテーブルの特別な機能や自動的な操作は完全に失われます。このリレーションを削除する際には、この操作の結果を警告するダイアログが表示され、必要であれば操作をキャンセルすることができます。

リレーションが取り除かれると、2つの"サブテーブルリレーション"タイプのフィールドは標準の倍長整数タイプのフィールドに変更されます。

リソースの管理

4D v11 ではリソースの管理が変更されました。

Apple 社の推奨および最近の Mac OS バージョンの実装に適合させるため、本来の意味でのリソースのコンセプトはサポートされなくなり、徐々に使用されなくなります。

リソースが使用されたニーズをサポートするため、新しいメカニズムが実装されました：文字列の翻訳に使用する XLIFF ファイル、.png ピクチャファイルなどです。リソースファイルは標準タイプのファイルに置き換えられていきます。

4D v11 はこの進化をサポートし、現在のシステムの互換性を保ちつつ、データベースの翻訳を管理するための新しいツールを提供します。

既存のリソースの互換性

互換性を保ちつつ、既存のアプリケーションを徐々に対応させることを可能にするため、幾つかの差異を除き、4D v11 でも旧来のリソースメカニズムは引き続き動作します：

- 4D はリソースファイルをサポートし、"リソースファイルチェーン"（複数リソースファイルの一連のオープン）はいぜんとして有効です。"リソースファイルチェーン"には、自動的に開かれる変換されたデータベースの.rsr や.4dr ファイルと、"リソース"テーマのコマンドを使用して開かれるカスタムリソースが含まれます。

- しかしながら、内部的なアーキテクチャの変更のため、"リソース"テーマのコマンドやダイナミック参照を使用して、4D アプリケーションのリソースやシステムのリソースに直接アクセスすることはできなくなりました。開発者の中には 4D の内部リソースをインターフェースに利用している方もいます (例えば月の名前やランゲージコマンド名を含むリソースなど)。既に推奨されていないこの方法は、今バージョンで明確に禁止されます。ほとんどのケースで、4D の内部リソースの代わりに他の方法 (定数, ランゲージコマンド) などを使用できます。既存のデータベースにおけるこの変更によるインパクトを制限するため、最も頻繁に使用されるリソースを公開する代わりにの方法が実装されました。しかしながら変換されたデータベースを変更して、4D の内部リソースをコールする部分を取り除くことを、強くお勧めします。

注： 4D の特定の内部リソースを使用する場合は、それをユーザーリソースファイルにコピーして、ランゲージコマンドで明示的にそのファイルをロードします。

- 4D v11 で作成されたデータベースにはデフォルトで、.RSR (ストラクチャリソース) や .ADR (データリソース) ファイルは含まれません。

リソース管理の新しい原則

4D v11 では、"リソース"という用語はより広い意味である、"アプリケーションインターフェースの翻訳に必要なファイル"と理解されなければなりません。

リソースの新しいアーキテクチャは、ストラクチャファイル (.4db or .4dc) と同階層に置かれる Resources という名前のフォルダに基づきます。

このフォルダには、翻訳や、アプリケーションインターフェースのカスタマイズに必要なすべてのファイル (ピクチャファイル, テキストファイル, XLIFF ファイルなど) が置かれます。

またカスタムの "前世代" データベースリソースファイル (.rsr ファイル) を置くこともできます。これらのファイルは自動ではリソースチェーンに組み込まれないことに注意してください; これらは 4D のリソースコマンドを使用して開く必要があります。

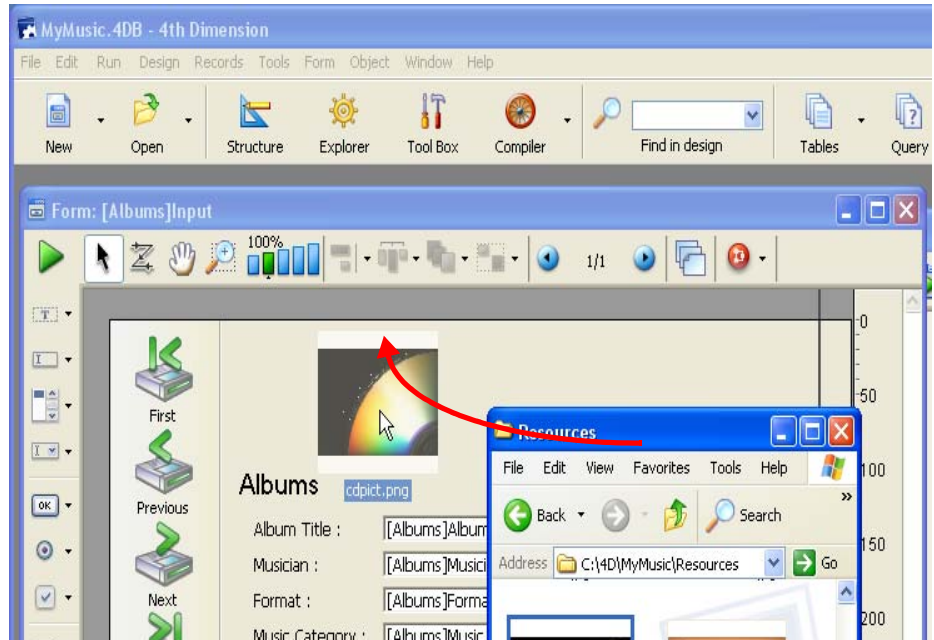
4D はこのフォルダに含まれるファイルを処理する際、特に XLIFF ファイルを使用する際には、自動メカニズムを使用します (98 ページ "XLIFF 標準のサポート" の節を参照)。

ピクチャの自動参照

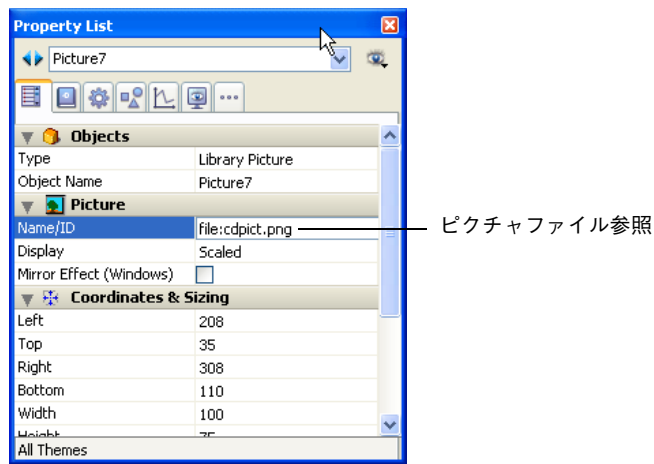
フォームで使用するピクチャを Resources フォルダに格納し、参照によってそれらを使用することができます。ピクチャをライブラリにおく必要はありません。これによりピクチャの表示が最適化されます。

特に .png (bitmap) や .svg (ベクター) ピクチャを使用することができます。4Dにおけるピクチャのネイティブサポートについては、[157 ページ "ネイティブフォーマットのサポート"](#) の節を参照してください。

フォームにこれらのピクチャを挿入するには、Resources フォルダからフォームにピクチャをドラッグ&ドロップします。:



4D は自動で、以下のようなピクチャ参照をフォームに挿入します:
"file:{pathname+}filename":



Resources フォルダのサブフォルダにピクチャファイルを格納することもできます。特に lproj folder メカニズムを使用して異なる言語用にピクチャを用意することができます。

新しいコンポーネントアーキテクチャ

4D のコンポーネントアーキテクチャは 4D v11 で大きく変更され、開発とインストールがより簡易になりました。同時に最大限のセキュリティも提供されています。

"以前の世代" のコンポーネントは 4D v11 でサポートされません。詳細は [23 ページ "コンポーネントを含むデータベースの変換"](#) の節を参照してください。

概要

4D のコンポーネントは、異なるデータベースにインストール可能な、1 つ以上の機能を持つ 4D オブジェクトの集合です。

以前のバージョンの 4D では、4D Insider を使用してコンポーネントを作成、生成、そしてインストールしていました。今バージョンでは 4D コンポーネントの作成とインストールは直接 4D v11 を使用して行われます。

4D v11 では、コンポーネントはプラグインのように扱われ、以下の原則が適用されます：

- コンポーネントは、標準のアーキテクチャまたはパッケージの形をした通常のストラクチャファイル (コンパイルまたは非コンパイル) で構成されます ([62 ページ "4dbase フォルダ"](#) の節を参照)。
- データベースにコンポーネントをインストールするには、データベースのストラクチャファイルと同階層に置かれる "Components" フォルダにコンポーネントをコピーします。ショートカット (Windows) やエイリアス (Mac OS) を使用することもできます。アンインストールするには、コンポーネントをフォルダから取り除きます。
- 以前のバージョンと異なり、4D v11 のコンポーネントにテーブルを含めることはできません。

これらの原則について、以下で詳細に説明します。

新しいコンセプト (定義)

4D v11 における新しいコンポーネント管理メカニズムでは、以下の用語とコンセプトを知っている必要があります：

- マトリクスデータベース: コンポーネント開発に使用する 4D データベース。マトリクスデータベースは特別な属性を持たない標準のデータベースです。マトリクスデータベースはひとつのコンポーネントを構成します。マトリクスデータベースは、コンポーネントを使用するデータベース (ホストデータベース) の Components フォルダにコピーされます。コンパイルされていなくてもかまいません。
 - ホストデータベース: (Components フォルダに) コンポーネントがインストールされるデータベース。
 - コンポーネント: ホストデータベースの Components フォルダにコピーされ、その内容がホストデータベースで使用されるマトリクスデータベース (コンパイル済みまたは非コンパイル)。
 - プロジェクトフォーム: テーブルにリンクされていないフォーム。プロジェクトフォームは 4D v11 の新しい機能で、特にコンポーネントの生成に使用できます。プロジェクトフォームに関する詳細は [135 ページ "プロジェクトフォーム" の節](#) を参照してください。
-
- 注: 4D v11 では、データベースに最低一つのテーブルを持つことは必須でなくなります。
-

- テーブルフォーム ("標準"フォーム): テーブルに属するフォーム。このタイプのフォームをコンポーネントで使用することはできません。

データベースは "マトリクス" にも "ホスト" にもなりえます。言い換えれば、マトリクスデータベースが 1 つ以上のコンポーネントを使用できます。しかしコンポーネントが "サブコンポーネント" を使用することはできません。

コンポーネントの保護: コンパイル

コンポーネントとしてインストールされたマトリクスデータベースのプロジェクトメソッドは、デフォルトでホストデータベース上で見ることが可能です。特に:

- 共有プロジェクトメソッドがエクスプローラの "メソッド" ページに存在し、かつホストデータベースのメソッドから呼び出し可能な場合 ([54 ページ "プロジェクトメソッドの共有" の節](#) を参照)。エクスプローラのプレビューエリアでそれらの内容を選択してコピーすることが可能です。またデバッガでも内容を見ることができます。しかしそれらをメソッドエディタ上で開いたり編集したりすることはできません。
- マトリクスデータベースの他のプロジェクトメソッドはエクスプローラに現れません。しかしホストデータベースのデバッガには内容が表示されます。

コンポーネントのプロジェクトメソッドを効果的に保護するには、マトリクスデータベースをコンパイルして、インタプリタコードを含まない .4dc ファイルとして提供します。コンパイルされたマトリクスデータベースがコンポーネントとしてインストールされると：

- 共有プロジェクトメソッドはエクスプローラの " メソッド " ページに表示され、ホストデータベースのメソッドから呼び出すことができます。しかしその内容はプレビューエリアにもデバッグにも表示されません。
- マトリクスデータベースの他のプロジェクトメソッドは全く表示されません。

コンポーネントのインストール

4D v11 データベースにコンポーネントをインストールするには、マトリクスデータベースのストラクチャをホストデータベースの Components フォルダにコピーします。エイリアス (Mac OS) やショートカット (Windows) を使用することもできます。

クライアント-サーバモードではこの処理は透過的に行われます。

Components フォルダ

Components フォルダはホストデータベースのストラクチャファイルと同階層に置かれなければなりません。コンパイルされ、4D Volume Desktop とマージされるデータベースの場合、Components フォルダは実行ファイルと同階層 (Windows)、およびパッケージの Contents フォルダの中 (Mac OS) に置かれます。

4D は Components フォルダ内で、.4db (インタープリタマトリクスデータベース)、.4dc (コンパイル済みマトリクスデータベース) または .4dbase (パッケージタイプのマトリクスデータベース、[62 ページ "4dbase フォルダ" の節](#)を参照) マトリクスデータベースを探します。データファイルやユーザストラクチャファイル (.4DA) などは無視されます。

これらのマトリクスデータベースのショートカットやエイリアスを使用できます。これはコンポーネントの開発時に便利です。マトリクスデータベースに対する更新はすぐにホストデータベースで有効になります。

Components フォルダに、それ自身がホストデータベースである他のデータベースのショートカット (Windows) やエイリアス (Mac OS) を置くことができます。エイリアスを使用することで、ホストデータベースがコンポーネントとなったり、またその逆が可能となります。この場合、1つのレベルのコンポーネントのみがロードされることに注意してください。それ自身がコンポーネントとして使用されるホストデータベースのコンポーネントはロードされません。

Components フォルダには、(xliff, ピクチャなど) コンポーネントの動作に必要なカスタムファイルやフォルダを置くことができます。他方、プラグインや Component サブフォルダを置くことはできません。これらが置かれていた場合、4D はそれらを無視します。

コンポーネントで使用されるプラグインは、ホストデータベースかまたは 4D にインストールします。

インタプリタ / コンパイル済み / Unicode

インタプリタモードで動作するホストデータベースは、インタプリタまたはコンパイル済みどちらのコンポーネントも、また Unicode モードであっても無くても、使用できます (60 ページ "Unicode のサポート" の節を参照)。一つのホストデータベースに、インタプリタとコンパイル済み両方のコンポーネントをインストールできます。しかし、複数のコンパイル済みコンポーネントが存在する場合、それらは同じ Unicode のモードで実行されなければなりません。

コンパイルモードで実行されるホストデータベースでは、インタプリタのコンポーネントを使用できません。この場合、コンパイル済みコンポーネントのみが利用可能です。また Unicode のモードはホストデータベースとコンポーネントで同じでなければなりません。

以下の表でこの点についてまとめます：

		インタプリタコンポーネント		コンパイル済みコンポーネント	
		Unicode	非Unicode	Unicode	非Unicode
インタプリタホストデータベース	Unicode	○		○ (*)	
	非Unicode				
コンパイル済みホストデータベース	Unicode	-	-	○	-
	非Unicode	-	-	-	○

(*) 複数のコンパイル済みコンポーネントがインストールされる場合、それらは同じ Unicode モードで動作しなければなりません。

注：・インタプリタコンポーネントがインストールされたインタプリタホストデータベースは、それがインタプリタコンポーネントのメソッドを呼び出さなければコンパイルできます。そうでない場合、コンパイル...メニューを選択すると、警告ダイアログが表示され、コンパイルはできません。
・インタプリタメソッドはコンパイル済みメソッドを呼び出せます。が逆はできません。これを行うためには、**EXECUTE METHOD** や **EXECUTE FORMULA** コマンドを使用します。

コンポーネント間及びホストデータベースとコンポーネント間の相互作用については 53 ページ "コンポーネントとホストデータベースの相互作用" の節を参照してください。

Mac OS / Windows

Mac OS で開発されたインタプリタコンポーネントを Windows 環境にインストールすること、及びその逆が可能です。
他方、コンパイルされたコンポーネントは、コンパイルされたのと同じプラットフォームでのみ利用可能です。両プラットフォームで利用可能にするためには、両プラットフォーム用にコンパイルします。

クライアント - サーバ

サーバデータベースにインストールされたコンポーネントは、プラグインと同様のメカニズムを使用して、自動でクライアントマシンに転送されます。

他方、クライアント / サーバモードで、クライアント側のコンポーネントのコピーを更新することはお勧めできません。変更はローカルに保存され、サーバマシンのコンポーネントは更新されません。

スタートアップ時のコンポーネントのロード

コンポーネントはホストデータベースが開かれるときにロードされます。

コンポーネントが一致しないコンパイル済みコードとインタプリタコードを含んでいる場合、エラーメッセージが表示され、そのコンポーネントはホストデータベースにロードされません。

スタートアップ時にコンポーネントが失われている場合、ホストデータベースは通常通り開かれます。これはオプションのコンポーネントを使用するアプリケーションを作成できることを意味します。新しい **COMPONENT LIST** コマンドを使用して、コンポーネントが存在するか調べることができます。

名前の衝突 (メソッドのマスク)

他の共有オブジェクトと異なり (53 ページ "共有及び非共有オブジェクト" の節を参照)、共有プロジェクトメソッドはデータベースに "物理的に" 存在します。つまりコードの実行時に作成されるわけではありません。

ゆえに、コンポーネントの共有プロジェクトメソッドと、ホストデータベースのプロジェクトメソッドが同じ名前を持つと、名前の衝突が発生します。この場合、ホストデータベースのコンテキストでコードが実行されると、ホストデータベースのメソッドが呼び出されます。これはつまり、コンポーネントメソッドをカスタムメソッドで "マスク" して、例えば異なる機能を実装することが可能であることを示しています。もちろん、コンポーネントの中でコードが実行されている場合は、コンポーネントのメソッドが呼び出されます。

このマスクングは、ホストデータベースのコンパイル時に警告によって通知されます。

注： 2 つのコンポーネントが同じ名前のメソッドを持つ場合は、ホストデータベースのコンパイル時にエラーが生成されます。

コンポーネントの開発

コンポーネントは 4D データベースの形で構成されるので、コンポーネントの開発はデータベースへの機能実装と同じです。

しかしながら、コンポーネントの持つ性質のため、制限や特別なルールがあります。

利用可能又は利用不可能なオブジェクト

コンポーネントはプロジェクトメソッド、プロジェクトフォーム、メニューバー、選択リスト、ライブラリピクチャなど、4D のほとんどのオブジェクトを呼ぶことができます。

以下のオブジェクトのみ、コンポーネントで使用することはできません：

- テーブルとフィールド。
- テーブルフォームとそのフォームメソッド（他方コンポーネントはホストデータベースのテーブルフォームを呼び出すことはできます）。
- ユーザフォーム。
- データベースメソッドとトリガ

マトリクスデータベースにこれらの要素が含まれてい折る場合でも、それらを削除する必要はありません。利用できないオブジェクトは単に無視されます。

注： マトリクスデータベースに設定されたユーザ & グループやアクセス権は、ホストデータベースでは無視されます。

コンポーネント側で "共有" されているプロジェクトメソッドのみがホストデータベースのデザインモードに表示され、選択することができます。他方、ホストデータベース側で "共有" されているプロジェクトメソッドをコンポーネントから呼ぶことが可能です。この件に関する詳細は、[54 ページ "プロジェクトメソッドの共有"](#) の節を参照してください。

他のコンポーネントオブジェクト (プロジェクトフォーム、選択リスト、メニューなど) はコンポーネントで利用することができますが、ホストデータベースのからストラクチャオブジェクトとしてアクセスすることはできません。ホストデータベースとコンポーネント間で、あるオブジェクトは分離され、他のオブジェクトは共有されることに注意してください。この件に関する詳細は、[53 ページ "共有及び非共有オブジェクト"](#) の節を参照してください。

コンポーネントは 4D アプリケーション又はホストデータベースにインストールされたプラグインを使用することができます。コンポーネントフォルダにプラグインをインストールすることはできません ([47 ページ "コンポーネントのインストール"](#) の節を参照)。

マトリクスデータベースのデータベースメソッドや、一般設定 (Web フォルダ、環境設定、その他) は一切参照されません。

使用できないコマンド

(読み込みのみで開かれるため、) ストラクチャファイルを更新する以下のコマンドをコンポーネントで使用することはできません。コンポーネント中で以下のコマンドを実行すると -10511, "CommandName コマンドをコンポーネントでコールすることはできません" のエラーが生成されます：


- ON EVENT CALL

- Method called on event
- SET PICTURE TO LIBRARY
- REMOVE PICTURE FROM LIBRARY
- SAVE LIST
- ARRAY TO LIST
- EDIT FORM
- CREATE USER FORM
- DELETE USER FORM
- CHANGE PASSWORD
- EDIT ACCESS
- Set group properties
- Set user properties
- DELETE USER
- CHANGE LICENSES
- BLOB TO USERS
- SET PLUGIN ACCESS

注： Current form table コマンドは、プロジェクトフォームのコンテキストで呼び出されると Nil を返します。ゆえにこのコマンドをコンポーネントで使用することはできません。

フォームの使用

特定のテーブルに属さない "プロジェクトフォーム" のみが、コンポーネントで利用できます。マトリクスデータベースのすべてのプロジェクトフォームをコンポーネントで使用することができます。

マトリクスデータベースでプロジェクトフォームを作成するには、エクスプローラの "フォーム" ページに移動して、プロジェクトフォームを選択し、追加ボタンをクリックします 。プロジェクトフォームは、連結メニューバーやフォームメソッドなど、標準のフォームと同じプロパティを持っています。

コンポーネントのプロジェクトフォームを、ホストデータベースでのデザインモードで表示させたり、明示的に呼び出すことはできません。これらのフォームはエクスプローラの "フォーム" ページやメソッドエディタには現れません。これらはコンポーネントのプロジェクトメソッドからのみ呼び出すことができます。

プロジェクトフォームに関する詳細は、[135 ページ "プロジェクトフォーム" の節](#)を参照してください。

コンポーネントはホストデータベースのテーブルフォームを使用できません。この場合、コンポーネントのコードでフォームを指定する際に、テーブル名ではなく、テーブルへのポインタを使用しなければなりませんことに注意してください。

注： コンポーネントで ADD RECORD コマンドを使用すると、ホストデータベースのコンテキストで、ホストデータベースのカレントの入力フォームが表示されます。その結果、フォームに変数が含まれていると、コンポーネントはその変数にアクセスできません (53 ページ "コンポーネントとホストデータベースの相互作用" の節を参照)。

リソースの使用

コンポーネントはリソースを使用することができます ("伝統的な" Mac OS リソースや XLIFF タイプのファイル)。

新しいリソース管理の原則に従い (42 ページ "リソースの管理" の節を参照)、コンポーネントのリソースファイルは、コンポーネントの .4db や .4dc ファイルと同階層の Resources フォルダに置かれなければなりません。コンポーネントが .4dbase 形式の場合 (推奨されるアーキテクチャ)、Resources フォルダは .4dbase フォルダの中に置かれます。

自動メカニズムが有効となり、コンポーネントの Resources フォルダ内で見つかった XLIFF ファイルは、このコンポーネントによってロードされます。(互換性のため) コンポーネントは、.4db や .4dc と同階層にある .rsr ファイルに格納された "伝統的な" Mac OS リソースも自動的に使用します。

Resources フォルダに置かれた "伝統的な" リソースファイルは、"リソース" テーマのコマンドを使用して、明示的にコンポーネントにロードしなければなりません。

1 つ以上のコンポーネントを含むホストデータベースでは、ホストデータベースと同様それぞれのコンポーネントが固有のリソースチェーンを持っています。リソースは異なるデータベース間で分離されます。コンポーネント A のリソースにコンポーネント B やホストデータベースからアクセスすることはできません (53 ページ "共有及び非共有オブジェクト" の節を参照)。

コンポーネントのオンラインヘルプ

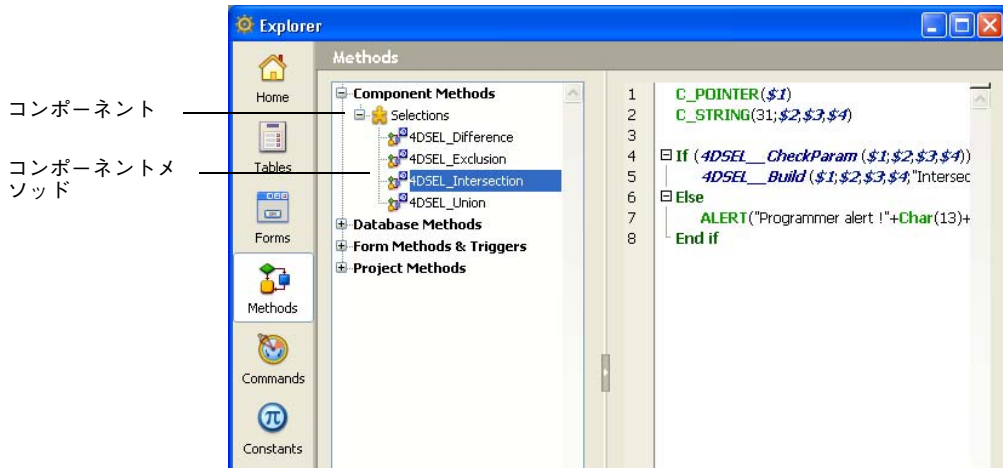
コンポーネントにオンラインヘルプを追加できるようにするために、特別なメカニズムが実装されました。原理は 4D データベースに提供されているものと同じです：

- コンポーネントヘルプは拡張子が .htm, .html または (Windows のみ) .chm で提供される。
- ヘルプファイルはコンポーネントのストラクチャファイルと同階層に置かれる。
- 結果、このファイルは自動でアプリケーションのヘルプメニューに、" ヘルプ : ヘルプファイル名" のタイトルでロードされます。

コンポーネントとホストデータベースの相互作用

コンポーネントの表示

ホストデータベースにコンポーネントがインストールされると、ホストデータベースのエクスプローラの方法ページ内、コンポーネントメソッドテーマにコンポーネント名が表示されます。共有プロジェクトメソッドが階層リストとして表示され、コンポーネントがインタプリタであれば、プレビューエリアに内容が表示されます。



注：共有メソッド定義についての詳細は、[54 ページ " プロジェクト メソッドの共有 "](#)の節を参照してください。

共有及び非共有オブジェクト

コンポーネントで定義される特定のタイプのオブジェクトは、自身の実行空間で展開されます。これによりホストデータベースや他のコンポーネントのオブジェクトとのコンフリクトの可能性を取り除いています。これらのオブジェクトは "非共有または"分割された"オブジェクトと呼ばれます。例えば変数は分割されたオブジェクトであり、あるコンポーネントにおける倍長整数タイプの <>Myvar 変数は、ホストデータベースや他のコンポーネントで使用されるテキストタイプの <>Myvar 変数と同時に使用することができます。

他方のオブジェクトはホストデータベースとコンポーネントで同じ実行空間を共有します。これらのオブジェクトを使用するにはより注意が必要となりますが、他方ホストデータベースとコンポーネント間で通信を行うことが可能です。これらのオブジェクトは "共有" または "分割されない" オブジェクトと呼ばれます。

例えばセットは分割されないオブジェクトであり、コンポーネントが作成した "mySet" を、ホストデータベースで実行される CLEAR SET("mySet") で消去することが可能です。

- 以下は非共有 (分割された) オブジェクトです:
 - スタイルシート
 - ヘルプ Tips
 - 選択リスト
 - ライブラリピクチャ
 - メニューエディタで作成されたメニューやメニューバー
 - " コンポーネントとホストデータベースで共有する " 属性を持たないプロジェクトメソッド
 - セマフォ
 - プロセス
 - 変数 (ローカル, プロセス, インタープロセス)
 - システム変数 (OK, Document, など)
 - プロジェクト及びテーブルフォーム
 - リソースと開いたリソースファイルの参照
- 以下は共有 (分割されない) オブジェクトです:
 - セット
 - 命名セレクション
 - 参照を使用した階層リスト (Qhz #hw Ordg#hw Frs|#hwまたは EORE#r#hw コマンドで作成されたもの)
 - 新しい **Create menu** コマンドから返される ID を使用したメニューやメニューバー
 - " コンポーネントとホストデータベースで共有する " 属性を持つプロジェクトメソッド
 - XML ストラクチャ参照
 - ファイル参照
 - Open resource file references (except for resources files)
 - ポインタ

注: もちろん、マトリクスデータベース内で見つかった利用できないオブジェクトは、ホストデータベース側で無視されます (50 ページ " 利用可能又は利用不可能なオブジェクト " の節を参照)。

プロジェクト メソッドの共有

マトリクスデータベースのすべてのプロジェクトメソッドは、コンポーネントに含まれます。つまり、マトリクスデータベースのプロジェクトメソッドは、すべて自身のコンポーネントから呼び出して実行することができます。

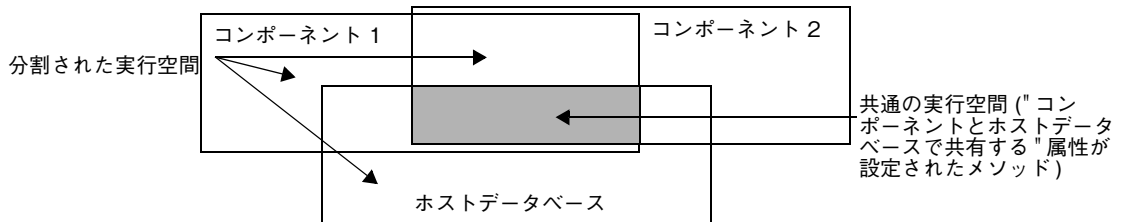
他方、デフォルトで、これらのプロジェクトメソッドはホストデータベースに表示されず、呼び出すこともできません。ホストデータベースでもプロジェクトメソッドを共有するためには、マトリクスデータベース

側でそのメソッドを共有されるよう設定しなければなりません。設定することで、それらのプロジェクトメソッドはホストデータベースのエクスプローラのメソッドページに表示され、(53 ページ "コンポーネントとホストデータベースの相互作用" の節を参照) 呼び出すことができるようになります (しかしホストデータベースのメソッドエディタで編集することはできません)。これらのメソッドはコンポーネントのエントリーポイントとなります。

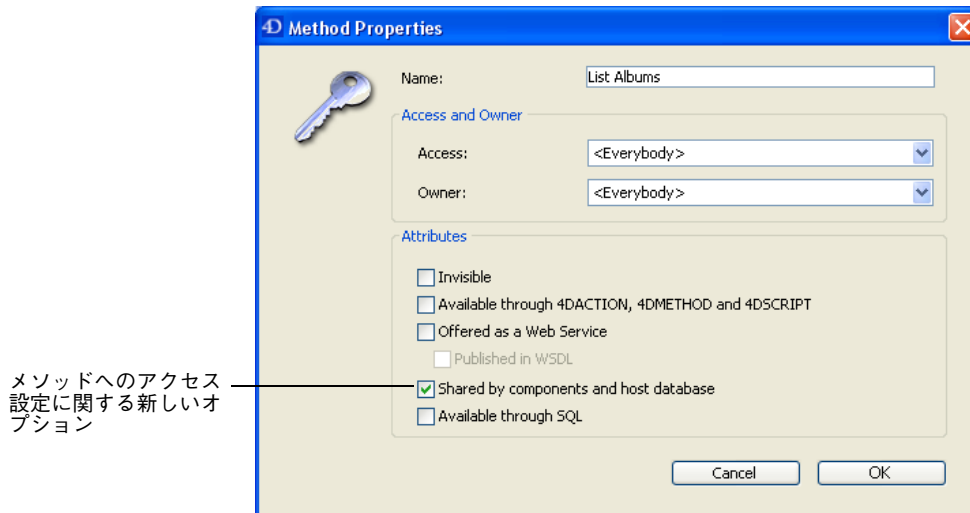
セキュリティのため、デフォルトでコンポーネントはホストデータベースのプロジェクトメソッドを実行することはできません。特定のケースで、コンポーネントがホストデータベースのプロジェクトメソッドにアクセスできるようにする必要があるかもしれません。

そうするためには、ホストデータベースのプロジェクトメソッドで、コンポーネントからのアクセスを可能にするよう明示的に指定しなければなりません。

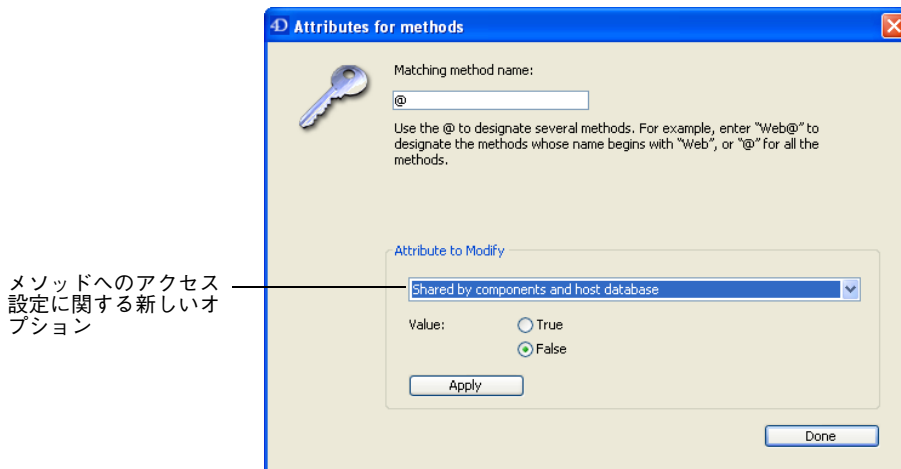
メソッドの共有



これはメソッドプロパティの新しいプロパティ、コンポーネントとホストデータベースで共有するで設定します：



メソッド属性の一括設定ダイアログを使用して、複数のメソッドに対し一気にこの属性を設定することもできます (このダイアログはエクスプローラのコンテキストメニューからアクセス可能です):



このオプションの効果は、データベースがどのように使用されるかにより異なります: データベースがコンポーネントとして利用される場合、メソッドはホストデータベースからアクセス可能で、エクスプローラで見ることができます。データベースがホストデータベースの場合、メソッドはコンポーネントから利用可能です。

変数を渡す

ローカル、プロセス、インタープロセス変数は、コンポーネントとホストデータベース間で共有されません。ホストデータベースからコンポーネントの変数、またはその逆の変数にアクセスする唯一の方法はポインタを使用することです。

▼ 配列を使用した例:

■ ホストデータベース側:

```
ARRAY INTEGER (MyArray;10)
AMethod (->MyArray)
```

■ コンポーネント側の AMethod:

```
ADD ELEMENT($1->; 2)
```

▼ 変数を使用した例:

```
C_TEXT(myvariable)
component_method1(->myvariable)
```

```
C_POINTER($p)
$p:=component_method2(...)
```

ホストデータベースとコンポーネント間でポインタを使用して通信を行うには、以下の点を考慮する必要があります:

- **Get pointer** をコンポーネントで使用した場合、このコマンドはホストデータベースの変数へのポインタを返しません。また逆にこのコマンドをホストデータベースで使用した場合も同様です。
- 新しいコンポーネントアーキテクチャでは、インタプリタデータベースで、インタプリタ及びコンパイル済み両方のコンポーネントを利用できます。(コンパイル済みデータベースでは、コンパイルされたコンポーネントしか使用できません。)

この場合、ポインタの利用は以下の原則を守らなければなりません：インタプリタ版では、コンパイルモードにビルトインされたポインタを解釈できます。逆にコンパイルモードではインタプリタモードにビルトインされたポインタを解釈することはできません。

以下の例でこの原則を説明します：同じホストデータベースにインストールされた2つのコンポーネント C (コンパイル済) と I (インタプリタ) があります：

- コンポーネント C が定義する変数 `myCvar` があるとき、コンポーネント I はポインタ `->myCvar` を使用して変数の値にアクセスすることができます。
- コンポーネント I が定義する変数 `myIvar` があるとき、コンポーネント C はポインタ `->myIvar` を使用しても変数の値にアクセスすることはできません。このシンタックスは実行時エラーを起こします。
- **RESOLVE POINTER** を使用したポインタの比較はお勧めできません。変数の分割の原則により、ホストデータベースとコンポーネント (あるいは他のコンポーネント) で同じ名前を持つ変数が存在することができますが、根本的にそれらは異なる内容を持ちます。両コンテキストで、変数のタイプが違うことさえあります。

ポインタ `myptr1` と `myptr2` がそれぞれ変数を指すとき、以下の比較は正しくない結果となるかもしれません：

```
RESOLVE POINTER(myptr1;vVarName1;vtablename1;vfieldnum1)
RESOLVE POINTER(myptr2;vVarName2;vtablename2;vfieldnum2)
If(vVarName1=vVarName2)
```

　　`変数が異なっているにもかかわらず、このテストは True を返します。

このような比較を行うためには、ポインタを比較しなければなりません：

```
If(myptr1=myptr2)
```

　　`このテストは False を返します

注： プロセス変数へのポインタの管理が 4D v11 で変更されました ([360 ページ "プロセス変数へのポインタ" の節](#)を参照)。

ホストデータベースの テーブルへのアクセス

コンポーネントでテーブルを使用することはできませんが、以下のコマンドをコンポーネントの中で呼び出すことはできます：

- DEFAULT TABLE
- **NO DEFAULT TABLE**
- Current default table

実際、コンポーネントがホストデータベースのテーブルを使用する必要がある場合に、これらのコマンドはとても便利です。この方法で、ホストデータベースとコンポーネントはポインタを使用して通信を行うことができます。

例えば、以下はコンポーネントで実行可能なメソッドです：

```
C_LONGINT($1) `ホストデータベースのテーブル番号
$stablepointer:=Table($1)
DEFAULT TABLE($stablepointer->)
CREATE RECORD `ホストデータベースのデフォルトテーブルを使用
$fieldpointer:=Field($1;1)
$fieldpointer->:="value"
SAVE RECORD
UNLOAD RECORD
```

ランゲージコマンドのス コープ

禁止されているコマンドを除き (50 ページ "使用できないコマンド" の節を参照)、コンポーネントではすべての 4D ランゲージコマンドが使用できます。

コマンドがコンポーネントから呼ばれると、コマンドはコンポーネントのコンテキストで実行されます。ただし新しい **EXECUTE METHOD** コマンドは除きます。このコマンドはコマンドで指定されたメソッドのコンテキストを使用します。また "ユーザ & グループ" テーマの読み出しコマンドをコンポーネントで使用することができますが、ホストデータベースのユーザ & グループ情報を読み出すことに注意してください (コンポーネントに固有のユーザ & グループはありません)。

SET DATABASE PARAMETER と **Get database parameter** コマンドは例外となります：これらのコマンドの範囲はグローバルです。これらのコマンドがコンポーネントから呼び出されると、結果はホストデータベースに適用されます。

さらに、**Structure file** と **Get 4D folder** コマンドは、コンポーネントで使用できるように変更が行われています (see the 329 ページの **Structure file** コマンド と 329 ページの **Get 4D folder** コマンドを参照)。

注：新しい **FRP SRQHQW#DIVW** コマンドを使用して、ホストデータベースにロードされたコンポーネントのリストを取得できます (328 ページの **COMPONENT LIST** コマンドを参照)。

デバッグ

コンパイルされていないコンポーネントを使用すると、コードの内容がホストデータベースの標準デバッガに表示されます。

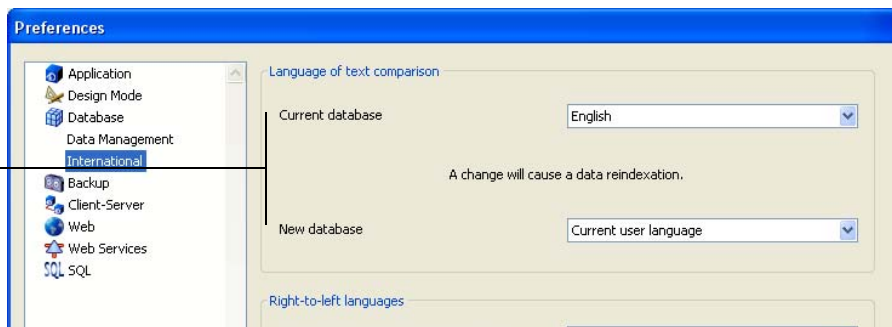
デバッガは分割されたオブジェクトの実行空間を考慮します。カスタムウォッチペインでホストデータベースの変数 `var1` の値を表示し、同じく `var1` 変数を含むコンポーネントのコードを実行しても、表示される値は更新されません。カレントコンテキストの変数値を表示させるためには、変数のインスタンスを別にカスタムウォッチペインに登録しなければなりません。

ランゲージの管理

文字列の比較や処理に使用されるランゲージのサポートが、4D v11 で拡張されました。

ランゲージ設定は環境設定ダイアログボックスのデータベース / インターナショナルページで行います。

ランゲージ選択オプション



デフォルトで、4D はシステムランゲージを使用します。ランゲージの選択は直接、テキストの検索や並び替え、大文字小文字の比較に影響します。ただしテキストの翻訳や日付、時刻、通貨フォーマットには影響せず、これらはシステムランゲージ設定が使用されます。

開いたデータベースのランゲージを選択すること (そのデータベース限定)、および新規データベースのデフォルトランゲージの選択が可能です。しかしながらランゲージの選択肢は、TRIC リソースベースから ICU ライブラリベースになったため、以前のバージョンに比べて多くなりました。

以前のバージョンと異なり、4D データベースはシステムと異なるランゲージで動作が可能です。データベースが開かれると、4D エンジン はデータファイルで使用されるランゲージを決定し、(インタープリタやコンパイルモードの) 言語に提供します。テキストの比較は、それがデータベースエンジンまたは言語で行われるかにかかわらず、同じランゲージを使用して行われます。

環境設定ダイアログボックスでランゲージを選択し、それを有効にした後は、ランゲージコードはストラクチャとデータに保存され、インデックスが即座に再構築されます。テキストの比較が正しくこのランゲージを使用して行われるようにするためには、データベースを再起動しなければなりません。

新しいデータファイルを作成する際、4D は環境設定で指定されたランゲージ設定を使用します。ストラクチャと異なるランゲージ設定のデータファイルを開くと、データファイルに設定されたランゲージが使用され、そのランゲージコードがストラクチャにコピーされます。

Unicode のサポート

4D v11 では Unicode 文字セットのサポートが拡張されました。今バージョンから、データベースエンジンとランゲージは Unicode 文字列をネイティブで処理します。この新しい機能により文字列処理の速度が改善され、4D アプリケーションの国際化が促進されます。

互換性を保つため、環境設定のオプションを使用して、4D v11 を以前のモードで動作させることもできます。

Unicode とは？

Unicode は統合された標準文字セットで、実質的にすべての一般的な世界中の言語を扱うことができるものです。文字セットは文字と番号の対応表です。例えば "a" -> 97, "b" -> 98, "5" -> 53 などです。

以前のバージョンの 4D では、原則として 1 つの言語に対し 1 つの文字セットが利用可能でした。4D は起動時に、システム言語に対応する文字セットを選択します。つまり、同時に複数の言語を管理することは不可能でした。

us-ascii では基本的な文字コードが 1 から 127 の間に含まれているのに対し、Unicode の上限は 65,000 を超えていて、実質上すべての言語のすべての文字を表現できます。

Unicode には幾つかの種類があります: UTF-16 は 16 ビット整数で、UTF-32 は 32 ビット整数で、UTF-8 は 8 ビット整数で符号化を行います。UTF-7 もあります。

4D v11 は主に (Windows や Mac OS のように) UTF-16 を使用します。時にはインターネットのニーズに合わせ、UTF-8 が使用されることもあります。これは Ascii キャラクタを扱う際によりコンパクトで、可読性があるというメリットがあります。

Unicode の 1 から 127 は完全に us-ascii 文字セットに対応していることに注意してください (Mac and Windows)。これらの文字は最も使用される文字なので、Unicode への変更はほとんど直接的な影響はありません。

4D データベースにおける Unicode の互換性

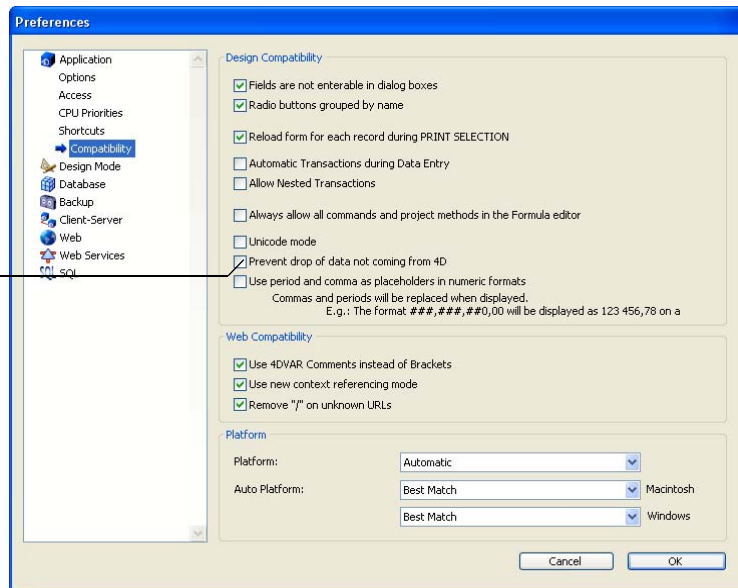
4D アプリケーションの Unicode 文字セットへの変更はユーザに対し透過的です；プログラムは内部的に必要な変換を処理します。

しかしながら、いくつかの言語で、特に文字を扱うコマンドを使用している場合には、多少の変更が必要な場合があります。例えば、Char(200) は Unicode と Ascii で異なる結果を返します。

4D 言語における Unicode サポートの詳細は [247 ページ "Unicode 関連の変更"](#) の節を参照してください。

既存のコードの互換性を保つために、環境設定のアプリケーション / 互換性ページに、4D データベースで Unicode 利用の有効 / 無効を切り替える新しいオプションが追加されました：

データベースで Unicode を有効にする



デフォルトで、前のバージョンの 4D から変換されたデータベースでは、このオプションは選択されていません。4D v11 以降で作成されたデータベースでは、このオプションはデフォルトで選択されます。

このオプションはデータベースごとに固有です。インタープリタモードでは、Unicode データベースで、非 Unicode のコンポーネントを使用したり、あるいはその逆が可能です。4D v11 におけるコンポーネントの詳細は、[45 ページ "新しいコンポーネントアーキテクチャ"](#) の節を参照してください。

注： `Get database parameter` や `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。この件の詳細は [331 ページ "SET DATABASE PARAMETER, Get database parameter"](#) の節を参照してください。

"キーボードレイアウト" プロパティ Unicode モードが有効になると、フォームオブジェクトの "キーボードレイアウト" プロパティは無視されます。

.4dbase フォルダ

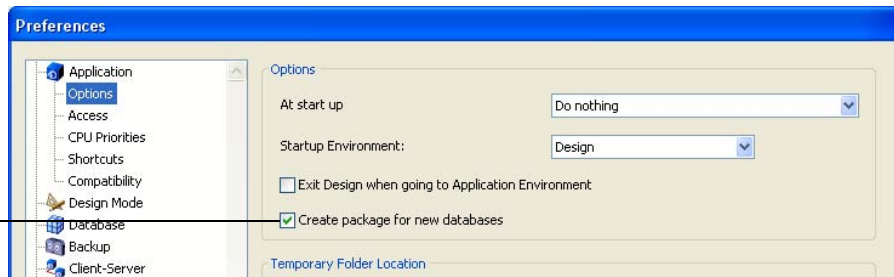
バージョン 11 で作成された 4D データベースは自動で .4dbase という接尾辞のついたフォルダに置かれます。例えば "Invoices" という名前のデータベースは [Invoices.4dbase] フォルダに作成されます。このフォルダにはデータベースの動作に必要なすべての要素が格納されます。

Mac OS ではデータベースフォルダはパッケージとして表示されます。直接パッケージをダブルクリックして 4D やデータベース、カレントのデータファイルを起動することも可能です。またパッケージを 4D アイコンにドラッグアンドドロップすることもできます。このことは、データベースを CVS や Subversion などのバージョン管理システムに置くことができることを意味します。

Windows では、この変更による影響は特にありません。

環境設定の "アプリケーション / オプション" ページにある新規データベースでパッケージを作成する オプションを使用してこのデフォルトの機能を無効にできます：

パッケージを作成するオプション



4

デザインモードのインタフェースとナビゲーション

4D v11 開発環境のインタフェースは大きく作り直され、より簡潔にそして使いやすくなりました。デザインモードとユーザモードは統合され、数多くの新機能が追加されました。

この章では以下の新機能について説明します：

- 4D データベースの作成と開始
- 4D のデザインモードとメニュー
- エクスプローラ
- コンパイラ
- データベースメソッドの実行
- デザインモードでのオブジェクトの移動
- データベース全体の検索 / 置換
- デザインモードでのショートカットの管理
- スタイルシートエディタ
- データベースの翻訳
- 新しい " キーワードを含む " 検索演算子

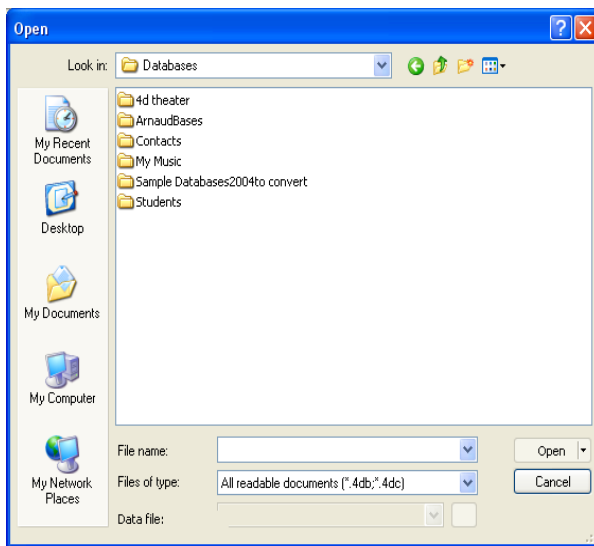
データベースの作成と開始

データベースの作成と開始が 4D v11 で変更されました。これらの操作はシステムダイアログボックスを使用して行われます。

開くダイアログボックス

データベースの作成と開始の手順が、4D v11 で簡素化されました。ようこそダイアログボックスはシステムダイアログボックスに置き換えられました。"お気に入り"のコンセプトは"最近使用したデータベース"と同じものであり、取り除かれました。バックアップファイルに関する情報と、検査のオプションは新しい Maintenance & Security Center (177 ページ "[Maintenance & Security Center](#)" の章を参照) に移されました。

4D アプリケーションアイコンをダブルクリックすると、デフォルトで標準のファイルを開くダイアログボックスが表示されます：



データベースを開くダイアログボックス (Windows)

- 注：
- ・ダイアログボックス上でキャンセルをクリックすると、4D はデータベースを開かずに起動します。その後ファイルメニューから作成や開く、またはヘルプメニューから Maintenance & Security Center コマンドを選択できます。
 - ・データベースを開く際の表示を環境設定で設定できます (67 ページ "[起動時の環境設定](#)" の節を参照)。

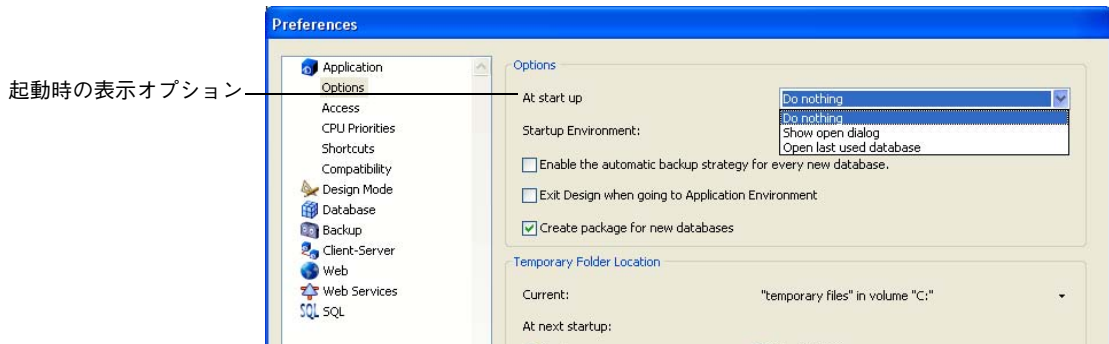
このダイアログボックスには、4D データベースを開く際に選択可能な標準のオプションが用意されています：

- 開く: このメニューはデータベースをコンパイルモードまたはインタプリタモードで開く際に使用します。このメニューで有効なオプションは、選択されたファイルのタイプとコンテンツによって変わります。
- 開くボタンに関連付けられたメニュー: このメニューは、データベースを開くモードを設定するために使用します。以下の選択肢があります: コンパイル済み (コンパイルコードが存在する場合)、インタプリタ (デフォルトモード) そして Maintenance & Security Center (必要に応じて修復を行うために、壊れたデータベースを開くことが可能)。このオプションに関する詳細は、177 ページ "[Maintenance & Security Center](#)" の章を参照してください。
- データファイル: このオプションは、データベースで使用するデータファイルを特定するために使用します。デフォルトでカレントのデータファイルが選択されます。開くをクリックすると、このファイルを使用してデータベースが開きます。

データファイルを変更するには、他のデータファイルを選択オプションを選択して、開くをクリックします。この場合、標準のデータファイルを選択するダイアログボックスが表示され、データファイルを選択または作成することができます。

起動時の環境設定

アプリケーション / オプション ページに、直接 4D Application が起動された場合のデフォルトの表示を設定する、新しい環境設定が追加されました:



以下のオプションを選択できます:

- 何もしない: アプリケーションウィンドウが空の状態が表示されます。
- 開くダイアログを表示 (デフォルトのオプション): 4D は標準のファイルを開くダイアログボックスを表示します。
- 最後に使用したデータベースを開く: 4D は直接最後に使用したデータベースを開きます; 開くダイアログボックスは表示されません。

注：最後に使用したデータベースを開くオプションが選択されている時に、ファイルを開くダイアログボックスを強制的に表示させるには、Alt (Windows) または Option (Mac OS) キーをデータベース起動中に押し続けます。

ファイルメニュー

4D アプリケーションが起動されると、ファイルメニューの新しいコマンドを使用してデータベースを開いたり作成したりできます：

- **新規>テンプレートを使用したデータベース...**：このコマンドはアプリケーションテンプレートウィザードを表示させるために使用します。前のバージョンの 4D では、ようこそダイアログのテンプレートを使用ボタンを使用してウィザードを表示させていました。このウィザードを使用して、様々なテーマにグループ化された定義済みテンプレートから、動作するデータベースを生成することができます。
- **最近使用したデータベースを開く>最近使用したデータベースのリスト**：このコマンドは、このマシン上で過去に開いたデータベースのリストをサブメニューに表示します。メニュークリアコマンドを選択すると、このメニューの内容をリセットできます。
- **データベースを閉じる**：このコマンドを選択すると、カレントのデータベースが閉じられます。4D アプリケーションは起動されていますが、データベースは開かれていません。ファイルメニューから**新規**や**開く**を選択したり、ヘルプメニューから **Maintenance & Security Center** コマンドを選択したりできます。

注：バージョン 11 で作成された 4D データベースは自動で .4dbase 接尾辞がついたフォルダに置かれます。詳細は [62 ページ "4dbase フォルダ"](#) の節を参照してください。

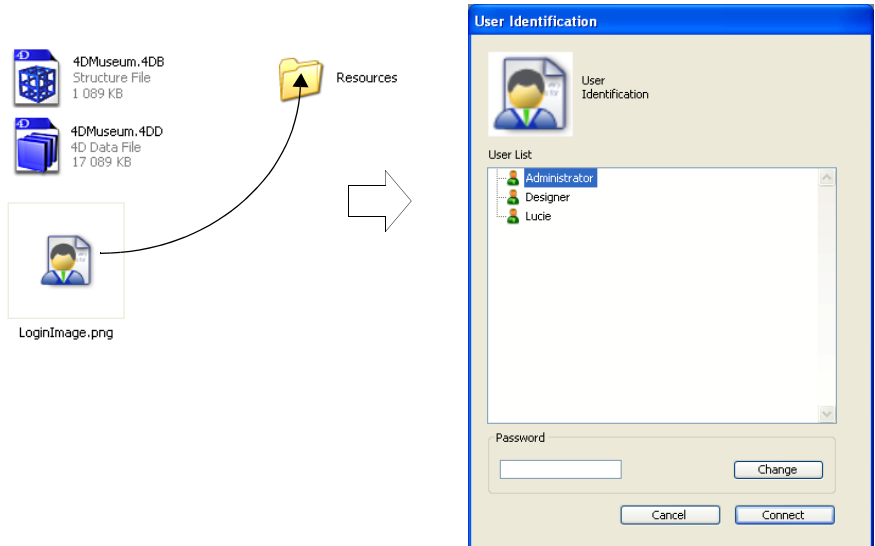
ID ウィンドウアイコンのカスタマイズ

データベース接続ダイアログボックスに表示されるアイコンをカスタマイズできます。デフォルトでアイコンには 4D ロゴが使用されます。

このアイコンを置き換えるには、LoginImage.png という名前のファイルを、(データベースストラクチャファイルと同階層にある) データベースの Resources フォルダに置きます。

カスタムファイルは "png" タイプで、サイズは 80x80 ピクセルでなければなりません。

データベースが開かれると、4D はデフォルトのアイコンの代わりに、このピクチャをロードします



新しいデザインモードインタフェース

4D 開発環境の全体的なインタフェースが変更されました。まず、デザインモードとユーザモードの違いが取り除かれ、カスタムモードの名前が変更されました。今バージョンでは2つのモードのみが存在します: デザイン (このモードにはユーザモードの機能が統合されています) とアプリケーション (以前のカスタムモードに対応) です。

デザインモード

新しいデザインモードは、以前のデザインモードとユーザモードの機能を一緒にしたものです。アプリケーション開発のエディタ (フォーム、メニューなど) やテーブルのレコードにアクセスするためのウィンドウが、1つのメニューバーに収められています。モードメニューを使用したデザインモードとユーザモード間の移動は行われません。代わりに、機能により表示されるウィンドウが決定されます。例えば、フォームエディタで作業を行っている時に、レコードメニューから**新規レコード**を選択すると、カレントテーブルの入力フォームが最前面に表示され、レコードの入力が可能となります。

4D v11 では、すべてのメニューが常に有効です。前のバージョンでは、メニューバーやツールバーはモードによって変更されていました。今バージョンからは、特定のエディタメニューがカレントのメニューバーに追加されます。デザインモードのメニューバーには常に**ファイル**、**編集**、**実行**、**デザイン**、**レコード**、**ツール**そして**ウィンドウ**メニューが表示され、4Dのすべての部分へのアクセスを提供します ([71 ページ "メニューの再構成" の節](#)を参照)。

アプリケーションモード

新しいアプリケーションモードは、前のバージョンのカスタムモードと全く同じです。アプリケーションモードにアクセスするには、**実行**メニューの**アプリケーションテストコマンド**を選択します。このコマンドはアプリケーションに最低1つのカスタムメニューが作成されていると有効になります。

標準アクション

2つの新しい標準アクション、デザインモードに戻るとアプリケーションを使用して、アプリケーションモードとデザインモード間のナビゲーションを管理できます。

バージョン 11 以降で作成されたデータベースでは、ユーザとデザイン標準アクションを選択することはできません。

変換されたデータベースでは、メニューに割り当て済みの標準アクションはそのまま使用できますが、新たにメニューコマンドに割り当てることはできません。詳細は、[176ページ "標準アクション"の節](#)を参照してください。

メニューの再構成

以前のデザインモードとユーザモードを統合したフレームワークにおいて、4D v11 のメニューバーのメニューが再構成されました。今バージョンより、すべてのメニュー（とツールバーのアイコン）はデザインモードで常に有効です。

ファイル	説明	以前のバージョンでのモード
新規 > データベース	前のバージョンと同じ	すべて
新規 > テンプレートを使用したデータベース	テンプレート選択ダイアログボックスを表示 (68 ページ "ファイルメニュー" の節 を参照)	4D v11 の新機能
新規 > ストラクチャ定義を使用したデータベース	XML ストラクチャ定義ファイルからデータベースを作成 (127 ページ "ストラクチャ定義からのデータベース作成" の節 を参照)	4D v11 の新機能
新規 > データファイル	前のバージョンと同じ	ユーザ
新規 > オブジェクトライブラリ	前のバージョンと同じ	デザイン
新規 > テーブル / フォーム / メソッド	前のバージョンと同じ	デザイン
開く > データベース	前のバージョンと同じ	すべて
開く > データファイル	前のバージョンと同じ	ユーザ
開く > オブジェクトライブラリ	前のバージョンと同じ	デザイン
開く > フォーム / メソッド	前のバージョンと同じ	デザイン
最近使用したデータベースを開く	最近使用したデータベースのリストを表示 (68 ページ "ファイルメニュー" の節 を参照)	4D v11 の新機能
データベースを閉じる	カレントのデータベースを閉じる (66 ページ "開くダイアログボックス" の節 を参照)	4D v11 の新機能
閉じる (オブジェクト)	前のバージョンと同じ	デザイン
すべてを閉じる	前のバージョンと同じ	デザイン
保存 (オブジェクト)	前のバージョンと同じ	デザイン

すべてを保存	前のバージョンと同じ	デザイン
データバッファをフラッシュ	前のバージョンと同じ	ユーザ
元に戻す	前のバージョンと同じ	デザイン
読み込み > ファイルから	前のバージョンと同じ	ユーザ
読み込み > ODBC ソースから	前のバージョンと同じ	ユーザ
書き出し > データをファイルに	前のバージョンと同じ	ユーザ
書き出し > データを ODBC ソースに	前のバージョンと同じ	ユーザ
書き出し > ストラクチャ定義を XML ファイルに	データベースストラクチャ定義ファイルを作成 (124 ページ "ストラクチャ定義の書き出し" の節を参照)	4D v11 の新機能
書き出し > ストラクチャ定義を HTML ファイルに	データベースストラクチャ定義ファイルを作成 (124 ページ "ストラクチャ定義の書き出し" の節を参照)	4D v11 の新機能
バックアップ	前のバージョンと同じ	ユーザ
ログファイルをチェック	前のバージョンと同じ	ユーザ
用紙設定	前のバージョンと同じ	すべて
プリント	前のバージョンと同じ	すべて
終了	前のバージョンと同じ	すべて

編集	説明	以前のバージョンでのモード
取り消し / やり直し / カット / コピー / ペースト / クリア / すべてを選択	前のバージョンと同じ	すべて
複製	選択されたオブジェクトを複製	4D v11 の新機能
デザインモードを検索	前のバージョンと同じ	デザイン
検索 >... (検索コマンド)	前のバージョンと同じ	デザイン
クリップボード表示	前のバージョンと同じ	すべて
環境設定	前のバージョンと同じ	すべて

実行	説明	以前のバージョンでのモード
アプリケーションテスト	データベースを再起動することなしに、アプリケーションモードを実行する (以前のカスタムコマンドと同じ)	ユーザ / デザイン
メソッド	前のバージョンと同じ	すべて

ランタイムエクスプローラ	前のバージョンと同じ	すべて
Web サーバ開始 (停止)	前のバージョンと同じ	すべて
Web サーバテスト	前のバージョンと同じ	すべて
SQL サーバ開始 (停止)	4D に統合された SQL サーバを開始または停止する (204 ページ "4D SQL サーバの起動と停止" の節を参照)	4D v11 の新機能
インタプリタ再起動 / コンパイル済再起動	データベースを再起動し、(可能なら) アプリケーションモードをインタプリタまたはコンパイル済みで実行する ¹	4D v11 の新機能

1. これらのコマンドは、以前の4Dに存在したインタプリタとコンパイル済みを切り替えるコマンドとは異なり、データベースをアプリケーションモード (以前のカスタムモード) で実行します。

デザイン	説明	以前のバージョンでのモード
このメニューは前のバージョンのデザインメニューに対応します。新しいコマンドが1つ追加されています。		デザイン
コンパイル開始	データベースメソッドの実行 ダイアログを経ずに、現在の設定を使用して直接コンパイルを開始する	4D v11 の新機能

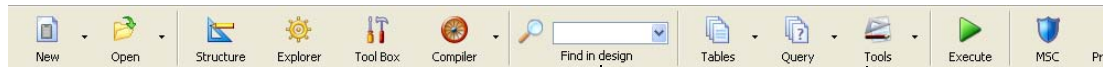
レコード	説明	以前のバージョンでのモード
カレントテーブル表示 (Table_Name)	カレントの出力フォームを使用してカレントテーブルのレコードを表示する (カレントテーブル名が括弧の間に表示される)。カレントテーブルがなければ、コマンドは選択不可になる。	4D v11 の新機能
最後に使用したテーブル >... (使用したテーブルの名前)	サブメニューにはデータベースで使用されたテーブルが15こまで表示される。テーブルを選択すると、カレントの出力フォームを使用して、そのテーブルのレコードが表示される。	4D v11 の新機能
テーブルのリスト	前のバージョンと同じ	ユーザ
新規レコードをリストに追加 新規レコード レコード修正	前のバージョンと同じ	ユーザ

セレクション削除	ウィンドウ上で選択したレコードをデータファイルから削除する (= クリアコマンド)	4D v11 の新機能
すべてを表示 サブセットを表示 クエリ >... 並び替え フォーミュラで更新	前のバージョンと同じ	ユーザ

ツール	説明	以前のバージョンでのモード
このメニューは以前のバージョンのツールメニューに対応します。		ユーザ

ツールバー

メニューバーと同様、4D v11 のツールバーも (アプリケーションモードを除き) 常に同じ内容が表示され、4D の主要な機能へのすばやいアクセスを提供します:

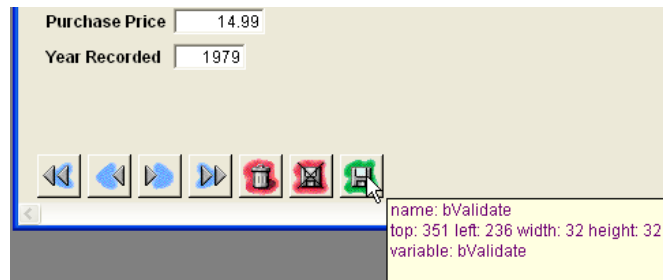


新しいテーブルボタンは、レコードメニューの最後に使用したテーブルコマンドと同じ機能を提供します。(上記参照)

新しい MSC ボタンは Maintenance & Security Center ウィンドウを表示します ([177 ページ "Maintenance & Security Center" の章](#) を参照).

フォームオブジェクトの情報

キアサインを使用して、開発者はフォームオブジェクトのさまざまな情報 (名前、座標など) を得ることができます。この情報は、Ctrl+Shift (Windows) または Command+Shift キーを押しながらマウスカーソルをオブジェクト上に持ってくると、ヘルプチップとして表示されます:

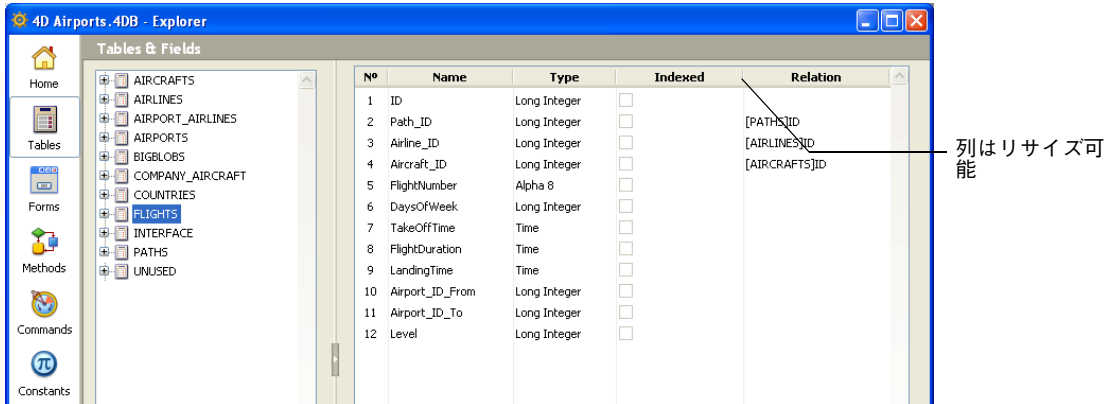


このアシスタンス機能は、開発環境が開かれているときに、フォームに表示される個々のオブジェクトに対して有効です。

エキスプローラ

テーブルとフィールドの表示

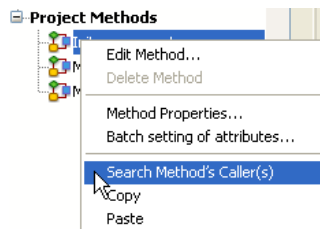
テーブルまたは Home ページでテーブルが選択されると、エキスプローラのプレビューエリアにテーブル定義が表形式で表示されます：



このエリアは表示を行います；値を変更することはできません。フィールド番号、名前、型に加えこの表にはテーブルに関連付けられたインデックス（タイプにかかわらず）やこのテーブルを n テーブルとするリレーションも表示されます。リレーションが張られたフィールド（n フィールド）ごとに、リレート先（1フィールド）がリレーション列に表示されます。

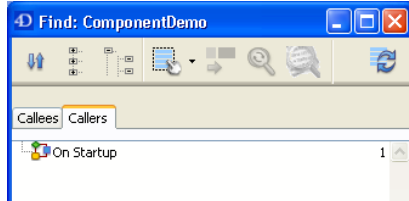
呼び出し元の検索

エキスプローラの新機能を使用して、プロジェクトメソッドを呼び出すオブジェクト（他のメソッドやメニュー）を検索できます。この機能はエキスプローラのコンテキストメニューからメソッドの呼び出し元を検索コマンドを選択することで利用できます：



注：メソッドの呼び出し元を検索コマンドは、メソッドエディタの件テキストメニューからも選択できます（130ページ "呼び出し元の検索"の節を参照）。

検索結果は新しいウィンドウに表示されます。見つかったオブジェクトは呼び出し元リストに表示されます:



このウィンドウに関する詳細は、[91 ページ "新しい結果ウィンドウ"](#) の節を参照してください。

プロジェクトメソッドのコピー、ペーストおよび複製

プロジェクトメソッドのコピー、ペースト、複製をエクスプローラのメソッドリスト上で直接行うことができます。これはつまり、データベース内でメソッドを複製したり、他のデータベースにメソッドのコピーを作成できるようになることを意味します。

メソッドを複製するには、メソッド名のリスト上でマウスを右クリックし、コンテキストメニューから複製コマンドを選択します。

メソッドをコピーするには、メソッド名のリスト上でマウスを右クリックし、コンテキストメニューからコピーコマンドを選択します。ペーストを行うには、同じコンテキストメニューからペーストを選択します。

メソッドを複製する場合や、ペースト先に同じ名前のメソッドが既に存在する場合、追加されるメソッドに番号が付加されます。この番号は必要に応じてインクリメントされます (例えばMymethod2, Mymethod3のように)。

メソッドページ

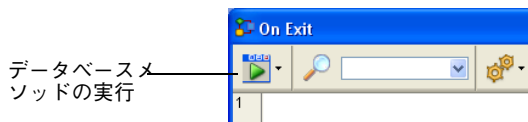
エクスプローラのメソッドページは再構成され、プロジェクトフォームメソッドが追加されました ([135 ページ "プロジェクトフォーム"](#) の節を参照)。以下の要素が用意されています:

- データベースメソッド (変更なし)
- プロジェクトフォームメソッド : データベースプロジェクトフォームのリストを表示 (4D v11 の新機能).
- プロジェクトメソッド (変更なし)
- テーブルフォームメソッド: テーブルごとに、データベーステーブルフォームのリストを表示 (以前のバージョンのフォームメソッド&トリガ要素のフォームメソッドに相当)。

- トリガ： データベーステーブルのリストを表示（以前のバージョンのフォームメソッド&トリガ要素のトリガに相当）。

データベースメソッドの実行

特定のデータベースメソッドは、メソッドエディタから直接実行することができるようになりました。このためデータベースを再起動することなしに機能をテストすることができます：



以下のデータベースメソッドが実行可能です：

- On Startup
- On Exit
- On Server Startup
- On Server Shutdown

デザインモードでのオブジェクトの移動

4D v11 では、2つのデータベース間でオブジェクトを移動させることができます。つまりあるデータベースに作成したテーブルやフォーム、メソッド等を他のデータベースにコピーできます。これによりデータベース開発がより簡易になります。

注： 前のバージョンでは、この機能は 4D Insider アプリケーションにより提供されていました。

オブジェクトの移動は個々のオブジェクトにとどまらず、依存関係にあるオブジェクトなど、オブジェクトから参照されるすべてのオブジェクトも対象とでき、機能全体のコピーが可能となります。例えば、カスタムクエリダイアログボックスを作成した場合、そのダイアログボックスとして使用するフォームの他に、すべてのメソッド、ピクチャ、そして使用するすべてのオブジェクトをコピーできます。フォームを他のデータベースやライブラリに、データベースで現在使用されている機能をグループ化してコピーできます。

特定のオブジェクトもやはり分離することはできず、"親"オブジェクトとともにコピーされなければなりません。分離できないオブジェクトは以下に示します。

移動可能なオブジェクト

オブジェクトはツールボックス、エクスプローラ、そしてフォームエディタから移動できます。更に、デザインモードを検索の結果を表示するウィンドウから移動を行うことも可能です。

ストラクチャの整合性を保つために、あるオブジェクトをコピーすると、そのオブジェクトと分離できないオブジェクトもすべてコピーされます。例えば、フォームをコピーすると、フォームメソッドやフォームに置かれたオブジェクトのオブジェクトメソッドもコピーされます。これら分離できないオブジェクトはそれ自身を直接移動することはできません。

以下は移動可能なオブジェクトと、そのオブジェクトに付随する分離できないオブジェクトのリストです：

移動可能なオブジェクト	分離不可オブジェクト
ツールボックス	
リスト	
スタイルシート	
フォーマット / フィルタ	
ライブラリのピクチャ	

移動可能なオブジェクト	分離不可オブジェクト
ヘルプ tips	
エクスプローラ	
プロジェクトフォーム	フォームメソッド
テーブルフォーム	フォームメソッド
プロジェクトメソッド	
フォルダ / サブフォルダ	
テーブル	フィールド, トリガ
フォームエディタ	
すべてのフォームオブジェクト (ボタン, 変数 など) フォーム移動時には、フォームに含まれるすべてのオブジェクトが合わせて移動されます。	オブジェクトメソッド

移動について

オブジェクトの移動はドラッグ&ドロップまたはコピー / ペーストで行います。

ドラッグ&ドロップを使用して2つのデータベース間でオブジェクトを移動するには、4D Developer のコピーを作成するか、4D Developer と 4D Client を使用します。

データベース間の移動では、移動されるオブジェクトは移動元と同じ環境 (ツールボックスやエクスプローラ) またはアプリケーションの他のエリアにペーストまたはドロップすることができます。4D は内容に応じて可能な限り適切なアクションを行います。例えばフォームをメソッドエディタにドロップできます。この場合、フォーム名がメソッドに挿入されます。

移動時に、同じタイプで同じ名前を持ったオブジェクトが移動先データベースに存在する場合は、デフォルトでは、既存のオブジェクトが移動されるオブジェクトで置き換えられます。この場合移動ダイアログボックスが表示され、置き換えられるオブジェクトを表示し、行うアクションを変更できます。

次の点に注意する必要があります：

- ビューとレベル: 移動されたフォームオブジェクトはエディタ上で元の位置情報を保持します。ビュー中の位置とフォームのレベルレベルも保持されることに留意してください。

- 継承されたフォーム：継承されたフォームは、ソースフォームと一緒に移動されません。しかしながら、そのフォームへの参照は保持されます。更に継承されたフォームは依存するオブジェクトとして扱われ、移動を行う際に、他の（既存の）フォームを継承されたフォームとして使用することができます。（[81 ページ "フォームの依存オブジェクト" の節](#)を参照）。
- アクセス権：移動されたフォームやプロジェクトメソッドは元のアクセス権を保持しません。自動でデフォルト値（"すべてのグループ"）が設定されます。
- フォルダ：エクスプローラの Home ページからフォルダを移動すると、フォルダのほかにその内容（テーブル、フォーム、プロジェクトメソッド）も移動されます。この操作は大量のデータの移動を伴うことがあるため、このタイプの移動を行う際には警告ダイアログボックスが表示されます。

注： エクスプローラのゴミ箱ページからオブジェクトをドラッグすることはできません。

フォームの依存オブジェクト

フォーム（テーブルやプロジェクト）はリストやピクチャなど様々な他のオブジェクトを参照することがあります。これらのオブジェクトは依存オブジェクトと呼ばれます。

特定のケースではすべての依存オブジェクトを移動する必要があるのに対して、他のケースでは、そのうちのいくつかのみあるいはまったく移動させたくない場合もあるでしょう。

4D では移動環境設定や特別な移動ダイアログボックスを使用して、依存するフォームオブジェクトの移動をコントロールすることができます。

移動環境設定では、依存オブジェクトの移動に関するデフォルト値を設定します。オブジェクトのタイプごとに様々なデフォルトオプション値を選択できます。詳細は [86 ページ "移動環境設定" の節](#)を参照してください。

移動ダイアログボックス

2つの 4D データベース間やデータベースとオブジェクトライブラリ間で、複数のオブジェクトを（ドラッグ&ドロップやコピー / ペーストで）移動すると、ダイアログが表示されることがあります。ダイアログボックスには移動されるすべてのオブジェクトと適用されるアクションが一覧表示されます。

このダイアログボックスは"移動ダイアログボックス"と呼ばれ、以下の1つに当てはまる場合に表示されます：

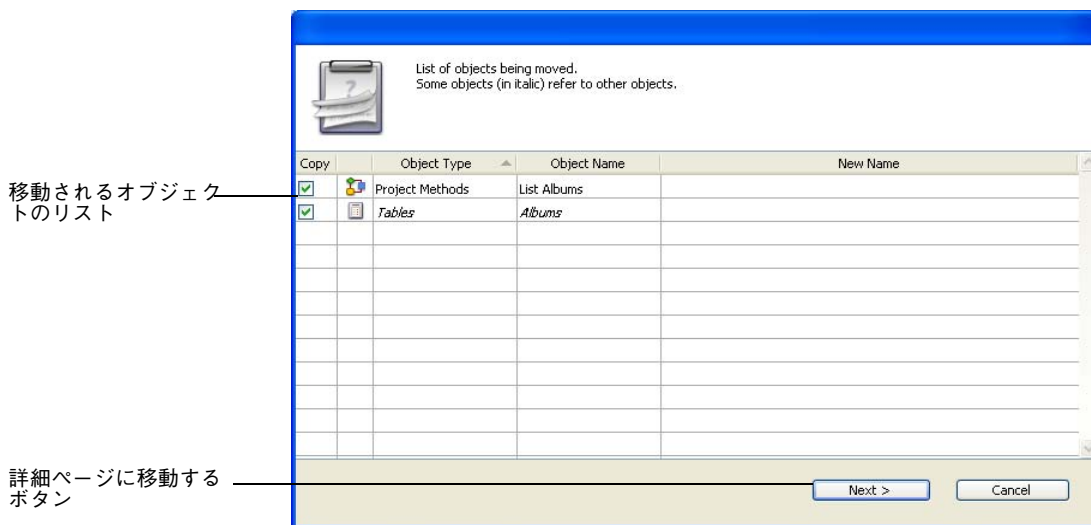
- 移動環境設定で "常に表示" オプションが選択されている (86 ページ "移動環境設定" の節を参照)。
- 1 つ以上のオブジェクトが移動先データベース内で名前のコンフリクトを起こしているとき
- 移動される依存オブジェクトのうち少なくとも1つのタイプに他のオブジェクトを使用デフォルトアクションが設定されている時。

これらのケースに対し、オブジェクトの移動が全くコンフリクトを起こさない場合、移動ダイアログボックスは表示されず、オブジェクトのコピーが実行されます。

このダイアログボックスではコンテキストに応じ、移動設定を表示させたり、変更したりすることができます。ダイアログにはメインと詳細の2つのページがあります。これらのページは **次 >** と **< 戻る** ボタンで切り替えることができます。

メインページ

メインページには移動されるオブジェクトがリスト表示されます。



名前がコンフリクトしているオブジェクトは太字で、依存オブジェクトはイタリックで表示されます。他の列ではオブジェクトのタイプや名前、そしてそのオブジェクトの"新しい"名前(移動先データベースでのデフォルトの名前)が表示されます。この名前は、必要であれば、詳細ページで変更できます(以下の節を参照)。

"コピー"列のチェックボックスは、オブジェクトが移動先データベースにコピーされるかされないかを示します。特定のオブジェクトが関連するコンフリクトをすばやく解決するために、チェックを外すことができます。コンフリクトが依存オブジェクトで発生している場合、その参照(名前)は移動先データベースに保持されます。

チェックボックス上でAlt+click (Windows) または Option+click (Mac OS) を行うと、すべてのチェックボックスのチェックの状態を切り替えることができます。また、ダイアログボックスのコンテキストメニューを使用して同じことを行えます。

デフォルトの移動オプションに問題がなければ、OK をクリックしてオブジェクトの移動を実行します。

注：少なくとも1つのオブジェクトに他のオブジェクトを使用アクションが割り当てられていると、使用するオブジェクトを選択するか、そのオブジェクトのチェックを外すまで、コピーを実行することはできません。

特定のアクションを変更するには、次> ボタンをクリックして詳細ページを表示します。移動をキャンセルするにはキャンセルをクリックします。

詳細ページ

詳細ページにはコピーされるオブジェクトがリストされていて (メインページでチェックがつけられているもの)、移動設定を変更できます:

移動対象のオブジェクトのリスト

Object Type	Object Name	Action	New Name or Other Object
Project Methods	List Albums	Create	
Tables	Albums	Do not create	

メインページに移動

"アクション"列のポップアップメニューを使用して、オブジェクトのアクションを変更できます。メニューで提供されるアクションは選択されたオブジェクトのタイプや後に説明するルールに基づきます。

複数のオブジェクトのアクションを一度に変更できます。これを行うには、変更するオブジェクトを選択して、選択したいずれかのオブジェクトの "アクション" を変更します。変更はすべての選択されたオブジェクトに適用されます。オブジェクトに互換性のないアクションが選択されると、そのオブジェクトは変更されず、警告ダイアログが表示されます。

"新しい名前または他のオブジェクト"列には、オブジェクトが移動先データベースにコピーされた際に適用される名前が表示されています。この名前を変更することができますが、移動先データベースに既に存在する名前を使用しないよう注意してください。それを行うと新たな名前のコンフリクトが発生します。

依存オブジェクトの場合は、この列を使用して移動先データベースの他のオブジェクトを指定することができます (他のオブジェクトを使用するアクションが選択されている場合)。例えばテーブルフォームを移動する場合、テーブルを作成する代わりに、移動先データベースに既存のテーブルを、フォームの所属先テーブルとして指定できます。

指定可能なアクション

以下のアクションを指定できます：

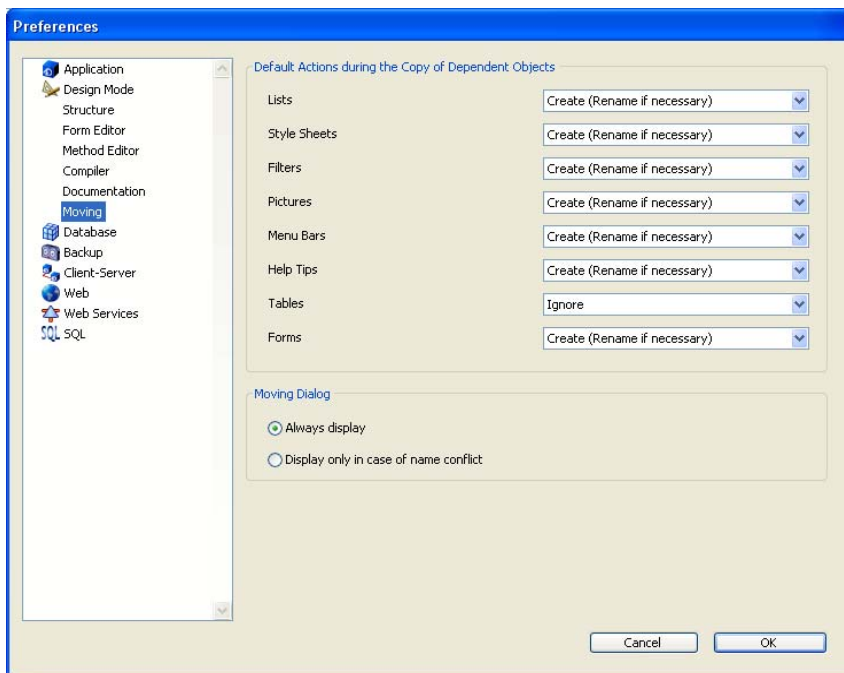
- **作成しない:** オブジェクトはコピーされません。依存オブジェクトの場合で、データベースに同じ名前のオブジェクトが既に存在する時、その参照 (名前) は保持されます (この場合、既存のオブジェクトがメインオブジェクトで使用されます)。同じ名前のオブジェクトが存在しない場合、参照は削除されます。
- **置換:** このオプションは、データベースに同じタイプで同じ名前のオブジェクトが既に存在する場合に提示されます。この場合、移動先データベースのオブジェクトは、移動元データベースのオブジェクトで置き換えられます。
- **作成:** 依存オブジェクトはその属性とともに移動先データベースにコピーされます (名前のコンフリクトがない場合に提示されるオプション)。
- **作成して名称変更:** このオプションは、移動先データベースに同じタイプで同じ名前のオブジェクトが既に存在する場合に提示されます。デフォルトで、オブジェクトに番号を付加することで名前が変更されます。この場合、"新しい名前または他のオブジェクト"列を使用して名前を指定できます。オブジェクト参照は移動先データベースで更新されます。
- **他のオブジェクトを使用:** このオプションは依存オブジェクトにのみ使用可能です。このオプションは移動先データベースの既存のオブジェクトを参照として利用可能にします。この場合、"新しい名前または他のオブジェクト"列には利用可能な他のオブジェクトがリストされます。
- **同じ名前のテーブルを使用:** このオプションは、同じ名前のテーブルが既にデータベースに存在する場合に提示されます。この場合、"新しい名前または他のオブジェクト"列には依存テーブルの代わりに利用可能なテーブルのリストが表示されます。

依存オブジェクトが他のオブジェクトを参照している場合、設定に従いリストが更新されます。

移動設定が終了したら、OKをクリックしてオブジェクトの移動を実行します。 <戻る をクリックするとメインページに戻ります。移動をキャンセルするにはキャンセルをクリックします。

移動環境設定

環境設定のデザインモード / 移動ページで、オブジェクト移動のデフォルト値を設定できます：



これらの設定は、データベースが移動先データベースとして使用される場合に、オブジェクトをデータベースにドロップしたりペーストしたりする際に適用されます。

依存オブジェクトコピー時のデフォルトアクション

ダイアログボックスの上部では、依存オブジェクト (移動されるフォームにリンクされたオブジェクト) の移動時設定を行います (81 ページ "フォームの依存オブジェクト" の節を参照)。依存オブジェクトのタイプごとにアクションを設定できます。

これらのデフォルトアクションは、オブジェクトの移動がコンフリクトを発生させず、名前が衝突する場合のみ表示オプションが選択されていると、自動で適用されます (以下の節を参照)。もしくは、移動ダイアログ上でデフォルトで選択されます。

無視、作成 (必要なら名前を変更)、作成 (必要なら置換)、そして他のオブジェクト塩用アクションが、それぞれのタイプに対して選択可能です。移動ダイアログボックスでは、より特化したアクションが提供されます。これらのアクションについての詳細は[85ページ "指定可能なアクション"の節](#)を参照してください。

以下はこれらのアクションについての説明です：

- **無視**: このタイプの依存オブジェクトは移動先データベースにコピーされません；。移動ダイアログボックスでは、デフォルトで作成しないアクションが選択されます。
- **作成 (必要なら名前を変更)**: このタイプの依存オブジェクトは常に移動先データベースにコピーされます。移動ダイアログボックスでは、オブジェクトが移動先データベースに存在しない場合デフォルトで作成アクションが選択されます。
移動先データベースで名前のコンフリクトが生じる場合、コピーされるオブジェクトは "_x" が名前の後ろに付加されます ([80 ページ "移動について"の節](#)を参照)。この場合、移動ダイアログボックスでは**名称変更**アクションがデフォルトで選択されます。
- **作成 (必要なら置換)**: このタイプの依存オブジェクトは常に移動先データベースにコピーされます。移動ダイアログボックスでは、オブジェクトが移動先データベースに存在しない場合デフォルトで作成アクションが選択されます。
移動先データベースで名前のコンフリクトが生じる場合、既存のオブジェクトは移動されるオブジェクトで置き換えられます。この場合、移動ダイアログボックスでは**置換**アクションがデフォルトで選択されます。
- **他のオブジェクト使用**: このオプションが選択されていると、"名前が衝突する場合のみ表示"オプションが選択されていても、常に移動ダイアログボックスが表示されます。オブジェクトを移動する際、コピーされる依存オブジェクトの代わりに使用する移動先データベースのオブジェクトを指定しなければなりません。

注： これらのオプションは依存オブジェクトにのみ適用されます。移動されるその他のオブジェクトには作成 (必要なら名前を変更) がデフォルトアクションとして使用されます。

移動ダイアログ

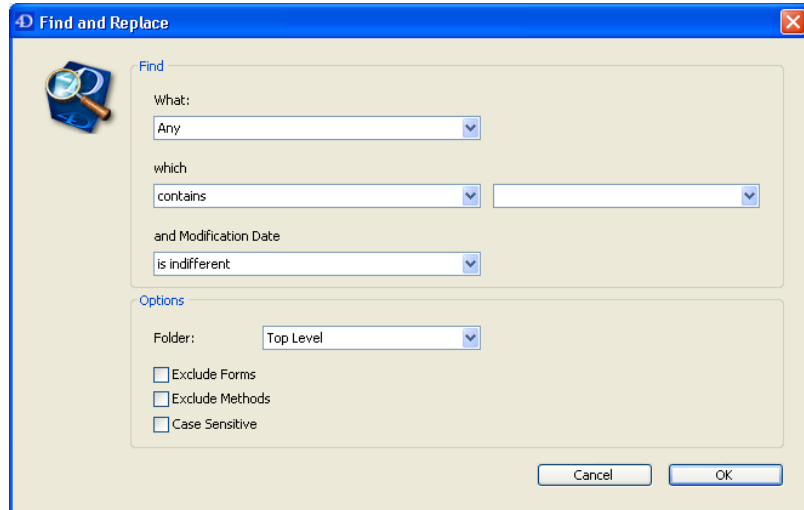
このオプションは移動ダイアログボックスの表示設定を行います。常に移動ラジオボタンが選択されていると、オブジェクトが移動されるたびにダイアログボックスが表示され、操作をより詳細にコントロールすることができます。名前が衝突するばあいのみ表示ラジオボタンが選択されていると、移動される依存オブジェクトまたはメインオブジェクトが、移動先データベースで名前のコンフリクトを起こす場合のみ、ダイアログが表示されます。

データベース内の検索と置換

ストラクチャ内の検索機能が拡張されました。新しい検索オプションが追加され、また検索された要素の名前を変更したり接頭辞をつけることが可能になりました。

新しい検索オプション

編集メニューのデザインモードを検索メニュー項目を選択すると、次のダイアログボックスが表示されます。



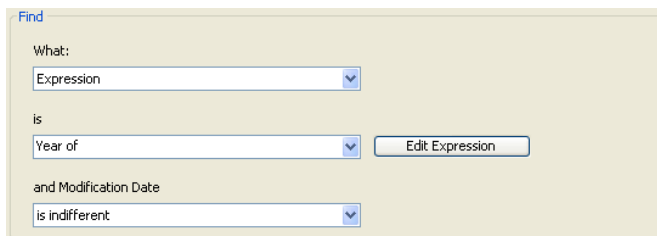
新しいオブジェクトタイプと、2つの新しい検索条件が追加されました。つまり1つから3つの条件を使用して検索を行うことができます。

- **タイプ:** 新しいオブジェクトタイプ "プロジェクトメソッド" がポップアップメニューに追加されました。このタイプを選択すると、データベースのプロジェクトメソッド内を検索対象とすることができます。このタイプが選択されると、"メソッドを除外" オプションは選択不可になります。更にこの場合、結果ウィンドウには "呼び出し先" と "呼び出し元" のリストが含まれます (93ページ "呼び出し先および呼び出し元のページ" の節を参照)。
- **検索条件:** この新しいメニューでは、比較モード ("含む", "等しい", "前方一致", "後方一致") を選択できます。

オブジェクト名を検索条件に含めたくないときは、"なし" オプションを選択します。この場合検索はオブジェクトタイプ、または更新日、あるいはその両方に基づき行われます。

注： 前のバージョンと異なり、アットマーク (@) をワイルドカードとして使用できます。例えば、("m@base" のような)" 最初と最後の文字を指定する "タイプの検索条件を設定する場合。

検索するオブジェクトのタイプが式の場合、このメニューは検索対象の文字列を含むエリアに置き換えられ、右側に式を編集ボタンが表示されます。



- 変更日: この新しいメニューは、検索条件の1つとして変更日を加える時に使用します。この条件は、メソッドやフォームにおける検索だけに有効で、単独またはその他の条件と合わせて使用することもできます。変更日を検索条件として使用したくない場合は、"is indifferent" オプションを選択すると、オブジェクトのタイプ、名前、またはその両方の組み合わせだけに基づいて検索が行われます。

注： 前のバージョンの 4D にあった "オブジェクト名全体" オプションは、検索モードメニューに置き換えられました。

フォルダーオプション

- フォルダー: この新しいオプションを使用すると、特定のファイルレベルに限定して検索できます。デフォルトでは、すべてのフォルダー内で検索を行うことができます。

検索の範囲

4D v11 ストラクチャの検索には、より多くのオブジェクトが含まれます。今後は、指定される文字列のタイプによって、次のような項目の検索を行うことができます。

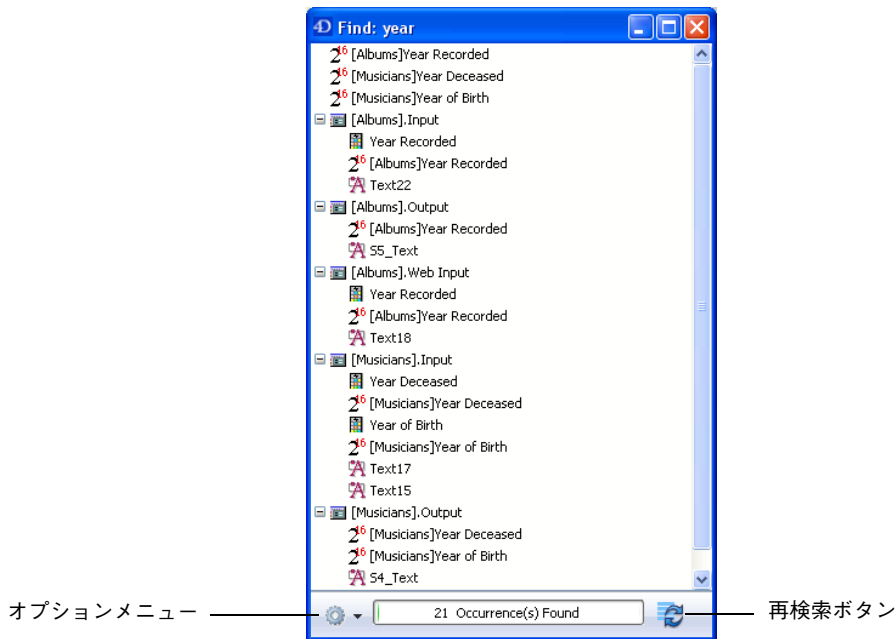
- フォーム (除外可能)
- あらゆるタイプのメソッド
- メニューとメニューコマンド
- 選択リスト
- テーブルとフィールド (サブテーブルとサブフィールドを含む)
- コメント
- ライブラリのピクチャ

- 静的なテキスト
- ヘルプとティップス
- フォーマット/フィルター (バージョン 11 の新機能)
- スタイルシート (バージョン 11 の新機能)
- プラグインコマンド (バージョン 11 の新機能)
- コマンド (バージョン 11 の新機能)
- コンスタント (バージョン 11 の新機能)
- フォルダー (バージョン 11 の新機能)

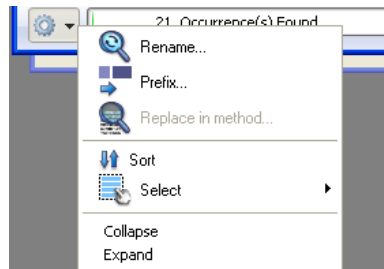
検索結果は、オブジェクトのタイプ別、その次にアルファベット順にグループ化され、結果ウィンドウに表示されます。

新しい結果ウィンドウ

結果ウィンドウに、含まれるオブジェクトに対して実行できるさまざまな捜査を呼び出すためのメニューが追加されました。



オプションメニューのコマンドは、選択されたオブジェクトの種類によって選択可能になったり、不可になったりします。たとえば 4D コマンドが選択に含まれる場合、名称変更と接頭辞機能は無効化されます。




- 名称変更: このコマンドは、選択したオブジェクトの名称を変更するために使用できます。この操作の詳細は後述します。
- 接頭辞: このコマンドは、リスト中で選択されたすべてのオブジェクトの名前に、接頭辞をつけるために使用できます。この操作の詳細は後述します。
- メソッド内を置換: このコマンドは、一つ以上のメソッド中で、検索と置換を実行するために使用します。この操作は後ほど説明します。

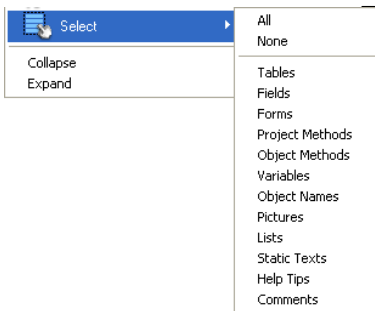
並び替え: このコマンドは、ウィンドウの内容を文字順に並び替えます。Shift キーを押しながらこのコマンドを選択すると、昇順に並び替えられます。

選択>: このサブメニューは、ウィンドウに表示されているオブジェクトの種類ごとに選択するために使用します (手動で Shift+click や Ctrl/Command+click を行って、複数選択することも可能です)。この機能は、リスト中の複数のオブジェクトの名称を変更したり、接頭辞をつけたりする際に便利です。

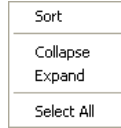
すべて を選択すると、ウィンドウの内容全てが選択されます。また Ctrl+A (Windows) や Command+A (Mac OS) ショートカット、さらにリストのコンテキストメニューを使用できます。

オブジェクトのタイプごとに選択することもできます。これを行うには、サブメニュー内で選択したいオブジェクトタイプを指定します。

- 折りたたむ / 展開する: これらのコマンドは、階層リストとして表示されている結果リストを、折りたたんだり、展開したりするために使用します。
- 検索再実行 : このボタンをクリックすると、同じ検索条件とオプション設定で検索が再実行されます。この機能は例えば、すべての置き換え処理が正しく行われたかを確認するのに役立ちます。



ウィンドウのコンテキストメニューを使用して、標準アクションを実行させることができます。

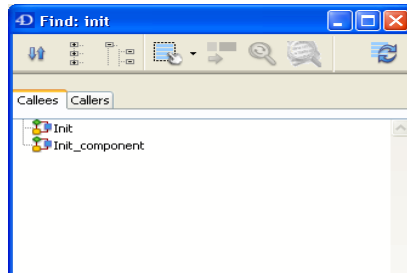


オブジェクトや文字列の複数のオカレンスが一つのメソッド中で検索されると、その数がオブジェクト名の隣に表示されます。



呼び出し先および呼び出し元のページ

プロジェクトメソッド内でだけ検索が行われた場合 (89 ページ "新しい検索オプション" の節を参照)、結果ウィンドウには2つのリストが表示されます。これらは、呼び出し先と呼び出し元のリストで、それぞれの専用タブを使用してアクセスできます。



- 呼び出し先のリストには、検索で見つかったプロジェクトメソッドが表示されます。言い換えると、それらの名称や変更日は検索条件に対応します。
- 呼び出し元のリストには、検索で見つかったプロジェクトメソッドを呼び出すオブジェクト(メソッド、フォームなど)が表示されます。

注：プロジェクトメソッドを呼び出すオブジェクトは、エクスプローラまたはメソッドエディタから直接検索することもできます。詳細は、75 ページ "テーブルとフィールドの表示" の節を参照してください。

接頭辞および名称変更

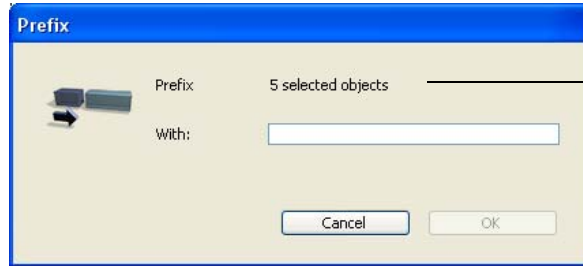
4D v11 では検索結果ウィンドウを使用して、オブジェクトの接頭辞付けや名称変更、または選択メソッドにおける文字列の置換を行うことができます。

接頭辞

接頭辞はコードの可読性を高めたり、データベース内の関連オブジェクトを素早く検索するのに役立ちます。

オプションメニューの接頭辞メニューは、接頭辞をつけることが可能なオブジェクトのみが選択されている場合にのみ有効になります。変数、プロジェクトメソッドそして(テーブルやプロジェクト)フォームに接頭辞を付けられます。

このコマンドを選択すると、以下のダイアログが表示されます。



一つのオブジェクトが選択されているとき、その名前が表示されます。

入力エリアには、選択したオブジェクトにつける接頭辞文字列を入力します。

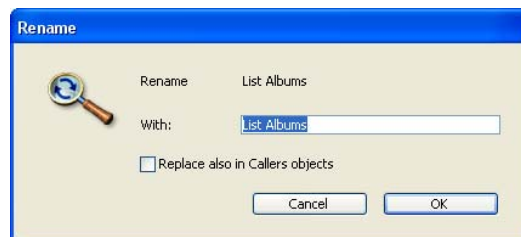
- 呼び出し元オブジェクトも置換(このオプションは呼び出し元リストのオブジェクトに接頭辞をつけると表示されます): 例えば、接頭辞をつけるメソッドの呼び出し元メソッドなど、選択オブジェクトの呼び出し元オブジェクトに対しても接頭辞を適用したい場合は、このオプションにチェックを入れてください。これは、接頭辞付けが適用された後の正常なコード操作を維持するためのオプションです。

注: オブジェクト名は31文字までに限られているため、接頭辞をつけた結果オブジェクト名が長くなりすぎる場合もあり得ます。そのような場合は、警告メッセージが表示され、操作を中断できます。

名称変更

この操作では、選択オブジェクトの名称を変更します。名称変更コマンドは、選択されたオブジェクトが1つで、それが名称変更可能なオブジェクトである場合に限り有効です。変数、プロジェクトメソッド、および(テーブルとプロジェクト)フォームは、すべて名称変更が可能です。

このボタンをクリックすると、次のダイアログボックスが表示されます。



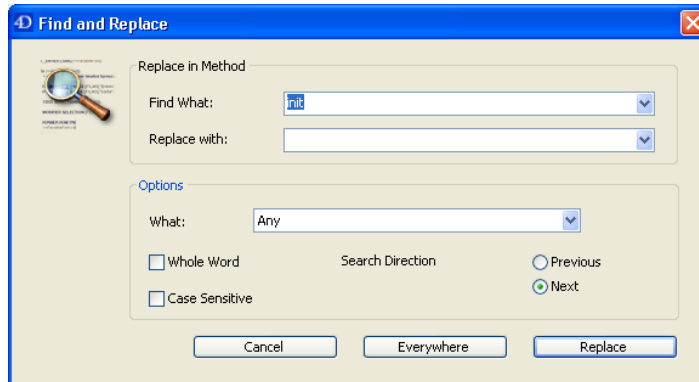
- 入力エリアには、選択したオブジェクトの新しい名称文字列を入力します。
- 呼び出し元オブジェクトも置換 (このオプションは呼び出し元リストのオブジェクトの名称を変更すると表示されます): オブジェクトの呼び出し元オブジェクトも名称変更したい場合は、このオプションにチェックを入れてください。これは、オブジェクトの名称変更が行われた後の正常なコード操作を維持するためのオプションです。

メソッド内を置換

この操作では、1つまたは複数のプロジェクトメソッド内で全体的な " 検索と置換 " を実行します。

このボタンは、1つまたは複数のプロジェクトメソッドが選択されている時に限って有効です。

このボタンをクリックすると、メソッドの「検索と置換」標準ダイアログボックスが表示されます。



このダイアログボックスを使用して、メソッド内における文字列の検索と置換を行うことができます。操作に関する詳細は、「4D デザインリファレンス」マニュアルを参照してください。

- 注： 4D v11 のこのダイアログボックスには、関係オブジェクトのタイプを選択する追加メニューがあります。詳細は、[130 ページ " 選択可能な置換オプション " の節](#) を参照してください。

キーボードショートカットの管理

4D v11 では、デザインモードでのキーボードショートカットは、インタフェースの使用と基準により適合するよう、変更されています。例えば、4th Dimension で選択したメソッドを開く時に使用していた従来のショートカット Ctrl キー +P (Windows) または Command キー +P (Mac OS) は、他のアプリケーションの殆どと同様、現在ではプリントコマンドと連動しています。

注： 今バージョンでは、メソッドを開くにはショートカット Ctrl キー +K / Command キー +K を使用します。

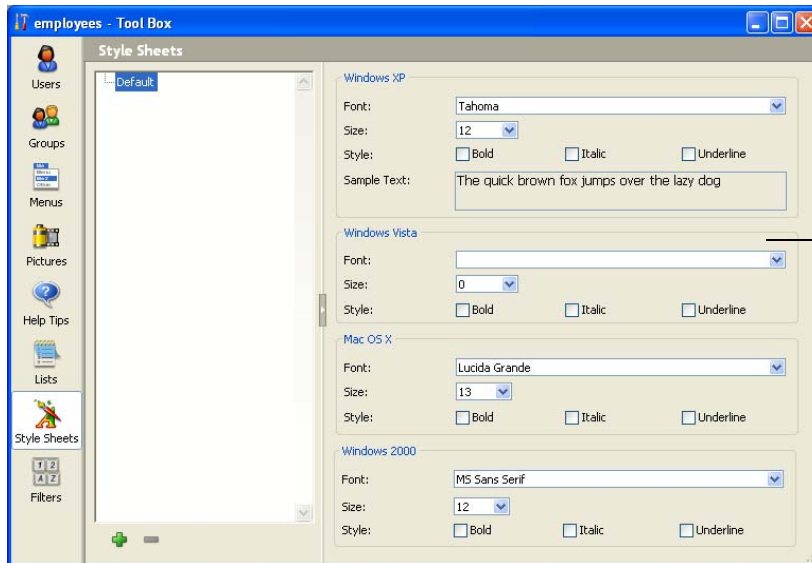
しかしながら、キーボードショートカットは作業習慣に合わせて、そのデフォルトをカスタマイズすることができます。デザインモードのショートカットはすべて、4D Extension フォルダにある 4DShortcuts.xml という XML ファイルで参照されています。オリジナルのショートカットを指定するには、このファイルをコピーし、XML エディタを使ってその複製を変更します(複数の動作に同じショートカットを指定しないよう注意してください)。変更が完了した複製は、4D 環境設定のカレントフォルダに入れます。4D はデフォルトファイルではなく、Preferences フォルダにある 4DShortcuts.xml ファイルをロードします。

警告: 4D の Extension フォルダ内の 4DShortcuts.xml ファイルを直接変更しないよう注意してください。

スタイルシートエディタ

Windows^(R) VistaTM のサポート

4D v11 は、Microsoft オペレーティングシステムの新バージョン Windows^(R)VistaTM のインタフェースをサポートします。そのため、スタイルシートエディタには Windows^(R)VistaTM 属性の新しいセットが用意されています。



Windows Vista プラットフォーム用のスタイルシート定義エリア

このエディタの機能に関する詳細は、「4D デザインリファレンス」マニュアルを参照してください。

XLIFF 標準のサポート

4D v11 は、インタフェースのテキストやタイトルをローカライズする XLIFF 標準をサポートします。このテクノロジーは、4D アプリケーションの内部で使用されるものです。4D デベロッパもプラグインデベロッパも、独自にカスタマイズされたアプリケーションまたはプラグインにおいて、この新しい方法の恩恵を受けることができます。

4D v11 は、互換性の理由から、リソースに基づく従来のシステムもサポートします。ただし、4D の内部リソースには直接アクセスできなくなった点に注意してください。(42 ページ "リソースの管理" の節を参照)。

XLIFF とは？

XLIFF (XML ローカライゼーション・インタチェンジ・ファイル・フォーマット) は、ローカライゼーションプロセス専用の標準です。これを使用して、XML ファイル内でソースランゲージとターゲットランゲージ間の対応を記述できます。

実際、XLIFF 標準はリソースに基づくローカライゼーションシステムに新しく代わるものです。いくつかのフリーウェアも含み、ファイル管理のための様々なツールを探することができます。

XLIFF 標準に関する詳細は、下記 URL にある XLIFF 公式仕様書、XLIFF 1.1 を参照してください。

<http://www.oasis-open.org/committees/xliff/documents/xliff-specification.htm>

警告: XLIFF スタンドアードは大文字・小文字を区別します。

4D データベースからの XLIFF 文字列呼び出し

原則として、XLIFF は 4D の前バージョンでリソース参照を使用できた場所であれば、どこでも使用できます。

- フォームエディタのプロパティリストにある"オブジェクト名"フィールド
- エクスプローラからアクセス可能な「フォームプロパティ」ダイアログボックスにある "ウィンドウ名" フィールド
- 静的なテキストオブジェクト
- メニューエディタ
- ヘルプティップスエディタ

注: XLIFF 参照をヘルプ Tips エディタに表示させることはできません。XLIFF に基づくヘルプ Tips を使う場合は、プロパティリストの "ヘルプティップス" フィールドに直接 XLIFF 参照を入力してください。

リソースに基づくシンタックスは ":xxxx,yyy" で、xxxx の部分は STR# リソースの ID 番号、yyy の部分は項目番号を表します。例えば、シンタックス :1015,3 の場合、4D は STR# リソースの ID 番号 1015 と項目番号 3 を文字列として使う必要があることを表します。

注：これらの動的な参照に関する詳細は、2004 年度版「4D デザインリファレンス」マニュアルを参照してください。

4D v11 では、XLIFF 参照を同じ条件の下で使えます。2 つのシンタックスを使うことができます。

■ :1015,3 シンタックス

このシンタックスは、4D の 2004 年度版とそれ以前のバージョン対象の標準リソース呼び出しと全く同じものです。

これを XLIFF ファイルと一緒に使うと、最初の値 (例の 1015) は group 要素の ID 属性を指定します。

2 番目の値 (例の 3) は、trans-unit 要素の ID 属性を指定します。

このシンタックスは、XLIFF に基づくシステムへのスムーズな移行を促進すると同時に、既存のシステムとの互換性を維持するために組み込まれました。具体的には、4D は最初にロードされている全 XLIFF ファイルの中で 1015,3 に対応する値を探しますが、該当する文字列がない場合、次はロードされている全リソースファイルの中で探します。

このメカニズムでは、既存の動的文字列参照を編集することなく XLIFF に基づくシステムをより高度なアプリケーションに組み込むことができます。XLIFF ファイルを正確な場所 (下記参照) にコピーさえすれば、4D がそれを認識します。

このメカニズムにより、4D v11 でリソースに基づくシステムを保持しながら、XLIFF に基づく新システムも使うことができます。実際は両方のシステムを同時に使えますが、同じ場所に同じ文字列がある場合は、原則として、リソースファイルよりもロードされている XLIFF ファイルの方が優先されます。

■ :xliff:OKButton シンタックス

この代用シンタックスは、ロードされている XLIFF ファイルでのみ使用できます。その場合、参照された値 (例の OKButton) は trans-unit 要素の resname 属性を指定します。

"リソース" テーマのコマンドのいくつかは、XLIFF ドキュメントの特性を利用できるよう、変更されました。詳細は、[360 ページ "リソース" の節](#)を参照してください。

カスタマイズされた XLIFF ファイルのインストール

ローカライズされたアプリケーションに XLIFF アーキテクチャを組み込むには、有効な XLIFF ファイルを 1 つまたは複数作成してデータベースフォルダ (Mac OS はパッケージ) の中に保存します。

Mac OS、Windows のどちらにおいても、カスタマイズされた XLIFF はデータベースの「Resources」フォルダに保存する必要があります。下記は、使用されるパス名です。

Windows:

MyDatabase\Resources\Lang.lproj\MyLocJA.xlf

Mac OS:

MyDatabase:Resources:Lang.lproj:MyLocJA.xlf

- *MyDatabase* は、データベースファイルを含むフォルダです。
- *Lang.lproj* は、Lang 言語用の XLIFF ファイルを含むフォルダです。フォルダ名は国際基準 ([101 ページ ".lproj フォルダ名" の節](#)を参照) に従う必要があります。例えば、日本語バージョンの場合は *ja.lproj* というフォルダ名をつけます。

4D は、データベースのカレントランゲージに対応するフォルダから、自動的に XLIFF ファイルをロードします。データベースのカレントランゲージを設定するため、4D はデータベースの「リソース」フォルダ内で、下記の項目に対応するランゲージを連続的に検索します (上から優先順位の高い順)。

1. システムランゲージ (Mac OS では、環境設定で複数のランゲージの順番を設定できます。4D はこの設定を使っています。)
2. 4D アプリケーションのランゲージ
3. 英語
4. 検索の結果何も見つからない場合は、「Resources」フォルダで最初に見つかったランゲージがロードされます。

ランゲージバリエーションが使用され、XLIFF ファイルで使用不可の場合は、次に近い言語が使用されます。

XLIFF ファイルの名前は自由につけることができますが、拡張子 ".xlf" をつける必要があります。同じ言語フォルダにいくつかの XLIFF ファイルを収めることができ、それらはファイル名のアルファベット順にロードされます。

.lproj フォルダ名

".lproj" のフォルダ名は、以下に述べる基準のどれかに従う必要があります。4D は、これらの基準に基づき、次の優先順位により有効なフォルダ名を探します。

1 RFC 3066 bis

この RFC は現在ドラフト作成中の段階ですが、4D は RFC の次の部分をサポートします。言語は、国コード (ISO639-1) + ハイフン + 地域コード (ISO3166). で表します。

例えば、"jp-ja" (つまり、jp-ja.lproj) は、Japan-Japanese 言語を意味します。

2 ISO639-1

この基準は各言語を 2 つの文字で定義します。例えば、"ja" (つまり、ja.lproj) は、日本語言語を意味します。

Ref: http://www.loc.gov/standards/iso639-2/php/English_list.php

3 Legacy name

この慣習では、言語名は英語で完全にスペルアウトされます。例えば、"japanese" (つまり、japanese.lproj) は、日本語言語を意味します。

注：最初の 2 つの協約は、Mac OS の 10.4 以上のバージョンだけがサポート対象です。この OS の以前のバージョンで使用されるのは、"Legacy" 名だけです。

4D がサポートする言語コードをまとめた表は、[ページ 403](#)、[Appendix A "ランゲージコード"](#) に掲載されています。

いくつかの言語定義が見つかった場合、4D は常に最も正確と思われる略語から探します。例えば、OS のカレント言語設定が "日本語" の場合、4D は最初に略語 "jp-ja" を探し、見つからなければ、次に略語 "ja" を探します。

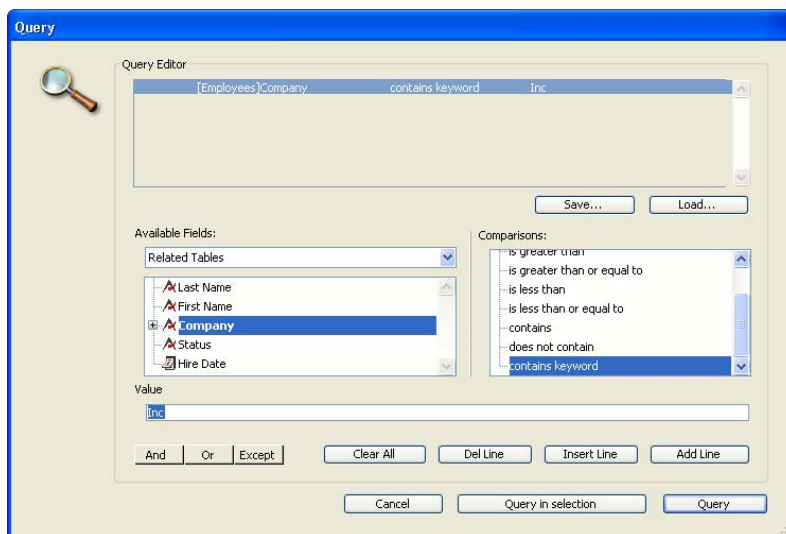
注：これと同じ法則が、XLIFF ファイルの "target-language" タグに適用されます。"target-language=ja" タグを持つ "jp-ja.lproj" フォルダにあるファイルは、"jp-ja" ではなく "ja" に略されたものと認識されるので、XLIFF ファイル内でこの属性を設定する際には十分に注意する必要があります。

カスタマイズされた XLIFFファイルのリロード

XLIFF ファイルは、アプリケーションがローカライズされている間に "動的に" にリロードされるため、ローカライズされた単語や文章がオブジェクトフレーム (ボタン、グループボックスなど) に適合するかどうかを確認できます。このリロードは、4D が前面に戻り、変更日や変更時間が最後のロード時から更新される度に行われます。カレントフォームも同時にリロードされます。

レコードのクエリ

"キーワードを含む"という新規の比較演算子が、4D の標準クエリエディタに追加されました。



この演算子を使用して、文字またはテキストタイプのフィールドに対するキーワード検索を実行できます。パワフルで早いキーワードによる検索は、定義されたキーワードを含むセットタイプのフィールドを持つレコード選択するために使用できます。

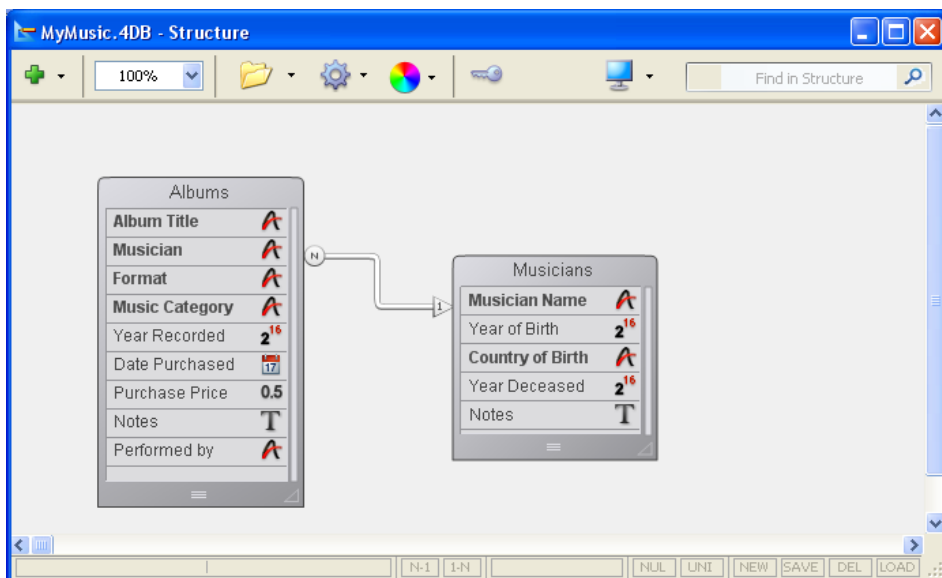
プログラミングによってキーワード検索を実行することもできます。この種の検索に関する詳細は、[261 ページ "キーワード検索" の節](#)を参照してください。

5

ストラクチャエディタ

4D v11 のストラクチャエディタには、インターフェース、機能、編集など、エディタに関するあらゆる観点において、数多くの新機能と改良が加えられました。

4D v11 のストラクチャエディタ



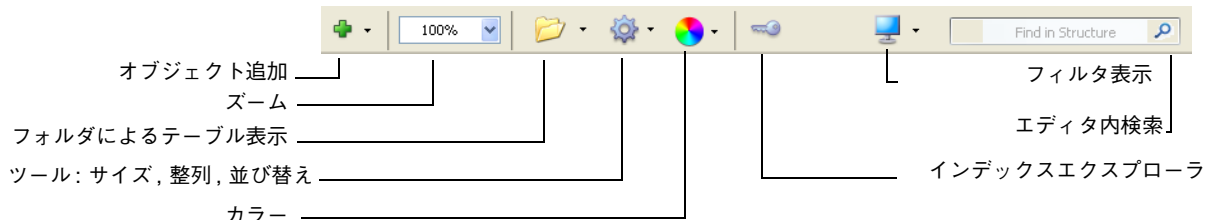
ストラクチャエディタの全体的なアピランスは、テーブル、リレーションなど、4Dの他のエディタとより調和するように変更されました。また、インスペクタのデザインが一新され、人間工学の観点から見直しが行われました。

加えて、ストラクチャ定義をテキストファイルとして書き出し、このタイプのファイルに基づくストラクチャをオンザフライで作成できるようになりました。

ツールバー、インフォメーションバーおよび新規インスペクタ

このエディタには、オブジェクトの追加、オプションのナビゲーションや表示などの機能を含むツールバーが追加されました。

ストラクチャエディタツールバー



ツールバーの機能については次の段落で説明しています。

エディタウィンドウの下部は、テーブルやフィールド、またはリレーションなど、マウスを動かしているエリアに対応したデータを表示するインフォメーションバーです。

I テーブルに関する情報

[Addresses]	1	N-1	1-N		NUL	UNI	NEW	SAVE	DEL	LOAD
テーブル名・テーブル番号										
							テーブルトリガ:			
							NEW = 新規レコード保存			
							SAVE = 既存レコード保存			
							DEL = レコード削除			
							LOAD = レコード読み込み			

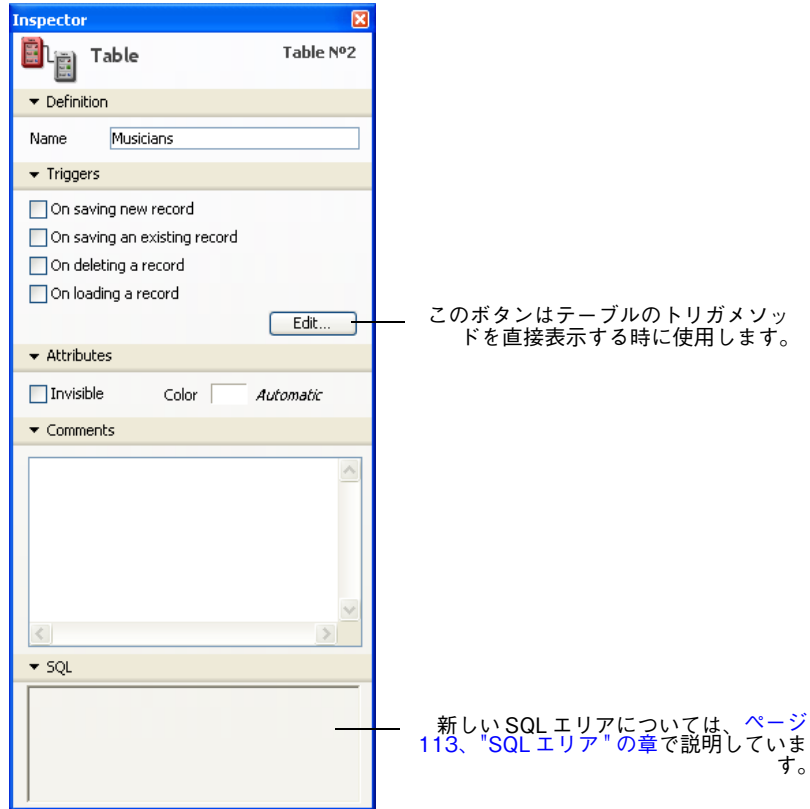
フィールドに関する情報

[Addresses]	1	City	2	N-1	1-N	Alpha - 255	NUL	UNI	NEW	SAVE	DEL	LOAD
テーブル名・テーブル番号		フィールド名・フィールド番号				フィールドタイプ						
							フィールド属性:					
							NUL = Never Null					
							UNI = Unique					

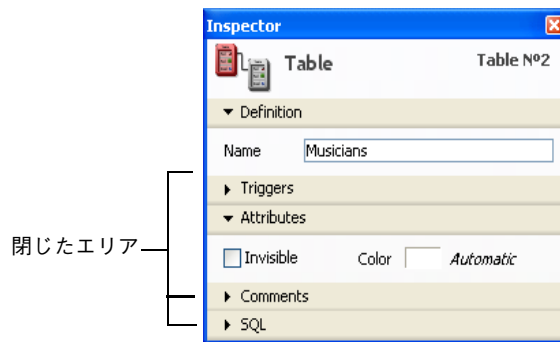
リレーションに関する情報

[Customers]Hometown	[Addresses]City	N-1	1-N		NUL	UNI	NEW	SAVE	DEL	LOAD
フィールド開始点 (複数フィールド)	フィールド終点 (単一フィールド)						リレーション属性:			
							N-1 = 自動 N 対 1			
							1-N = 自動 1 対 N			

インスペクタウィンドウも、1 ページにすべての情報を表示できるよう変更されました。:



ウィンドウ上の異なるエリアは、隣接する三角をクリックすると、それぞれの情報を表示する / 隠すために拡張する / 閉じることができます。



次のショートカットを使用できます。

- 閉じたパネルのタイトルバーを **Shift** キーを押しながらクリックすると、そのパネルを拡張して他のパネルをすべて閉じます。
- 閉じたパネルのタイトルバーを **Alt** キー (Windows) または **Option** キー (Mac OS) を押しながらクリックすると、すべてのパネルを拡張します。
- 拡張したパネルのタイトルバーを **Alt** キー (Windows) または **Option** キー (Mac OS) を押しながらクリックすると、すべてのパネルを閉じます。

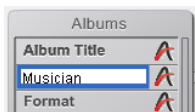
ウィンドウの位置および拡張した / 閉じたパネルの状態は保存されます。

互換性メモ テーブルインスペクタを使用して、アクセスグループにテーブルのデータ操作 (ロード、保存、追加および削除) を指定したり、テーブルオーナーを設定したりすることができなくなりました。詳細は、[ページ 25](#)、"[保持されないメカニズム](#)" の章を参照してください。

グラフィックモードでの直接編集

ストラクチャエディタの多くの機能 (新旧両方を含む) へは、グラフィックモードで直接アクセスできるようになりました。

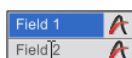
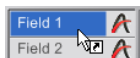
- **タイトルの変更:** グラフィックモードで、直接テーブルやフィールドのタイトルを変更できます。これを行うには、タイトルをクリックして編集モードに切り替えます。:



マウスを動かすタイトルのエリアによっては、カーソルの形が矢印 (例えば、リレーションを特定できる "選択" モードを示す)、または "I" ("編集" モードを示す) に変わる点に注意してください。

選択モード

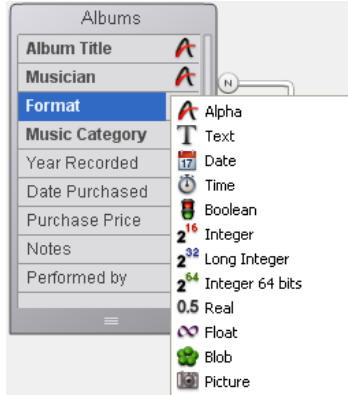
編集モード



- **フィールドの追加:** グラフィックモードで、直接フィールドを追加できます。これを行うには、テーブルが選択されている状態で、テーブルの空白エリアをダブルクリックするか、またはキャリッジリターンキーを押すと、新しい文字フィールドがテーブルの下端に追加されます。なお、フィールドのタイトルはデフォルトにより既に入力可能になっています。また、エディタツールバーにある追加ボタンを使ってフィールドを作成することもできます。

互換性メモ テーブルから削除したいフィールドを選択して Backspace キーを押す (または、エディタのコンテキストメニューから Delete コマンドを選択する) 方法でフィールドを削除できるようになりました。これに関する詳細は、[ページ 30](#)、"[テーブルとフィールドの削除](#)" の章を参照してください。

- フィールドタイプの変更: フィールド名の右側にあるポップアップメニューを使用して、直接フィールドタイプを変更できます。



- 同タイプオブジェクトの同時選択: 例えば、インスペクタウィンドウで通常のプロパティを表示または変更する場合など、複数のテーブルやフィールド、またはリレーションを同時に選択できるようになりました。複数のオブジェクトを選択する時は標準コマンドを使用してください。
- 隣接する複数のオブジェクトを選択する時は、Shift キーを押しながらクリックします。
- 隣接しない複数のオブジェクトを選択する時は、Ctrl キー (Windows) または Command キー (Mac OS) を押しながらクリックします。

互換性メモ 異なるテーブルからフィールドを選択することができます。

- テーブルを選択する範囲の囲み線を引きます (テーブルの選択時のみ)。
- 編集メニューまたはコンテキストメニューのすべてを選択コマンドを選択する (テーブルの選択時のみ)。
- Tab キーの使用: Tab キーを押す、または Shift キーを押しながら Tab キーを押すと各テーブルを選択できます。テーブルは、作成順ではなく、(画面表示に基づく) ポジション順に選択されます。

- コピー & ペースト: ストラクチャエディタを使用して、テーブルやフィールドをコピー & ペーストできます。

1 つまたは複数のオブジェクトをコピーするには、対象オブジェクトを選択し、標準の "コピー" コマンド (4D コンテキストメニューの編集メニューまたは Ctrl/Command キー +C のショートカット) を選択します。

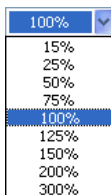
テーブルをペーストするには、編集メニューまたはエディタのコンテキストメニューのペーストコマンドを選択します。テーブルは、そのフィールドすべてと一緒にペーストされ、デフォルトにより名称が "Copy(X)_of_TableName" に変更されます。変更された名称の TableName の部分は元々のテーブル名、X の部分はテーブルのコピー数を表します。フィールドをコピーするには、フィールドを挿入したいテーブルを選択し、標準の "ペースト" コマンド (4D コンテキストメニューの編集メニューまたは Ctrl/Command キー +V のショートカット) を選択します。フィールドは、そのプロパティすべてと一緒にペーストされます。テーブルに既に同名のフィールドがある場合、デフォルトによりペーストされたフィールドの名称が "Copy(X)_of_FieldName" に変更されます。変更された名称の FieldName の部分は元々のフィールド名、X の部分はフィールドのコピー数を表します。

インタフェースおよび表示

ストラクチャエディタには、開発インタフェースおよびテーブルに関する新しいナビゲーション機能や表示機能を高めるため、フィールドやリレーションが用意されています。

ズーム

ストラクチャツールバーのズームメニューを使用して、実際のストラクチャ画面に拡大/縮小パーセンテージ表示が設けられました。:



データベースを開いた時のデフォルト値は 100% です。ズーム設定が選択されている場合は、そのパーセンテージがフォーカスされます。

カレントのズーム設定は各ユーザの設定によります。ズーム設定は、ウィンドウが閉じる時に記憶されます。

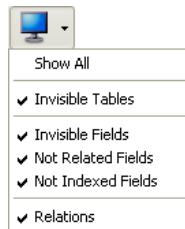
スクロールする

新機能により大型ストラクチャにおけるナビゲーションが簡単になりました。

- スクロールホイールを使用して、エディタウィンドウの内容を垂直方向に動かすことができます。また、テーブルのフィールド上にカーソルを持っていくと、フィールド内でも動かせます。
- Shift キーを押しながらスクロールホイールを使うと、ウインドウの内容を水平方向に動かすことができます。
- Shift キーを押すと "ハンド" ツールが有効になります。このツールを使用すると、空白エリアのクリックによりウインドウの内容すべてをドラッグできます。

オブジェクトタイプの表示

ツールバーにある新しい表示ボタンはメニューと連動しており、ストラクチャ内に表示するオブジェクトを選択できます。



デフォルトでは、すべてのオブジェクトが表示されます。

この機能は様々な表示を可能にします。つまり、最も単純なものから完璧なものまで、要求される情報のレベルに基づく分析ツールをストラクチャに供給します。

選択はすべてのテーブルとフィールドに適用されます。なお、選択は各ユーザーによって保存され、ウインドウを閉じる時に記憶されます。

特定のオブジェクトを隠すことでテーブルのポジションが変わることはありません。

- 複数のタイプのフィールドにチェックが入っている場合、表示するオブジェクトを決定するには論理演算子 OR が使用されます。例えば、不可視フィールドと索引なしフィールドのオプションにチェックが入っている場合、すべての索引なしフィールド(可視かどうかは問わない)、およびすべての不可視フィールド(索引の有無は問わない)が表示されます。

- テーブルの方がフィールドより優先されます。つまり、テーブルが表示されなければ、そのフィールドは表示されません。
- 表示されないオブジェクト(テーブル、フィールドまたはリレーション)を追加した場合、オブジェクトはエディタに表示されます。それを隠すには、表示ボタンと連動するメニューにある該当オプションを再度選択します。

フォルダごとにテーブルをハイライト

エクスプローラウィンドウで定義されたフォルダに基づいて、ストラクチャエディタのテーブル群を隠すことができます。

ツールバーにある新しいフォルダボタンを使用して、テーブル表示を構成できます。



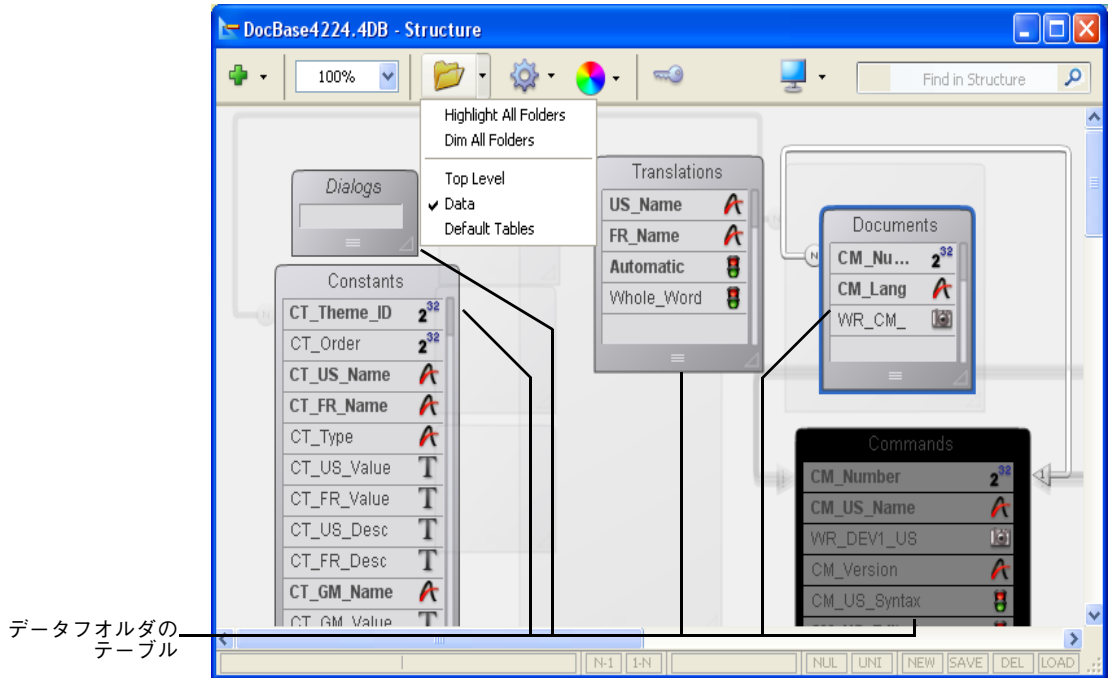
このボタンと連動するメニューは、表示を管理するコマンド、およびデータベースで指定されたフォルダリストを表示します。表示されているフォルダ名の隣にはチェックが入っています。カレント表示設定は、このメニューを使ってフォルダを選択または選択解除することで変更できます。

すべてのフォルダをハイライト」/すべてのフォルダを薄暗く表示コマンドを使用して、データベースのフォルダにあるすべてのテーブルを表示/隠すことができます。



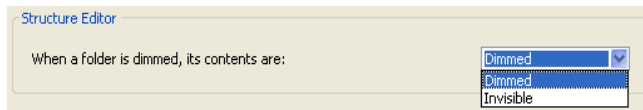
ボタンをクリックする度に、ハイライトは薄暗い表示またはその逆に、テーブルの表示が切り替わります。

テーブルが薄暗い表示の場合、デフォルトで、ストラクチャウィンドウにはアウトラインだけが表示されます。薄暗い表示になっていないテーブルのみ、その全体を見ることができます。



フォルダごとにテーブルを薄暗く表示

新しいオプションにより、ストラクチャエディタにある薄暗く表示されたテーブルの Appearance (フォルダ別) を、薄暗い表示または非表示のいずれかに設定できるようになりました。このオプションについては、環境設定のデザインモード/ストラクチャの節で説明しています。



互換性メモ この環境設定を有効にするには、ストラクチャエディタウィンドウを一度閉じて再度開く必要があります。

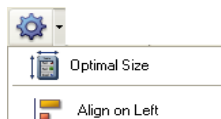
テーブルのサイズ変更

選択したテーブルの高さを自動変更する方法は2通りあります。

- 最適サイズ: これにより、選択されたテーブルを、そこに含まれるフィールドのサイズに完全対応するサイズに自動変更します (空の行は表示されません)。

1つまたは複数のテーブルに最適サイズを適用するには、

- エディタのツールバーボタンと連動するメニューの最適サイズコマンドを選択します (このコマンドは選択されたテーブルがない時は有効化されません)。



- エディタのコンテキストメニュー (テーブル上をクリック) の最適サイズコマンドを選択します。
- サイズ変更するテーブルの名前エリアを **Shift** キーを押しながらダブルクリックします。このショートカットを繰り返すと、オリジナルサイズ->最適サイズ->折りたたむ、という一連の選択が繰り返されます。

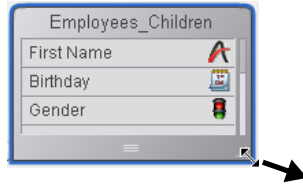
- 折りたたむ: この機能は、選択されたテーブルを折りたたみ、テーブル名だけを表示します。これは、大型ストラクチャの場合に便利です。

1つまたは複数のテーブルに折りたたむコマンドを適用するには、

- エディタのコンテキストメニュー (テーブル上をクリック) の折りたたむコマンドを選択します。
- サイズ変更するテーブルの名前エリアを **Shift** キーを押しながらダブルクリックします。このショートカットを繰り返すと、オリジナルサイズ->最適サイズ->折りたたむ、という一連の選択が繰り返されます。

-
- 互換性メモ**
- ・ テーブル名のエリアで通常のダブルクリックをすると、インスペクタを開きます。
 - ・ **Alt** キー (Windows) または **Option** キー (Mac OS) を押しながらダブルクリックすると、トリガメソッドを開きます。
 - ・ **Ctrl** キー (Windows) または **Command** キー (Mac OS) を押しながらダブルクリックすると、フォームページ上にエクスプローラを開きます。
-

ストラクチャエディタのテーブルは、左下の角をクリックすることにより手動で幅も変更できるようになった点に注意してください。



フィールドの順序

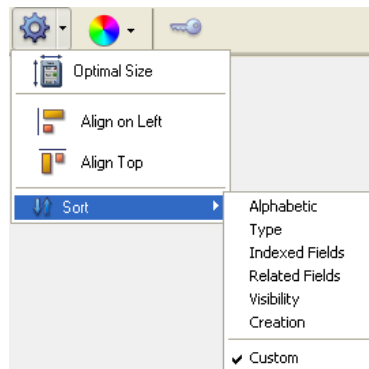
4D v11 では、テーブル内でフィールドが現れる順番を変更できます。その方法は2通りあります。

- フィールドをドラッグ & ドロップして、手動でテーブルの内容を再構成します。Alt キー (Windows) または Option キー (Mac OS) を押すとカーソルが手の形になるので、フィールドを任意の場所に移動させます。:



この方法で行われた独自の並び替えは記憶され、必要に応じて、並び替えメニュー (下記参照) のカスタムコマンドを使用すると再度適用できます。

- 並び替えの基準を適用します。エディタのツールボタンと連動するサブメニューには、並び替えの基準が幾つか用意されています。



あるオプションを選択すると、それはウィンドウで選択されているテーブルすべてに適用されます。このメニューは、選択されたテーブルがない時は無効です。

また、このメニューでは、現在適用されている基準にチェックマークがつきます。選択テーブル内で異なる複数の基準が使用されている場合は、複数のチェックマークが表示されます。


並び替えの基準は、次の中から選択できます。

- 文字コード順: フィールドを文字コード順に表示します。
- タイプ: フィールドをタイプ別に表示します。
- Iインデックスフィールド: インデックス付フィールドを上部に表示します。
- リレートしたフィールド: プライベートキーと次に外部キーを表示します。
- 表示設定: 可視フィールドを上部に表示します。
- 作成: フィールドを作成順に表示します(デフォルト)。
- カスタム: 特定の並び替えは適用されません。このオプションは、ドラッグ & ドロップにより手動で行った以前の並び替えを復帰させます。先にドラッグ & ドロップを行っていない場合、このオプションは効力を持ちません。

-
- 互換性メモ
- ・ テーブルにフィールドを追加した場合、そのフィールドは現在適用されている並び替えの基準に拘わらず、常に最後のフィールドの後に配置されます。
 - ・ ストラクチャエディタで行われたテーブルの並び替えは、アプリケーションの他のエディタにあるフィールド表示に対して、なんの影響も与えません。
-

カラー

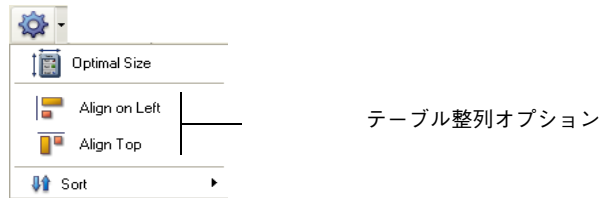
テーブルやフィールド、またはリレーションにカラーを設定できます。1つまたは複数のオブジェクトにカラーを設定するには、下記により表示される標準カラーパレットを選択してカラーを指定します。

- エディタのツールバーにある  カラーボタン
- エディタのコンテキストメニュー(テーブル、フィールドまたはリレーションの上でクリック)のカラーコマンド
- インспекタパレットのカラーオプション

-
- 互換性メモ
- 200x 度版 4D の環境設定で利用できたカラー名や背景カラーのオプションは、4D v11 からなくなりました。背景カラーは新しくテーブルのカラーに、カラー名は新しくフィールドのカラーになりました。
-

テーブルの整列

ツールメニューには、エディタウィンドウで選択されたテーブルを縦方向または横方向に揃えるオプションが2つあります。



テーブル整列オプション

これらのコマンドは、少なくとも2つのテーブルが選択されている時だけアクティブになります。

ストラクチャの描画 クォリティ

このオプションは、環境設定のデザインモード/ストラクチャページにあります。このオプションは、ストラクチャエディタの描画レベルを変更するために使用できます。



デフォルトで、クォリティは高に設定されています。標準クォリティを選択して、表示速度を向上させることができます。特にズーム機能を使用する際、その効果が顕著に現れます。

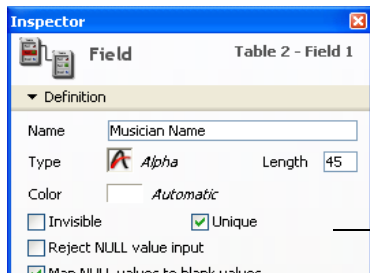
SQL 属性のサポート

4D v11 には、パワフルな統合 SQL エンジンが搭載されています ([ページ 199](#)、"[4D SQL エンジンの使用](#)"の章を参照)。

SQL 言語は特にテーブル、フィールドまたはリレーションの作成に使用されるので、この新しい言語の特性をサポートするためインスペクタウィンドウに変更が加えられました。

NULL値サポートオプション

フィールドインスペクタに、4D フィールドで NULL 値を管理するための新しい二つのオプションが追加されました。



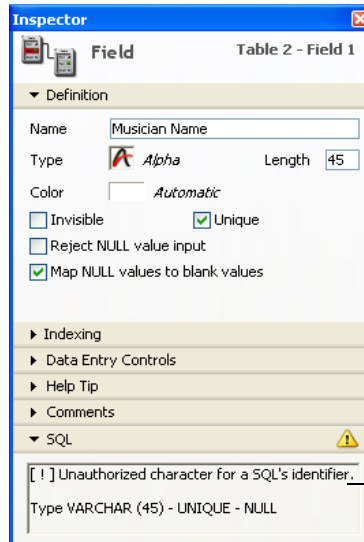
新しいフィールドオプション

互換性メモ NULL 値は SQL のコンセプトです。これに関する詳細は [ページ 211](#)、"[4D での NULL 値](#)" の章を参照してください。

- NULL 値の入力を拒否: このオプションはフィールドが NULL を受け入れるかどうかを指定します。これは SQL 言語の NOT NULL 属性にあたります。このオプションがチェックされていると、4D はこのフィールドに対する NULL 値を許可しません。
- NULL 値を空値にマップ: この互換オプションは、フィールドに存在する可能性のある NULL 値を強制的にも、4D の空値として扱うために使用します。これにより 4D エンジンによるクエリや処理がフィールド中の NULL 値により壊されることがなくなります。
つまり、このオプションがチェックされていると、`MyAlphaField=""` のクエリは `MyAlphaField` フィールドに格納された NULL 値も検索します。

SQL エリア

インスペクタパレットには、テーブル、フィールドおよびリレーションのための、SQL という名の新エリアが追加されました。このエリアは、SQL 言語での使用という観点から、これらオブジェクトに関する様々な情報を提供しています。:



SQL 情報エリア

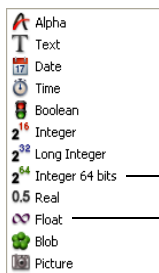
このエリアは、特にテーブル名やフィールド名またはリレーション名が SQL 命名法のルールに従っているかどうかを示します (例えば、4D と異なり、SQL ではスペースを含むフィールド名は許されません)。

フィールドに関しては、SQL 情報エリアは SQL 属性 (タイプおよびプロパティ) も表示します。

リレーションについては、このエリアは FOREIGN KEY プロパティと REFERENCES を表示します。

64ビット整数タイプおよび浮動タイプ


フィールドタイプのリストに新しい2つのタイプ、Integer 64 bits タイプおよび Float タイプが追加されました。:

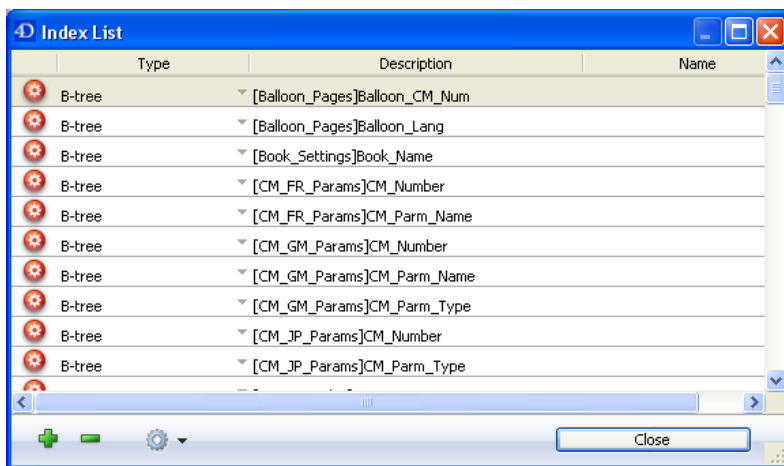


新しいフィールドタイプ

これらのフィールドタイプは、データベースのSQLエンジンの使用に関係しています。これに関連して、各タイプはそれぞれ64ビット整数と浮動小数点型数値(ページ30、"新しいフィールドタイプ"の章を参照)を格納する時に使用します。4D エンジンを使ってこれらフィールドの作業をする場合、フィールドに含まれるデータは内部的に実数値に変換される点に注意してください。

インデックスエクスプローラ


ストラクチャエディタのツールバーにある  ボタンは、インデックスエクスプローラウィンドウを表示します。このエクスプローラは、ストラクチャの全インデックスおよびそれぞれのプロパティのリストを表示します。:




互換性メモ インデックスタイプに関する詳細は、ページ35、"インデックスの管理"の章を参照してください。

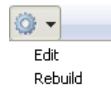
インデックスエクスプローラは、インデックスの主要プロパティを表示する時に使用します。

- **タイプ:** インデックスタイプを表示します。インデックスの各タイプ (B-Tree、クラスタ B-Tree、キーワード) は、それぞれ異なるアイコンで描写されます。
逆三角形をクリックしてポップアップメニューの値を指定すると、インデックスエクスプローラのインデックスタイプを変更できます。
- **説明:** インデックスのテーブルおよびフィールドを表示します。複合インデックスに関しては、このリストはインデックスの全フィールドを含みます。
- **名前:** インデックス名を表示します。このプロパティは、特に新しい言語コマンドで使用します (ページ 363、"ストラクチャアクセス"の章を参照)。データベースが変換されたインデックスは、デフォルトにより名前がありません。
このエリアをダブルクリックすると、インデックス名を変更または追加できます。

 ボタンは、インデックスプロパティダイアログボックスを表示します (次節を参照)。

 ボタンは、選択されたインデックスを削除します (確認ダイアログボックスが表示されます)。

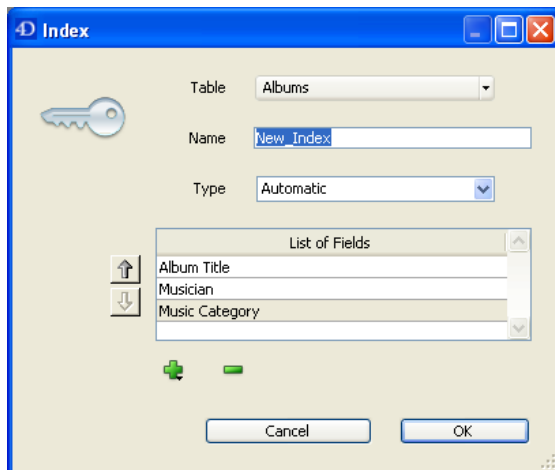
ツールボタンと連動するメニューに、2つのコマンドが追加されました (インデックスが選択されている時に有効)。




- **編集:** 選択されたインデックスのプロパティを表示します。このコマンドには、リストの列 (名前エリアを除く) をダブルクリックする時と同じ効果があります。
- **再構築:** 選択されたインデックスを削除して再構築する時に使用します。このコマンドを選択すると、確認ダイアログボックスが表示されます。

インデックスプロパティダイアログボックス

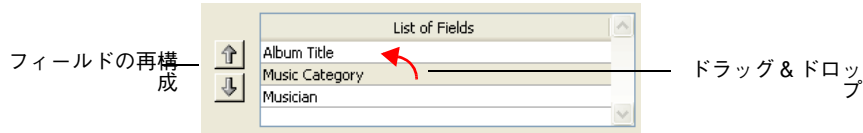
インデックスエクスプローラ、オブジェクト追加メニューまたはエディタのコンテキストメニューでインデックスを追加する時は、インデックス作成ダイアログボックスが表示されます。:




このダイアログボックスは、インデックスを構成する時に使用し、次の項目が含まれます。

- テーブル: データベースの全テーブルのリストを表示します。このメニューからインデックスが所属するテーブルを選択します。
- 名前: インデックス名を入力するエリアです。
- タイプ: 作成するインデックスのタイプを選択するメニューです。"自動"オプションを保持すると、4D はフィールドの内容に従って自動的にインデックスのタイプを選択します。
- フィールドのリスト: このエリアは、インデックスを作成するフィールドを指定する時に使用します。デフォルトでは、インデックスに割り当てられたフィールドが表示されます。
インデックスにフィールドを追加するには、 ボタンをクリックします。選択したテーブルのフィールドリストが表示されるので、インデックスに追加するフィールドを指定できます。

- 複合インデックスを作成する場合は、インデックスに入れるフィールドを連続して追加します。
リストが完成したら、矢印ボタンまたはドラッグ & ドロップを使ってフィールドを再配列できます。



- "キーワードインデックス"タイプを選択した場合、指定できるフィールドは文字またはテキストだけです。またこの場合、インデックスには1つのフィールドだけを割り当てることができます。

インデックスからフィールドを削除するには、リストから選択して  ボタンをクリックします。

インデックスの構成が完了したら、OK をクリックしてインデックスを作成します。

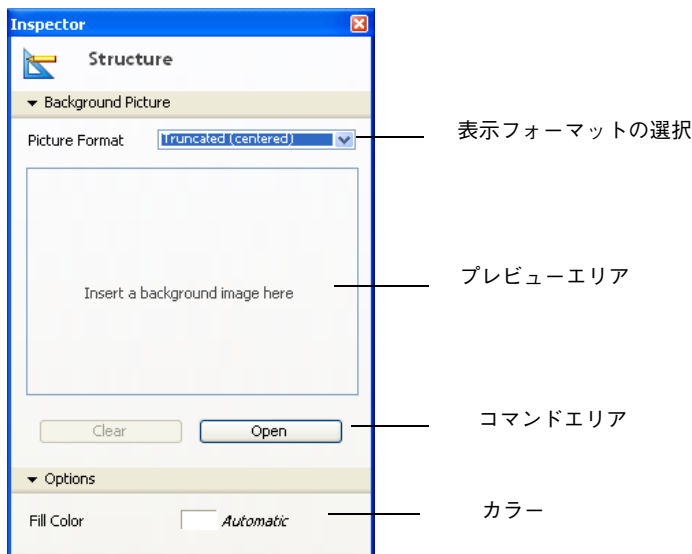
エディタウィンドウのカスタマイズ

ストラクチャエディタウィンドウには、インスペクタを使って設定できる特定のプロパティがあります。

ストラクチャエディタウィンドウのプロパティを表示させるには、次のいずれかを実行します。

- ウィンドウの空白エリアをダブルクリック (または、ウィンドウが既に表示されている時はクリック)する。
- ウィンドウの空白エリアを右クリックし、コンテキストメニューからストラクチャプロパティコマンドを選択する

インスペクタウィンドウにストラクチャプロパティが表示されます。:



次のようなプロパティを設定できます。

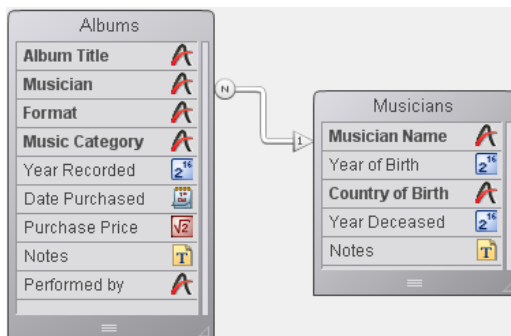
- 背景のピクチャ: 使用する背景のピクチャ、およびその表示フォーマットを変更できます。
 - ピクチャを変更するには、「開く ...」ボタンをクリックするか、またはプレビューエリアを右クリックしてコンテキストメニューから「開く ...」を選択します。それから、表示するピクチャを含むファイルを選択します。あらゆるピクチャフォーマットが使用できます。選択されたピクチャは、プレビューエリアおよびエディタウィンドウにすぐに表示されます。
 - ピクチャの表示フォーマットを変更するには、ピクチャフォーマットメニューから値を選択します。用意されているフォーマットは、4D の標準ピクチャ表示フォーマットです。
 - カスタムピクチャを削除するには、「クリア」ボタンをクリックするか、またはプレビューエリアを右クリックしてコンテキストメニューから「クリア」を選択します。
- 塗りつぶしカラー: エディタウィンドウの背景に使われているカラーを変更できます。これを行うには、カラー選択エリアをクリックして選択メニューからカラーを指定します。

リレーション

4D v11 では、リレーションの外観とインターフェースが変更されました。

外観

リレーションは、特に文字の N および 1 が追加された点など、その外観が変更されました。



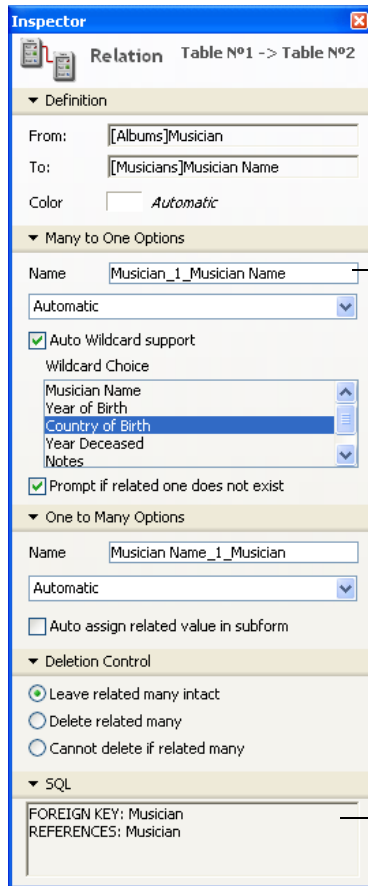
リレーションは、SQL (正方形のコネクタ) で作成された場合は、その外観が異なります。



SQL リレーション

リレーションプロパティ

リレーションが選択されている時、インスペクタはそのプロパティを表示します。



リレーション名

SQL エリア

リレーションの構成要素 (1 対 N リレーションおよび N 対 1 リレーション) は、どちらも命名が可能になりました。この新しいプロパティの特性は、4D v11 の今後のバージョンに活かせるようになります。

コンテキストメニュー

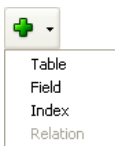
各リレーションと関連するコンテキストメニューを使用して、リレーション関係の主要編集機能に直接アクセスし、"自動"プロパティを有効化または無効化することができます。:



リレーション用のコンテキストメニューを表示させるには、コネクタの1つを右クリックします。

ツールバーからリレーションを追加する

ストラクチャエディタのツールバーにある [+] ボタンは、異なるタイプのオブジェクト (テーブル、フィールド、インデックス、またはリレーション) を追加する時に使用します。



2つのフィールドが選択されており、それぞれのタイプが互いのフィールド間で直接リレーションを作成することを許可する場合は、リレーションコマンドを使用できます。

最初に選択されるフィールドは外部キーフィールド (N フィールド)、2番目に選択されるフィールドは主キーフィールド (1 フィールド) として認識されます。

リレーションが引かれ、インスペクタがリレーションプロパティを表示します。

3つ以上のフィールドが選択された場合、このコマンドは何もしません。

リレーションの削除

4D v11 では、新しいリレーションを開始ポイントから引いて空白エリアで放す方法によるリレーションの削除はできなくなりました。今後、リレーションを削除するには、次のいずれかを実行します。

- リレーションを選択し、Delete キーまたは Backspace キーを押します。
- または、リレーションに関連するコンテキストメニューから削除コマンドを選択します。

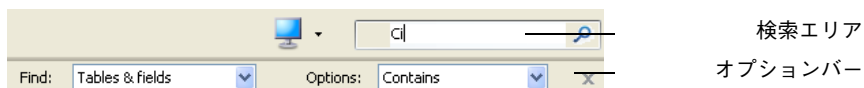
いずれの場合も警告ダイアログボックスが表示されるので、動作の確認または取り消しができます。

ストラクチャ内での検索

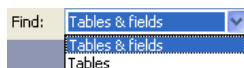
4D v11 では、ストラクチャエディタウィンドウ内で 検索を実行できます。検索は、次のような要素から行われます。

- フィールド名
- テーブル名
- テーブル番号

検索を行うには、ストラクチャエディタの「ストラクチャ内で検索」エリアに、検索する文字列またはテーブル番号を入力します。このエリアに値を入力すると、その下にオプションバーが表示されます。これを使用して、要求される検索の範囲およびタイプを指定できます。



- 「検索」メニューは、検索の範囲(テーブルおよびフィールド、またはテーブルのみ)を設定する時に使用します。

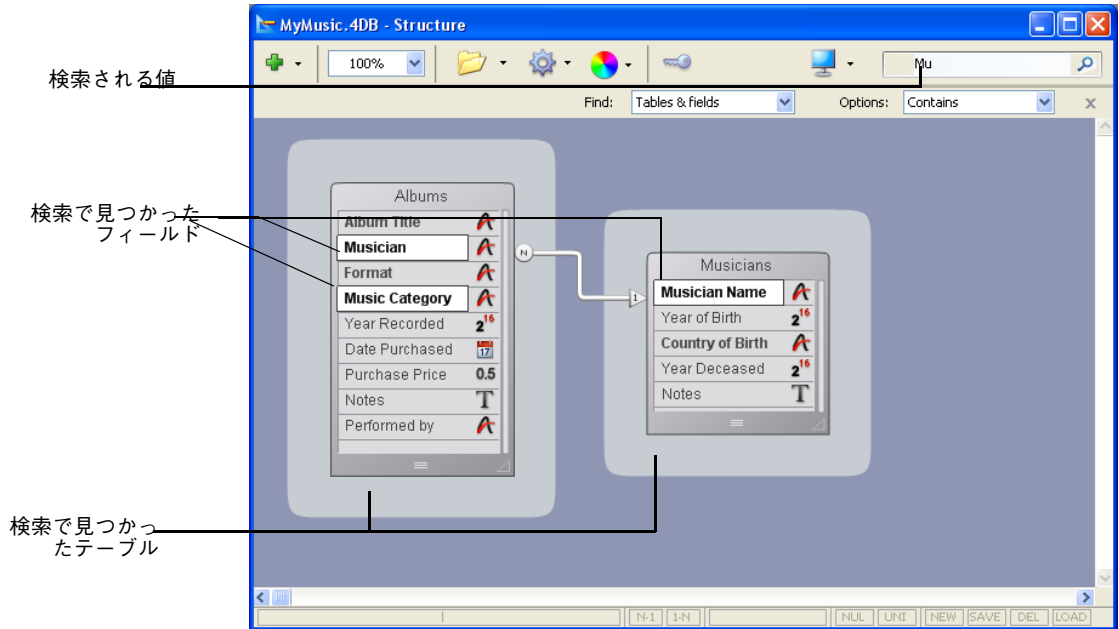


- 「オプション」メニューは、検索のタイプを設定する時に使用します。



- 含む (デフォルト): "le" で検索すると、"Table"、"Letter"、"Elements" などを探します。
- 以下で始まる: "pa" で検索すると、例えば "paper" や "paid" などを探しますが、"repair" は探しません。
- 番号: "2" で検索すると、例えばテーブル番号 2、12、20、21 などを探します。

検索は、値を入力するリアルタイムで行われます。検索で見つかったテーブルやフィールドは"明るい"表示になります。



何も見つからない場合は、検索エリアが赤色の表示になります。:



"検索"モードを終了するには、検索オプションバーの **X** ボタンをクリックするか、または検索エリアの文字をすべて削除します。

ストラクチャ定義の書き出しおよび読み込み

4D v11 では、データベースのストラクチャ定義を XML ファイルまたは HTML ファイルとして書き出すことができます。逆に、XML フォーマットに保存したストラクチャ定義を使って新しい 4D データベースを作成することもできます。

これらの新しい機能は、次のような異なった必要性に対応します。

- ストラクチャを、カスタムフォーマット(レポートやテーブルなど)で表記できる、または他の環境で分析できるようにする。
- データベースを、定義ファイルから作成できるようにする。

4D ストラクチャ定義のフォーマット

4D v11 のストラクチャ定義は XML フォーマットに基づきます。単純なテキストエディタを使用してストラクチャ定義を表示できます。また、XML フォーマットは、特に XSL 変換など、他の方法で使用できます。更に、4D がストラクチャ定義を HTML フォーマットで書き出す時は XSL ファイルを使用します。

ストラクチャ定義には、テーブル、フィールド、インデックス、リレーションとそれらの属性、および定義の完成に必要な様々な特性が含まれます。4D ストラクチャ定義の内部の "文法" は、XML ファイルの有効化にも使用される DTD ファイルを通してドキュメント化されます。

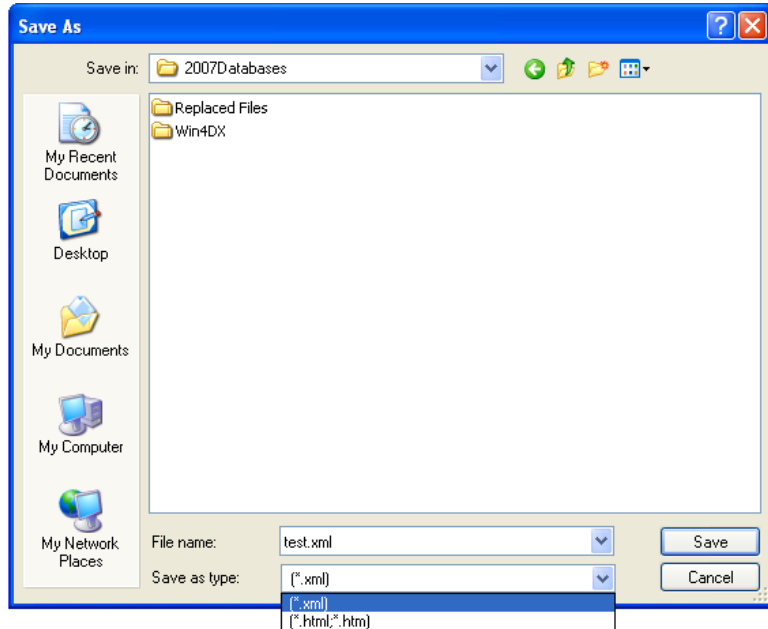
4D で使用される DTD ファイルは、4D アプリケーションと同階層にある DTD フォルダにグループ化されています。ストラクチャ定義には、`base_core.dtd` ファイルおよび `common.dtd` ファイルが使用されます。4D ストラクチャの定義に関する詳細は、これらのファイル、およびファイル中にあるコメントを参照してください。

ストラクチャ定義の書き出し

4D v11 では、XML フォーマットまたは HTML フォーマットでストラクチャを書き出すことができます。必要に応じてフォーマットを選択します:

- XML フォーマット: XML フォーマットのストラクチャは、単純なテキストエディタを使用する他、様々な方法 (カスタム XSL 変換、読み込み、他のソフトウェアでの分析など) で表示することができます。
新しいデータベースを作成するためにストラクチャ定義を使用する場合は、このフォーマットを選択します。
- HTML フォーマット: このフォーマットのストラクチャは、ブラウザで表示およびプリントできるレポートの形で表記されます。
- ▼ XML でストラクチャ定義を書き出すには:
 - 1 4D 「ファイル」メニューの「書き出し」からストラクチャ定義を XML ファイルに書き出し ... コマンドを選択する。

標準の「別名で保存」ダイアログボックスが表示されたら、ファイルの名前、場所および書き出しファイルのタイプを指定します。



2 書き出しファイルの名前と場所を指定し、ダイアログボックスを確定します。

▼ HTMLでストラクチャ定義を書き出すには、

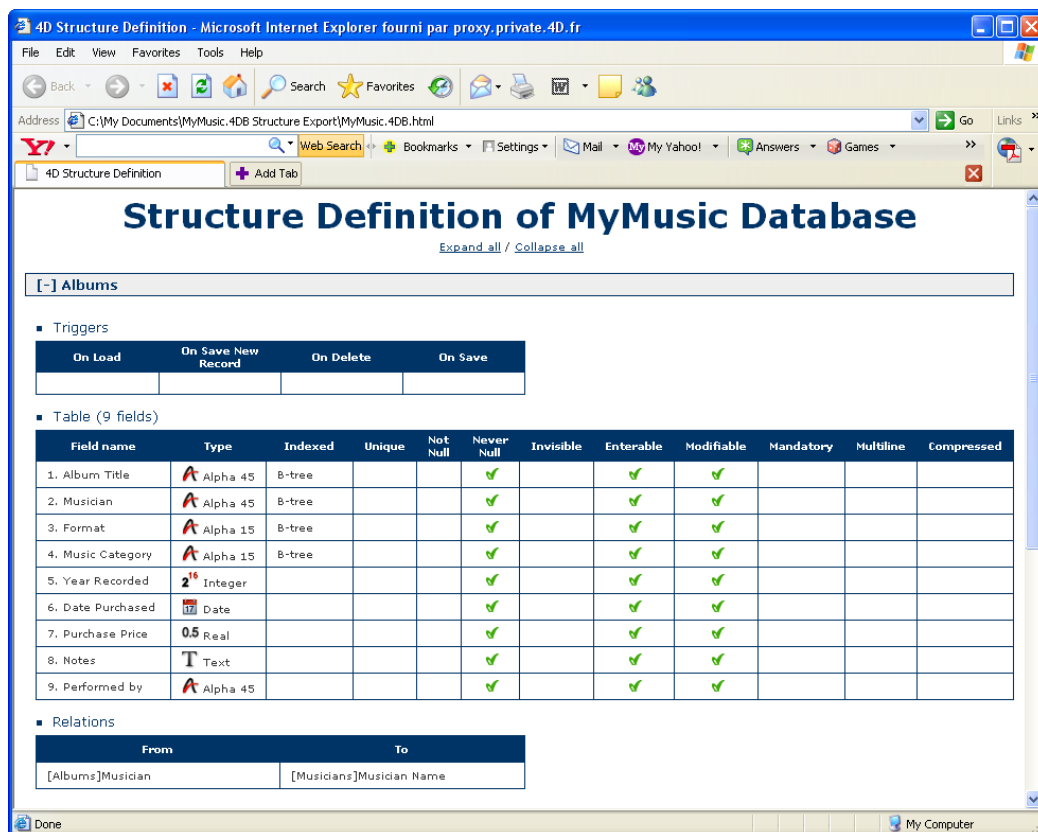
1 4D「ファイル」メニューの「書き出し」からストラクチャ定義をHTMLファイルに書き出し... コマンドを選択します。

フォルダ選択時に使用したダイアログボックスが表示されたら、HTMLファイルを格納するファイルの場所を指定します。

2 「新規フォルダ作成」をクリックするか、または既存のフォルダを指定します。

4D v11は、指定された場所に自動的にフォルダを作成します。この、書き出し項目を含むフォルダの名前は、例えば "Mystructure.4db Structure Export" となります (Mystructure.4db の部分はデータベースストラクチャファイルの名前)。

ダイアログボックスにより、書き出しの結果をデフォルトのブラウザで直接表示できます。HTML フォーマットによるストラクチャ定義のサンプルは下記の通りです。



XSL 変換のカスタマイズ 4D は、ストラクチャ定義の HTML ページを作成するため、デフォルトではアプリケーションの「/Resource/language.lproj」サブフォルダにある "Structure_to_html.xml" ファイルを使って XSL 変換を実行します。

互換性メモ このファイルが存在しない場合、書き出しダイアログボックスを使用した HTML フォーマットによる書き出しはできません。

この変換は、カスタム XSL スタイルシートファイルを使用してカスタマイズできます。これを行うには、"Structure_to_html.xml" という名前のファイルを作成 (またはデフォルトのファイルを複製) して、それを「.4db」ファイルと同階層に配置します。すると、4D はこのファイルを使用して HTML フォーマットでストラクチャ定義を作成します。

ストラクチャ定義からのデータベース作成

XML フォーマットで書き出されたストラクチャ定義を使用して、全く同じデータベースを新しく作成することができます。この場合のストラクチャ定義は、ストラクチャテンプレートの役割を果たし、コピーを作成することができます。

XML ストラクチャ定義は、そのまま、または XML エディタで予め変更して使用できます。つまり、プログラミングによるストラクチャ作成に使われるあらゆるメカニズムの使用を考慮できます。

更に、4D ストラクチャ定義 XML ファイル内部のフォーマットは公開されているので (ページ 124、"4D ストラクチャ定義のフォーマット" の章を参照)、4D データベースを自動的に作成するため、他のデータベース環境またはあらゆるデザインアプリケーションからこの種類のファイルを構築することができます。

- ▼ ストラクチャ定義からデータベースを作成するには、
 - 1 4D 「ファイル」メニューの「新規」からストラクチャ定義を使用したデータベース ... コマンドを選択します。

標準の「ドキュメントを開く」ダイアログボックスが表示されたら、開く定義ファイルを指定します。その際は、4D ストラクチャ定義の "文法" に従う XML フォーマットを選択する必要があります (プログラムは DTD を通してファイルを有効化します)。
 - 2 ストラクチャ定義 XML ファイルを選択して「OK」をクリックします。

4D のダイアログボックスが表示されたら、作成するデータベースの名前と場所を指定します。
 - 3 作成するデータベースの名前と場所を指定して「保存」をクリックします。

XML ファイルが有効の場合、4D は (もし、あれば) カレントのデータベースを閉じ、ストラクチャ定義に基づく新しいストラクチャを作成し、エクスプローラウィンドウを表示します。また、デフォルトで空白のデータファイルも作成されます。

6

メソッドエディタ

4D v11のメソッドエディタには、次のような新しい機能が加えられました。

- SQL コードの入力
- コンテキストメニューのタイプアヘッド機能による定数の表示
- 新しい検索と置換オプション
- 新しいマクロアーキテクチャ

メソッドエディタへの SQL コード入力

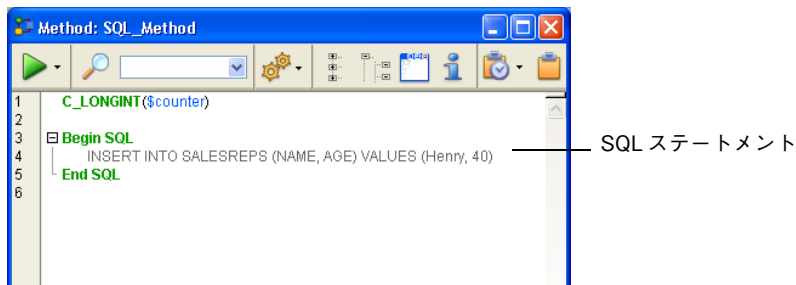
4D v11 には、ネイティブな SQL エンジンが搭載されています。([ページ 199](#)、"[4D SQL エンジンの使用](#)" の章を参照)。

メソッドエディタへの SQL コードの直接入力可能な点は、この新しいエンジンから恩恵を受ける方法の1つです。

その他の2つは、ODBC([ページ 249](#)、"[外部データソース](#)" の節を参照)と新しい[ページ 249](#)、"[外部データソース](#)" の節の[ページ 254](#)、**QUERY BY SQL コマンド** を使用できる点です。

4D メソッドエディタに SQL コードを挿入する原則は簡単です。SQL コードを **Begin SQL** タグと **End SQL** タグの間に入れます。

4D の標準インタープリタは、2つのタグ間に入力された SQL ステートメントを解析しません。:



これらのタグは、次の原則に従って働きます。

Begin SQL/End SQL タグで設定されるブロックは、同じメソッド内に1つまたは複数配置できます。SQLコードのみの構成でメソッドを作成できますし、また同じメソッド内で4DコードとSQLコードを組み合わせることも可能です。

- 複数のSQLステートメントを1行または複数行に置く場合は、";" (セミコロン) で区切ります。次は、その例です。

```
Begin SQL
  INSERT INTO SALESREPS (NAME, AGE) VALUES ("Henry",40);
  INSERT INTO SALESREPS (NAME, AGE) VALUES ("Bill",35)
End SQL
```

または、

```
Begin SQL
  INSERT INTO SALESREPS (NAME, AGE) VALUES
("Henry",40);INSERT
INTO SALESREPS (NAME, AGE) VALUES ("Bill",35)
End SQL
```

ただし、注意点として、4D デバッガは行毎にSQLコードを解析するので、場合によっては複数行を使う方が便利かもしれません。

SQLコードのデバッグ

標準の4Dデバッガは **Begin SQL/End SQL** タグ内のブロックをデバッグします。複数行に置かれたSQLステートメントは、通常の4Dステートメントと同様、行毎に解析されます。1行に複数のSQLステートメントがある場合、4Dデバッガはそれらすべてをまとめて解析します。

デバッガを使用して、単純な4Dステートメントに関するものと同じ情報を得ることができます。4Dデバッガについての詳細は、「4Dランゲージリファレンス」マニュアルを参照してください。

SQL関連のエラーは **ON ERR CALL** メソッドによってトラップされます。エラーが発生し、エラーメソッドが何もインストールされていない場合、4Dはカレントコマンドおよびカレントメソッドの実行を中止します。

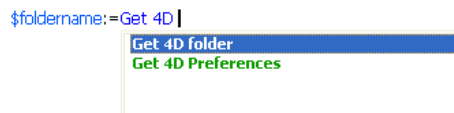
4Dストラクチャの名前

次の例のような4D列(フィールド)を含むSQLステートメントの場合、
SELECT eName FROM PEOPLE INTO <<[MYTABLE]Name>>

4D のテーブル名またはフィールド名がストラクチャレベルで変更された場合、その変更はコード中では認識されないため、各 SQL ステートメントの中で変更する必要があります。

コンテキスト定数

4D v11 のメソッドエディタでは、タイプアヘッド機能を使った定数の挿入がコンテキストメニューで実行できるようになりました。タイプアヘッドメニューに表示される定数は、入力内容によって変わります。つまり、入力するコマンドパラメータに関連のある定数だけがウィンドウに表示されます。



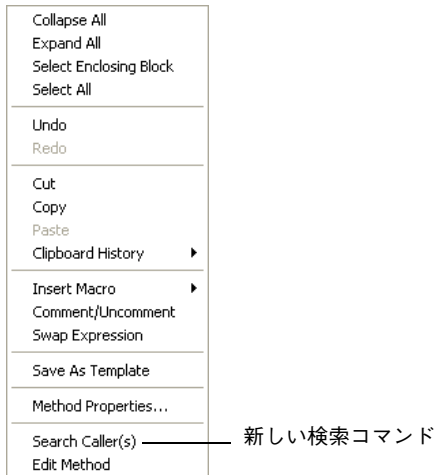
このメカニズムは、各コマンドのパラメータおよび機能に対して動作します。

検索

メソッドエディタには検索と置換に関する 2 つの新しい機能、つまり特定のメソッドの "呼び出し元" を検索する機能、および置換を選択する機能が追加されました。

呼び出し元の検索

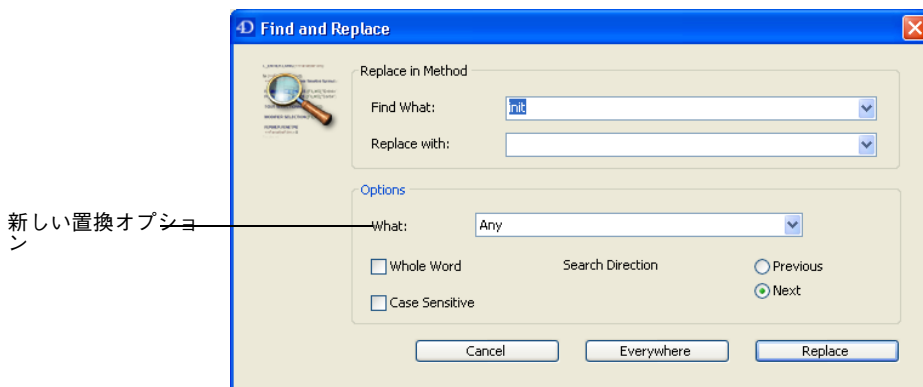
プロジェクトメソッドが選択されている場合、メソッドエディタのコンテキストメニューに「呼び出し元の検索」という新しいコマンドが追加されました。



このコマンドは、結果ウィンドウで選択されたプロジェクトメソッドを呼び出すオブジェクト (メソッドまたはメニュー) のリストを表示します。これは、エクスプローラからも可能です ([ページ 75](#)、["呼び出し元の検索" の節を参照](#))。

選択可能な置換オプション

4D v11 では、メソッドエディタの標準 "検索と置換" ダイアログボックスに、置換を行う時にターゲットオブジェクトを1つのタイプに制限できる新しいオプションが追加されました。



次のようなタイプがあります。

- すべて (標準操作)

- 変数: 変数の名前にある文字列だけを置き換えます。
- プロジェクトメソッド: プロジェクトメソッドの名前にある文字列だけを置き換えます
- コメント: コメントの中にある文字列だけを置き換えます。

新しいマクロアーキテクチャ

4D v11 の マクロコマンドファイルは、XML 標準と完全に互換性を持つようになりました。具体的には、マクロファイルをロードするためには、マクロファイルの最初にXML宣言文<?xml version="1.0" ...?>およびドキュメント宣言文 `<!DOCTYPE macros SYSTEM "http://www.4d.com/dtd/v11/macros.dtd">` を置くことが義務づけられました。異なったタイプの XML エンコーダをサポートしますが、互換性のあるエンコーディング Mac/PC (UTF-8) を推奨します。

4D には、マクロファイルの検証に使用する DTD が備わっています。このファイルは、次の場所に置かれています。

- Windows 版: 4D Developer/Resources/DTD/macros.dtd
- Mac OS 版: 4D Developer.app:Contents:Resources:DTD:macros.dtd

マクロファイルが宣言ステートメントを含まない、または検証されない場合、それはロードされません。

この新しいメカニズムは前バージョンのマクロフォーマットと互換性がありません。従って、4D v11 は "Macros v2" という新しいフォルダからマクロをロードします。マクロは、このフォルダに配置される 1 つまたは複数の XML ファイルという形で表示されなければなりません。

"Macros v2" フォルダが配置される場所は次の通りです。

- マシンのアクティブな 4D フォルダ (4D の前バージョン同様)。マクロはすべてのデータベースで共有されます。
- データベースのストラクチャファイルと同階層。マクロはこのストラクチャにのみロードされます。
- コンポーネント用はデータベースの「Components」フォルダ。マクロは、コンポーネントがインストールされている場合のみロードされます。

これら 3つの場所は、それぞれに "Macros v2" フォルダをインストールすれば、同時に使用できます。マクロは、4D フォルダ、ストラクチャファイル、コンポーネント 1 からコンポーネント X の順にロードされます。

旧マクロからの移行

4D v11 と前バージョンで定義されたマクロの互換性を保つため、自動変換メカニズムが導入されました。4D が起動すると、プログラムはアクティブな 4D フォルダの中の "Macros v2" フォルダの存在を確認します。

存在する場合、その中のマクロファイルはロードされます。

存在しない場合は "Macros v2" フォルダが作成され、4D は "Macros.xml" ファイル (4D v2003/2004) または "Macros" フォルダ (4D 2004) の存在を確認します。このファイルまたはフォルダの内容は変換され、新しい "Macros v2" フォルダにコピーされます。既存のマクロファイルを変換する時、XML 宣言ステートメントおよびドキュメント宣言ステートメントが追加されます。

アクティブな 4D フォルダに前バージョンのマクロファイルまたはマクロフォルダが全く含まれない場合、デフォルトのマクロを含む "Macros v2" フォルダが作成されます。

XML 標準に関連する非互換性

マクロファイルが XML 標準に準拠するよう、厳密なシンタックスルールに従う必要があります。

これは、既存のマクロコードとの非互換性の原因となり、XML ファイルのロードを阻害します。機能障害の主な原因は次の通りです。

- 以前のマクロフォーマットの `<macro>` 要素の中では許可されていた `"// my comment"` タイプのコメントと XML シンタックスは互換性はありません。
コメントの行は、標準の `"<!-- my comment -->"` に準拠する必要があります。
- 特にインタープロセスオブジェクトの名前に使用されたシンボルの `<>` はコード化する必要があります。例えば、「`<>params`」の変数は「`<>params`」と記述しなければなりません。
- 最初の `<macros>` 宣言タグは、4D の以前のバージョンでは省略が可能でしたが、このタグの使用は義務づけられ、これがない場合はファイルがロードされません。

<macro> 要素の新しい属性

<macro>要素は2つの新しい属性、つまり `method_event` および `version` をサポートします。

- `method_event`: この属性を使用すると、各メソッドでのカレントの処理段階(作成する、閉じるなど)に応じてマクロの自動呼び出しをトリガできます。次のような値のいずれかをとります。
 - `on_load`: マクロは各メソッドを開く時にトリガされます。
 - `on_save`: マクロは各メソッドを保存する時にトリガされます(変更されたメソッドを閉じる時、または「ファイル」メニューの「保存」コマンドを使って保存する時)。
 - `on_create`: マクロは各メソッドを作成する時にトリガされます。
 - `on_close`: マクロは各メソッドを閉じる時にトリガされます。

`on_save` と `on_close` は一緒に使用できます。つまり、この2つのイベントは両方とも変更されたメソッドが閉じられる時に起こります。一方、`on_create` と `on_load` は、連続して起こることはありません。

この新しい属性を使用すると、例えば、メソッドを作成する時(ヘッダエリアにコメント)にそれをプリフォーマットしたり、メソッドを閉じる時に日付や時間などの情報を記録することができます。

- `Version`: この属性を使用すると、マクロのテキスト選択をサポートする新しいモードを有効化できます(次節を参照)。この新しいモードを有効化するには、マクロ要素の「`version="2"`」の属性を渡します。この属性を省略した場合、またはマクロ要素の「`version="1"`」を渡した場合、前のモードが保持されます。

メソッドのテキスト 選択変数

4Dの前バージョンでは、プログラムは、<method> タグを使用する時にメソッド内でテキストを処理するプロセス変数のセットを自動的に保存していました。<method> タグとは、テキストを回復する入力変数(`_textSel` や `_blobSel` など)、およびテキストを挿入する出力変数(`_textReplace` や `_blobReplace` など)です。詳細は、「4D デザインリファレンス」マニュアルを参照してください。

このメカニズムは、互換性の目的から 4D v11 にも対応しますが、次のような理由から、使用される頻度は少なくなっています。

- サイズが32,000文字以上のテキストを処理する BLOB 変数は、使用する必要がなくなりました ([ページ 29](#)、" [容量の拡張](#) " の節を参照)。
- 変数の管理と、変数の実行スペースが区分されている新しいコンポーネントアーキテクチャは互換性がありません ([ページ 45](#)、" [新しいコンポーネントアーキテクチャ](#) " の節を参照)。事前定義済みの変数を使用して、バージョン 11 のコンポーネントがホストデータベースメソッドのテキストに (または、その逆も) アクセスすることはできません。

従って、**GET MACRO PARAMETER** コマンドおよび **SET MACRO PARAMETER** コマンドを使って、テキスト選択をサポートする新しいモードを使用することを推奨します。これらのコマンドを使用すると、ホストデータベース / コンポーネントの実行スペースが区分問題が解決するため、マクロ管理専用のコンポーネントを作成できます。

マクロのこの新しいモードを有効化するためには、新しい「Version」属性をマクロ要素の「value 2」で宣言する必要があります。この場合、4D は、_textSel や _textReplace などの事前定義済み変数を処理せず、**GET MACRO PARAMETER** コマンドおよび **SET MACRO PARAMETER** コマンドが使用されます。この属性は、次のように宣言する必要があります。

```
<macro name="MyMacro" version="2">  
--- Text of the macro ---  
</macro>
```

この属性を渡さない場合は、前のモードが保持されます。

7

フォームおよびオブジェクト

4D v11 では、フォームおよびそのオブジェクトの作成と管理に関して数多くの変更が行われました。

- テーブルから独立したインタフェースを作成するプロジェクトフォームが新しく追加されました。
- リストボックスに数多くの新機能が追加されました。
- ピクチャフィールドおよび変数の管理が最適化されました。
- 新しいシールド、ページモードにおけるサブフォームの表示、ドラッグ & ドロップのオプション、メタルルック (Mac OS) など、フォームエディタに様々な新機能が追加されました。
- 日付時間の表示フォーマットおよび数字のプレースホルダに関して数々の新機能が追加されました。

プロジェクトフォーム

4D v11 では、どのテーブルにも所属しない "独立した" フォームを作成、使用できます。このフォームをプロジェクトフォームと呼びます。

これと明確に区別するため、標準の 4D フォーム、つまりテーブルに所属するフォームを テーブルフォームと呼びます。

注 プロジェクトフォームの導入により、4D ランゲージに変更が加えられました。詳細は、[ページ 262](#)、"[プロジェクトフォーム](#)" の節を参照してください。

プロジェクトフォーム 使用の理由

4D v11 では、新しいコンポーネントアーキテクチャの導入により、プロジェクトフォームの作成が必要になりました([ページ45](#)、"[新しいコンポーネントアーキテクチャ](#)"の節を参照)。このアーキテクチャでは、コンポーネントにフォームを格納することはできますが、テーブルは格納できません。従って、テーブルから独立したフォームを作成できる機能は不可欠です。

さらに、プロジェクトフォームは、インタフェース(一般的にダイアログコマンドを使って呼び出されます)専用で特にテーブルに関係ないフォームの要求を満たします。4D の前バージョンでは、ダイアログボックス用のフォームを作成する目的のみで [Interface] テーブルを作成することが慣例でした。

最後に、プロジェクトフォームを使用すると、OS 標準に対応するインタフェースをより簡単に作成できます。特に、プロジェクトフォームを DIALOG コマンドで呼び出して、レコードのリスト表示にサブフォームを使用することを 4D は推奨します。多少のプログラミングを加えれば、この組み合わせは MODIFY SELECTION コマンドおよび DISPLAY SELECTION コマンドを改良できます。

特性

プロジェクトフォームは、次のような点でテーブルフォームと異なります。

- プロジェクトフォームに許されるのは、詳細(ページ)タイプだけです。出力(リスト)フォームのメカニズムとプロジェクトフォームは互換性がありません。
- プロジェクトフォームはテーブルリストに表示されず、カレントの入力/出力フォームとして指定することはできません。また、プロジェクトフォームはラベルエディタや4Dの読み込み/書き出しエディタで使用できません。プロジェクトフォームは、ダイアログコマンドを使用した時、または継承されたフォームとしてのみ表示されます。
- プロジェクトフォームは、フィールドを含み、テーブルフォームと同じタイプのオブジェクトを格納できます。
フィールドが使用されると、プロジェクトフォームはテーブル番号とフィールド番号を保存します。フォームがデータベースから他のデータベースに、またはコンポーネント内でコピーされる時は、その参照もコピーされます。使用されるテーブルとフィールドは、ターゲットになるデータベースのテーブルとフィールドです。(テーブルが存在しない、フィールドタイプが間違っているなどの理由から) 適合しない場合、フォームは正確に機能しません。
プロジェクトフォームの目的は、主に **DIALOG** コマンドのコンテキストで使用するためなので、デフォルトでは、エディタや新しいフォームウィザードにレコード管理の標準操作(次のレコード、レコードを削除など)のボタンは備わっていません。レコード表示およびフォーム内のデータ修正を管理するには、ランゲージコマンドを使う必要があります。
一方、プロジェクトフォームがテーブルフォームの継承されたフォームとして使用される場合、自動レコード管理メカニズムの使用は可能です。
- プロジェクトフォームはテーブルフォーム同様、フォームメソッドを持つことができます。この新しいタイプのフォームメソッドを反映させるため、エクスプローラの"メソッド"ページが変更されました。([ページ 76](#)、"[メソッドページ](#)"の節を参照)。

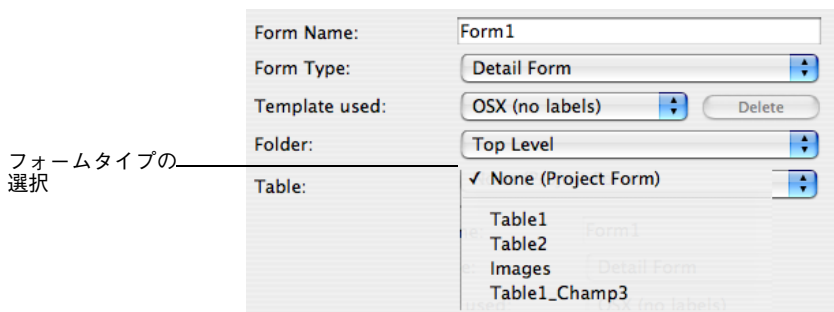
作成

プロジェクトフォームはテーブルフォームと全く同じ方法で作成できます。つまり、エクスプローラを使って "空白の" プロジェクトフォームを作成し、フォームエディタまたは新しいフォームウィザードを使って内容を埋めます。

フォームのタイプは作成時に設定されますが、プロジェクトフォームをテーブルフォームに(または、その逆に)変換するために随時タイプを変更できます。

新しいフォームウィザードの使用

新しいフォームウィザードを使用してプロジェクトフォームを作成するには、ウィザードの 1 ページ目にある "テーブル" で「なし (プロジェクトフォーム)」を選択します。:



この場合、全テーブルのリストだけが対象になります。

このオプションを選択すると、"フォームタイプ"のポップアップメニューには「詳細フォーム」と「プリントタイプの詳細フォーム」だけが表示されます。実際、リストタイプのプロジェクトフォームを作成することはできません(ページ 137、"特性"の節を参照)。

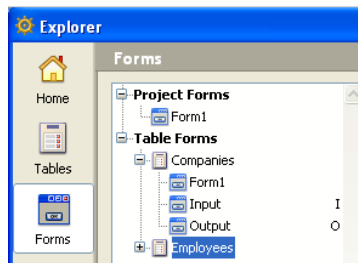
ウィザードの詳細設定ページ(「詳細設定...」ボタンをクリックすると表示されます)では、プロジェクトフォームの概念に適合しない特定のオプションの選択はなくなりました。


- 例えば、ウィザードの 1 ページ目では、"フィールド"のページはマスターテーブルを表示せず、全テーブルのリストだけを表示します。"リレートフィールド入力可"オプションの選択はありません。
- "オプション"のページには、"レコード番号/レコード総数"オプションの選択はありません。

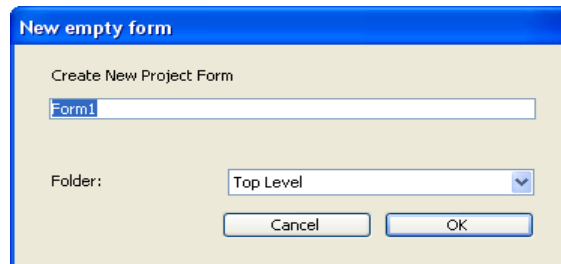
- " ボタン " のページでは、レコードに関する事前定義済みアクション (次のレコード、最後のレコードなど) の選択はありません。
- " サブフォーム " のページは、テーブルフォームのそれと全く同じです。" サブフォーム " のページは、テーブルにリレーションが何も含まれない場合や、どのような場合も使用できるようになった点に注意してください。

エクスプローラの使用

エクスプローラから空白のプロジェクトフォームを作成できます。4D v11 では、" フォーム " ページに2つの新しいデフォルト項目が追加されました。これらは、リストの上部にある「プロジェクトフォーム」および「テーブルフォーム」です。



空白のプロジェクトフォームを作成するには、リストの「プロジェクトフォーム」を選択して追加ボタンをクリックします。次のダイアログボックスが表示されます。



注 テーブルフォームが選択されている場合、追加ボタンは薄暗い表示になります。

プロジェクトフォームは、データベースにある既存のテーブルフォームと同じ名前をつけることができます。新しい **NO DEFAULT TABLE** コマンドは、同じ名前を持つテーブルフォームがある時に、プロジェクトフォームが使われていることを確認するために使用します ([ページ 262](#)、" [プロジェクトフォーム](#)" の節を参照)。ただし、2つのプロジェクトフォームが同じ名前を持つことはできません。

注 入力フォーム・プロパティおよび出力フォーム・プロパティは、テーブルの概念と関係があります。従って、これらのプロパティは、プロジェクトフォーム用には選択できません。プロジェクトフォームのリストに、文字の I および O は表示されません。

テーブルフォームからプロジェクトフォームへ(または、その逆の)変換

テーブルフォームをプロジェクトフォームに変換する(または、その逆の処理を実行する)ことは随時可能です。

テーブルフォームをプロジェクトフォームに変換する際、テーブルフォームにあるデータ管理関連の自動機能は、いったんフォームが変換されるとすべて機能しなくなるので、注意してください。同様に、"詳細(リスト)"または"印刷用詳細"タイプのフォームは、"ページ"タイプのプロジェクトフォームに変換されます。

フォームのタイプは、エクスプローラの「フォーム」ページにあるドラッグ&ドロップまたはコピー/ペーストを使って変更できます。これは、同じデータベース内で、あるいは2つのデータベース間で可能です。

▼ プロジェクトフォームをテーブルフォームに(または、その逆に)変換するには、

1 エクスプローラの「フォーム」ページで、変換するフォームをクリックして行先の項目でドロップします。

プロジェクトフォームをテーブルフォームに変換する場合は、それが移動されるテーブルの名前の上にドロップしなければなりません。

デフォルトでは、同じデータベース内でドラッグ&ドロップ操作を実行した時にフォームが移動します。フォームをコピーする場合は、Alt キー (Windows) または Option キー (Mac OS) を押しながらドラッグ&ドロップします。2つのデータベース間でドラッグ&ドロップする場合、フォームはコピーされるだけです。

エクスプローラのコンテキストメニューにある標準のコピー/ペーストコマンドも使用できます。

プロパティ

フォームプロパティ・ダイアログボックスに、フォームが所属するテーブルを示す "テーブル" フィールド (入力不可) が追加されました。プロジェクトフォームでは、このフィールドは "なし (プロジェクトフォーム)" の値を含みます。

テーブルフォーム

The screenshot shows the 'Form Properties' dialog box for a 'Table Form'. The 'Table' field is set to 'Companies'. An arrow points to the 'Table' field.

プロジェクトフォーム

The screenshot shows the 'Form Properties' dialog box for a 'Project Form'. The 'Table' field is set to 'None (Project Form)'. An arrow points to the 'Table' field.

この新しい機能は、特にプロジェクトフォームとテーブルフォームを区別するために使用します。

リストボックス

4D v11 では、リストボックスタイプのオブジェクトに関して次のような新機能が追加されました。

- フィールドまたは式とリストボックスのカラムを関連付ける機能
- リストボックスに SQL クエリの結果を表示する機能

注 これらの新機能により、4D v11 のランゲージにおいて "リストボックス" テーマコマンドレベルで変更が行われました。この件に関する詳細は、[ページ 311](#)、"[リストボックス](#)" の節を参照してください。

フィールドまたは式とリストボックスとの関連付け

4D v11 では、フィールドまたは式とリストボックスのカラムを関連付けられるようになりました。前バージョンでは、リストボックスと関連付けられるのは配列のみでした。

リストボックスはフィールドと関連付けられると、データベースのカレントセレクションまたは命名セレクションと相互に働きます。

カレントセレクションの場合、データベース側から行われた変更はすべて自動的にリストボックス(または、その逆)に反映されます。従って、両方のカレントセレクションは常に同じです。

リストボックスのカラムと式を関連付けることもできます。式は、1 つまたは複数のフィールドに基づく場合(例えば、`[Employees]FirstName+" [Employees]LastName`)、または単純な式(例えば、`String(Milliseconds)`)の場合があります。プロジェクトメソッド、変数、または配列項目も可能です。

注 同じリストボックス内でフィールド(または、式)と配列を組み合わせることはできません。

新しいリストボックスプロパティ

フィールドおよび式との関連付けをサポートするため、新しいリストボックス・プロパティが追加されました。リストボックスのカラムにも新しいプロパティが追加された点に留意してください(この点については次節で説明します)。

■ データソース

作業時にリストボックスからカレントセレクション、命名セレクション、または配列のいずれかを選択できるよう、新しいデータソースプロパティが"オブジェクト"テーマに追加されました。

Objects	
Type	List Box
Object Name	List Box1
Variable Name	List Box1
Data Source	Arrays
List Box	
Number of Columns	Current Selection
Number of Static Columns	Named Selection

■ 値がテーブルのカレントセレクションの各レコードごとに評価される式、フィールド、またはメソッドを使用する場合は、カレントセレクション・オプションを選択します。

- 値が命名セレクションの各レコードごとに評価される式、フィールド、またはメソッドを使用する場合は命名セレクション・オプションを選択します。
- 配列オプションを選択した場合、リストボックスの動作は 4D の前バージョンと同じです。
配列オプションは、リストボックスの SQL クエリの結果を表示する場合は必須です。([ページ 154](#)、"[SQL クエリ結果の表示](#)" の節を参照)。

■ マスターテーブル

この新しいプロパティは、カレントセレクションデータソースが選択されている時、"データソース" テーマに追加されます。

このプロパティは、カレントセレクションが使用されるテーブルを定義します。このテーブルとそのカレントセレクションは、リストボックスのカラムと関連付けられたフィールドの参照 (フィールド参照、またはフィールドを含む式) を形成します。他のテーブルのフィールドを含むカラムがあったとしても、表示される列数はマスターテーブルによって定義されます。

このプロパティと連動するメニューは、フォームとテーブルが関連している (テーブルフォーム) か、関係していないか (プロジェクトフォーム) に拘わらず、すべてのデータベーステーブルを表示します。デフォルトでは、プロパティはデータベースの最初のテーブルを表示します。

このプロパティの動作に関する詳細は、[ページ 152](#)、"[データ入力](#)" の節を参照してください。

■ 命名セレクション

この新しいプロパティは、命名セレクションのデータソースが選択されている時に、"データソース" テーマに追加されます。

有効な命名セレクションの名前を入力する必要があります。その命名セレクションは、プロセスでもインタープロセスでも可能です。

リストボックスの内容は、このセレクションに基づいたものになります。選択された命名セレクションは、リストボックスが表示された時に存在し、有効でなければなりません。そうでない場合、リストボックスは空白に表示されます。名前エリアに入力しない場合も、リストボックスは空白になります

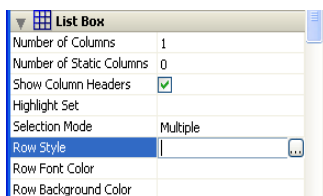
注 命名セクションはレコードを順序付けしたリストで、セクションの順序とカレントレコードをメモリに残すために使用します。詳細は、「4Dランゲージリファレンス」マニュアルを参照してください。

■ スタイルプロパティ (スタイル、フォントカラー、背景色)

これらのスタイルプロパティの名前は、カレントセクションまたは命名セクションのデータソースが選択されると変更されます。この場合、行スタイル配列プロパティ、行フォントカラー配列プロパティ、および行背景色配列プロパティは、それぞれ行スタイル・プロパティ、行フォントカラー・プロパティ、および行背景色プロパティに変わります。

実際、"セクション"タイプのデータソースでは、配列をスタイルに使用することはできません。スタイル、フォントカラー、および背景色のためには、式または変数 (配列を除く) を使う必要があります。式または変数は、表示される各行ごとに評価されます。

各プロパティにフォーミュラエディタを使用することができます。フォーミュラエディタを使用してプロパティの式を定義するには、エリアを選択する時に表示される [...] ボタンをクリックします。



以前のバージョンと同様、"Font Styles" テーマの定数を使用して行スタイルプロパティを設定できます。また RGB カラー値 ("SET RGB COLORS" テーマの定数を使用できます) を使用して行フォントカラー や 行バックグラウンドカラー プロパティを設定できます。

さらに、新しい **Choose** コマンドも使用できます。

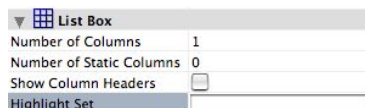
変数名を使用する例 : CompanyStyle を 行スタイル プロパティエリアに、CompanyColor を 行フォントカラー プロパティエリアに入力します。フォームメソッドでは、以下のコードを記述します。

```
CompanyStyle:=Choose([Companies]ID;Bold;Plain;Italic;Underline)
CompanyColor:=Choose([Companies]ID;Default background color;
Default light shadow color;Default foreground color;
Default dark shadow color)
```

■ ハイライトセット

この新しいプロパティは、カレントセレクションまたは命名セレクションのデータソースが選択されている時に、"リストボックス"テーマに追加されます。これは、リストボックスでハイライト表示されたレコードを管理するために使うセットを指定する時に使用します (4D の以前のバージョン同様、配列データソースが選択されている時は、リストボックスと同じ名前のブール型配列が使用されます)。

4D は、必要に応じて修正できるデフォルトのセットを作成します。(プロセスまたはインタープロセスセット)。セットは 4D が自動で管理します。ユーザがリストボックスの 1 行または複数行を選択すると、4D は即座にそのセットを更新します。プログラミングにより 1 行または複数行を選択する場合は、"セット"テーマのコマンドをハイライト表示されたレコードの管理に使用されるセットに適用します。



- 注
- ・ リストボックス行のハイライトステータスと、テーブルレコードのハイライトステータスは、完全に独立しています。
 - ・ "ハイライトセット" プロパティに名前が入力されていない場合、リストボックスの行を選択することはできません。

■ セレクションモード

この新しいプロパティは、どのデータソースが選択されている時でも、"リストボックス"テーマに追加されます。これは、以前のバージョンで使われた複数選択プロパティに取って代わります。選択なしモードが追加されました。次の 3 つのモードがあります。

- 選択なし: 行を選択できず、データは入力できません。行の選択およびデータ入力はプログラミングによってのみ管理できます。リスト上をクリックやダブルクリックしても、(入力可能オプションにチェックが入っている時でさえ)何も起こりませんが、On Clicked イベントおよび On Double Clicked イベントを生成できます。このモードでは、(ハイライトセットを使用する) 選択、および (EDIT ITEM コマンドを使用する) データ入力をデベロッパが完全にコントロールします。On Selection Change イベントおよび On Before Data Entry イベントは生成できません。一方、ユーザが EDIT ITEM コマンドでデータ入力した場合は、On After Edit イベントを生成できます。

- 単一選択: 以前のバージョン同様、1度に1行だけ選択できます。
- 複数選択: 以前のバージョン同様、複数行 (隣接しているかどうかは問わない) を選択できます。

新しいリストボックスカラムプロパティ

リストボックスにカレントセレクションまたは命名セレクションのデータソースが選択されている時は、フィールドまたは式と各カラムを関連付けることができます。

この場合、変数名プロパティおよび変数タイプ・プロパティは " オブジェクト " テーマにありません (これらは、配列データソースリストボックスには使用できます。).

注 リストボックスタイプのオブジェクトにも新しいプロパティが追加されました (この点については、前の節で説明しています)。

■ 式

この新しい式プロパティは、リストボックスのデータソースがカレントセレクションまたは命名セレクションの時に、" テータソース " テーマに追加されます。

Data Source	
Expression	
Data Type	Alpha
Choice List	<None>

このプロパティで入力できるのは次の通りです。

- 4D 式 (単純な表現式、計算式、または4D メソッド) を入力できます。式の結果は、アプリケーションモードに切り替えると自動的に表示されます。
- 単純な変数を入力できます (この場合、コンパイラに対して変数を明示的に宣言する必要があります)。BLOB および配列を除き、すべてのタイプの変数を使用できます。変数の値は、通常 On Display Detail イベントで計算されます。

■ 標準の [テーブル] フィールドシンタックス (例えば、[Employees]Last Name) を使用してフィールドを入力できます。使用できるフィールドのタイプは次の通りです。

- 文字
- テキスト
- 数値
- 日付
- 時間
- ピクチャ
- ブール型

マスターテーブルまたは他のテーブルのフィールドを使用できます。

いずれの場合も、4D フォーミュラエディタを使い、「プロパティリスト」の [...] ボタンをクリックすることにより、式を指定できます。



式を使用した場合は、入力可能なオプションにチェックが入っていたとしても、カラムには入力できません。

フィールドまたは変数を使用した場合は、入力可能オプションの設定に従い、カラムへの入力が可または不可になります。

デザインモードでは、データソースのタイプはカラムの最初の行に表示されます。Field=[Table1]MyFld は、その例です。

入力された式が正確でない場合、リストボックスのカラムはアプリケーションモードでエラーメッセージを表示します。

■ データタイプ

この新しいデータタイプ・プロパティは、カレントセレクションまたは命名セレクションのデータソースがリストボックスに選択されている時に、"データソース" テーマで使用できます。

このメニューは、カラムと関連付けられた式または変数のタイプを定義するために使用します。これを使用すると、適用する表示フォーマットを示し、"表示" テーマの「表示タイプ」メニューを更新できます。

式エリアにフィールドが入力された場合、データタイプ・プロパティは表示されません。フィールドタイプに対応する表示フォーマットが使用されます。

フィールドの表示

リストボックスのカラムと、マスターテーブルのフィールドや他のテーブルのフィールドを関連付けることができます。マスターテーブルに関する詳細は、[ページ 142](#)、"[新しいリストボックスプロパティ](#)"の節を参照してください。

ただし、いかなる場合も、リストボックスの内容はリストボックスのマスターテーブルのカレントコレクション(または命名セレクション)に基づきます。

- マスターテーブルのフィールドだけを使用する場合、リストボックスの行の内容はマスターテーブルセレクションの内容に基づいて形成されます。
- マスターテーブルに所属しないフィールドを使用する場合、これらの "外" の "テーブルは、N 対 1 リレーションを使ってマスターテーブルに関連づける必要があります。そうしない場合、"外" のフィールドは空白に表示されます。マスターテーブルセレクションの各レコードのために自動リレーションが有効化され、リストボックスは関連フィールドの対応するデータを表示します。

手動によるリレーションを使用する場合は、リストボックスのデータを表示するためにリレーションの有効化をプログラムする必要があります。

リストボックスの定義に不一致が生じた結果カラムが空白に表示された場合、アプリケーションモードでカラムにエラーメッセージが表示されます。

- ▼ 異なるケースの説明に、例を用います。
 - 2つのテーブル、[Companies] および [Employees] を持つデータベースがあるとします。

[Companies] テーブル用のカレントセレクションは次の通りです。

Company Name
 Big Encyclopedias
 Tiny Computers
 Boring Travel Company

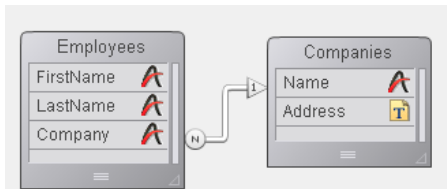
[Employees] テーブル用のカレントセレクションは次の通りです。

First Name	Last Name	Company Name
Carla	Packard	Boring Travel Company
Andrew	Black	Tiny Computers
Vincent	Laughter	Boring Travel Company
Oliver	Dawson	Boring Travel Company

First Name	Last Name	Company Name
Sylvia	Fairview	Tiny Computers
Robert	Lanzel	Big Encyclopedias
Arnold	Schmitt	Boring Travel Company
Elizabeth	Jones	Big Encyclopedias
Yolanda	Court	Tiny Computers
Pascal	Pratt	Tiny Computers

[Companies] テーブルの名前フィールドはリストボックスの最初のカラムと関連付けられています。[Employees] テーブルの名前および苗字のフィールドは、その次の2つのカラムと関連付けられています。リストボックスのデータソースはカレントセレクションです。

- ケース 1: 2つのテーブルは、自動リレーションの使用により関連付けられています。



1) リストボックスのマスターテーブルは [Employees] です。リストボックスは [Employees] テーブルのカレントセレクションを表示し、各従業員の会社名を表示する自動リレーションを有効化します。

Company Name	First Name	Last Name
Boring Travel Company	Carla	Packard
Tiny Computers	Andrew	Black
Boring Travel Company	Vincent	Laughter
Boring Travel Company	Oliver	Dawson
Tiny Computers	Sylvia	Fairview
Big Encyclopedias	Robert	Lanzel
Boring Travel Company	Arnold	Schmitt
Big Encyclopedias	Elizabeth	Jones
Tiny Computers	Yolanda	Court
Tiny Computers	Pascal	Pratt

2) リストボックス用に選択されたマスターテーブルは [Companies] です。リストボックスは [Companies] テーブルのカレントセレクションを表示します。このセレクションのレコードは3つだけなので、リストボックスに表示されるのは3行だけとなります。

[Employees] テーブルの名前フィールドおよび苗字フィールドのカラムは空白になります。

Company Name	First Name	Last Name
Big Encyclopedias		
Tiny Computers		
Boring Travel Company		

- ケース 2: 2つのテーブルは関連付けられていません (または、手動のリレーションにより関連付けられています)。



1) リストボックスのマスターテーブルは [Employees] です。リストボックスは [Employees] テーブルのカレントセクションを表示します。[Companies] テーブルの名前フィールドのカラムは空白になります。

Company Name	First Name	Last Name
	Carla	Packard
	Andrew	Black
	Vincent	Laughter
	Oliver	Dawson
	Sylvia	Fairview
	Robert	Lanzel
	Arnold	Schmitt
	Elizabeth	Jones
	Yolanda	Court
	Pascal	Pratt

2) リストボックス用に選択されたマスターテーブルは [Companies] です。リストボックスは [Companies] テーブルのカレントセクションを表示します。このセクションのレコードは3つだけなので、リストボックスに表示されるのは3行だけとなります。[Employees] テーブルの名前フィールドおよび苗字フィールドのカラムは空白になります。

Company Name	First Name	Last Name
Big Encyclopedias		
Tiny Computers		
Boring Travel Company		

注 当然、プログラミングにより異なったテーブルのセクションも管理できます。その場合、マスターテーブルに属さないフィールドと関連付けられたカラムを表示できます。

式の表示

集計カラムを使用するには、カラムプロパティの"データソース"テーマの式エリアに有効な 式を入力します。式エリアの隣にある [...] ボタンをクリックすると、フォーミュラエディタを使用できます。

式は値を返さなければなりません。同じテーマにあるデータタイプ・プロパティを使って値のタイプを示す必要があります。

式はマスターテーブルのセレクション (カレントまたは命名) にある各レコードごと評価されます。このセレクションが空白の場合、カラムに結果は表示されません。

標準並べ替え

レコードのセレクションに基づくリストボックスでは、標準の並べ替え機能 (カラムヘッダのクリックによる並べ替え) は、次のような場合のみ使用できます。

- データソースがカレントセレクションの場合。
- カラムがフィールド (文字、数値、日付、時間、またはブール型タイプ) と関連付けられている場合。

これ以外のケース (命名セレクションや式と関連付けられたカラムに基づくリストボックス) では、標準の並べ替え機能は使用できません。

リストボックスの標準並べ替えは、データベースのカレントセレクションの順序を修正します。ただし、カレントレコードおよびハイライト表示されたレコードは変更されません。

標準並べ替えは、集計カラムを含み、リストボックスのすべてのカラムが同期されます。

行の移動

セレクションを関連付けたリストボックスでは、マウスを使用して行を入れ替えることはできません。行の移動可属性は選択できなくなります。

データ入力

レコードのセレクションに基づくリストボックスでは、フィールドと関連付けられているカラムのデータを入力、修正できます。式と関連付けられているカラムには決して入力できません。

フィールドと関連付けられているカラムに入力するためには、そのフィールドについてストラクチャエディタの "変更不可" プロパティにチェックが入っていないことを確認します。また、"入力可" プロパティをカラムに適用する必要があります。

ユーザがリストボックスを通してフィールドの値を変更すると、その変更はデータベースのレベルで即座に反映されます。

注 フィールドがリストボックス内やフォーム内他所で表示されている場合、リストボックスを通してフィールドに加えられた変更は他の場所にはず

ぐに反映されません。

逆に、同じフィールドが同じリストボックスの異なる場所で参照されている場合、フィールドのインスタンスの1つに変更が加えられると、リストボックス内のすべてのインスタンスに即座に反映されます。

一般に、ストラクチャエディタでフィールド用に設定されたデータ入力コントロール(必須入力、変更不可など)は、リストボックスで考慮されます。

注 ストラクチャエディタで定義されたフィールドプロパティは、カラムのフィールドプロパティより優先されます。例えば、ストラクチャエディタの " 変更不可 " プロパティを持つフィールドは、" 入力可能 " プロパティがカラムに適用されたとしても、変更することはできません。

レコードがデータベースレベルでロックされている場合(マルチプロセスまたはクライアント/サーバモードに関連するロック)、リストボックスレベルでそれを変更することはできません: 「レコードがロックされています」という標準エラーメッセージが表示されます。

レコードが、例えば「DELETE RECORD」などのコマンドを使ってセレクションから削除された場合、対応する行がリストボックスから自動的に削除されます。複数のプロセスで使用されるリストボックスの場合、カレントプロセス以外のプロセスで削除されたレコードに対して空白の行が表示されます。

ある処理でレコードが追加または削除される場合、その処理が有効化された時だけ、リストボックスに変更が反映されます。

注 リストボックスがフィールドまたは式と関連付けられている場合、ランゲージコマンド使用による行の追加または削除はできません([ページ 311](#)、" [リストボックス](#) " の節を参照)。

ユーザセレクションの管理

レコードに基づくリストボックスにおいて、リストボックス中のユーザセレクションの管理(クリックやキーボード入力)は、GET LISTBOX CELL POSITION コマンドと、GOTO SELECTED RECORD などのカレントセレクションコマンドとの組み合わせで行います。

SQL クエリ結果の表示 4D v11 は、データのクエリを実行できるパワフルな SQL エンジンを搭載しています(ページ 199、"4D SQL エンジンの使用" の章を参照)。

リストボックスに直接 SQL クエリの結果を表示できます。SELECT タイプのクエリのみ使用できます。

これは、次の原則に従って機能します。

- クエリの結果を受け取るリストボックスを作成します。SQL クエリの結果を考慮して、リストボックスには同じカラム数を割り当てることを推奨します(下記を参照)。
- リストボックスのデータソースは、配列に設定しなければなりません。
- SELECT タイプの SQL クエリを実行し、その結果をリストボックスと関連付けられている変数に割り当てます。Begin SQL/End SQL のタグを使用できます(ページ 200、"4D と SQL エンジン間のデータ送信" の節を参照)。
- リストボックスの内容が SQL クエリに由来する場合、ユーザはカラムを並び替えたり、変更したりできません。
- リストボックスに SELECT リクエストを新しく実行することは、カラムのリセットにつながります(複数のSELECTリクエストを使用して同じリストボックスを漸次満たすことはできません)。

注 このメカニズムは外部 SQL データベースで使用することはできません。

- ▼ 「PEOPLE」というテーブルの全フィールドを検索し、それらの内容を変数名「vlistbox」を持つリストボックスに入れると想定します。(例えば) ボタンのオブジェクトメソッドには、次のように書きます。

Begin SQL

```
SELECT * FROM PEOPLE INTO <<vlistbox>>
```

End SQL

カラムの数

取得したSQLカラムの数がリストボックスに設定されたカラム数を超える場合、4D はリストボックスに自動的に必要カラムを追加し、SQL フィールドの内容に従ってそれらを型宣言します。追加されたカラムは読み取り専用なので、プログラミングによる管理はできません。プログラミングを使ってリストボックスの全カラムにアクセスしたい時は、カラム数が正確に対応しているかを確認する必要があります。

リストボックスで定義されたカラムの数がSQLクエリで取得したカラムの数を超える場合、リストボックスでは未使用カラムは隠されます。

ピクチャフィールドおよび変数の最適化

ピクチャフィールドおよび変数に関する管理および表示が変更されました。

最初に、2つのタイプのオブジェクトの操作は調和されました: これらのインタフェースと提供可能な機能はまったく同じになりました。

さらに、これらのオブジェクトに関連する新機能が追加されました。

最後に、4Dのピクチャ処理に関して、内部の最適化が複数回行われました。

警告: フォームに挿入されたピクチャの変数はプロパティリストで型宣言しなければなりません。 [ページ 162](#)、"[フォーム上のピクチャ変数を型宣言する](#)"の節を参照してください。

スクロールバー

ピクチャタイプのオブジェクトにも スクロールバーが使えるようになりました。

重要: ピクチャ用のスクロールバーを有効化するには、ピクチャの表示フォーマットは " トランケート (中央合わせしない) " に設定する必要があります。

この場合、水平スクロールバー・プロパティおよび垂直スクロールバー・プロパティが、ピクチャ・オブジェクト用のプロパティリストで選択可能になります。各プロパティは、次の3つのオプションを持つメニューを使って設定されます。

- **はい:** スクロールバーは、必要ない時 (つまり、ピクチャのサイズがフレームより小さい時) でも、常に表示されます。
- **いいえ:** スクロールバーが表示されることはありません。
- **自動:** スクロールバーは、必要時 (つまり、ピクチャのサイズがフレームより大きい時) には常に自動で表示されます。

スクロールバーの可視設定コマンドを使用するプログラミングによってピクチャタイプオブジェクトのスクロールバーを管理することもできます。

ネイティブフォーマットのサポート

4D v11 では、特に JPEG、PNG、BMP および TIFF フォーマットなど、よく使用される ピクチャフォーマットをネイティブにサポートできるようになりました。

つまり、ピクチャは 4D で何の障害もなく表示され、オリジナルフォーマットで保存されます。異なったフォーマットの特定の特性 (明暗、透明エリアなど) は保持され、変わることなく表示されます。特に、アニメーション GIF も使用できます。4D の以前のバージョンでは、PICT フォーマットに変換したピクチャのみ使用可能でした。

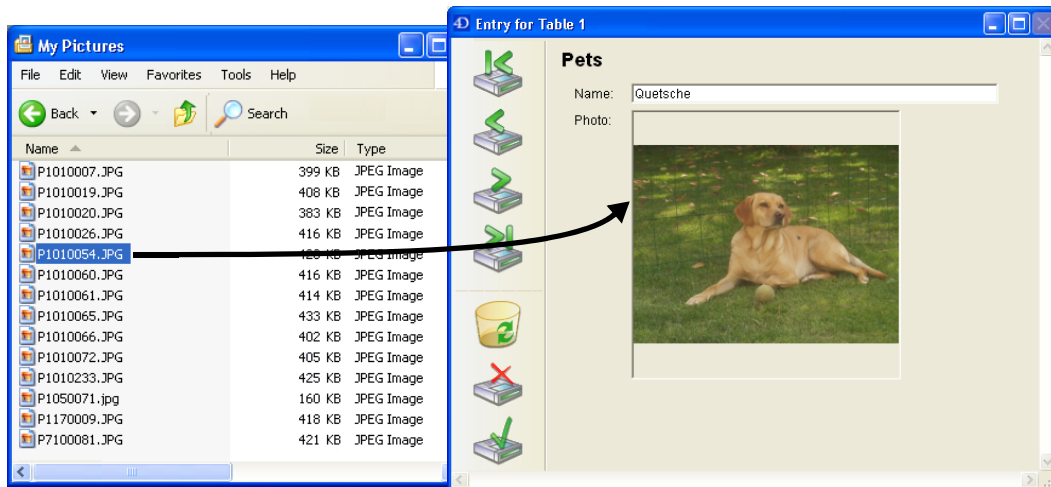
このネイティブサポートは、4D に保存されたピクチャすべてに有効です：つまり、ライブラリピクチャ、デザインモードでフォームにペーストされたピクチャ、アプリケーションモードでフィールドや変数にペーストされたピクチャなどです。新しい **PICTURE CODEC LIST** コマンドを使用すると、マシン上の新しいネイティブタイプを探すことができます。

-
- 注
- ・ 4D では、アニメーション GIF はスタティックピクチャまたはライブラリピクチャとして (実行モードで) 表示されている時のみ、フォーム上で " 再生 " されます。フィールドや変数に保存されたアニメーション GIF は再生できません。
 - ・ 4D がピクチャフォーマットを解釈できない場合、4D は QuickTime ルーチンを呼び出します。この原則が、特定のピクチャフォーマットを使用するアプリケーションの互換性を維持します。
-

自動ドラッグ & ドロップ

サードパーティーアプリケーション (オペレーティングシステム、ウェブブラウザなど) と 4D v11 の間で、ピクチャの自動ドラッグ & ドロップができるようになりました。これを行うには、取り出すピクチャファイルをクリックして、4D (例えば、ピクチャライブラリ、デザインモードでのフォーム、アプリケーションモードでのフィールドや変数) の上にドロップします。

ピクチャは直接 4D で表示されます。

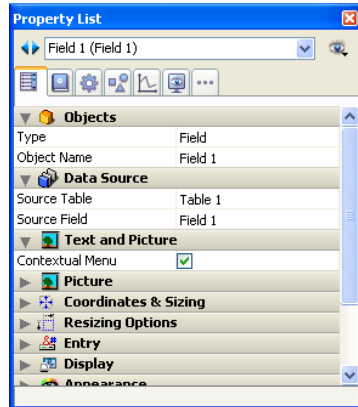


オリジナルのピクチャは、4D がネイティブサポートするフォーマットで保存されていなければなりません (前節を参照)。ピクチャはネイティブフォーマットで自動的に貼り付け先エリアにペーストされます。

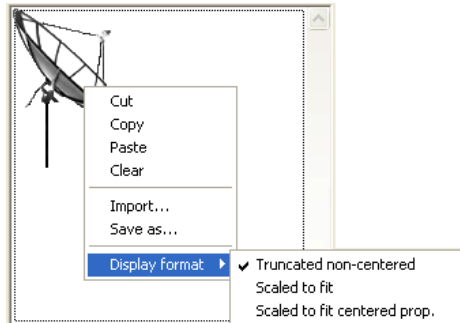
ピクチャが(アプリケーションモードで)実行中のフォームにドロップされた場合は、On After Edit フォームイベントが生成されます。

コンテキストメニュー

自動コンテキストメニューをピクチャタイプのフィールドや変数に関連付けられるようになりました。これを行うには、プロパティリストの "テキストとピクチャ" テーマにあるコンテキストメニュー・オプションにチェックします。



コンテキストメニューが有効化されると、ユーザはアプリケーションモードでピクチャを右クリックして編集および表示コマンドにアクセスできます。



標準の編集コマンド(カット、コピー、ペーストおよびクリア)に加えて、メニューにはファイルに保存されたピクチャを読み込む時に使用する「読み込み...」コマンド、およびディスクにピクチャを保存する時に使用する「別名で保存...」コマンドもあります。これら2つのコマンドは、ネイティブのピクチャ管理機能を活用します:これらは4Dがサポートするすべてのネイティブフォーマットにおいて、ピクチャをそれぞれ、開くまたは保存することができます。

メニューは、ピクチャの表示フォーマットを変更する時にも使用できます:トランケート(中央合わせしない)、スケーリング、およびスケーリング(中央合わせ・プロポーションナル)のオプションが用意されています。このメニューを使った表示フォーマットの変更は一時的なもので、レコードと一緒に保存されません。

注 ピクチャのフィールドまたは変数が入力不可の場合は、「コピー」、「別名で保存...」、およびフォーマットのコマンドだけ使用できます。

最適化

以上の節で説明した新機能に加えて、4D v11 のピクチャタイプオブジェクトに次のような最適化が行われました。

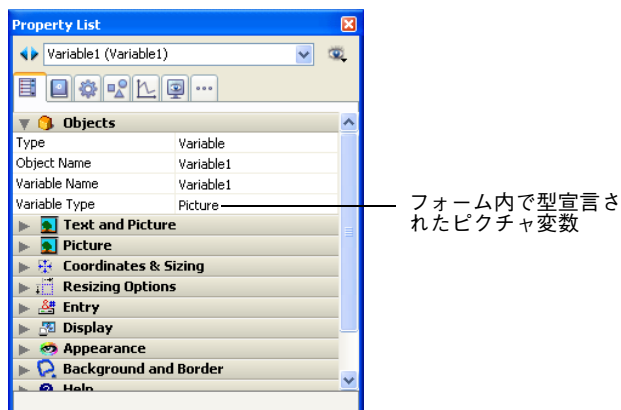
- **インタフェース**：フォームで選択されたピクチャは、その他のフォームオブジェクト同様、長方形(フォーカス表示)で囲まれるようになりました。4D の以前のバージョンでは、選択されたピクチャは反転表示されていました。
- **同じピクチャの複数表示**：同じフォーム内の異なる数箇所ですべて同じピクチャ(フィールドまたは変数)が使用される場合(例えば、配列で表示される黒丸など)、以前と異なり、メモリに置かれるピクチャはひとつだけになりました。つまり、使用メモリの量が減る一方、ピクチャの表示は促進されることとなります。

フォーム上のピクチャ変数を型宣言する

4D v11 では、新しいネイティブのメカニズムを使用してフォーム上のピクチャ変数の表示を管理します。これらの新しいメカニズムは、変数の設定時により高い正確さを求めます：今後、これらのメカニズムはフォームをロードする前 – つまり、On Load フォームイベントよりも前に、予め宣言されなければなりません。

これを行うには、次のいずれかが必要です。

- フォームをロードする前に、ステートメント `C_PICTURE(varName)` の実行を完了します (一般的には、`DIALOG` コマンドを呼び出すメソッドで実行します)。
- または、プロパティリストの変数タイプ・ポップアップメニューを使って、事前にフォームレベルで変数を型宣言します。:



そうしないと、4D v11 においてピクチャ変数は正常に表示されません (インタープリタモードのみ)。

- 注 4D の以前のバージョンは変数の初期設定に関して、あいまいが許されていたため、この新しい原則は、変換されたデータベースにおける表示の不具合につながるかもしれません。不具合が起きた場合は、以上で述べた2つの解決法のいずれかにより、ピクチャ変数が正確に宣言されているかどうか確認してください。

フォームの編集

4D v11 では、フォームの編集や動作に関する様々な新機能が追加されました。

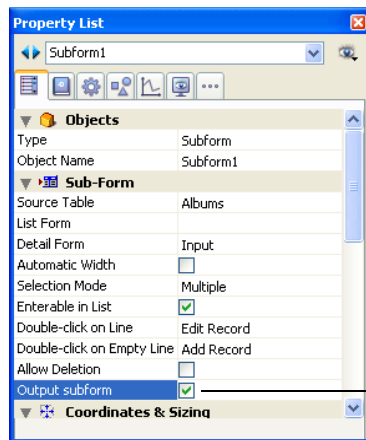
ページモードでのサブフォーム

4D の以前のバージョンでは、サブフォームはレコードのセレクションをリストとしてのみ表示できました。

4D v11 では、ページフォームで、DIALOG コマンドを使ってサブフォームを使用できます。

この場合、サブフォームはコンテキスト (変数、ピクチャなど) によって、カレントサブレコードのデータ、または関連値のすべてのタイプを表示することができます。このサブフォームをデータ入力に使用することも可能です。

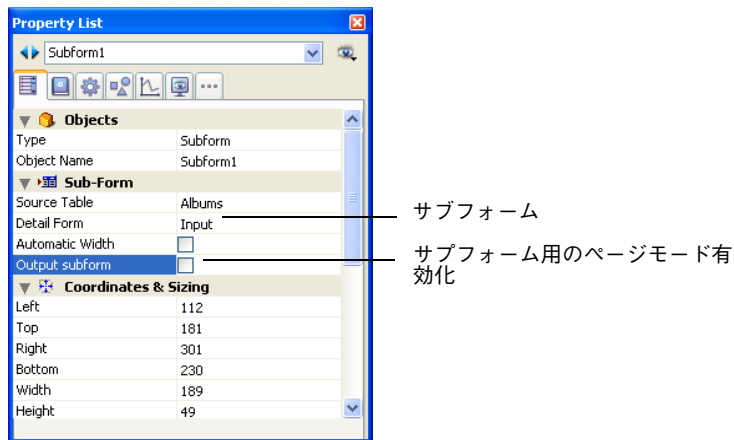
新しい出力サブフォームプロパティを使用すると、この操作を有効化できます。



サブフォーム用の使用モード
オプションのチェック = リストモード (標準)

このオプションにチェックが入っている場合、サブフォームは (4D の以前のバージョン同様) リストモードで使用されます。デフォルトでは、このオプションにチェックが入っています。

ページモードを有効化するには、このオプションのチェックを外します。この場合、サブフォームのリスト(セレクションモード、行のダブルクリックなど)としての設定に関連するプロパティは表示されなくなります。

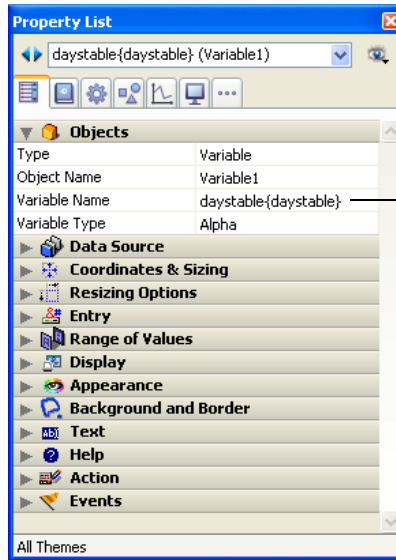


ページモードでのサブフォームは、"詳細フォーム"プロパティに示される入力フォームを使用します。リストモードでのサブフォームとは異なり、親フォームと同じテーブルの中のフォームを使用できます。また、プロジェクトフォーム(ページ135、"プロジェクトフォーム"の節を参照)も使用できます。

実行中、ページモードでのサブフォームには入力フォームと同じ標準の表示特性があります。出力フォームのメカニズム(特に、マーカの管理関連)は、有効化されません。

変数名に表現式を使用する

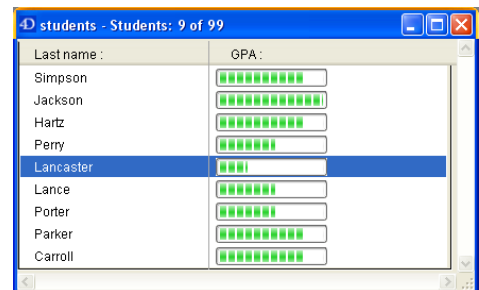
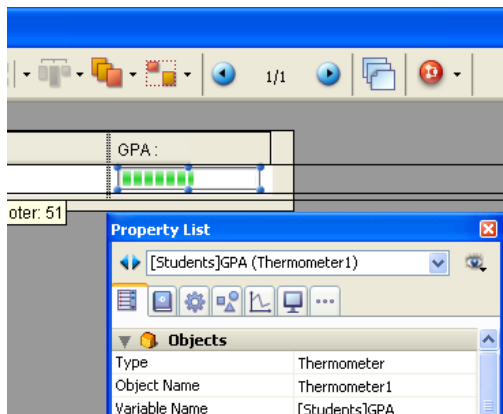
フォームオブジェクトに関連付けられた変数に、変数名だけでなく、値を返す表現式を使用できるようになりました。オブジェクトのプロパティリストの変数名エリアに、直接表現式を入力できます。



変数名として使用される表現式

有効な 4D 式なら何でも利用できます。単純な式、フォーミュラ、4D 関数、プロジェクトメソッド名、標準の [Table]Field シンタックスを使用したフィールドなど。表現式はフォームが実行される際に評価されます。

この原則により複数の可能性が生まれます。たとえば、サーモメータに数値フィールドを関連付けて、リストに値をグラフィカルに表示できます。



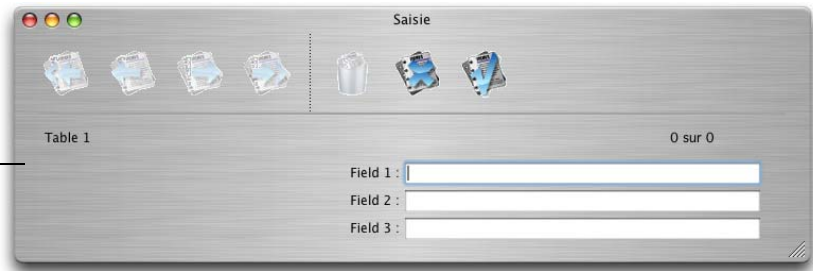
オブジェクトが入力可で、表現式がそれを許可している場合、このタイプのオブジェクトに値を入力することも可能です。

メタルルック

4D v11では、MacOSの場合にメタルルックのウィンドウを作成できます。

この外観はMac OS X インタフェース全体で見ることができます。:

Mac OSのメタルルックウィンドウ

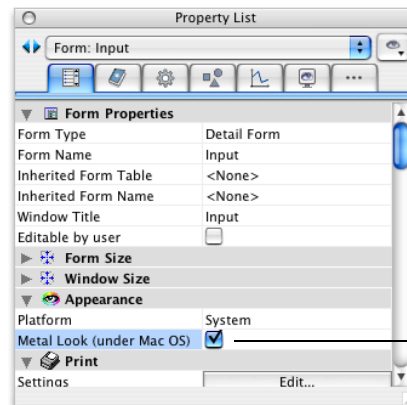


メタルルックのウィンドウを作成するには、次の2つの方法があります。

- 新しいメタルルック・プロパティ (詳細は下記参照) を設定したフォームを **Open form window** コマンドで開きます。#
- 新しい **Metal Look** 定数を指定して **Open window** コマンドを使用します。(詳細は、[ページ 317](#)、"**Open window**" の節を参照してください。)

"メタルルック"プロパティ (Mac OS 用)

"アピアランス" テーマの新しいフォームプロパティ「メタルルック (Mac OS 用)」は、Mac OS 上で **Open form window** コマンドを使用して、メタルルックのフォームを表示するために使用します。



メタルルック・プロパティ

注 Windows の場合、このプロパティは効果はありません。

Open form window コマンド以外で作成したウィンドウ (例えば、デザインモードなど) にフォームが表示された場合、プロパティは考慮されません。

Mac OSでは、メタルルック・オプションにチェックが入っていて、フォームの境界が表示されている時、フォームエディタでメタルルックをプレビューできます。

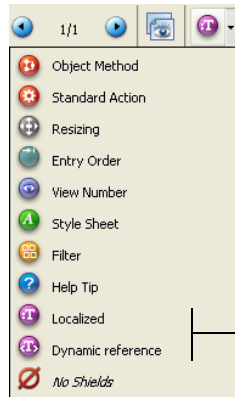
新しいドラッグ&ドロップオプション

4D v11 では、ドラッグ & ドロップ機能に変更が加えられました。特に、ネイティブコントロール(フィールド、変数、リストボックス、およびコンボボックス)のテキストオブジェクトには、ドラッグ & ドロップを管理する機能が追加されました: 自動ドラッグ、自動ドロップ、およびオブジェクトのドロップを許可する("動作"テーマ)などです。



新しく導入されたこれらのオプションおよびメカニズムについては、"ランゲージ"の章の[ページ264](#)、"[ドラッグ&ドロップ](#)"の節で説明しています。

新しいシールド

フォームエディタに、新しい2つのタイプの シールドが追加されました。:



新しいシールド

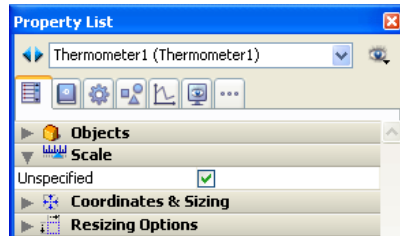
- ローカライズ済み : このシールドは、ラベルに ":" (コロン) で始まる参照を使用したオブジェクトに表示されます。参照はリソース (STR#) または XLIFF タイプです。
XLIFF 参照に関する詳細は、[ページ 98](#)、"[XLIFF 標準のサポート](#)" の節を参照してください。
- 動的参照 : このシールドは、フィールド、テーブルまたは変数 ("`<label>`" タイプのシンタックス) への動的参照を含むオブジェクトに表示されます。

バーバーシヨッフ サーモメータ

4D v11 フォームに "バーバーシヨッフ" タイプの引数を使用できます。このタイプのサーモメータは、連続アニメーションを表示します。"バーバーシヨッフ" サーモメータは、通常はユーザにプログラムが長時間操作を実行中であることを示すために使用します。



- ▼ "バーバーシヨッフ" をフォームに追加するには、
 - 1 "サーモメータ" を作成します。
 - 2 プロパティリストにあるスケール・プロパティの "未定義" にチェックを入れます。
 このプロパティを有効化すると、サーモメータ用の他のスケールオプションが隠されます。



フォームが実行されても、サーモメータは動画になりません。動画は、サーモメータに関連付けられた変数に値を渡すことで処理されます。

- 1 = 動画開始
 - 0 = 動画停止
-
- 注 "バーバーシヨッフ" サーモメータは、アピアランスが「システム」または「プリント」に設定されている時に作動します。
-

SET FORMAT コマンドは、"バーバーシヨッフ" サーモメータを動的に標準プログレスバーに変更する時に使用できます。

日付と時間の表示フォーマット

4D v11 では、国際アプリケーションの展開を促進するために、リージョン引数のサポートが強化されました。

この構想において、日付と時間の表示フォーマットに変更が加えられました。これらのフォーマットはシステム引数に基づく部分が大きくなり、新しいフォーマットが追加されました。

注 このリージョン引数のサポートを完全にするため、少数点および千単位区切りを設定するための新しい参照を使用できるようになりました([ページ 175](#)、"[数値のシステム環境設定](#)" の節を参照)。また、同様の目的から、**Num** 機能に変更が加えられました。最後に、新しい **GET SYSTEM FORMAT** コマンドにより、システムで定義された複数のリージョン引数を使用できるようになりました。

日付フォーマット

より明確にするため、日付表示フォーマットは名前が変更されました。

- システムで定義されたフォーマットに基づくフォーマットの名前は、"System" で始まります。これらのフォーマットにより表示される結果は、システムの地域別設定により変わります。
- 4D アプリケーション内部の引数に基づくフォーマットの名前は、"Internal" で始まります。

次のテーブルは、4D v11および以前のバージョンそれぞれの日付表示フォーマットの対応を表しています。

4D v11 フォーマット名	例 (英語表記)	以前のバージョンの名前
System date short	3/25/99	Short
System date abbreviated	Wed, Mar 25, 1999	Abbreviated
System date long	Wednesday, March 25, 1999	Long
Internal date short special	03/25/99 but 04/25/2032	MM DD YYYY
Internal date long	March 25, 1999	Month Day Year
Internal date abbreviated	Mar 25, 1999	Abbr Month Day
Internal date short	03/25/1999	MM DD YYYY Forced
ISO Date	1999-03-25T00:00:00	ISO Date Time

注 4D v11 のフォーマット定数はこれらの変更を反映します (ページ 383、"表示フォーマット定数" の節を参照)。

- Null Dates: デフォルトでは、通常 null date は "00/00/00" のように表示されます。日付がゼロまたは SQL NULL 属性を含む場合、プロパティリストの新しいヌルの時ブランクにするオプション ("表示" テーマ) を使って空エリアを表示できます。

時間フォーマット

4D v11 では (以前のバージョンからの既存のフォーマットに加えて)、新しい 時間表示フォーマットが用意されています。これらのフォーマットは、次のテーブルで説明しています。:

新しい 4D v11 のフォーマット	説明	04:30:25 の例
MM SS	00:00:00 からの経過時間の表記	270:25
Min Sec	00:00:00 からの経過時間の表記	270 分 25 秒
ISO Time	ISO コンポーネントフォーマットの " 時間 " 部分	0000-00-00T04:30:25
System time short	システムで定義された標準時間フォーマット	04:30:25
System time long abbreviated	Mac OS のみ : システムで定義された略表記時間フォーマット Windows では、このフォーマットはシステム時間短表記フォーマットと同じです。	4 · 30 · 25 AM
System time long	Mac OS のみ : システムで定義された長表記時間フォーマット Windows では、このフォーマットはシステム時間短表記フォーマットと同じです。	4:30:25 AM HNEC

注 4D v11 のフォーマット定数はこれらの変更を反映します (ページ 383、" 表示フォーマット定数 " の節を参照)。

- Null Times: 例えば、null time は "00:00:00" のように表示されます (表示はオブジェクトに適用したフォーマットにより変わります)。時間がゼロまたは SQL NULL 属性を含む場合、プロパティリストの新しいヌルの時ブランドにするオプション (" 表示 " テーマ) を使って空エリアを表示できます。

数値のシステム環境設定

4D v11 より、数値表示フォーマットは自動で地域のシステム環境パラメタに基づきます。4D は自動で数値表示フォーマットの M、m と E、e 文字を、OS に設定された千の位区切り文字と小数点文字に置き換えます。つまりピリオドやカンマはプレースホルダとして使用されます。

以前のバージョンでは、数値表示フォーマットはシステムの地域パラメタを考慮しませんでした。たとえば "###,##0.00" フォーマットは日本のシステムでは有効です。しかしこの表示フォーマットをフランスのシステムに適用すると、結果は正しくありませんでした。

変換されたデータベースでは、互換性のために、このメカニズムは無効になっています。これを有効にするには、環境設定のアプリケーション / 互換性ページにある、「数値フォーマットにシステム設定を使用する」をチェックしなければなりません。

次の表では、互換性オプションの設定により、新しいメカニズムの結果がどのように異なるかを示します。

表示する値	15353.33	
表示フォーマット	###,##0.00	
"数値フォーマットにシステム設定を使用する" オプション	チェックなし (デフォルト)	チェックあり
日本システム上での表示 千の位区切り文字 = , 小数点 = .	15,353.33	15,353.33
フランスシステム上での表示 千の位区切り文字 = <space> 小数点 = ,	<<<<<<<<<< (error)	15 353,33
スイスシステム上での表示 千の位区切り文字 = 1 小数点 = ,	<<<<<<<<<< (error)	15 1 353,33

このオプションがチェックされている場合、この機能は 4D v11 で作成されたデータベースと同じになります。

注 Num コマンドは、特定の小数点を指定する時に使用する新しい引数に対応するようになりました ([ページ 353](#)、[コマンド Num](#) を参照)。さらに、新しい [GET SYSTEM FORMAT](#) コマンドの使用により、システムのリージョン引数の現在値を求めることができるようになりました。

8

メニューエディタ

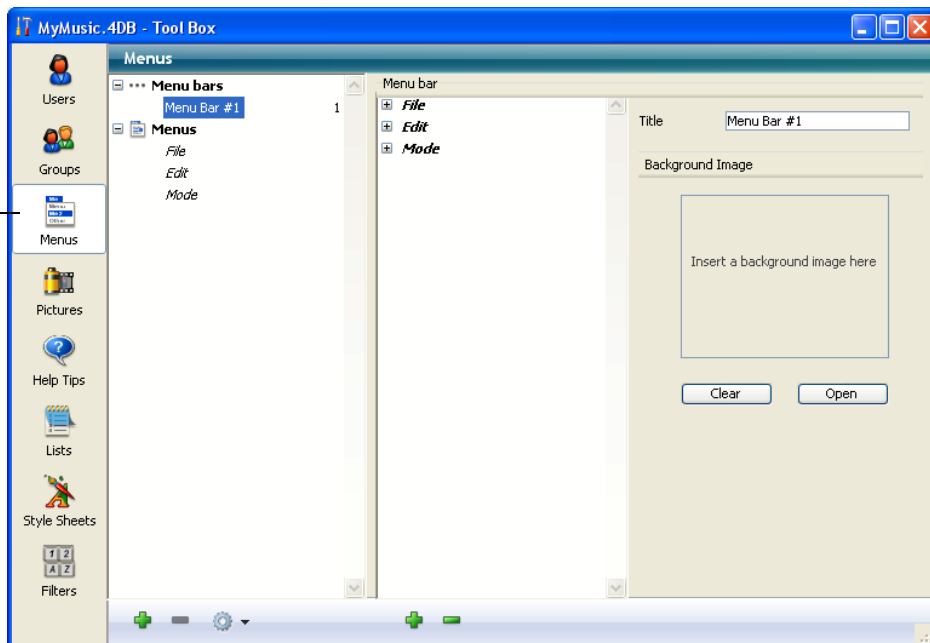
4D v11 のメニューエディタは、メニューおよびメニューバー管理の柔軟性を強化するために変更されました。更に、新機能も追加されました。

注：メニューのプログラム管理も変更されました。詳細に関しては、[ページ 277](#)、"[メニュー](#)"の節を参照してください。

インタフェース

メニューエディタは、「ツール」ボックスの「メニュー」ボタンを使用してアクセスできます。

新しいメニュー
エディタ



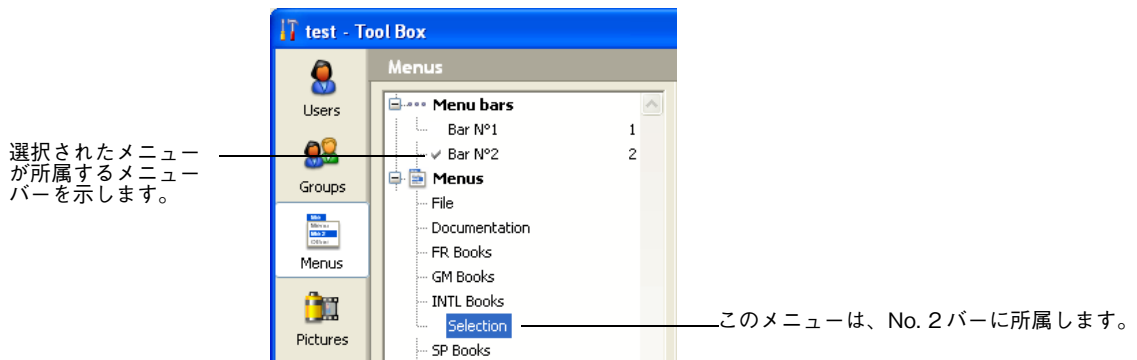
メニューおよびメニューバーは、ダイアログボックスの左側にある同じ階層リストに、2つの項目として表示されるようになりました。最初にメニューバーを選択する必要なく、データベースで定義されたメニューが即座に表示されるようになりました。メニューはアルファベット順でリスト表示されます。

各ソースメニューを簡単に特定できるため、追加メニュー（複数箇所で使用されるメニュー）の管理が容易になりました。

注： 4D v11 では、"連結"メニューを"追加"メニューと呼びます。

各メニューを、メニューバーまたはもう1つのメニューに追加することができます。第2のケースとして、メニューはサブメニューになります（[ページ 172](#)、"[階層サブメニュー](#)"の節を参照）。

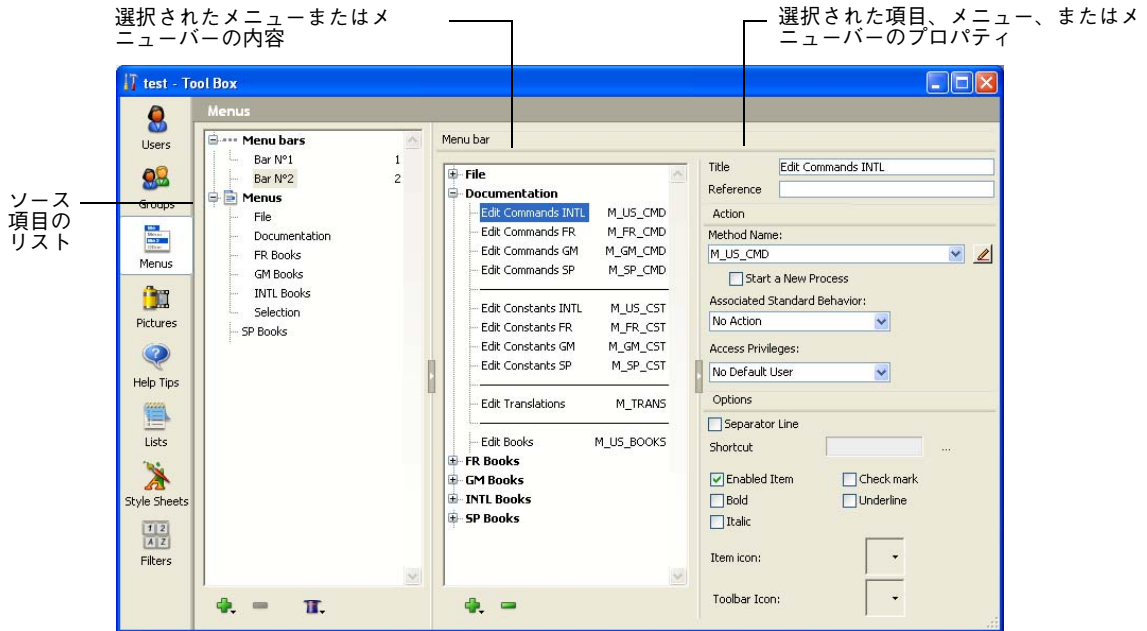
チェックマーク（√）は、選択されたメニューが所属する要素（メニューバーまたはメニュー）を示します。メニューが複数項目と関連付けられている場合は、複数のチェックマークが表示されます。



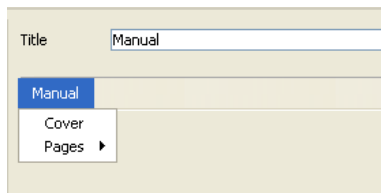
メニューが使用されない場合、チェックマークは表示されません（[ページ 172](#)、"[独立メニュー](#)"の節を参照）。

メニューバーまたはメニューの内容を表示させるには、エディタの左側リストにあるタイトルをクリックします。メニューバーまたはメニューに所属する項目のリストが中央エリアに表示されます。

メニューバーまたはメニューのプロパティも、ウィンドウの右側に表示されます。メニュー項目のプロパティを表示させるには、ウィンドウの中央エリアでそれを選択します。

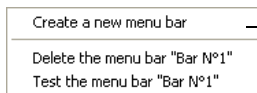


メニューが選択されると、右側のエリアでもメニューをプレビューできます。



コンテキストメニューおよびウィザードメニュー

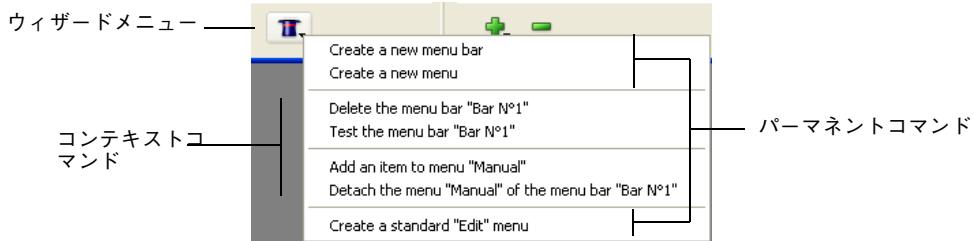
メニューエディタは、クリックされた項目のタイプ (メニューバー、メニュー、または項目) によって、実行可能な動作に直接アクセスできるコンテキストメニューを含みます。



メニューバーを右クリックすると表示されるコンテキストメニュー

コンテキストメニューは、項目を追加または削除する時、リストを展開または折りたたむ時などの他、特定の動作を実行するために使用できます。

また、メニューエディタは、左側リストの下にある帽子のような形のボタンをクリックするとアクセスできる"ウィザード"メニューも含まれます。このメニューは、パーマネントコマンドおよびコンテキストコマンドの両方を含みます。



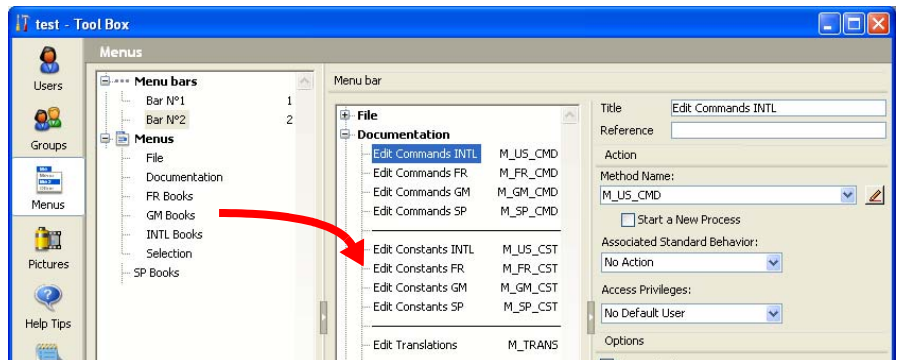
パーマネントコマンドを使用すると、新しいメニューバーや新しいメニューの他、標準編集メニューを作成できます ([ページ 177](#)、"編集メニュー (互換性)" の節を参照)。コンテキストコマンドを使用すると、選択された項目(メニューバーまたはメニュー)によって、適切な処理動作が提示されます。

メニューの管理

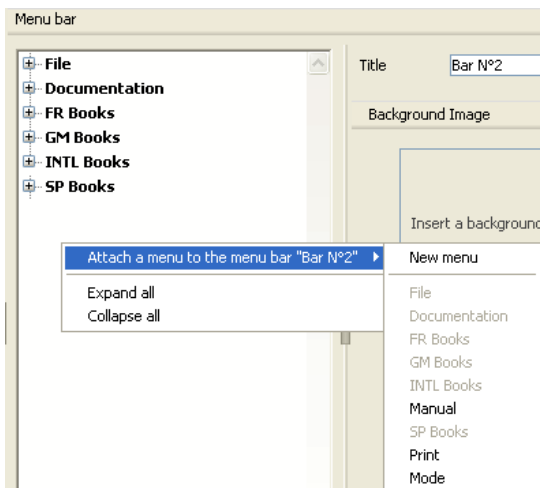
メニューをメニューバーに追加する

メニューをメニューバーに追加するには、"ウィザード" ボタンまたは中央エリアのコンテキストメニューを使ってドラッグ & ドロップをします。

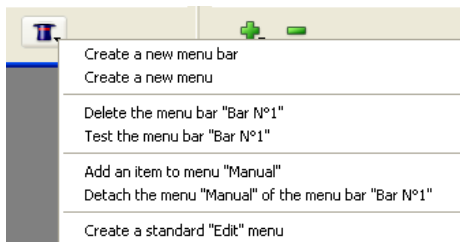
- ドラッグ & ドロップの使用: メニューバーの内容を中央リストに表示させるため、メニューバーをクリックします; 左側リストからメニューを選択し、それを中央リスト内の任意の場所までドラッグします。



- コンテキストメニューの使用: メニューバーの内容を中央リストに表示させるため、メニューバーをクリックします; このエリアで右クリックして「メニューをメニューバー "バー名" に追加する」コマンドを選択し、それからサブメニューとして使用するメニューを選択します。



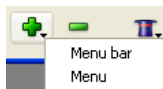
- “ウィザード”メニューの使用: 左側リストからメニューを選択してから、リストの下にある“ウィザード”ボタンをクリックします; 「メニューをメニューバー "バー名" に追加する」コマンドを選択し、それからサブメニューとして使用するメニューを選択します。:



独立メニュー

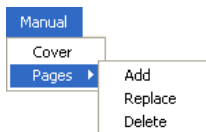
"独立"メニューを設定できるようになりました;つまり、メニューバーや他の特定のメニューに所属しないメニューです。これらのメニューはメニューエディタで設定できますが、ランゲージコマンドを使用して処理する必要があります。

独立メニューを作成するには、メニューバー / メニューリストにある作成ボタンと関連付けられているメニューからメニューを選択します。:



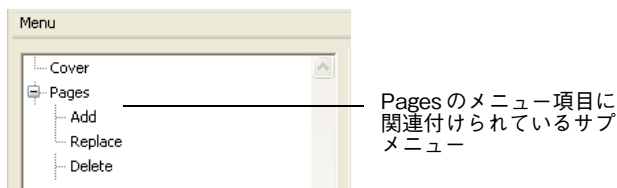
階層サブメニュー

4D v11 では、階層サブメニューを設定できるようになりました。サブメニューを使用すると、メニューバーで、同じメニュー内のサブジェクトに従って整理された機能をグループ化できます。

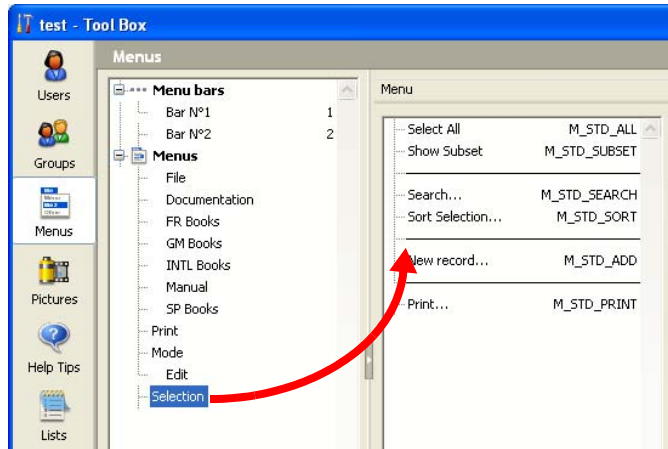


サブメニューとその項目は、もともとのメニューと同じ属性(動作、メソッド、ショートカット、アイコンなど)を持つことができます。

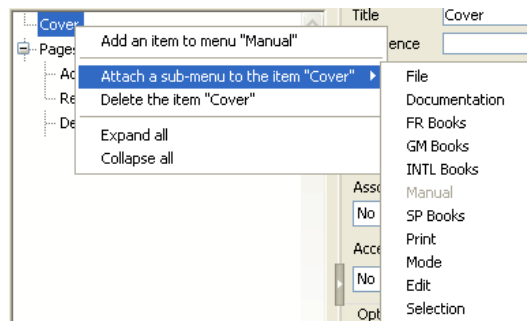
メニューエディタでは、サブメニューは階層リストの項目として表示されます。:



- ▼ サブメニューを作成するには、既存のメニューをもう1つのメニューの項目に関連付けます(追加します)。これを行う方法は、次の通り2つあります。
- ドラッグ & ドロップの使用: 左側リストからメニューを選択し、それを中央リスト内のサブメニューを追加したい項目の上までドラッグします。



- コンテキストメニューの使用: 中央リスト内のサブメニューを追加したい項目を右クリックします。コンテキストメニューから「サブメニューを項目 "項目名" に追加する」コマンドを選択し、それからサブメニューとして使用するメニューを選択します。



結果、追加されるメニューがサブメニューになります。項目のタイトルは保持されます(最初のサブメニュー名は無視されます)が、このタイトルは変更できます。

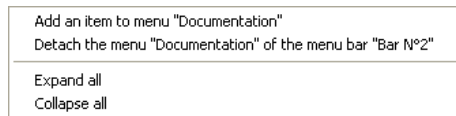
サブメニューの項目はもともとの特徴やプロパティを保持します。また、サブメニューの機能は標準メニューの機能と全く同じです。

サブメニューのサブメニューを作成することもできます。これを行うには、中央リストのサブメニューを展開し、サブ項目の1つにメニューを追加します。サブメニューは、ほぼ限らない階層まで追加できます。ただし、インタフェースの人間工学的考慮から、通常は2段階を超えるサブメニューはお奨めしません。

メニューまたはサブメニューの削除

メニューバーからメニューを削除したり、メニューからサブメニューを削除することは随時できます。メニューは削除されると、場合によってはメニューバーまたはサブメニューで使用不可になりますが、メニューのリストには存在します。

メニューを削除するには、中央リスト内の削除したいメニューまたはサブメニューを右クリックし、「メニュー "メニュー名" をメニューバー "バー名" から削除する」または「サブメニューを項目 "項目名" から削除する」を選択します。:



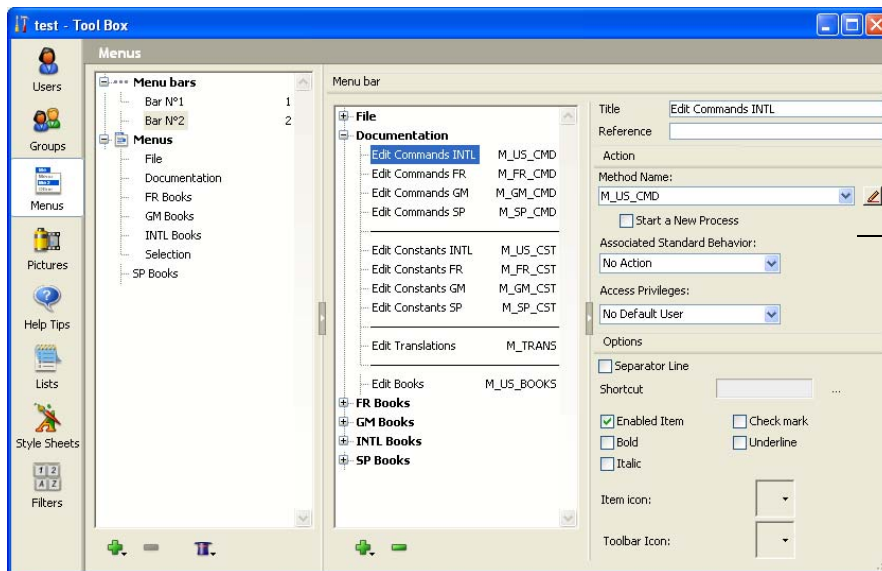
参照を使用したタイトル

4D の以前のバージョン同様、参照によってメニューコマンドを定義できます (エディタでイタリック体で表示されるタイトル)。4D v11 は今でもリソースのタイトルと互換性がありますが、現在は XLIFF 表記法が推奨されています ([ページ 98](#)、"[XLIFF 標準のサポート](#)" の節を参照)。

メニュー参照のみ保存され、タイトルはアプリケーションのカレントランゲージに従って表示されます。これは、多言語インタフェースを持つアプリケーションの配布を促進します。

新しいプロパティ

メニューおよびメニュー項目に新しいオプションおよびプロパティが追加されました。新しいプロパティは、エディタの中央エリアで要素が選択されると右側に表示されます。



プロパティ表示および入力エリア

メニュー項目参照

各メニュー項目に独自の参照を割り当てることができます。メニュー項目参照は、内容を自由に選択できる文字列です。

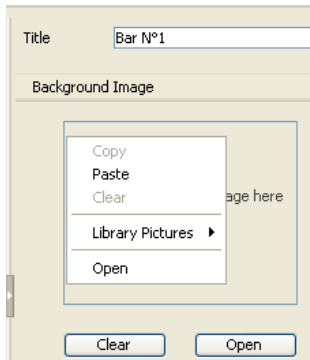
メニュー項目参照は、主にメニューのプログラム管理において、特に新しい **Dynamic pop up menu** コマンドを使う時に便利です。

ネイティブの背景画像

ネイティブフォーマットの背景画像を指定できるようになりました。これを行うには、左側リストからメニューバーを選択して「開く...」をクリックします。標準の「ファイルを開く」ダイアログボックスを使用して、JPEG、PNG、GIF など、背景画に使う様々な ネイティブタイプを選択できます。

注： 4D v11 ネイティブフォーマットのサポートについては、[ページ 155](#)、"[ピクチャフィールドおよび変数の最適化](#)" の節で詳細を説明しています。

コンテキストメニューを使用して、4D ピクチャライブラリから背景画を選択することもできます。



標準アクション

"標準アクション" ポップアップメニューで選択可能なアクションは、主に新しいデザインモードの新しい操作原理に関して、変更されました (ページ 70、"新しいデザインモードインタフェース" の節を参照)。

- デザインモードに戻る: 4D の新しいデザインモードのウィンドウおよびメニューバーを最前面に持ってきます。
この動作は、以前の 2 つの動作「ユーザ」および「デザイン」にとって代わります。データベースがインタプリタモードで稼動している場合、これはデザインモードのカレントウィンドウを表示します。また、データベースがコンパイルモードで稼動している場合、これはカレントテーブルのレコードウィンドウを表示します (コンパイルモードでは、レコードへのアクセスのみ可能)。
- アプリケーション: 4D アプリケーションモードのウィンドウおよびメニューバーを最前面に持ってきます。この動作は、以前の「カスタムメニュー」動作に相当します。
- MSC: Maintenance & Security Center ウィンドウを表示します。

変換データベースでの標準動作

変換データベースでは、ユーザ標準動作およびデザイン標準動作は既存のメニューバーに保持されますが、"標準アクション" ポップアップメニューでは選択できません。「カスタムメニュー」動作は、名前が「アプリケーション」に変更されました。

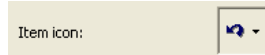
ユーザ標準動作およびデザイン標準動作の機能は以前のバージョンと同じですが、例外として、ユーザ動作は新しいデザインモードのレコードリストのウィンドウを表示できるようになりました。

チェックマーク

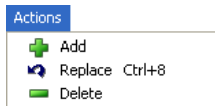
この新しいオプションを使用して、デフォルトでシステムチェックマーク (✓) とメニュー項目を関連付けることができます。それから、ランゲージコマンド (**SET MENU ITEM MARK** および **Get menu item mark**) を使用してチェックマークの表示を管理できます。

項目アイコン

新しい項目アイコンのオプションを使用して、メニュー アイコンと選択された項目を関連付けることができます。



このアイコンは、一度設定されると、項目の隣のメニューに直接表示されます。:



編集メニュー (互換性)

4D v11 のメニューでは、v 6.8 オプションがなくなりました。このオプションは、変換データベースにおいて、システムの管理による編集メニューの自動追加に基づく以前の操作を維持するために使われていたものです。

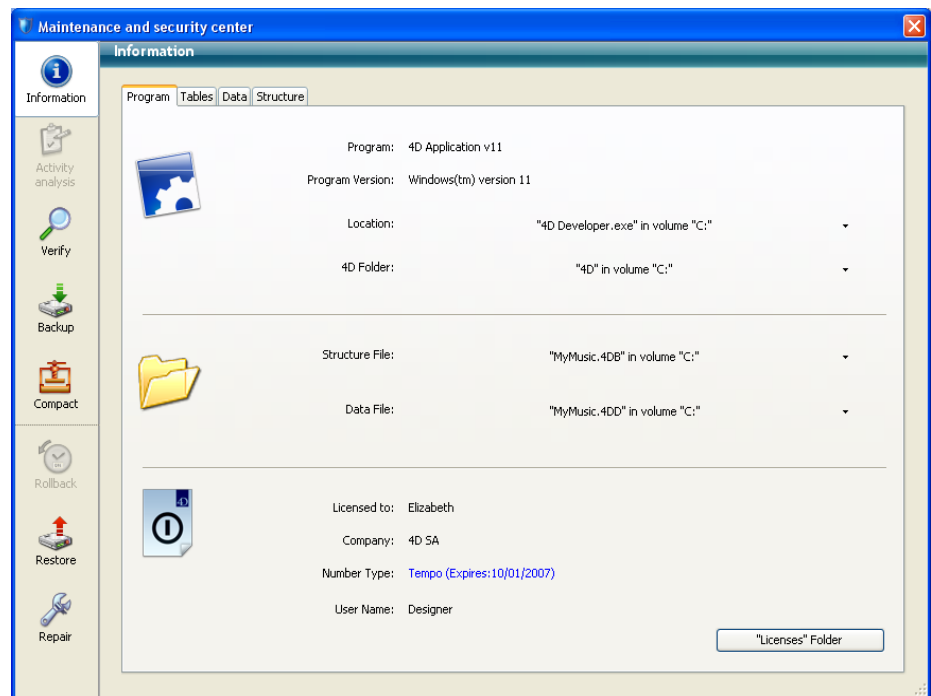
今後、「編集」メニューは他のメニューのように完全に管理されなければなりません。変換ウィザードは自動で編集メニューを以下のメニューバーに追加します。

- バージョン 6.8 から変換されたデータベース
- v6.8オプションが設定されたバージョン200xから変換されたデータベースメニューバーが変更されたことを通知するダイアログボックスが表示されます。メニューバーについて留意しなければなりません。これらのメニューバーのメニュー管理や実行ステートメントに使用される menu 引数の値を増加させなければならないからです。実際、以前の機能では、編集メニューはカウントされていませんでした。

9

Maintenance & Security Center

4D v11 は、データとストラクチャファイルの両方をチェック、保守、バックアップおよび圧縮ツールのフルセットを提供する新しいウィンドウを含みます。この新しいウィンドウには、Maintenance & Security Center (MSC) という名前がつけられています。



ウィンドウの左端にあるナビゲーションコントロールパネルにより、表示する操作または情報のテーマを選択できます。

MSC ウィンドウは、4D 旧バージョンの異なる環境間 ("About" ダイアログボックス、「データベースを開く」ダイアログボックス、バックアップ機能、および 4D Tools アプリケーション) でやりとりされた数々の機能を統合しています。

注：4D Tools MSC は、以前は 4D Tools アプリケーションを通して利用できた機能を含みます。この統合により、単独アプリケーションとしての 4D Tools は必要なくなり、バージョン 11 で提供されません。

MSC は、4D Developer、4D Server、4D Interpreter Desktop、4D Desktop、および 4D Unlimited Desktop などの 4D アプリケーションで使用できます。

MSC の表示

MSC ウィンドウを開く方法は幾つかあります。アクセスの方法により、"メンテナンス" モードまたは "標準" モード、いずれによってデータベースを開くかも決定されます。メンテナンスモードの場合、4D はデータベースを開かず、その参照だけが MSC に供給されます。標準モードの場合、4D はデータベースを開きます。

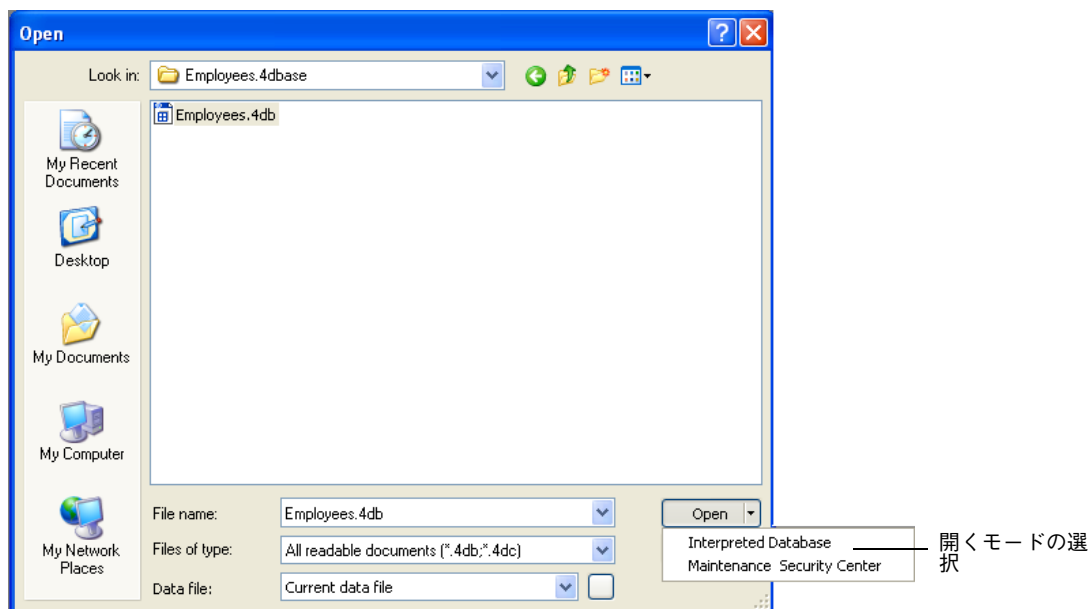
メンテナンスモードでの表示

メンテナンスモードでは、MSC ウィンドウだけが表示されます (4D アプリケーションはデータベースを開きません)。つまり、損傷が激しいため 4D が標準モードで開けないデータベースにもアクセスできるということです。さらに、特定の操作 (圧縮、修復など) はデータベースをメンテナンスモードで開くことを要求します ([ページ 180](#)、"[アクセス権](#)" の節を参照)。

メンテナンスモードの場合、次の 2 つの場所から MSC を開くことができます。

■ 標準の開くダイアログボックス

標準のデータベースを開くダイアログボックスは、「開く」ボタンと関連付けられているメニューの Maintenance & Security Center オプションを含みます。:



検査するデータベースを指定し、「開く」をクリックします。



■ 「ヘルプ」メニューまたはツールバーの MSC ボタン

データベースが開かれていない状態でこの機能呼び出すと、標準の「ファイルを開く」ダイアログボックスが表示され、検査するデータベースを指定できます。

注： 4D v11 では、開いているデータベースがない状況でアプリケーションを開始できます (ページ 66、"データベースの作成と開始" の節を参照)。

標準モードでの表示

標準モードではデータベースが開いています。このモードでは、特定の保守機能を使用できません。次の通り、可能性は幾つかあります。



■ 「ヘルプ」メニューまたはデザインモードでのツールバーの「MSC」ボタンから、このコマンドは MSC ウィンドウを開きます。この機能はアプリケーションモードにはありません。

- "MSCを開く"標準アクションにより、作成したメニューコマンドと関連付けることができます。
- 新しい **OPEN SECURITY CENTER** コマンドを使用できます。

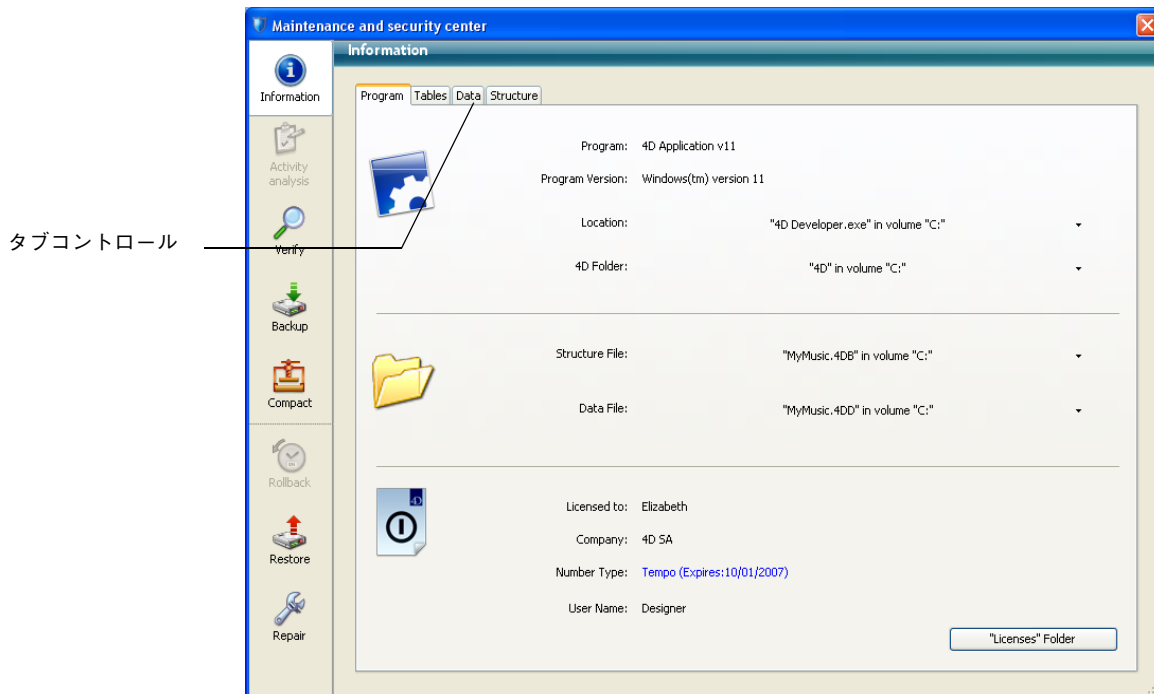
アクセス権

MSC の特定の機能は、MSC が開かれたモード、または (パスワードシステムが有効な場合) ユーザプロファイルにより、無効化されます。

- アプリケーションストラクチャに影響を及ぼす機能 (確認、修復および圧縮) は、4D Developer および 4D Server アプリケーションからのみアクセスできます。
4D Client アプリケーションおよび 4D Desktop アプリケーションでは、対応するボタンおよびタブは隠されています。
- データおよびストラクチャの内容に関する情報は、データベースが開いている時 (MSC が標準モードで開かれているとき) のみ提供されます。
- データの圧縮、ロールバック、復元および修復の機能は、開いていないデータファイル (MSC はメンテナンスモードで開かれていなければなりません) にのみ使用できます。データベースが標準モードで開かれている時にこれらの機能を試みた場合は、メンテナンスモードでアプリケーション再起動を促すダイアログボックスが表示されます。
- パスワードが有効化されている場合、データの圧縮、ロールバック、復元および修復の機能へは Administrator および Designer によってのみアクセスできます。

「情報」 ページ

"情報" ページは 4D 環境、システム環境、データベースおよびアプリケーションファイルについての情報を提供します。各ページは、ウィンドウ上部にあるタブコントロールを使って切り替えできます。

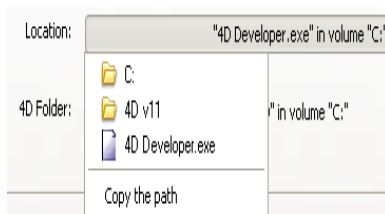


プログラムおよびテーブル

最初の 2 ページは、以前は 4D の「4th Dimension について」ダイアログボックスにあった、環境に関する情報を提供しています。

- プログラム: アプリケーションならびに 4D フォルダの名前、バージョンおよび場所を表示します; ウィンドウの中央部は、データベースストラクチャならびにデータファイルの名前および場所を表示します。ウィンドウの下部は、4D ライセンスフォルダの名前、ライセンスのタイプ、およびパスワードが有効化されている時のデータベースユーザの名前を表示します。
"ライセンス"フォルダボタンは、新しいシステムウィンドウにあるアクティブなライセンスフォルダの内容を表示します。

- **パス名の表示および選択**：「プログラム」ページでは、パス名はディスク上で検索された一連のフォルダで構成されるポップアップメニューに表示されます。



メニュー項目 (ディスクまたはフォルダ) を選択した場合、パス名は新しいシステムウィンドウに表示されます。

パスをコピーコマンドは、完全なパス名をクリップボードにテキストとしてコピーします。

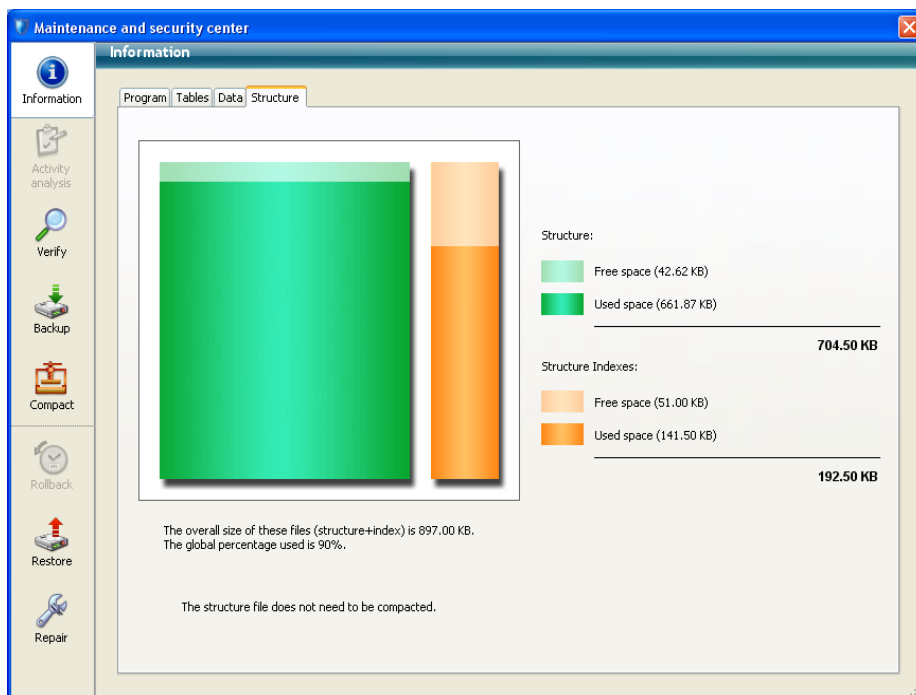
- **テーブル**: データベースの全テーブル、ならびにそれらの特徴(各テーブルの番号、各テーブルのレコード総数、フィールド総数およびインデックス総数)をリストアップします。

データおよびストラクチャ

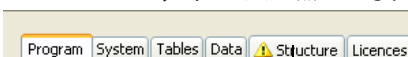
「データ」および「ストラクチャ」ページは、データベースストラクチャおよびデータファイルの使用率についての情報を提供しています。

- 注：
- ・ これらのページは、メンテナンスモードではアクセスできません。
 - ・ 「ストラクチャ」ページは、4D Developer および 4D Server アプリケーションでのみ提供されます。

この情報はグラフの形で提供されます。



断片化が激し過ぎるファイルは、ディスク、それからデータベースのパフォーマンスを低下させます。使用率が低すぎる場合、4D はこれを (情報ボタンおよび対応するファイルタイプのタブの上に表示される) 警告アイコンによって表示し、圧縮が必要であることを指摘します。



ストラクチャファイルの圧縮を要求

警告アイコンは「圧縮」ページの下部にも表示されます (ページ 190、「圧縮」ページ"の節を参照)。



「アクティビティ解析」ページ

このページは、ログファイルに記録されたすべての操作を表示する時に使用します。

The list below shows all the performed operations recorded in the log file since the last backup.

Operation#	Action	Table	Record/BLOB	Process	Size	Date	Hour
1	0 Creation of a ...			20		2007-02-01	15:09:51
2	1 Addition	AIRPORT_AI...	0	20	8	2007-02-01	15:09:51
3	2 Addition	AIRPORT_AI...	1	20	8	2007-02-01	15:09:51
4	3 Addition	AIRPORT_AI...	2	20	8	2007-02-01	15:09:52
5	4 Addition	AIRPORT_AI...	3	20	8	2007-02-01	15:09:55
6	5 Addition	COMPANY_AI...	0	20	12	2007-02-01	15:09:58
7	6 Addition	COMPANY_AI...	1	20	12	2007-02-01	15:09:58
8	7 Addition	COMPANY_AI...	2	20	12	2007-02-01	15:09:58
9	8 Addition	COMPANY_AI...	3	20	12	2007-02-01	15:10:01
10	9 Start of Trans...			20		2007-02-01	15:11:11
11	10 Validation of ...			20		2007-02-01	15:11:12
12	11 Start of Trans...			20		2007-02-01	15:11:12
13	12 Validation of ...			20		2007-02-01	15:11:12
14	13 Deleting a BL ...	AIRPORTS	22	20		2007-02-01	15:11:12
15	14 Deletion	AIRPORTS	22	20		2007-02-01	15:11:12
16	15 Start of Trans...			20		2007-02-01	15:11:12
17	16 Cancellation o...			20		2007-02-01	15:11:12
18	17 Start of Trans...			20		2007-02-01	15:11:12
19	18 Cancellation o...			20		2007-02-01	15:11:12
20	19 Start of Trans...			20		2007-02-01	15:11:12
21	20 Cancellation o...			20		2007-02-01	15:11:12
22	21 Addition	AIRCRAFTS	854	20	48	2007-02-01	15:12:05
23							

選択されたデータベースのカレントログファイル (4D v11 でのデフォルトの名前は datafilename.journal) の内容を表示するため、「解析」をクリックします。

「選択...」ボタンを使って、データベースの別のログファイルを選択して開くことができます。

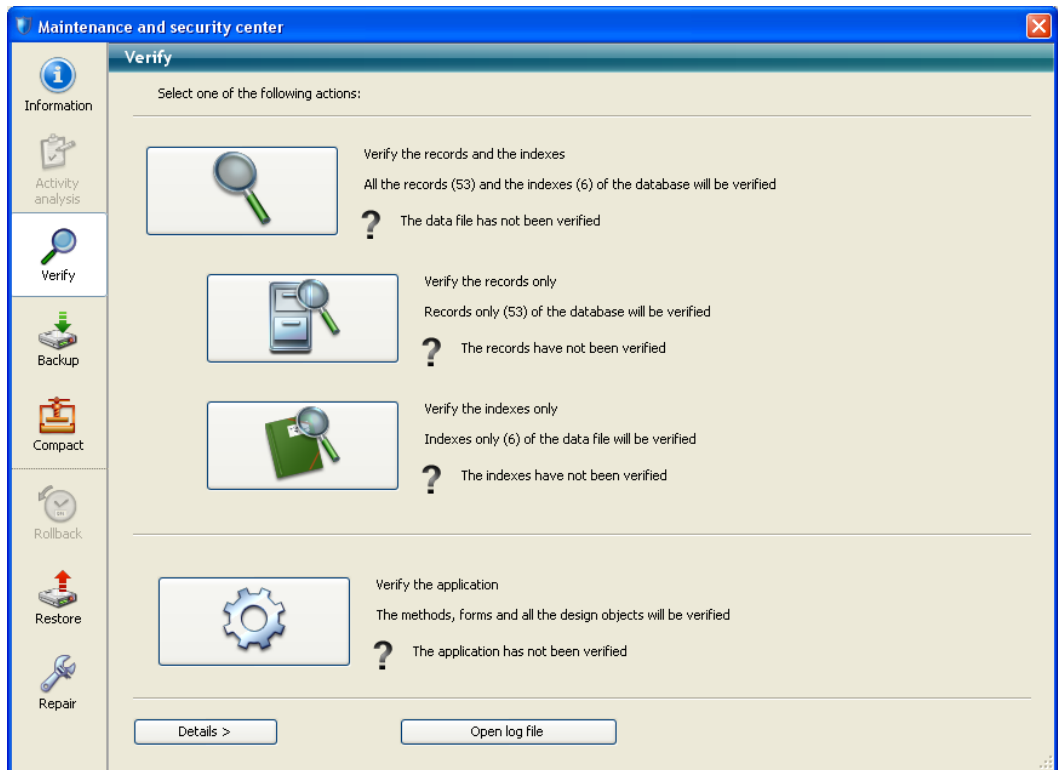
「書き出し...」ボタンを使って、ファイルの内容をテキストとして書き出すことができます。

このページは、4D の " ログをチェック " ダイアログボックスにある情報、即ちログファイルで記録されたデータベース操作 (レコードならびにトランザクションの追加、変更または削除) のリストと連動します。

さらに、データベースを開いた / 閉じた日付および時間も表示されます。

「検証」 ページ

このページでは、データおよび構造上の整合性を検証できます。検証は、レコードやインデックス、またデザインオブジェクト (メソッド、フォームなど) について実行できます。



注： この機能は検証のみをおこないます。エラーが見つかり修復が必要な場合は、「修復」 ページの使用を忠告する表示が出ます (下記を参照)。

動作

このページには、検証機能に直接アクセスするための、次の4つのボタンが置かれています。




- レコードとインデックスを検証: 全体のデータ検証処理を開始します。
- レコードのみを検証: レコードのみの検証処理を開始します (インデックスは検証されません)。
- インデックスのみを検証: インデックスのみの検証処理を開始します (レコードは検証されません)。

注：レコードおよびインデックスの検証は、詳細モードでテーブル毎に実行することもできます（[ページ 187](#)、"[詳細](#)"の節を参照）。

- アプリケーションを検証：デザインモードで定義されたすべてのオブジェクト（テーブル、メソッド、フォームなど）の検証処理を開始します。

"検証結果" エリア

"検証結果"のエリアは、各タイプの検証のカレントステータスを表示します。各ステータスのタイプは、特定の記号で説明、表記されます。:

Verify status:	
ステータス記号	 All the records are valid
	 Some indexes are damaged
	 The application has not been verified

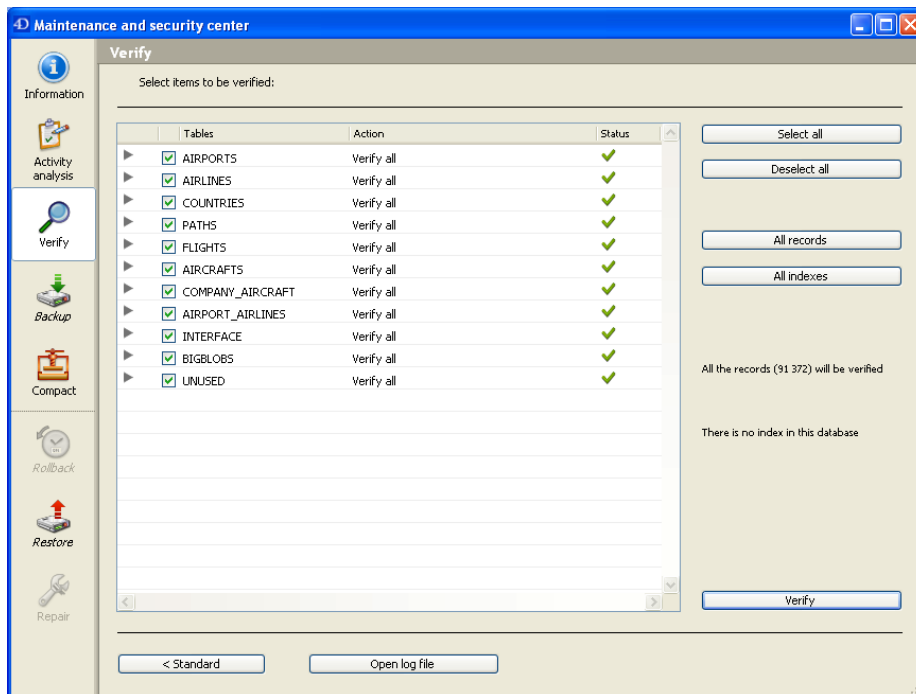
ログファイルを開く

要求された検証に関係なく、4D はデータベースフォルダにログファイルを生成します。このファイルは XML フォーマットで作成されます。しかしこのボタンをクリックすると、HTML 形式のファイルが瞬時に作成され、実行されたすべての検証を表示できます。

「ログファイルを開く」ボタンをクリックすると、4D はデータベースフォルダの内容を新しいウィンドウに表示し、ログファイルを選択します。

詳細

「詳細」ボタンは、検証するレコードおよびインデックスを表示して、選択する時に使用する詳細ページを表示します。



検証する項目を指定することにより、検証処理にかかる時間を節約できます。

リストには、データベースの全テーブルが表示されます。各テーブルに対して、検証対象をレコードやインデックスに限定できます。三角形のアイコンをクリックしてテーブルまたはインデックス付フィールドの内容を展開し、要求に応じてチェックボックスにチェックマークを入れたり外したりします。デフォルトでは、すべての項目にチェックマークが入っています。ショートカットボタン「すべて選択」、「すべての選択をはずす」、「すべてのレコード」および「すべてのインデックス」も使用できます。

テーブルの各行に対して、「アクション」カラムは実行する操作を表示します。テーブルが展開されると、「レコード」および「インデックスフィールド」の行は関連する項目の数を表示します。

「ステータス」カラムは、記号を使用して各項目の検証ステータスを表示します。

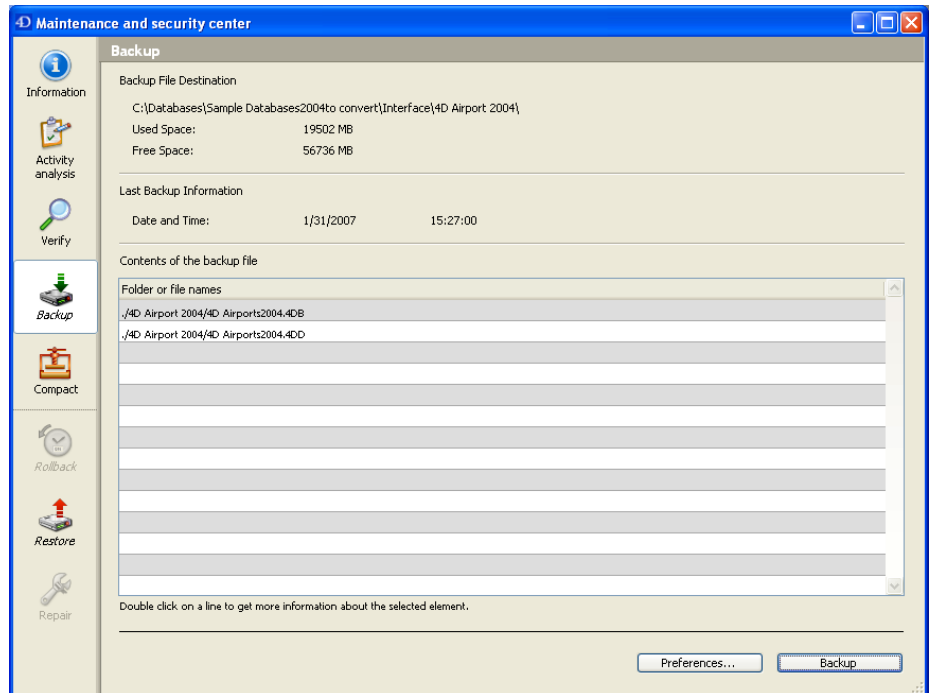
- ✓ — 問題なく実行された検証
- ✗ — 実行中に問題が発生した検証
- ▬ — 部分的に実行された検証
- ？ — 実行されていない検証

検証を開始するには「検証」をクリック、標準ページに戻る時は「< 標準」をクリックします。

注：標準ページは、詳細ページで行われた変更はすべて考慮しません。標準ページの照合ボタンをクリックすると、すべての項目が照合されます。逆に、詳細ページで行われた設定は、セッションから次のセッションに保持されます。

「バックアップ」ページ

「バックアップ」ページは、データベースのバックアップ設定を表示し、手動のバックアップ作業を開始する時に使用します。



このページは、バックアップ設定を変更するためには使用できません。これを行うには、「環境設定...」ボタンをクリックします。

■ バックアップファイルの保存先

このエリアは、データベースのバックアップファイルの場所に関する情報を表示します。また、ここはバックアップディスクの空いている / 使用されているスペースも表示します。

■ 前回のバックアップの情報

このエリアは、データベースで最近行われた (自動または手動の) バックアップの日付および時間を提供します。

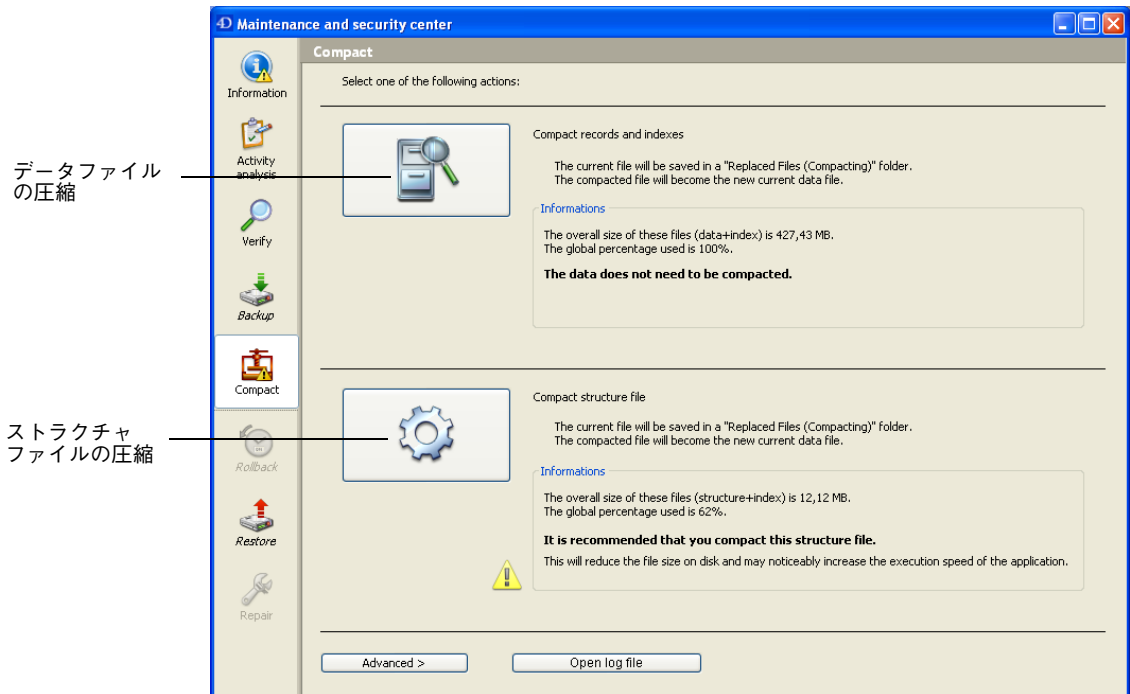
■ バックアップファイルの内容

このエリアは、バックアップファイルに含まれるファイルおよびフォルダをリストアップします。

「バックアップ」ボタンは、手動のバックアップを開始する時に使用します。

「圧縮」 ページ

このページは、データおよびストラクチャファイルの圧縮機能にアクセスする時に使用します。



ファイルを圧縮する理由

ファイルには使っていないスペースがあるかもしれません。実際、レコードやフォームなどを削除すると、それらがファイル上で占有していたスペースが空になります。4D はできる限り、こういったのスペースを再利用しますが、データのサイズは可変なため、連続的に削除や変更を行うと、必然的にプログラムにとって使用不可のスペースを作り出します。大量のデータが削除された直後についても同じことが言えます：空のスペースはそのままファイルに残ります。

データファイルのサイズと実際にデータに使われているスペースの比率を、データの 使用率と言います。使用率が低すぎると、スペースが無駄なだけでなく、データベースパフォーマンスの低下につながります。圧縮は、空きスペースを取り除き、データのストレージを再編成、最適化するために行います。

"情報" エリアにはフラグメンテーションに関するデータが要約され、必要な操作が表示されています。MSC の "情報" ページでは、データベースファイルのカレントのフラグメンテーションについて詳細を説明しています ([ページ 181](#)、"[「情報」 ページ](#)" の節を参照)。

- 注： 圧縮はメンテナンスモードでのみ可能です。標準モードでこの操作を実行しようとする、警告ダイアログボックスが表示され、データベースを終了してメンテナンスモードで再起動することを知らせます。ただし、データベースによって開かれていないデータファイルを圧縮することは可能です ([ページ 193](#)、"[上級モードでのデータファイルの圧縮](#)" の節を参照)。

データまたはストラクチャファイルの圧縮

データおよびストラクチャファイルの標準 圧縮プロシージャは全く同じです。

データまたはストラクチャファイルの圧縮を直接開始するには、MSC ウィンドウの対応するボタンをクリックします。:

データ ストラクチャ



- 注： 圧縮はオリジナルファイルのコピーを含むため、ファイルのあるディスクに十分なスペースがない場合、ボタンは使用不可になります。

この操作は、メインファイルの他、インデックスファイルもすべて圧縮します。

4D はオリジナルファイルをコピーし、それをオリジナルファイルの隣に作成された "Replaced Files (Compacting)" フォルダに置きます。操作が完了すると、圧縮ファイルは自動的にオリジナルファイルと置き換えられます。データベースは即座に操作可能になります。

- 注： ・ このデフォルト機能は、上級モードを使って変更できます (次の段落を参照)。
 ・ 圧縮操作を複数回実行すると、毎回新しいフォルダが作成されます。フォルダ名は、"Replaced Files (Compacting)_1" のようになります。

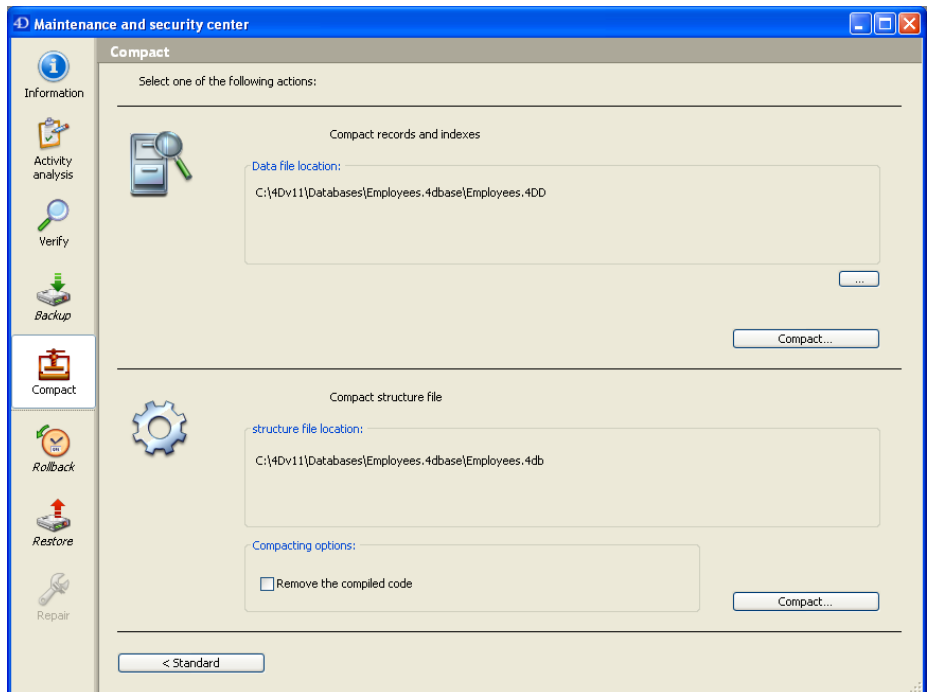
圧縮操作は毎回オリジナルファイルのコピーを伴うため、アプリケーションフォルダのサイズが大きくなります。アプリケーションのサイズが過剰に増加しないよう、これを考慮することが大切です（特に、4D アプリケーションがパッケージとして現れる Mac OS の場合）。パッケージのサイズを小さく保つためには、パッケージ内オリジナルファイルのコピーを手動で削除することも役立ちます。

圧縮が完了すると、4D はデータベースフォルダにログファイルを生成します。databasename_Compact_Log.xml と名付けられるこのファイルは、XML フォーマットで作成されます。これを使用して、実行されたすべての操作を表示できます。

「ログファイルを開く」をクリックすると、4D は新しいウィンドウに、データベースフォルダの内容およびログファイルを表示します。

上級モード

「圧縮」ページには、データおよびストラクチャファイルの圧縮に関するオプションページにアクセスする時に使用できる上級>ボタンがあります。



ページ上部には、データファイルのカレントパス名、および圧縮するファイルを変更する時に使用するボタンが表示されます。ページ下部には、ストラクチャファイルのカレントパス名、および圧縮オプションが表示されます。「圧縮...」 ボタンの1つをクリックすると、対応する圧縮操作が即座に開始されます。

上級モードでのデータ ファイルの圧縮

上級モードは、カレントデータファイル以外の データファイルを圧縮する時や、標準モードで導入された自動操作 (オリジナルファイルのコピーおよび置換) を無効にする時に使用します。

カレントデータファイル以外のデータファイルを選択するには、[...] をクリックします。標準の「ドキュメントを開く」ダイアログボックスが表示され、圧縮するデータファイルを指定できます。開かれたストラクチャファイルと互換性のあるデータファイルを選択する必要があります。このダイアログボックスで開くをクリックすると、圧縮するファイルのパス名がウィンドウに表示されます。

圧縮を開始するには、このエリアの「圧縮」 ボタンをクリックします。標準の「ドキュメントを保存」ダイアログボックスが表示され、圧縮ファイルの場所を指定できます。新しいファイルの名前と場所を指定し、「OK」 をクリックします。

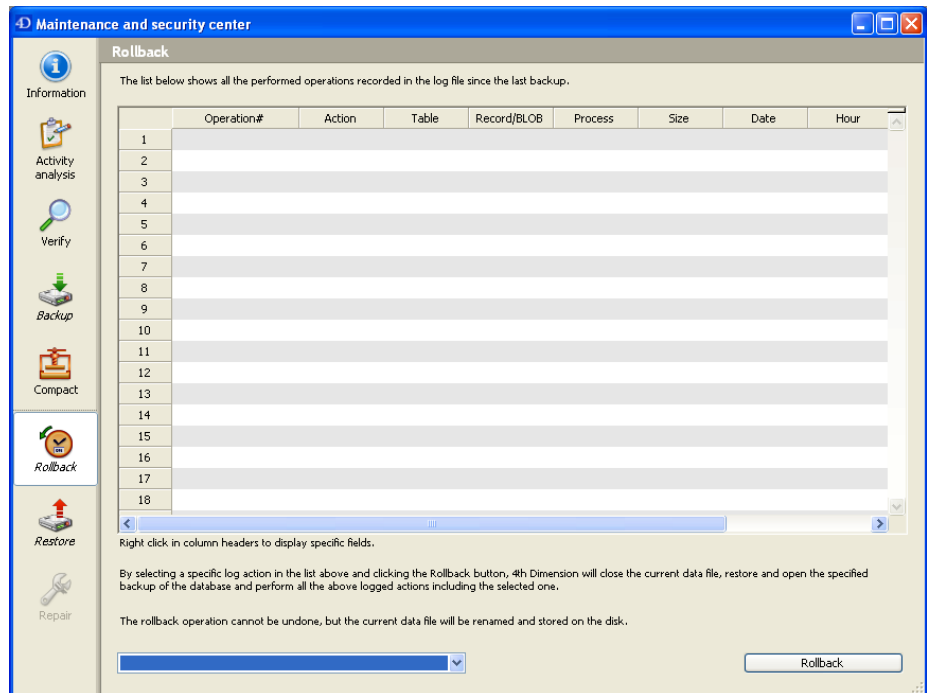
このモードでは特定の自動操作は導入されていない点に注意することが大切です。: オリジナルのデータファイルには移動も変更もありません。

注 : 指定されたデータファイルと関連付けられているインデックスファイルはすべて、自動的に圧縮されます。

「ロールバック」ページ

このページは、ログファイル上の、実行された操作に対するロールバック機能にアクセスする時に使用します。この機能は、複数レベルに適用された取り消し機能に似ています。

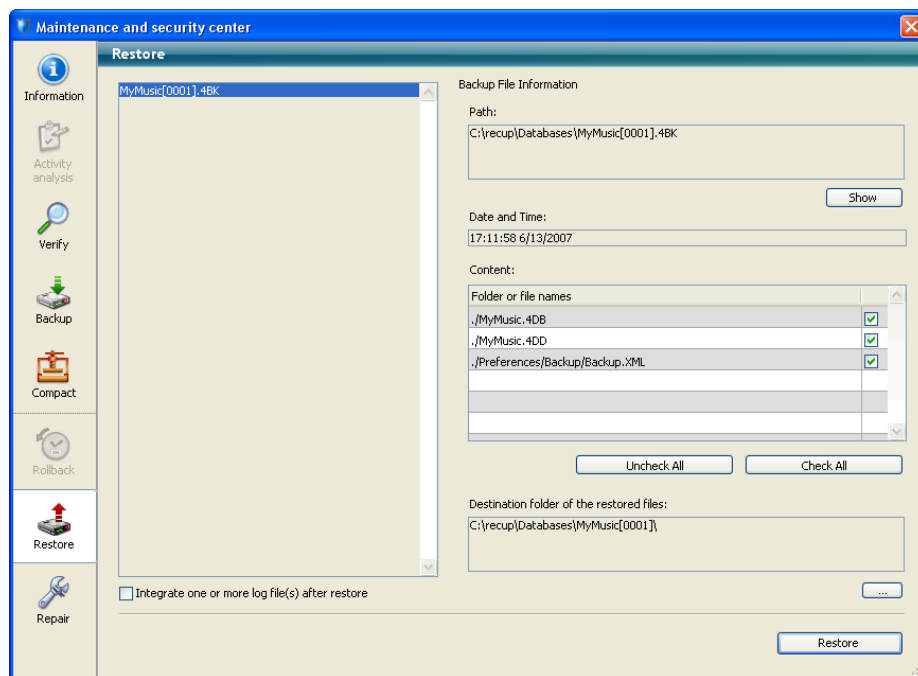
この機能は、データベースで間違っってレコードを削除した場合には特に便利です。



注： 4D の 11 バージョンでは、データベースのログファイルに、(プログラムの旧バージョンにあった ".4DL" の代わりに) ".journal" 拡張子がデフォルトで付けられます。

「復元」 ページ

このページは、データベースのバックアップを表示する時、また手動で復元する時に使用します。



ウィンドウ左側のリストは、既存のデータベースのバックアップを表示します。リストからバックアップを選択すると、ウィンドウの右側にこのバックアップに関する情報が表示されます。

- **パス:** 選択されたバックアップファイルの完全なファイルパス名。表示ボタンは、システムウィンドウを開いてバックアップファイルを表示します。
- **日付と時刻:** バックアップの日付と時刻。
- **内容:** バックアップファイルの内容。リスト内の各項目の隣にはチェックボックスがあり、それぞれの項目について復元実行の有無を指定できます。すべてを選択するボタンまたはすべての選択をはずすボタンを使用して、復元する項目のリストを設定します。
- **復元されたファイルの保存先フォルダ:** 復元ファイルが置かれるフォルダ。この場所を変更するには、[...] ボタンをクリックし、復元を実行するフォルダを指定します。

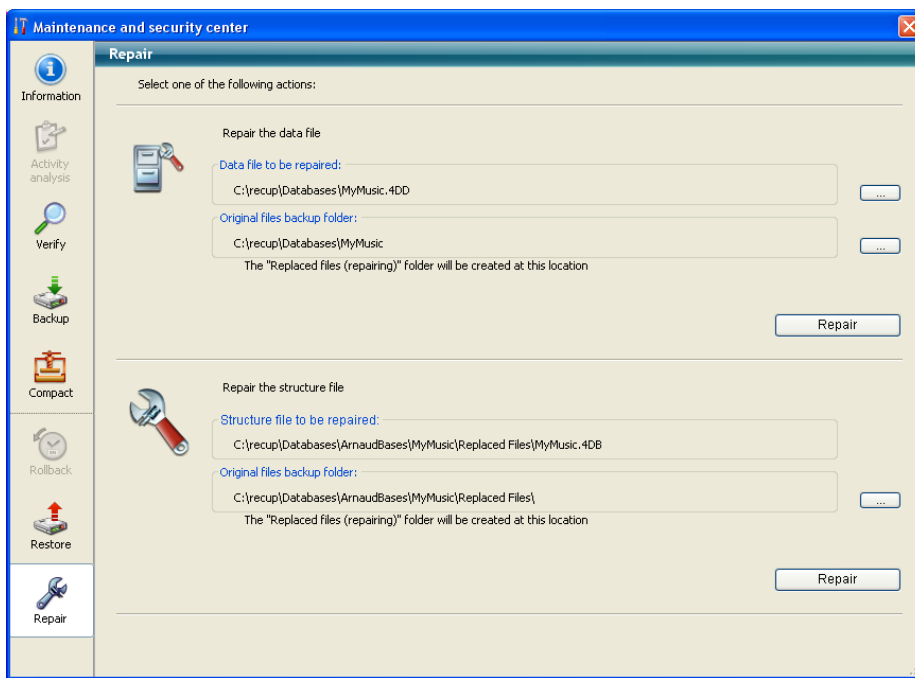
復元後に一つ以上のログファイルを統合 オプションを使用して、バックアップされた複数のログファイルをログファイルに統合できます。たとえば (データベースの4つのバックアップに対応する) 4つの連続したログファイルのバックアップがある時、まず最初のデータベースのバックアップを復元し、ひとつずつログファイルのバックアップを統合できます。これによりたとえば、最後のデータベースバックアップが失われているような場合でも、データファイルを復旧させることが可能です。

このオプションがチェックされていると、4D は復旧の後、標準のファイルを開くダイアログを表示し、統合するログファイルが選択可能になります。この開くダイアログは、キャンセルされるまで繰り返し表示されます。

「修復」 ページ

このページは、損傷したデータまたはストラクチャファイルを修復する時に使用します。通常は、データベースを開く時または検証後に異常が発覚し、4D の要求があった時のみ修復機能を使用します。

注： 検証に関する詳細は、[ページ185、「検証」ページ](#)の節を参照してください。



修復操作は毎回オリジナルファイルのコピーを伴うため、アプリケーションフォルダのサイズが大きくなります。アプリケーションのサイズが過剰に増加しないよう、これを考慮することが大切です (特に、4D アプリケーションがパッケージとして現れる Mac OS の場合)。パッケージのサイズを小さく保つためには、パッケージ内オリジナルファイルのコピーを手動で削除することも役立ちます。

4D v11 バージョンの最も重要かつパワフルな特徴の1つは、4D エンジンに実装された SQL 標準です。

SQL は、データベースに保存されたデータを作成、編成、管理および回収するためのランゲージです。SQL 自体はデータベース管理システムでも単独のプロダクトでもありません。SQL は DBMS と通信するためのランゲージおよびツールであり、データベース管理システムの不可欠な部分です。

新しい 4D の SQL 内臓エンジンは SQL92 (または SQL-2) 互換ですが、いくらか特定の实装、制限および差異があります。

この章では、次のトピックについて説明しています。

- SQL がどのように 4D に実装されているか、SQL へのアクセス方法および使用方法
- 純粋な「SQL92」ランゲージと「4D SQL」ランゲージとの違い

4D がサポートする SQL ランゲージに関する詳細は、4D SQL Reference マニュアルを参照してください。

4D SQL エンジンへのアクセス

4D SQL エンジンへのクエリの送信 4D の SQL エンジンには、次の 3 つの方法で呼び出すことができます。

- 新しい QUERY BY SQL コマンドを使う。SQL 「SELECT」 文の 「WHERE」 節をクエリ引数として渡します。

例:

```
QUERY BY SQL([OFFICES];"SALES > 100")
```

QUERY BY SQL コマンドの使用についての詳細は、[ページ 254](#)、**QUERY BY SQL コマンド**を参照してください。

- **"外部データソース"** テーマ (ODBC SET PARAMETER、ODBC EXECUTE など) に分類された 4D の ODBC コマンドを使用する。これらのコマンドは、カレントデータベースの 4D SQL エンジンと連動するよう変更されました。

4D の SQL カーネルに対するハイレベル ODBC コマンドの使用についての詳細は、[ページ 249](#)、**"外部データソース"** の節を参照してください。

- 4D の標準メソッドエディタを使用する。SQL ステートメントを、4D の標準メソッドエディタに直接記述できます。新しい 2 つのタグ、**Start SQL** タグと **End SQL** タグの間に SQL クエリを挿入します。含まれているコードは、4D インタプリタが解析するのではなく、SQL エンジンが実行します。4D メソッド内での SQL ステートメント使用についての詳細は、[ページ 127](#)、**"メソッドエディタへの SQL コード入力"** の節を参照してください。

4D と SQL エンジン間のデータ送信

4D 式の参照

SQL 式の 「WHERE」 節および 「INTO」 節内で、有効な 4D 式 (変数、フィールド、配列、式など) のタイプを参照できます。4D 参照を示すには、次のいずれかの表記を使用します。

- "<<" と ">>" のように、「小なり」記号 2 つと 「大なり」記号 2 つの間に参照を挿入します。
- 参照の前に ":" (コロン) を付けます。

例:

```
C_STRING(80;vName)
vName:=Request("Name:")
```

```
ODBC EXECUTE("SELECT age FROM PEOPLE WHERE
name=<<vName>>")
```

または:

```
C_STRING(80;vName)
vName:=Request("Name:")
Begin SQL
  SELECT age FROM PEOPLE WHERE name= :vName
End SQL
```

注： インタープロセス変数処理の場合は、括弧 [] の使用が要求されます (例えば、<<[<>myvar]>> or :[<>myvar])。

データを SQL リクエストから 4D に読み出す

「SELECT」ステートメントのデータ検索は、Begin SQL/End SQL タグの間で SELECT コマンドの「INTO」節を使うか、または外部データソース (ODBC) ランゲージコマンドを使って処理されます。

- Begin SQL/End SQL タグの場合、SQL クエリの「INTO」節を使い、値を求めるため有効な 4D 式 (フィールド、変数、配列) の参照を指定します。

```
Begin SQL
  SELECT ename FROM emp INTO <<[Employees]Name>>
End SQL
```

- ODBC EXECUTE コマンドの場合、追加の引数を使用することもできます。

```
ODBC EXECUTE("SELECT ename FROM emp";[Employees]Name)
```

詳細については、[ページ 250](#)、**ODBC EXECUTE コマンド**を参照してください。

SQL からデータを受信する 2 つの方法 (Begin SQL/End SQL タグおよび ODBC コマンド) の主な違いは、後者の場合は明示的に ODBC LOAD RECORD を使ってレコードを読み込む必要があるのに対し、前者の場合は 4D にすべての情報が 1 回で返信される点です。

例えば、PEOPLE テーブルに 100 レコードがあると仮定します。

- ODBC コマンドを使う場合:

```
ARRAY INTEGER(aBirthYear;0)
C_STRING(40;vName)
vName:="Smith"
$SQLStm:="SELECT Birth_Year FROM PEOPLE WHERE
ename= <<vName>>"
ODBC EXECUTE($SQLStm;aBirthYear)
```

```
While (Not (ODBC End Selection))
  ODBC LOAD RECORD(10)
End while
```

ここでは、100レコードをすべて検索するのに10回ループしなければなりません。すべてのレコードを1回で読み込むには、次のように書く必要があります。

```
ODBC LOAD RECORD(ODBC All Records)
```

■ **Begin SQL/End SQL** タグを使う場合：

```
ARRAY INTEGER(aBirthYear;0)
C_STRING(40;vName)
vName:="Smith"
Begin SQL
  SELECT Birth_Year FROM PEOPLE
  WHERE ename= <<vName>> INTO <<aBirthYear>>
End SQL
```

この状況では、「セレクト」ステートメントの実行後に「eBirthYear」配列のサイズが100になり、その要素は100レコードにあるすべての誕生日 (birth year) で満たされます。

配列の代わりに、カラム (つまり、4D フィールド) に検索データを保存する場合は、4D はデータすべての保存に必要な、できるだけ多くのレコードを自動的に作成します。PEOPLE テーブルに100レコードがあると仮定した前述の例では：

■ **ODBC コマンド** を使う場合：

```
C_STRING(40;vName)
vName:="Smith"
$$SQLStm:="SELECT Birth_Year FROM PEOPLE WHERE
ename= <<vName>>%"
ODBC EXECUTE($$SQLStm;[MYTABLE]Birth_Year)
While (Not (ODBC End Selection))
  ODBC LOAD RECORD(10)
End while
```

ここでは、100レコードをすべて検索するのに10回ループしなければなりません。1回につきMYTABLEテーブルに10レコードが作成され、PEOPLE テーブルから検索されたBirth yearが[MYTABLE]Birth_Yearフィールドに保存されます。

■ **Begin SQL/End SQL** タグを使う場合：


```

C_STRING(40;vName)
vName:="Smith"
Begin SQL
  SELECT Birth_Year FROM PEOPLE
  WHERE ename= <<vName>> INTO <<[MYTABLE]Birth_Year>>
End SQL

```

この状況では、「SELECT」ステートメントの実行中に MYTABLE テーブルに 100 レコードが作成され、Birth_Year フィールドに PEOPLE テーブルからの対応データ、Birth_Year のデータが格納されます。

リストボックスの使用

4D は、「SELECT」クエリからのデータをリストボックスに格納できる特定の自動機能を含みます。詳細については、[ページ 154](#)、「SQL クエリ結果の表示」の節を参照してください。

クエリの最適化

最適化の理由から、クエリには SQL 機能より、4D 式を使用する方が望ましいです。実際、SQL 機能は見つかった各レコードに対して評価されますが、4D 式はクエリ実行の前に 1 回計算されます。

例えば、次のステートメントでは：

```

ODBC EXECUTE("SELECT FullName FROM PEOPLE WHERE
FullName=<<vLastName+vFirstName>>")

```

... 「vLastName+vFirstName」の式は、クエリ実行の前に 1 回計算されます。次のステートメントでは：

```

ODBC EXECUTE("SELECT FullName FROM PEOPLE WHERE
FullName=CONCAT(<<vLastName>>,<<vFirstName>>)")

```

... 「CONCAT(<<vLastName>>,<<vFirstName>>)」は、テーブルの各レコードに対して呼び出されます；つまり、式は各レコードごとに評価されます。

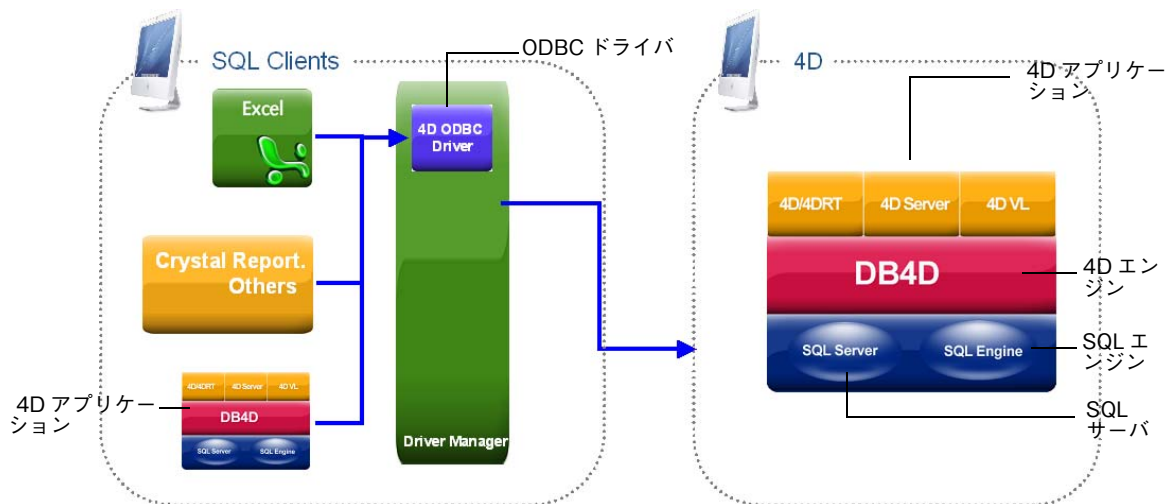
4D SQL サーバの設定

4D v11 は、4D データベースに格納されたデータに外部からアクセスできるパワフルな SQL サーバを含みます。このアクセスは、4D の ODBC ドライバによって行われます。

4D アプリケーションの SQL サーバは、随時停止または起動できます。更に、性能とセキュリティの理由から、TCP ポートならびに待機中の IP アドレスを指定し、4D データベースへのアクセスを制限することができます。

SQL サーバへの外部アクセス

4D SQL サーバへの外部アクセスは ODBC によって行われます。サードパーティーアプリケーション (Excel^(R) タイプのスプレッドシート、他の DBMS など) や 4D の他アプリケーションから 4D SQL サーバに接続させることができる新しい ODBC ドライバを提供されています。これについては、次の図でまとめています。:

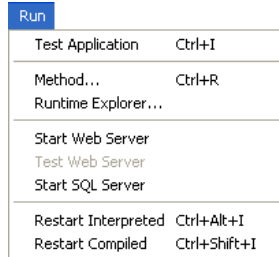


4D の ODBC ドライバは、SQL クライアントのマシンにインストールしなければなりません。4D の ODBC ドライバのインストールおよび設定については、別のマニュアルで詳細を説明しています。

4D SQL サーバの起動と停止

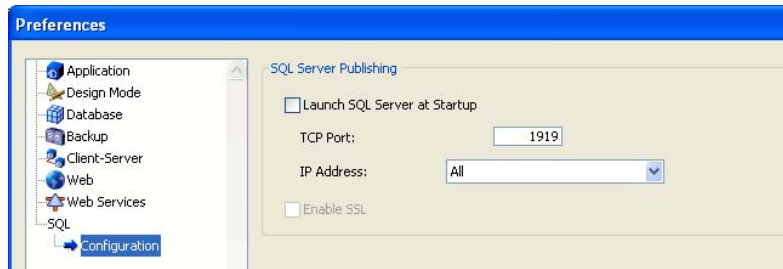
4D v11 では、SQL サーバを起動 / 停止する方法は次の 3 通りあります。

- 4D の「実行」メニューの SQL サーバを起動 /SQL サーバを停止コマンドを使って手動で起動 / 停止します。:



サーバが起動すると、メニュー項目は「SQL サーバを停止」に変わります。

- 環境設定を使用して、アプリケーションの起動時に自動的に起動する。これを行うには、「SQL / 設定」ページを表示させ、「開始時に SQL サーバを起動する」オプションにチェックを入れます。



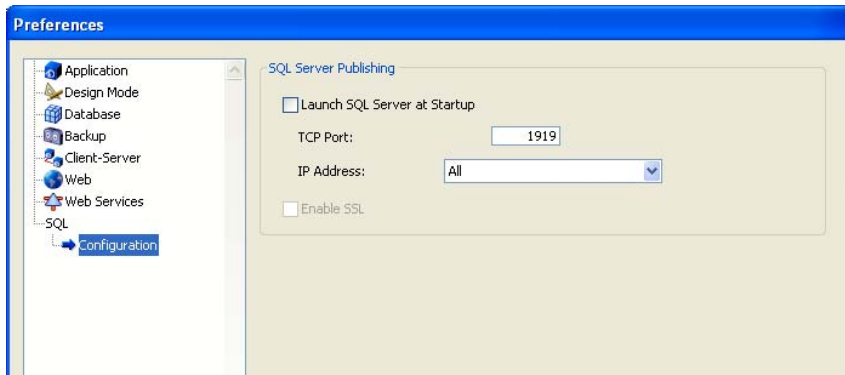
- 新しい START SQL SERVER コマンドおよび STOP SQL SERVER コマンド ("SQL" テーマ) を使用してプログラミングにより起動 / 停止します。

SQL サーバが停止している場合 (または、起動前の場合)、4D は外部の SQL クエリには一切反応しません。

注 : SQL サーバの停止は 4D SQL エンジンの内部の機能に影響を及ぼしません。SQL エンジンは内部クエリに常に応答します。

SQL サーバ公開環境設定

4D に統合された SQL サーバの公開を設定できます。これらの設定項目は、データベース環境設定の「SQL/ 設定」ページにあります。:



- 「開始時に SQL サーバを起動」オプションは、アプリケーションの開始時に SQL サーバを起動する時に使用します (ページ 204、"4D SQL サーバの起動と停止" の節を参照)。

- TCPポート: デフォルトで、4D SQLサーバはTCPポート1919で応答します。このポートが既に他のサービスに使用されている場合、または他の設定が必要な場合は、4D SQLサーバに使う TCP ポートを変えることができます。

注: [0] を渡すと、4D はデフォルトの TCP ポート番号 [1919] を使用します。

- IPアドレス: SQLサーバがSQLクエリを待ち受けするIPアドレスを設定できます。デフォルトでは、サーバはすべてのIPアドレスで応答します (「すべて」 オプション)。

"IP アドレス" ドロップダウンリストには、マシンに設定されたすべての IP アドレスがリストされます。特定のアドレスを選択すると、サーバはそのアドレスに送られたクエリにのみ応答します。

これは、複数の TCP/IP アドレスを持つマシンがホストする 4D アプリケーションを対象としています。

注: クライアント側の注意点として、アプリケーションが接続する SQL サーバの IP アドレスおよび TCP ポートは ODBC データソース定義に正確に設定されている必要があります。

- SSL を有効にする: このオプションは、SQL サーバが SQL 接続の処理のために SSL プロトコルを有効にするかどうかを指定します。

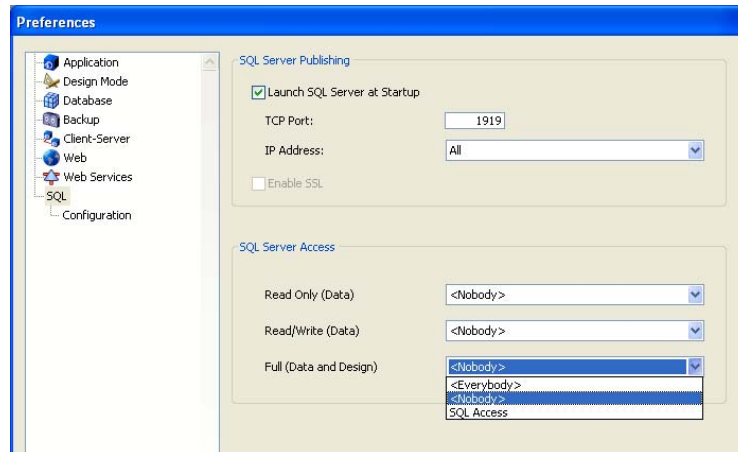
4D データベースの アクセス管理

セキュリティの理由から、外部からSQLサーバに送られたクエリが4Dデータベース内で行える動作を制限することができます。

これは、2つのレベルにおいて行うことができます。

- 許可される動作のタイプレベルで制限する。
- クエリを実行するユーザレベルで制限する。

これらの設定は、データベース環境設定の「SQL/ 設定」ページにあります。:



SQLサーバによる4Dデータベースへのアクセスについては、それぞれ独立した3種類の方法が設定できます。

- "読み込みのみ (データ)": データベース・テーブルからすべてのデータを読み取るアクセス権はありますが、レコードの追加、変更、削除、或いはデータベース・ストラクチャの変更は一切許可されません。
- "読み書き可能 (データ)": データベース・テーブルのすべてのデータに関して読み書き (追加、変更、削除) アクセス権はありますが、データベース・ストラクチャの変更は許可されません。
- "フルアクセス (データとデザイン)": データベース・テーブルのすべてのデータに関して読み書き (追加、変更、削除) アクセス権がある他、データベース・ストラクチャ (テーブル、フィールド、リレーションなど) の変更も許可されます。

それぞれのアクセスタイプによってユーザを指定できます。この目的では、次の3つのオプションがあります。

- <Nobody>: このオプションを選択した場合、当該のアクセスタイプは、クエリの発信元に関係なく、どんなクエリに対しても拒否されます。この引数は、4D のパスワードアクセス管理システムが有効化されていない場合でも使用できます。
- <すべてのグループ>: このオプションを選択した場合、当該のアクセスタイプはすべてのクエリに対して許可されます(制限はありません)。
- ユーザグループ: このオプションにより、当該のアクセスタイプを実行する独占権を持つユーザグループを指定できます。このオプションは、4Dパスワードが有効化されていることが前提です。クエリ発信元のユーザは、ODBC を通して SQL サーバに接続する時に名前とパスワードを提供します。

警告: このメカニズムは 4D パスワードに基づいています。SQL サーバアクセス管理が効力を発するためには、4D パスワードシステムは有効化されていなければなりません (Designer にパスワードを設定する)。

注: 各 4D プロジェクトメソッドのレベルで、追加のセキュリティ・オプションを設定できます。詳細については、[ページ 213](#)、"["SQL 利用可" オプション](#)" の節を参照してください。

4D SQL エンジンの実装と制限

基本的に、4D SQL エンジンは SQL92 に対応しています。つまり、使用するコマンド、機能、演算子またはシンタックスの詳細説明については、SQL92 リファレンスを参照できることとなります。これらは、例えばインターネットで検索できます。

しかし、4D SQL エンジンは SQL92 の機能を 100% サポートするわけではなく、特定の追加機能を提供します。

この節では、4D SQL エンジンの主な実装と制限について説明します。SQL92 の機能および 4D のサポートについては、[ページ 223](#)、"[SQL 関数のサポート](#)" の節に詳細なリストがあります。

一般的制限

4D の SQL エンジンが 4D データベースの核心に統合されたので、データベース当たりのテーブル、カラム (フィールド) およびレコードの最大数に関する制限の他、テーブルとカラムの命名ルールに関する制限も、標準の内部 4D カーネル (DB4D) の制限と同じです。これらについては、下記にリストアップします。

- テーブルの最大数: 理論上は 20 億テーブルですが、4D v11 との互換性を考慮して「32767」です。
- テーブル内のカラム (フィールド) の最大数: 理論上は 20 億カラム (フィールド) ですが、4D v11 との互換性を考慮して「32767」です。
4D v11 の "Standard Edition" では、カラムの最高数は「511」に制限されています。
- テーブル内の行 (レコード) の最高数: 「10 億」です。サブレコードのあるフィールドについては、全レコードに対して「10 億」サブレコードが限界です。
- インデックスキーの最大数: 「10 億 x 64」です。
- 主キーに NULL 値は許されず、また一意的でなければなりません。主キーのカラム (フィールド) にインデックスを付ける必要はありません。
- テーブル名およびフィールド名に許可される文字の最大数: 「31 文字 (4D の制限)」です。
異なるユーザが作成した同じ名前のテーブルは許可されません。標準の 4D 管理メカニズムが適用されます。

データタイプ

次の表は、4D SQL がサポートするデータタイプ、およびそれらが 4D で対応するタイプを表しています。:

4D SQL	説明	4D v11
Varchar	テキスト	テキスト
Real	+/-3,4E38 の範囲の浮動小数点数	実数
Numeric	+/- 2E64 の範囲の数	整数 64 ビット
Float	浮動小数点数 (ほぼ無限大)	実数
Smallint	-32 768 ~ 32 767 の数	整数
Int	-2 147 483 648 ~ 2 147 483 647 の数	倍長整数
Bit	TRUE(真) または FALSE(偽) の値のみ取るフィールド	ブール
Boolean	TRUE(真) または FALSE(偽) の値のみ取るフィールド	ブール
Blob	最大 2 GB; 図形、他のアプリケーション、任意のドキュメントなど、すべてのバイナリオブジェクト	Blob
Bit varying	最大 2 GB; 図形、他のアプリケーション、任意のドキュメントなど、すべてのバイナリオブジェクト	Blob
Clob	テキスト最大 2 GB 文字。このカラム (フィールド) にインデックスを付けることはできません。これはレコードそのものには保存されません。	テキスト
Text	テキスト最大 2 GB 文字。このカラム (フィールド) にインデックスを付けることはできません。これはレコードそのものには保存されません。	テキスト
Timestamp	日月 年時間 : 分 : 秒 : ミリ秒フォーマットの日付と時間	日付と時間の部分は別々に処理 (自動変換)
Duration	日 : 時間 : 分 : 秒 : ミリ秒フォーマットの時間	時間
Interval	日 : 時間 : 分 : 秒 : ミリ秒フォーマットの時間	時間
Picture	PICT ピクチャ最大 2 GB	ピクチャ

数値タイプ間には、自動データタイプ変換が実装されています。

数字を表す文字列は、対応する数値に変換されません。値を 1 つのタイプから別のタイプに変換する特別の **CAST** 機能があります。

次の SQL データタイプは実装されていません。

- NCHAR
- NCHAR VARYING.

4DでのNULL値

NULL値は、4D SQLのランゲージの他、4Dデータベース・エンジンにも実装されました。しかし、4DランゲージではNULL値をサポートしていません。

注： それでも、新しい **Is field value Null** コマンドおよび **SET FIELD VALUE NULL** コマンドを使用すると、4DフィールドでNULL値を読み書きすることができます。

NULL値を空値にマップの処理と互換性

4D v11での互換性の理由から、4Dデータベース・テーブルに保存されたNULL値は、4Dランゲージによって操作されると自動的にデフォルト値に変換されます。例えば、次のステートメントの場合：

```
myAlphavar:=[mytable]MyAlphafield
```

... 「MyAlphafield」フィールドがNULL値の場合、「myAlphavar」変数は""(空の文字列)が代入されます。

デフォルト値はデータタイプによって変わります。

- データタイプが文字およびテキストの場合：""
- データタイプが実数、整数および倍長整数の場合：0
- データタイプが日付の場合："00/00/00"
- データタイプが時間の場合："00:00:00"
- データタイプがブールの場合：FALSE(偽)
- データタイプがピクチャの場合：空のピクチャ
- データタイプがBlobの場合：空のblob

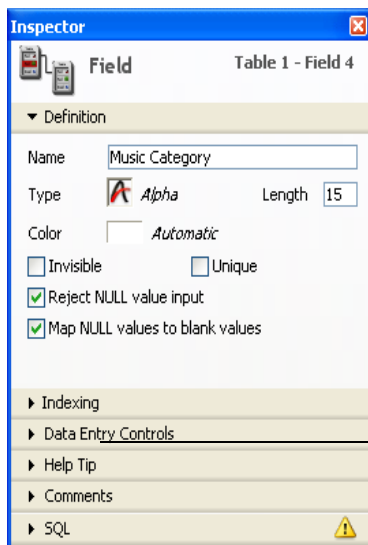
一方、このメカニズムは、例えばクエリなど、4D データベースエンジンのレベルで実行された処理には原則として適用されません。実際"空"の値の検索 (たとえば myvalue=0) は、NULL 値が格納されたレコードを検索しません。逆も同様です。両方の値 (デフォルト値と NULL) が同じフィールドに格納されている場合、特定の処理については代替えの手段もしくは追加のコードが必要になるでしょう。

これを避けるために、4D v11 ランゲージですべての処理を標準化することができるよう、新しいオプション、NULL 値を空値にマップ、が使用可能です。ストラクチャエディタのフィールドインスペクタウィンドウに用意されたこのオプションは、すべての処理にデフォルト値を適用するよう仕様を拡張するために使用されます。NULL 値を含むフィールドは、すべからずデフォルト値を含むものとして処理されます。このオプションはデフォルトで選択されています。詳細は [ページ 112](#)、"[ヌル値サポートオプション](#)"の節を参照してください。

NULL 値を空値にマッププロパティは、データベースエンジンのローレベルで動作します。このオプションは特に、新しい [ページ 257](#)、[Is field value Null コマンド](#) で有効です。

NULL 値の入力を拒否

新しいフィールドプロパティ NULL 値の入力を拒否 は NULL 値が格納されることを防止するために使用します。



新しいオプション

フィールドのこの属性が有効にされると、そのフィールドに NULL 値を格納することはできなくなります。このローレベルなプロパティは、SQL の NOT NULL 属性に対応します。

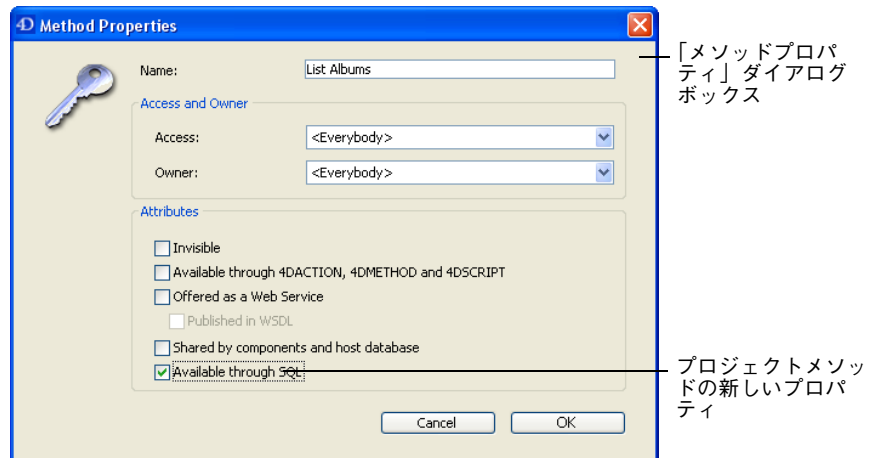
一般的に、4D データベースで NULL 値を使用可能にしたい場合は、常に 4D の SQL ランゲージを使用することをお勧めします。

注： 4D では、フィールドに必須入力属性を設定することが可能です。二つのコンセプトは同じですが、スコープが異なります。必須入力属性はデータの入力をコントロールし、NULL 値の入力を拒否属性はデータベースエンジンレベルで動作します。

この属性が設定されたフィールドが NULL 値を受け取ると、エラーが生成されます。

"SQL 利用可" オプション

4D プロジェクトメソッドにセキュリティプロパティ、SQL 利用可が追加されました。:



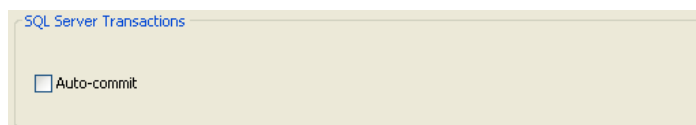
このオプションにチェックが入っていると、4D SQL エンジンがプロジェクトメソッドを実行できます。これがデフォルトで選択されていないということは、4D プロジェクトメソッドが保護されており、オプションにチェックを入れて明示的に許可しない限りは 4D SQL エンジンに呼び出されないことを意味します。

このプロパティは、内部および外部両方の SQL クエリすべてに適用されます。ODBC ドライバ、Begin SQL/End SQL タグ間に挿入された SQL コード、または QUERY BY SQL コマンドのいずれを使って実行したかは関係ありません。

-
- 注： ・ メソッドに "SQL 利用可" 属性が付加されている時でも、環境設定のレベルおよびメソッドプロパティのレベルで設定されたアクセス権は実行時に考慮されます。
・ ODBC の「*SQL Procedure*」機能は、"SQL 利用可" 属性を持つプロジェクトメソッドのみ返します。
-

自動コミット

4D 環境設定の「SQL/ 設定」ページにあるオプションを使用して、4D SQL エンジンの自動コミットメカニズムを有効にできます。



自動コミットモードの目的は、データの参照整合性を維持することです。このオプションにチェックが入っている場合、トランザクション内でまだ実行されていない「SELECT」、「INSERT」、「UPDATE」および「DELETE」(SIUD) クエリはどれも、自動的にアドホックトランザクションに加えられます。これは、クエリが完全な形で実行されること、またはエラーの場合は完全にキャンセルされることを保証するものです。

既にトランザクションに含まれるクエリ (参照整合性のカスタム管理) は、このオプションの影響を受けません。

このオプションにチェックが入っていない場合、一切の自動トランザクションは生成されません (「SELECT... FOR UPDATE」クエリの場合は除く。ページ 217、"「SELECT」オプション" の節を参照してください)。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。

VHW GDWDEDVH SDUDP HWHU コマンドを使用して、プログラムでこのオプションを管理できます (ページ 331、"SET DATABASE PARAMETER, Get database parameter" の節を参照)。

-
- 注： 4D SQL エンジンによって検索されたローカルデータベースのみがこの設定の影響を受けます。外部データベースの場合、自動コミットメカニズムはリモートの SQL エンジンが処理します。
-

システムテーブル

4D の SQL カタログは、読み込みアクセス権を持つ SQL ユーザなら誰でもアクセスできる 6 つのシステムテーブル、「_USER_TABLES」、「_USER_COLUMNS」、「_USER_INDEXES」、「_USER_CONSTRAINTS」、「_USER_IND_COLUMNS」および「_USER_CONS_COLUMNS」を含みます。

SQL の慣習に従い、システムテーブルはデータベース・ストラクチャを記述します。以下は、これらのテーブルおよびフィールドの説明です。:

_USER_TABLES		データベースのユーザテーブルを説明
TABLE_NAME	VARCHAR	テーブルの名前
TEMPORARY	BOOLEAN	一時的テーブルなら TRUE; それ以外は FALSE
TABLE_ID	INT64	テーブル番号

_USER_COLUMNS		データベースのユーザテーブルのカラムを説明
TABLE_NAME	VARCHAR	テーブルの名前
COLUMN_NAME	VARCHAR	カラムの名前
DATA_TYPE	INT32	カラムのタイプ
DATA_LENGTH	INT32	カラムの長さ
NULLABLE	BOOLEAN	カラムがヌル値を受け入れる場合は TRUE; それ以外は FALSE
TABLE_ID	INT64	テーブル番号
COLUMN_ID	INT64	カラム番号

_USER_INDEXES		データベースのユーザインデックスを説明
INDEX_ID	VARCHAR	インデックス番号
INDEX_NAME	VARCHAR	インデックスの名前
INDEX_TYPE	INT32	インデックスのタイプ
TABLE_NAME	VARCHAR	インデックス付テーブルの名前
UNIQUENESS	BOOLEAN	インデックスが一意制約を課す場合は TRUE; それ以外は FALSE
TABLE_ID	INT64	インデックス付テーブルの番号

_USER_IND_COLUMNS		データベースのユーザインデックスのカラムを説明
INDEX_ID	VARCHAR	インデックス番号
INDEX_NAME	VARCHAR	インデックスの名前
TABLE_NAME	VARCHAR	インデックス付テーブルの名前
COLUMN_NAME	VARCHAR	インデックス付カラムの名前
POSITION	INT32	インデックス付カラムの位置
TABLE_ID	INT64	インデックス付テーブルの番号
COLUMN_ID	INT64	カラムの番号

_USER_CONSTRAINTS		データベースのユーザ制約を説明
CONSTRAINT_ID	VARCHAR	制約番号
CONSTRAINT_NAME	VARCHAR	制約の名前
CONSTRAINT_TYPE	VARCHAR	制約のタイプ
TABLE_NAME	VARCHAR	制約付テーブルの名前
TABLE_ID	INT64	制約付テーブルの番号
DELETE_RULE	VARCHAR	削除ルール CASCADE または RESTRICT
RELATED_TABLE_NAME	VARCHAR	リレートされたテーブルの名前
RELATED_TABLE_ID	INT64	リレートされたテーブルの番号

_USER_CONS_COLUMNS		データベースのユーザ制約のカラムを説明
CONSTRAINT_ID	VARCHAR	制約番号
CONSTRAINT_NAME	VARCHAR	制約の名前
TABLE_NAME	VARCHAR	制約付テーブルの名前
TABLE_ID	INT64	制約付テーブルの番号
COLUMN_NAME	VARCHAR	制約付カラムの名前
COLUMN_ID	INT64	制約付カラムの番号
POSITION	INT32	制約付カラムの位置
RELATED_COLUMN_NAME	VARCHAR	制約内のリレートされたカラムの名前

RELATED_COLUMN_ID	INT32	制約内のリレートされたカラムの番号
-------------------	-------	-------------------

SQL DML

この節では、4D v11 SQL エンジンの DML (Data Manipulate Language) の主な実装と制限について説明しています。

サポート対象のデータタイプの説明については、[ページ 210](#)、"[データタイプ](#)"の節を参照してください。

サポート対象の機能および演算子の説明については、[ページ 223](#)、"[SQL 関数のサポート](#)"の節を参照してください。

有効なカラム名

有効なカラム名を使ってカラムのアドレスを指定できます。例えば、「CUSTOMERS」テーブルの「NAME」カラムにアクセスする時は、「CUSTOMERS.NAME」と書けます。

「SELECT」オプション

- "*" (アスタリスク) と明示的フィールドの混ざったクエリは、「SELECT」ステートメントでは許可されません。
- 次の例は許可されません:
SELECT *, SALES, TARGET FROM OFFICES
- 次の例は許可されます:
SELECT * FROM OFFICES
- 「FROM」節のテーブル名の場所にクエリ式を入れることはできません。
- グループ化されたクエリ (「GROUP BY」節) は、複数のグループカラムを受け入れます。
- グループ検索条件 (「HAVING」節): 「GROUP BY」節なしに「HAVING」節を使用できます。

4D SQL エンジンには、「SELECT... FOR UPDATE」クエリがトランザクション内でまだ実行されていない場合、それを自動的にトランザクションに加えるメカニズムがあります。これは、データの参照整合性を維持します。

「INSERT」、 「DELETE」 および 「UPDATE」

- 「INSERT」ステートメントは、1行および複数行のクエリでサポートされます。
ただし、複数行の「INSERT」ステートメントは、「UNION」および「JOIN」の操作を許可しません。
 - 次の例は許可されます：
INSERT INTO Table1 SELECT * FROM Table2
- 「DELETE」ステートメントはクエリにサポートされます。ただし、位置決めされた「DELETE」ステートメントはサポートされません。
削除ルール: 4Dに「CASCADE」削除が実装されましたが、「SET DEFAULT」の削除ルールはサポートされません。
- 「UPDATE」ステートメントはクエリおよびサブクエリにサポートされます。ただし、位置決めされた「UPDATE」ステートメントはサポートされません。
更新ルール: 4Dに「CASCADE」更新が実装されましたが、「SET NULL」および「SET DEFAULT」の削除ルールはサポートされません。

トリガ

トリガは4Dランゲージを通して実装され、SQLエンジンによって完全にサポートされています。

制約

- 次のようなデータの整合性制約が実装されました。
 - 必須データ (必須カラム): カラムは非ヌル値を含まなければなりません。
 - 実体整合性: 4Dカーネルに実装されました。
- SQL内蔵のデータ整合性制約の妥当性チェックは実装されていません。
- 次の4つのタイプの制約は実装されていません：
 - カラムの制約
 - ドメイン
 - テーブルの制約
 - アサーション (ASSERTION)

SQL TCL (トランザクション)

4D SQL には、「COMMIT」および「ROLLBACK」のステートメントが実装されました。トランザクションは、「START TRANSACTION」で開始し、「COMMIT」ステートメントまたは「ROLLBACK」ステートメントのいずれかで終了します。

4D のトランザクション・コンセプトも使用できます。4D エンジンには標準のトランザクション・メカニズムが実装されています。また、最大 32767 レベルまで、トランザクションの入れ子を持つことができます。

次は、対応するコマンドです。:

SQL	4D
START (TRANSACTION)	START TRANSACTION
COMMIT (TRANSACTION)	VALIDATE TRANSACTION
ROLLBACK (TRANSACTION)	CANCEL TRANSACTION

データベースのレベルでは、4D トランザクションと SQL トランザクションに違いはありません。両トランザクションのタイプは、同じデータとプロセスを共有します。Start SQL/End SQL の構造に含まれる SQL ステートメントと、ローカルデータベースに適用された **QUERY BY SQL** コマンドおよび ODBC コマンドは、常に 4D の標準コマンドと同じ状況で実行されます。

次の例は、トランザクションの異なる組み合わせを表しています。

- ▼ "John" および "Smith" とともに emp テーブルに追加されません。

ODBC LOGIN(SQL_INTERNAL ;"";"") `4D SQL エンジン`を初期化

START TRANSACTION `カレントプロセスでトランザクションを開始

Begin SQL

INSERT INTO emp

(NAME)

VALUES ('John');

End SQL

ODBC EXECUTE("START") `カレントプロセスでもう1つのトランザクション`

ODBC CANCEL LOAD

ODBC EXECUTE("INSERT INTO emp (NAME) VALUES ('Smith')")

`この文は同じプロセスで実行される`

ODBC CANCEL LOAD

ODBC EXECUTE("ROLLBACK") `プロセス内側のトランザクションをキャンセル`

CANCEL TRANSACTION `プロセスの外側のトランザクションをキャンセル`

ル
ODBC LOGOUT

- ▼ "John" のみが emp テーブルに追加されます。

```
ODBC LOGIN(SQL_INTERNAL ;"";"";)"
START TRANSACTION
Begin SQL
  INSERT INTO emp
    (NAME)
    VALUES ('John');
End SQL
ODBC EXECUTE("START")
ODBC CANCEL LOAD
ODBC EXECUTE("INSERT INTO emp (NAME) VALUES ('Smith')")
ODBC CANCEL LOAD
ODBC EXECUTE("ROLLBACK") `プロセス内側のトランザクションをキャンセル
VALIDATE TRANSACTION `プロセスの外側のトランザクションを有効化
ODBC LOGOUT
```

- ▼ "John" と "Smith" とともに emp テーブルに追加されません。外側のトランザクションが内側のトランザクションもキャンセルします。

```
ODBC LOGIN(SQL_INTERNAL ;"";"";)"
START TRANSACTION
Begin SQL
  INSERT INTO emp
    (NAME)
    VALUES ('John');
End SQL
ODBC EXECUTE("START")
ODBC CANCEL LOAD
ODBC EXECUTE("INSERT INTO emp (NAME) VALUES ('Smith')")
ODBC CANCEL LOAD
ODBC EXECUTE("COMMIT") `内側のトランザクションを有効にする
CANCEL TRANSACTION `外側のトランザクションをキャンセルする
ODBC LOGOUT
```

- ▼ "John" と "Smith" が emp テーブルに追加されます。

```
ODBC LOGIN(SQL_INTERNAL ;"";"";)"
START TRANSACTION
Begin SQL
  INSERT INTO emp
    (NAME)
    VALUES ('John');
End SQL
```

```

ODBC EXECUTE("START")
ODBC CANCEL LOAD
ODBC EXECUTE("INSERT INTO emp (NAME) VALUES ('Smith')")
ODBC CANCEL LOAD
ODBC EXECUTE("COMMIT") `内側のトランザクションを有効にする
VALIDATE TRANSACTION `外側のトランザクションを有効にする
ODBC LOGOUT

```

SQL DDL

この節では、4D v11 SQL サーバの DDL (Data Definition Language) コマンドの実装と制限について説明しています。DDL コマンドを使用してデータベース・ストラクチャを定義、処理することができます。

主な実装

次の DDL ステートメントが、全体的または部分的に実装されました。

- CREATE TABLE はサポートされます。CREATE TABLE は IF NOT EXISTS の制約を受け入れます：テーブルは、データベースに同じ名前のテーブルが存在しない場合のみ作成できます。そうでない場合、テーブルは作成されず、エラーも生成されません。
- DROP TABLE はサポートされますが、CASCADE コンセプトおよび RESTRICT コンセプトはサポートされません。DROP TABLE は IF EXISTS 制約を受け入れます：データベースにテーブルが存在しない場合、コマンドは何もせず、エラーも生成されません。
- ALTER TABLE はサポートされますが、CASCADE コンセプトおよび RESTRICT コンセプトはサポートされません。

実装されていない DDL ステートメント

次の DDL ステートメントは実装されていません。：

```

CREATE VIEW
DROP VIEW
ALTER VIEW
CREATE ALIAS
DROP ALIAS
CREATE SCHEMA
DROP SCHEMA
CREATE TRIGGER
DROP TRIGGER
ALTER TRIGGER
CREATE RULE
DROP RULE

```

SQL DCL (セキュリティ)

3つのSQLセキュリティ・コンセプト (ユーザ、データベース・オブジェクト、特権) は実装されていません。現在の実装では、4D セキュリティ・コンセプト (ユーザおよびグループ) が使用されています。

SQL ソースへの接続

マルチデータベース・アーキテクチャが、4D SQL サーバのレベルで実装されました。4D 内において、次のことが可能です。

- ODBC LOGIN コマンドを使用して、既存のデータベースに接続できます (詳細は、[ページ 249](#)、"[外部データソース](#)"の節を参照)。
- 4D の新しい **USE EXTERNAL DATABASE** コマンドおよび **USE INTERNAL DATABASE** コマンドを使用して、1つのデータベースから他のデータベースに切り替えられます。

標準のSQLコンセプト、カタログおよびスキーマは使用されていません。

プログラミングにおけるSQLの制限

次の「SQL-92」のステートメントおよびコンセプトは実装されていません。

- 位置決めされた「DELETE」および「UPDATE」のステートメント
- 「SQLCODE」および「SQLSTATE」のステートメント； 4D ランゲージ・メカニズムによってエラー処理が実装されました (ON ERR CALL)。
- 「GET DIAGNOSTICS」ステートメント。
- 「WHENEVER」ステートメント。
- 「NOT FOUND」条件。
- 動的なSQLコンセプト — 代わりに、4D ランゲージが使用されます。
- 「カーソル」および「カーソルをスクロール」のコンセプト
- 外部ストアードプロシージャ・コンセプト。しかし、(ウェブサービスに関するメソッドプロパティの類似メカニズムを使用して)、SQL 環境から4D プロジェクトメソッドを呼び出すことができます。SQL 環境そのものが外部プラグインに属す機能呼び出します。
- 内部ストアードプロシージャ・コンセプト；代わりに、4D のストアードプロシージャ・コンセプトが使用されています。

SQL 関数のサポート

次の表は SQL-2 の関数と演算子をまとめたものであり、4D の内臓 SQL エンジンサポートについて説明しています。また、必要箇所には追加情報も記載しています。

統計関数

SQL 関数	説明	4D SQL	コメント
AVG	平均	yes	カラムのヌル値はカラム機能により無視されます。
COUNT, COUNT(*)	総数	yes	
MAX	最大	yes	
MIN	最小	yes	
SUM	合計	yes	

システム関数

SQL 関数	説明	4D SQL	コメント
CURRENT_USER	カレントユーザ	no	
SESSION_USER	許容ユーザ	no	
SYSTEM_USER	システムユーザ	no	
CURRENT_DATE	カレントの日付	yes	
CURRENT_TIME	カレントの現地時間	yes	
CURRENT_TIMESTAMP	カレントの現地日付と時間	yes	
CURDATE	カレントの日付	yes	SQL-2 に該当なし
CURTIME	カレントの現地時間	yes	SQL-2 に該当なし

標準関数

SQL 関数	説明	4D SQL	コメント
CAST	型変更	yes	
COALESCE	非ヌル値	yes	
NULLIF	ヌル値	yes	
OCTET_LENGTH	長さ (バイト)	yes	

文字列関数

SQL 関数	説明	4D SQL	コメント
	連結	no	
CHAR_LENGTH	文字列の長さ	yes	

CHARACTER_LENGTH	文字列の長さ	no	
LENGTH	文字数	yes	
COLLATE	文字の並べ替え	no	
CONCATENATE	連結	yes	
CONCAT	連結	yes	SQL-2 に該当なし
CONVERT	タイプの変換	no	
LOWER	小文字	yes	
POSITION	Position of a string in another string	yes	
SUBSTRING	文字部分列の抽出	yes	
SUBSTR	文字部分列の抽出	no	
TRANSLATE	文字セットの変換	no	
TRIM	未使用文字を削除する	yes	
LTRIM	先頭の空欄を削除する	yes	SQL-2 に該当なし
RTRIM	末尾の空欄を削除する	yes	SQL-2 に該当なし
UPPER	大文字	yes	
CHAR	ASCII コード文字を返す	yes	SQL-2 に該当なし
LOCATE	文字列中の部分文字位置	yes	SQL-2 に該当なし
REPLACE	文字の置換	yes	SQL-2 に該当なし
SPACE	スペースの生成	yes	SQL-2 に該当なし
LEFT		yes	SQL-2 に該当なし
REPEAT		yes	SQL-2 に該当なし
RIGHT		yes	SQL-2 に該当なし
LIKE	部分的比較	yes	・ "%" は 0 個以上の文字に使用するワイルドカード文字です。 ・ "_" は 1 個の文字のみに使用するワイルドカード文字です。

ビット関数

SQL 関数	説明	4D SQL	コメント
BIT_LENGTH	bit で表した長さ	yes	4D 文字列は Unicode (1 文字 =2 バイト)

数値関数

SQL 関数	説明	4D SQL	コメント
ABS	絶対値	yes	SQL-2 に該当なし
ACOS	ラジアンのアークコサイン	yes	SQL-2 に該当なし
ASCII	ASCII コード	yes	SQL-2 に該当なし
ASIN	ラジアンのアークサイン	yes	SQL-2 に該当なし
ATAN	ラジアンのアークタンジェント	yes	SQL-2 に該当なし
ATAN2	ラジアンの xy アークタンジェント	yes	SQL-2 に該当なし
CEILING	引数以上の最小整数	yes	SQL-2 に該当なし
COS	ラジアンのコサイン	yes	SQL-2 に該当なし
COT	ラジアンのコタンジェント	yes	SQL-2 に該当なし
DEGREES	度数	yes	SQL-2 に該当なし
EXP	指数値	yes	SQL-2 に該当なし
FLOOR	引数以下の最大整数	yes	SQL-2 に該当なし
LOG(float_exp)	自然対数	yes	SQL-2 に該当なし
LOG10	10 を底とする対数	yes	SQL-2 に該当なし
MOD	剰余 (係数)	yes	SQL-2 に該当なし
PI	pi の定数値	yes	SQL-2 に該当なし
POWER	Power 値	yes	SQL-2 に該当なし
RADIANS	変換されたラジアンの数	yes	SQL-2 に該当なし
RAND	ランダム値	yes	SQL-2 に該当なし
ROUND	四捨五入値	yes	SQL-2 に該当なし
SIGN	渡された式の符号	yes	SQL-2 に該当なし
SIN	ラジアンサイン	yes	SQL-2 に該当なし
SQRT	平方根	yes	SQL-2 に該当なし
TAN	ラジアンタンジェント	yes	SQL-2 に該当なし
TRUNC (TRUNCATE)	トランケート	yes	SQL-2 に該当なし
MILLISECOND	ミリ秒で表した日付	yes	SQL-2 に該当なし

時間関数

SQL 関数	説明	4D SQL	コメント
EXTRACT	日付の一部	yes	
INTERVAL (operating)	持続時間	no	
OVERLAPS (Predicate)	重複時間	no	
DAY	日付の日	yes	SQL-2 に該当なし

DAYNAME	曜日	yes	SQL-2 に該当なし
DAYOFMONTH	月の何日目	yes	SQL-2 に該当なし
DAYOFWEEK	週の何日目	yes	SQL-2 に該当なし
DAYOFYEAR	年の何日目	yes	SQL-2 に該当なし
HOUR	時間	yes	SQL-2 に該当なし
MINUTE	日付 / 時間の分	yes	SQL-2 に該当なし
MONTH	日付の月	yes	SQL-2 に該当なし
MONTHNAME	月の名前	yes	SQL-2 に該当なし
QUARTER	15 分	yes	SQL-2 に該当なし
SECOND	秒	yes	SQL-2 に該当なし
WEEK	年の週	yes	SQL-2 に該当なし
YEAR	日付の年	yes	SQL-2 に該当なし

算術演算子

SQL 関数	説明	4D SQL	コメント
+ - * / ()	算術演算子および括弧	yes	2 つの演算子の優先順位は : 「*」 (アスタリスク) および 「/」 (スラッシュ) が、「+」 (プラス) および 「-」 (マイナス) より優先されます。
%	モジュロ演算	no	SQL-2 に該当なし

論理演算子と 2 項演算子

SQL 関数	説明	4D SQL	コメント
IS [NOT] TRUE	TRUE	no	
IS [NOT] FALSE	FALSE	no	
IS [NOT] UNKNOWN	Unknown	no	
IS [NOT] NULL	NULL	yes	TRUE AND NULL = NULL FALSE AND NULL = FALSE
AND	Yes	yes	NULL AND NULL = NULL
OR	Or	yes	TRUE OR NULL = TRUE FALSE OR NULL = NULL
NOT	No	yes	NULL OR NULL = NULL
&	論理 "and"	no	SQL-2 に該当なし
	論理 "or"	no	SQL-2 に該当なし
^	排他論理 "or"	no	SQL-2 に該当なし

比較演算子

SQL 関数	説明	4D SQL	コメント
=	等しい	yes	
>	より大きい	yes	
<	より小さい	yes	
>=	以上	yes	
<=	以下	yes	
<>	異なる	yes	

述語演算子

SQL 関数	説明	4D SQL	コメント
[NOT] IN	メンバーシップ	yes	テスト式がヌル値を返した場合、「IN」テストはヌル値を返します。
[NOT] BETWEEN	区間	yes	3つの NULL 例外は以下のように扱われます： ・範囲を定義する両方の式が NULL 値である場合、BETWEEN テストは NULL を返します。 ・下限を定義する式が NULL 値である場合、BETWEEN テストは、テスト値が上限を超える場合に FALSE を返し、そうでなければ NULL を返します。 ・上限を定義する式が NULL 値である場合、BETWEEN テストは、テスト値が下限未満である場合に FALSE を返し、そうでなければ NULL を返します。
[NOT] EXISTS	存在	yes	
ALL	セットのすべての値の比較	yes	
ANY	セットの少なくとも一つの値との比較	yes	
SOME	セットの少なくとも一つの値との比較	yes	

データ操作

SQL 関数	説明	4D SQL	コメント
INTERSECT	積 (セット)	no	
UNION	和 (セット)	no	

EXCEPT	差 (セット)	no	DIFFERENCE と同じ
DIFFERENCE	差 (セット)	no	EXCEPT と同じ
INNER JOIN	Internal join	yes	
SELECT FROM WHERE		yes	
SELECT INTO		no	
INSERT INTO VALUES		yes	
INSERT INTO SELECT FROM		yes	
DELETE		yes	
UPDATE		yes	
DISTINCT	重複しない値	yes	

データ定義

SQL 関数	説明	4D SQL	コメント
CREATE TABLE		yes	物理的ストレージ定義はサポートされません。
DROP TABLE		yes	「CASCADE」および「RESTRICT」コンセプトはサポートされません。
ALTER TABLE		yes	「CASCADE」および「RESTRICT」コンセプトはサポートされません。
CREATE DATABASE		no	ベータ・リリースには実装されていません。

データ分類

SQL 関数	説明	4D SQL	コメント
ORDER BY	ソート	yes	
ORDER BY DESC		yes	
ORDER BY ASC		yes	
GROUP BY		yes	
GROUP BY HAVING		yes	

Join

SQL 関数	説明	4D SQL	コメント
INNER JOIN	Internal join	yes	
LEFT, RIGHT, FULL OUTER JOIN	External join	no	
NATURAL JOIN	Natural join	no	
UNION JOIN	Union join	no	

LEFT, RIGHT, FULL OUTER NATURAL JOIN	External natural join	no	
--	-----------------------	----	--

条件式

SQL 関数	説明	4D SQL	コメント
CASE	制御フロー	yes	
UNIQUE	重複値なし	no	
MATCH UNIQUE	Matching values	no	
LIMIT	返された行数	yes	<p>SQL-2 に該当なし</p> <pre>SELECT select_list [INTO new_table] FROM table_source [WHERE search_condition] [GROUP BY group_by_expression] [HAVING search_condition] [ORDER BY order_expression [ASC DESC]] [LIMIT numeric-expression] [OFFSET numeric-expression]</pre> <p>Parameters:</p> <p>1 select_list: list of columns 2 new_table: table created 3 table_source: table link to column 4 search_condition: search condition 5 group_by_expression: group by expression 6 search_condition: search condition 7 order_expression: sort condition 8 num-expression: value to limit selection 9 num-expression: value to position selection</p>

サブクエリ

SQL 関数		説明	4D SQL	コメント
SELECT 節	nested		no	
	Correlated		no	
FROM 節	nested		no	
	Correlated		no	

WHERE 節	nested		yes	
	Correlated		yes	
HAVING 節	nested		yes	
	Correlated		yes	
IN 演算子	nested		yes	
	Correlated		yes	
ALL, ANY, SOME 演算子	nested		yes	
	Correlated		yes	
EXISTS 演算子	nested		yes	
	Correlated		yes	

インデックス

SQL 関数	説明	4D SQL	コメント
CREATE INDEX		yes	
DROP INDEX		yes	

トランザクション

SQL 関数	説明	4D SQL	コメント
START [TRANSACTION]		yes	
COMMIT [TRANSACTION]		yes	
ROLLBACK [TRANSACTION]		yes	

SQL エラーコード

SQL エンジン は下記に示すような特定のエラーを返します。これらのエラーには、ON ERR CALL コマンドでインストールされたエラー割り込みメソッドを使用して介入できます。

■ 一般的エラー

1001	INVALID ARGUMENT
1002	INVALID INTERNAL STATE
1003	NOT RUNNING
1004	ACCESS DENIED
1005	FAILED TO LOCK SYNCHRONIZATION PRIMITIVE
1006	FAILED TO UNLOCK SYNCHRONIZATION PRIMITIVE
1007	SQL SERVER IS NOT AVAILABLE

■ セマンティックエラー

1101	TABLE DOES NOT EXIST
1102	COLUMN DOES NOT EXIST
1103	TABLE NOT DECLARED IN FROM CLAUSE
1104	AMBIGUOUS COLUMN NAME
1105	TABLE ALIAS SAME AS TABLE NAME
1106	DUPLICATE TABLE ALIAS
1107	DUPLICATE TABLE IN FROM CLAUSE
1108	INCOMPATIBLE TYPES
1109	INVALID ORDER BY INDEX
1110	WRONG AMOUNT OF PARAMETERS
1111	INCOMPATIBLE PARAMETER TYPE
1112	UNKNOWN FUNCTION
1113	DIVISION BY ZERO
1114	ORDER BY INDEX NOT ALLOWED
1115	DISTINCT NOT ALLOWED
1116	NESTED COLUMN FUNCTIONS NOT ALLOWED
1117	COLUMN FUNCTIONS NOT ALLOWED
1118	CAN NOT MIX COLUMN AND SCALAR OPERATIONS
1119	INVALID GROUP BY INDEX
1120	GROUP BY INDEX NOT ALLOWED
1121	GROUP BY NOT ALLOWED WITH SELECT ALL

1122	NOT A COLUMN EXPRESSION
1123	NOT A GROUPING COLUMN IN AGGREGATE ORDER BY
1124	MIXED LITERAL TYPES IN PREDICATE
1125	LIKE ESCAPE IS NOT ONE CHAR
1126	BAD LIKE ESCAPE CHAR
1127	UNKNOWN ESCAPE SEQUENCE IN LIKE
1128	COLUMNS FROM MORE THAN ONE QUERY IN COLUMN FUNCTION
1129	SCALAR EXPRESSION WITH GROUP BY
1130	SUBQUERY HAS MORE THAN ONE COLUMN
1131	SUBQUERY MUST HAVE ONE ROW
1132	INSERT VALUE COUNT DOES NOT MATCH COLUMN COUNT
1133	DUPLICATE COLUMN IN INSERT
1134	COLUMN DOES NOT ALLOW NULLS
1135	DUPLICATE COLUMN IN UPDATE
1136	TABLE ALREADY EXISTS
1137	DUPLICATE COLUMN IN CREATE TABLE
1138	DUPLICATE COLUMN IN COLUMN LIST
1139	MORE THAN ONE PRIMARY KEY NOT ALLOWED
1140	AMBIGUOUS FOREIGN KEY NAME
1141	COLUMN COUNT MISMATCH IN FOREIGN KEY
1142	COLUMN TYPE MISMATCH IN FOREIGN KEY
1143	FAILED TO FIND MATCHING PRIMARY COLUMN
1144	UPDATE AND DELETE CONSTRAINTS MUST BE THE SAME
1145	FOREIGN KEY DOES NOT EXIST
1146	INVALID LIMIT VALUE IN SELECT
1147	INVALID OFFSET VALUE IN SELECT
1148	PRIMARY KEY DOES NOT EXIST
1149	FAILED TO CREATE FOREIGN KEY
1150	FIELD IS NOT IN PRIMARY KEY
1151	FIELD IS NOT UPDATEABLE
1153	BAD DATA TYPE LENGTH
1154	EXPECTED EXECUTE IMMEDIATE COMMAND
■ 実装	
1203	FUNCTIONALITY IS NOT IMPLEMENTED
1204	FAILED TO CREATE NEW RECORD

1205 FAILED TO UPDATE FIELD
1206 FAILED TO DELETE RECORD
1207 NO MORE JOIN SEEDS POSSIBLE
1208 FAILED TO CREATE TABLE
1209 FAILED TO DROP TABLE
1210 CANT BUILD BTREE FOR ZERO RECORDS
1211 COMMAND COUNT GREATER THAN ALLOWED
1212 FAILED TO CREATE DATABASE
1213 FAILED TO DROP COLUMN
1214 VALUE IS OUT OF BOUNDS
1215 FAILED TO STOP SQL_SERVER
1216 FAILED TO LOCALIZE
1217 FAILED TO LOCK TABLE FOR READING
1218 FAILED TO LOCK TABLE FOR WRITING
1219 TABLE STRUCTURE STAMP CHANGED
1220 FAILED TO LOAD RECORD
1221 FAILED TO LOCK RECORD FOR WRITING
1222 FAILED TO PUT SQL LOCK ON A TABLE

■ 構文解析

1301 PARSING FAILED

■ ランタイムランゲージアクセス

1401 COMMAND NOT SPECIFIED
1402 ALREADY LOGGED IN
1403 SESSION DOES NOT EXIST
1404 UNKNOWN BIND ENTITY
1405 INCOMPATIBLE BIND ENTITIES
1406 REQUEST RESULT NOT AVAILABLE
1407 BINDING LOAD FAILED
1408 COULD NOT RECOVER FROM PREVIOUS ERRORS
1409 NO OPEN STATEMENT
1410 RESULT EOF
1411 BOUND VALUE IS NULL
1412 STATEMENT ALREADY OPENED
1413 FAILED TO GET PARAMETER VALUE
1414 INCOMPATIBLE PARAMETER ENTITIES
1415 PARAMETER VALUE NOT SPECIFIED

- 1416 COLUMN REFERENCE PARAMETERS FROM DIFFERENT TABLES
- 1417 EMPTY STATEMENT
- 1418 FAILED TO UPDATE VARIABLE
- 1419 FAILED TO GET TABLE REFERENCE
- 1420 FAILED TO GET TABLE CONTEXT
- 1421 COLUMNS NOT ALLOWED
- 1422 INVALID COMMAND COUNT
- 1423 INTO CLAUSE NOT ALLOWED
- 1424 EXECUTE IMMEDIATE NOT ALLOWED
- 1425 ARRAY NOT ALLOWED IN EXECUTE IMMEDIATE
- 1426 COLUMN NOT ALLOWED IN EXECUTE IMMEDIATE
- 1427 NESTED BEGIN END SQL NOT ALLOWED
- 1428 RESULT IS NOT A SELECTION
- 1429 INTO ITEM IS NOT A VARIABLE (LANGUAGE RUNTIME)
- 1430 VARIABLE WAS NOT FOUND (LANGUAGE RUNTIME)

■ 日付の解析

- 1501 SEPARATOR_EXPECTED
- 1502 FAILED TO PARSE DAY OF MONTH
- 1503 FAILED TO PARSE MONTH
- 1504 FAILED TO PARSE YEAR
- 1505 FAILED TO PARSE HOUR
- 1506 FAILED TO PARSE MINUTE
- 1507 FAILED TO PARSE SECOND
- 1508 FAILED TO PARSE MILLISECOND
- 1509 INVALID AM PM USAGE
- 1510 FAILED TO PARSE TIME ZONE
- 1511 UNEXPECTED CHARACTER
- 1512 FAILED TO PARSE TIMESTAMP
- 1513 FAILED TO PARSE DURATION

■ 日付のフォーマット

- 1551 FAILED

■ Lexer errors

- 1601 NULL INPUT STRING
- 1602 NON TERMINATED STRING
- 1603 NON TERMINATED COMMENT

1604 INVALID NUMBER
1605 UNKNOWN START OF TOKEN
1606 NON TERMINATED NAME
1607 NO VALID TOKENS

■ 4D エンジン・エラー

1837 DB4D QUERY FAILED

■ キャッシュ

2000 CACHEABLE NOT INITIALIZED
2001 VALUE ALREADY CACHED
2002 CACHED VALUE NOT FOUND

■ プロトコル・エラー

3000 HEADER NOT FOUND
3001 UNKNOWN COMMAND
3002 ALREADY LOGGED IN
3003 NOT LOGGED IN
3004 UNKNOWN OUTPUT MODE
3005 INVALID STATEMENT ID
3006 UNKNOWN DATA TYPE
3007 STILL LOGGED IN
3008 SOCKET READ ERROR
3009 SOCKET WRITE ERROR
3010 BASE64 DECODING ERROR
3011 SESSION TIMEOUT
3012 FETCH TIMESTAMP ALREADY EXISTS
3013 BASE64 ENCODING ERROR
3014 INVALID HEADER TERMINATOR

4D バージョン 11 で用意された幾つかの新しい機能は 4D に組み込まれた Web サーバ機能に関連します：

- Webサーバへの接続認証に"Digest"モードを使用することができるようになりました。
- Webのログ書き出しに異なるログフォーマットがサポートされるようになりました。またスケジュール設定によるログの分割も可能になります。

Digest モードによる認証

概要

4D v11 では、統合された Web サーバに対する接続リクエストの認証方法で、Digest モードをサポートします。認証モードは、ユーザ名やパスワードの収集や処理の方法に関連します。

前のバージョンの 4D では、Basic モードだけがサポートされていました。Basic モードでは、ユーザが入力した名前とパスワードは暗号化されることなく HTTP リクエストにのせて送信されます。第三者が情報を途中で盗聴および利用可能であり、トータルなシステムセキュリティを確保することはできません。

Digest モードはより高いレベルのセキュリティを提供します。認証情報はハッシングと呼ばれる一方向の処理が施され、内容を解読することはできません。

ユーザにとり、どちらの認証モードが使用されるかは透過的です。

しかし Digest 認証は HTTP1.1 の機能であり、すべてのブラウザでサポートされているわけではありません。例えば Microsoft Internet Explorer ではバージョン 5.0 以降だけがこのモードを受け入れることができます。

この機能をサポートしないブラウザが Digest 認証を有効にしている Web サーバにリクエストを送信すると、サーバはリクエストを拒否して、エラーメッセージをブラウザに返します。

4D への実装

互換性のため、バージョン 11 に変換された 4D データベースはデフォルトで Basic 認証が使用されます (前のバージョンで "パスワードを使用" オプションが有効になっていた場合)。Digest モードを使用するためには、明示的にそれを有効にします。

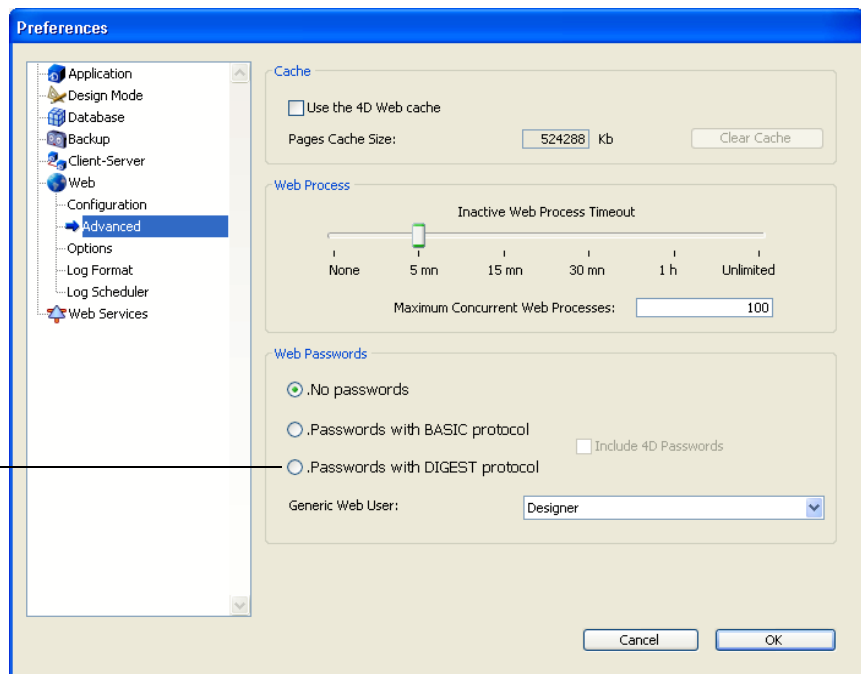
Digest モードを有効にするには以下の処理を行います：

- 4 環境設定の Web/ 詳細ページで新しい "DIGEST 認証のパスワード" オプションをチェックする。
- 5 新しい **Validate Digest Web Password** コマンドを `RqZheDxwkhq0wlfDwRq` データベースメソッドで使用して、接続の受け入れまたは拒否を行う。

環境設定

Digest モードを有効にするには、環境設定の Web/ 詳細ページ中、"DIGEST 認証のパスワード" ラジオボタンを選択します：

DIGEST モードを有効にする設定



4D v11 では、Web 認証に対し 3 つのオプションがあります：

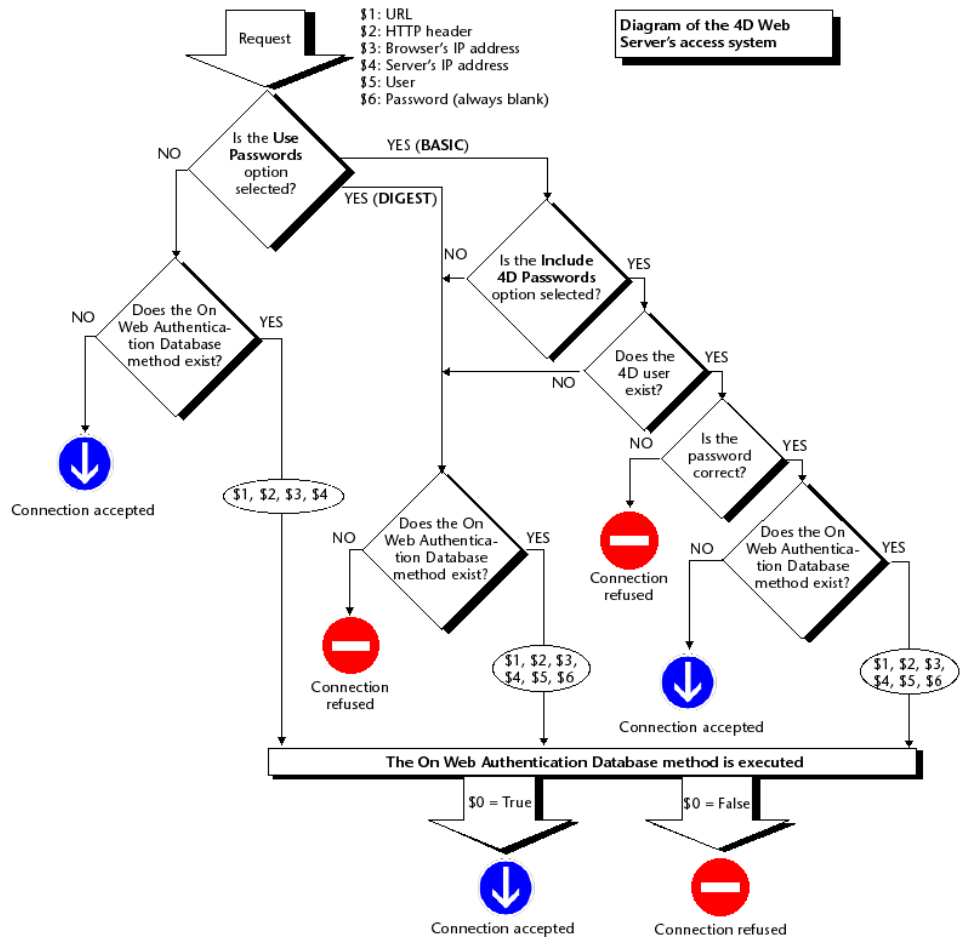
- パスワードなし：Web サーバへの接続に対する認証を行わない（前バージョンの 4D における "パスワードを使用" オプションにチェックを入れていない状態と同じ）。
- BASIC 認証のパスワード：BASIC モードによる標準認証（前バージョンの 4D における "パスワードを使用" オプションにチェックを入れている状態と同じ）。
- DIGEST 認証のパスワード：Digest モードの認証。この新しいモードは 4D v11 で使用可能になりました。このオプションをチェックしたら、新しいコマンド **Validate Digest Web Password** を On Web Authentication データベースメソッドで使用して、接続を管理しなければなりません。

注：これらのオプションを変更した後は、Web サーバを再起動しなければなりません。

Basic モードと違って、Digest モードは 4D の標準パスワードシステムと互換性がありません：4D パスワードを Web ID として使用することはできません。このモードが選択されると、"4D パスワードを含む" オプションは選択できなくなります。

Web ユーザ ID の管理は、（例えばテーブルを使用するなど）カスタマイズされた方法で行わなければなりません。

4D v11 Web サーバのアクセスシステム :



On Web Authentication データベースメソッド

Digest モードが有効にされると、On Web Authentication データベースメソッドの \$6 引数 (パスワード) には空の文字列が渡されます。実際このモードを使用しているとき、この情報はネットワーク上で暗号化されないクリアテキストとしては渡されません。

このため、Digest モード使用時は、新しい **Validate Digest Web Password** コマンドを使用して、接続リクエストを評価する必要があります。この新しいコマンドは [370 ページ "Web サーバ" の節](#) で説明されています。

Web ログの設定 (logweb.txt)

4D v11 では、Web リクエストのログファイル (logweb.txt) を生成するメカニズムの設定を行うことができます。

logweb.txt ファイルは、4D Web サーバに送信されたリクエストを、後で検査するために使用できます。このファイルは自動で以下の場所に置かれます：

- 4D Developer や 4D Server では、ストラクチャファイルと同階層。
- 4D Client または実行可能アプリケーションでは、アプリケーション (Windows) またはソフトウェアパッケージ (Mac OS) と同階層。

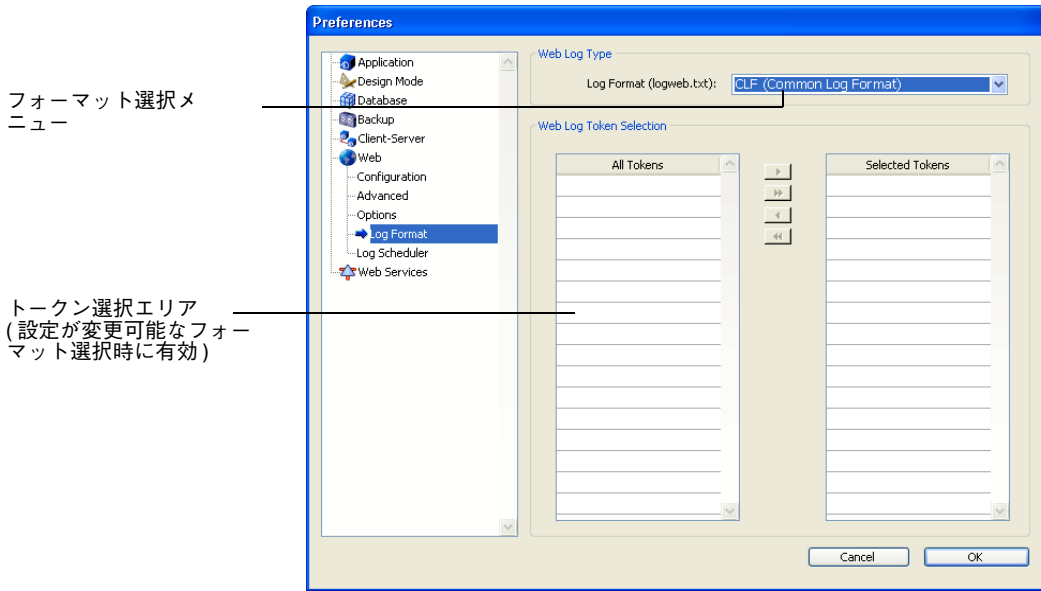
以前のバージョンの 4D では、このファイルへの記録を有効にするか無効にするかの設定のみが可能でした。4D v11 では、その他の設定が可能になります：

- 内部フォーマット
- バックアップ周期

ログフォーマット

以前のバージョンの 4D では、CLF (Common LogFile Format) フォーマットのみを使用して、logweb.txt ファイルに記録を行っていました。このフォーマットは 4D v11 でも使用できます。さらに 3 つの標準フォーマットも使用できるようになりました：DLF (Combined Log Format), ELF (Extended Log format) そして WLF (Webstar Log Format) です。最後の二つは設定を変更できます。

ログファイルの有効化と内容の設定は、環境設定の Web/ ログフォーマット ページで行えます：



注： Web ログファイルは SET DATABASE PARAMETER コマンドを使用して有効・無効を切り替えることもできます。このコマンドは 4D v11 で変更され、新しいファイルフォーマットを扱えるようになりました。詳細は [331 ページ "SET DATABASE PARAMETER, Get database parameter" の節](#) を参照してください。

ログフォーマットメニューでは以下のオプションを選択できます：

- ログファイルなし: このオプションが選択されると、4D は Web ログを生成しません。
以前のバージョンの 4D では、"ファイルにリクエストを保存する (log0 web.txt)" オプションのチェックが外れている場合と同じ設定です。
- CLF (Common Log Format): このオプションが選択されている場合、CLF フォーマットの Web ログが生成されます。このログフォーマットは以前のバージョンの 4D で使用されていたものと同じものです。CLF フォーマットに関する詳細は、4D Language Reference マニュアルを参照してください。
CLF フォーマットはカスタマイズできません。

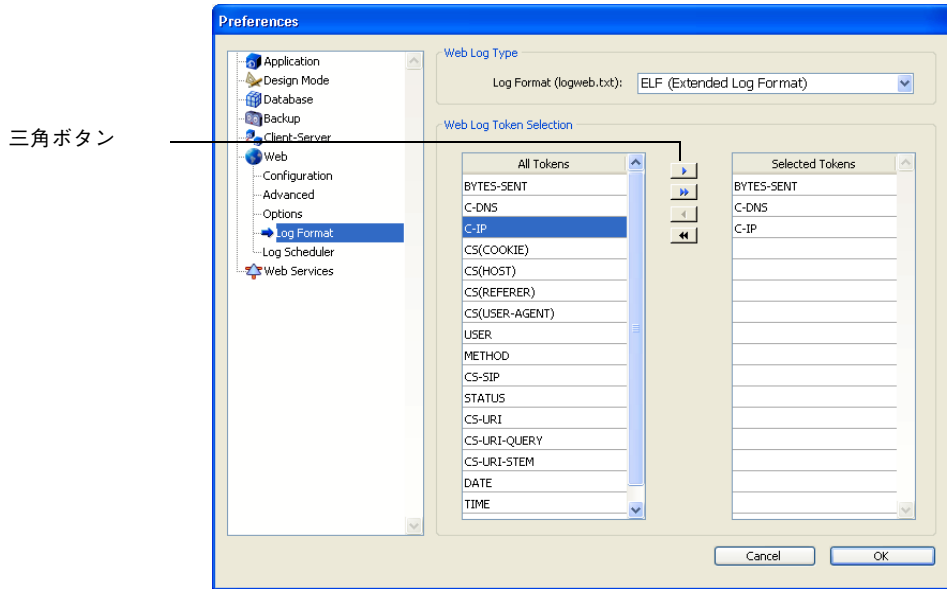
- **DLF (Combined Log Format):** このオプションが選択されている場合、DLF フォーマットで Web ログが生成されます。
DLF フォーマットは CLF フォーマットに似ていて、まったく同じアーキテクチャを持ちます。ただし 2 つの HTTP フィールド、`referer` と `user-agent` が追加されています。:
 - `referer`: リクエストされたドキュメントのリンク元ページの URL 情報。
 - `user-agent` : リクエストの発行元となったブラウザやソフトウェアの名前やバージョン情報。

DLF フォーマットはカスタマイズできません。
- **ELF (Extended Log Format):** このオプションが選択されている場合、ELF フォーマットで Web ログが生成されます。
ELF フォーマットは HTTP ブラウザの世界で広く使用されていて、特定の用途に合わせて洗練されたログを生成することができます。このため、ELF フォーマットはカスタマイズ可能です: ログに記録するフィールドやその順番を選択できます。
- **WLF (WebStar Log Format):** このオプションが選択されている場合、WLF フォーマットで Web ログが生成されます。
WLF フォーマットは WebSTAR サーバのために特別に開発されました。幾つか追加のフィールドがあることを除き、ELF フォーマットに似ています。ELF フォーマットと同様、このフォーマットもカスタマイズ可能です。

フィールドの設定

ELF (Extended Log Format) や WLF (WebStar Log Format) フォーマットを選択すると、"Web ログトークン" エリアに、選択されたフォーマットで利用可能なフィールドが表示されます。ここでログに含めるフィールドを選択します。

三角ボタンやドラッグアンドドロップで、"選択されたトークン" エリアに使用するフィールドを移動します：



注： 同じフィールドを2回選択することはできません。

次の表はそれぞれのフォーマットで利用可能なフィールドを示しています。またその内容を説明します：

フィールド	EL F	WL F	値
BYTES_RECEIVED		<input type="radio"/>	サーバが受信したバイト数
BYTES_SENT	<input type="radio"/>	<input type="radio"/>	サーバが送信したバイト数
C_DNS	<input type="radio"/>	<input type="radio"/>	DNS の IP アドレス (ELF: フィールドは C_IP フィールドと同じ)
C_IP	<input type="radio"/>	<input type="radio"/>	ユーザの IP アドレス (例: 192.100.100.10)
CONNECTION_ID		<input type="radio"/>	Connection ID 番号

フィールド	EL F	WL F	値
CS(COOKIE)	<input type="radio"/>	<input type="radio"/>	HTTP に含まれる cookie に関する情報
CS(HOST)	<input type="radio"/>	<input type="radio"/>	HTTP リクエストの Host フィールド
CS(REFERER)	<input type="radio"/>	<input type="radio"/>	リクエストされたドキュメントの参照元 URL
CS(USER_AGENT)	<input type="radio"/>	<input type="radio"/>	ユーザのソフトウェア及び OS の情報
CS_SIP	<input type="radio"/>	<input type="radio"/>	サーバの IP アドレス
CS_URI	<input type="radio"/>	<input type="radio"/>	完全なリクエスト URI
CS_URI_QUERY	<input type="radio"/>	<input type="radio"/>	リクエストクエリパラメタ
CS_URI_STEM	<input type="radio"/>	<input type="radio"/>	リクエストからクエリパラメタを除いた部分
DATE	<input type="radio"/>	<input type="radio"/>	DD: 日, MMM: 月 (Jan, Feb, 等), YYYY: 年
METHOD	<input type="radio"/>	<input type="radio"/>	サーバへのリクエストに使用された HTTP メソッド
PATH_ARGS		<input type="radio"/>	CGI パラメタ: "\$" 文字の後に続く文字列
STATUS	<input type="radio"/>	<input type="radio"/>	サーバからの返信
TIME	<input type="radio"/>	<input type="radio"/>	HH: 時, MM: 分, SS: 秒
TRANSFER_TIME	<input type="radio"/>	<input type="radio"/>	サーバがレスポンスを生成するのに要した時間
USER	<input type="radio"/>	<input type="radio"/>	認証された場合のユーザ名; そうでなければ - (マイナスサイン)。ユーザ名に含まれるスペースは _ (アンダーライン) に置き換え
URL		<input type="radio"/>	ユーザがリクエストした URL

注: DATE と TIME は GMT で記録されます。

バックアップの周期

Web ログファイルは非常に大きなサイズになるので、自動アーカイブを行うように設定できます。バックアップの実行は特定の時間経過後 (時間、日、週または月単位)、またはファイルサイズで指定でき、設定されたタイミングで 4D は自動的にカレントログファイルを閉じてアーカイブを行い、新しいログファイルを作成します。

アーカイブ名

Web ログファイルのバックアップが実行されると、バックアップファイルは "Logweb Archives" フォルダに置かれます。このフォルダは logweb.txt ファイルと同階層 (つまりデータベースストラクチャと同階層) に作成されます。

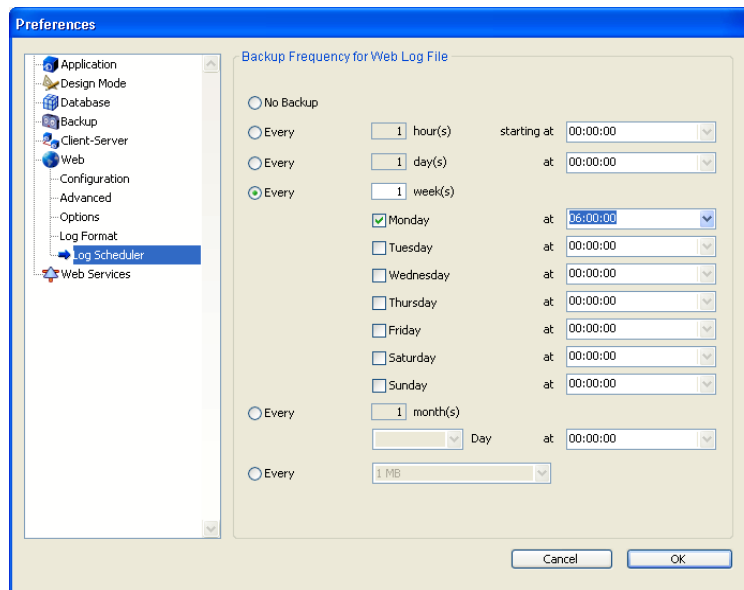
アーカイブされたファイルには以下のような名前がつけられます:

"DYYYY_MM_DD_Thh_mm_ss.txt"。例えば2006年9月4日午後3:50:07にアーカイブされたファイルには以下の名前がつけられます :

"D2006_09_04_T15_50_07.txt"

バックアップ設定項目

Web ログのバックアップ設定は環境設定の Web/ ログスケジューラ ページで行うことができます:



まずラジオボタンをクリックして周期 (日, 週, その他) で設定するか、ファイルサイズで設定するか、選択する必要があります。その後必要に応じて、バックアップを実行するタイミングを決定する特定の条件を指定します。

- バックアップしない: バックアップスケジュール機能を無効にします。
- X時間ごと: 時間単位でバックアップ周期を設定するには、このオプションを選択します。1 から 24 の値が指定可能です。
- 開始時刻: 最初のバックアップ時刻を設定するために使用します。

- **X日ごと**: このオプションは日数を指定してバックアップを行う場合に使用します。毎日バックアップを行うには1を指定します。このオプションを使用する場合、何時にバックアップを開始するか指定しなければなりません。
 - **X週ごと, 曜日 X**: このオプションは曜日を指定してバックアップを行う際に使用します。毎週バックアップを行うには1を指定します。このオプションを使用する場合、バックアップを行う曜日と時刻を指定しなければなりません。必要であれば複数の曜日を指定できます。例えば週に2回のバックアップを取るように設定できます。
 - **X月ごと X日 X**: このオプションは月毎にバックアップを行う際に使用します。毎月バックアップを行うには1を入力します。このオプションを選択する時は、バックアップを開始する日と時刻を指定しなければなりません。
 - **XMB**: このオプションは、カレントログファイルのサイズを元にバックアップを行う際に使用します。バックアップは、ファイルサイズが設定値になると自動で開始されます。サイズは1, 10, 100, 1000 MBから設定できます。
- 注: スケジュールバックアップを使用する場合、バックアップ開始時刻にWebサーバが開始されていない場合は、次回起動時に4Dはバックアップが失敗したものとして扱い、データベースの環境設定に設定された内容を適用します。

4D バージョン 11 のランゲージは、多数の新しいコマンドと定数によって充実し、またいくつかの既存コマンドも拡張されました。

- 4D ランゲージでの Unicode のサポートは、[ページ 247](#)、"[Unicode 関連の変更](#)" の節に詳細のとおり変更されました。
- "[外部データソース](#)" テーマのコマンドは、4D に内臓された新しい SQL エンジンをサポートするため変更されました。
- 新しい "[SQL](#)" テーマは、4D の SQL エンジンおよび統合 SQL サーバの有効化に関連したコマンドを1つにまとめグループ化しました。
- "[クエリ](#)" テーマのコマンドは、新しい 4D エンジンおよびキーワード検索の機能を活用するため変更されました。
- 新しいプロジェクトフォームのサポートに関する詳細は、[ページ 262](#)、"[プロジェクトフォーム](#)" の節で説明しています。
- 4D v11 ではドラッグ & ドロップ管理が改善、拡張されました。ドラッグ & ドロップは、"[ドラッグ&ドロップ](#)" テーマのコマンドおよび "[ペーストボード](#)" テーマ (以前の "[クリップボード](#)" テーマ) のコマンド両方によってサポートされるようになりました。
- 4D v11 ではメニュー管理が完全に再編成されました。例えば、階層メニューを使用してメニューバーを作成することもできます。結果として、"[メニュー](#)" テーマは新しいコマンドおよび変更された既存コマンドの両方を含みます。
- 階層リストにも重要な変更が加えられました。従って、"[階層リスト](#)" テーマは新しいコマンドおよび変更された既存コマンドを含みます。また、既存のインタフェースコマンドのいくつかが階層リストと連動できるようになった点に注意してください ([ページ 301](#)、"[階層リストに使用可能な既存のコマンド](#)" の節を参照)。
- リストボックスが新しい機能 (例えば、レコードのセレクションに基づく内容) の恩恵を受けるようになったため、"[リストボックス](#)" テーマは新しいコマンドを含みます。また、いくつかの既存コマンドが変更されました。
- Mac OS では [Open window](#) コマンドを使用して "[メタルルック](#)" のウィンドウを作成できるようになりました。

- "プリント" テーマに、プリント作業にコントロールを提供する2つのコマンド、[OPEN PRINTING JOB](#)および[CLOSE PRINTING JOB](#)が追加されました。
- "4D 環境" テーマに、4D データベースファイルの保守およびコンポーネントに関連する新しいユティリティ・コマンドが追加されました。また、様々な既存コマンドが変更されました。
- "システム環境" テーマに、新しい [Select RGB Color](#) コマンドおよび [GET SYSTEM FORMAT](#) コマンドが追加され、[PLATFORM PROPERTIES](#) コマンドが変更されました。また、Windows でもスクリーン管理のコマンドが使用できるようになりました。
- "ユーザインターフェース" テーマに新しいコマンドが追加され、いくつかの既存コマンドが変更されました。
- [Select folder](#) コマンド("システムドキュメント" テーマ)は、追加の引数を受け入れます。
- "ピクチャ" テーマは次にあげる新しいピクチャ管理コマンドを含みます: [PICTURE CODEC LIST](#)、[TRANSFORM PICTURE](#)、[COMBINE PICTURES](#) および [CONVERT PICTURE](#) です。
- "グラフ" テーマの二つのコマンドは、4D に統合された SVG 描画エンジンを使用するために追加されました。
- "文字列" テーマには、Unicode のサポートに関する様々な変更が加えられました。また、このテーマは、XLIFF テクノロジー使用する新しい [Get localized string](#) や、[Match regex](#) による正規表現コマンドを含みます。
- "BLOB" テーマの [TEXT TO BLOB](#) コマンドと [BLOB to text](#) コマンド、および関連する定数は、Unicode テキストをサポートするために変更されました。
- "ランゲージ" テーマに "[EXECUTE METHOD](#)" コマンドが追加されました。
- "リソース" テーマの2つのコマンドが、XLIFF アーキテクチャを使用できるよう変更されました。
- "通信" テーマの [SEND PACKET](#) コマンドおよび [RECEIVE PACKET](#) コマンドは引数としてBLOBを受け入れるようになり、[USE CHARACTER SET](#) コマンドは変更されました。
- "ストラクチャアクセス" テーマには様々な変更が加えられました: [Count fields](#) コマンドおよび [Count tables](#) コマンドは名前が変更され、[Is table number valid](#) コマンドおよび [Is field number valid](#) コマンドが追加されました。さらに、このテーマにはインデックス管理に関する新しい機能が追加されました。
- "Web サーバ" テーマに新しい [Validate Digest Web Password](#) コマンドが追加され、[SET HTML ROOT](#) コマンドは変更されました。

- "ツール" テーマは、新しいユーティリティコマンド (**Choose**) やマクロの管理に関するコマンドの追加によって改善されました。さらに、このテーマは以前は他のテーマにあったコマンドを1つにまとめグループ化しました。
- "トランザクション" テーマに、トランザクションの入れ子を管理するために使用する新しい **Transaction level** コマンドが追加されました。
- "XML" テーマのコマンドいくつかが変更され、DOMコマンドが追加されました。
- "Web サービス (クライアント)" テーマの **CALL WEB SERVICE** コマンドが最適化されました。
- 最後に、いくつかのコマンドやコマンドテーマに変更が行われました: この情報は [ページ 382](#)、"[その他の変更](#)" の節にまとめられています。

Unicode 関連の変更

4D v11 は Unicode 文字セット ([ページ 60](#)、"[Unicode のサポート](#)" の節を参照) に対する拡張サポートを提供します。特に、プログラミングランゲージはこの文字セットを式の処理および保存に使用できるようになりました。

この新しい特性によって、特に文字列に関するコマンドなど、特定のコマンドレベルにおける変更が生じました。この節では、データベースにおいて Unicode モードが有効化された時のこれらの変更点について説明しています。

注 環境設定の新しいオプションを使用して Unicode モードを無効にできません ([ページ 61](#)、"[4D データベースにおける Unicode の互換性](#)" の節を参照)。

変数の宣言と型

Unicode モードでは、変数を宣言するコマンドが変更されました。

- `C_TEXT` および `ARRAY TEXT` は UTF-16 を使用して変数と配列を宣言します。
- `C_STRING` および `ARRAY STRING` は、それぞれ `C_TEXT` および `ARRAY TEXT` と同様に機能します。文字数引数は無視されます。次のコードで具体的に説明すると:

```
C_STRING(1:myvar)
myvar:="abc"
thesize:=Length(myvar)
```

- ・ 互換性モード (以前のバージョン) では `thesize = 1` となります。
- ・ バージョン 11 の Unicode モードでは `thesize = 3` となります。

従って、4D v11 はデータベースの実行モードによって異なるタイプの文

字列配列と変数を返すことになります。次の表は、関数 `Type(value)` を使用した時の表記の違いについてまとめています。:

	互換モードでの <code>Type(x)</code>	Unicode モード (v11) での <code>Type(x)</code>
<code>C_STRING(x;10)</code>	<u>Is String Var</u> (24)	<u>Is Text</u> (2)
<code>C_TEXT(x)</code>	<u>Is Text</u> (2)	
<code>ARRAY STRING(10;x;1)</code>	<u>String array</u> (21)	<u>Text array</u> (18)
<code>ARRAY TEXT(x;1)</code>	<u>Text array</u> (18)	
<code>[Table]String_Field</code>	<u>Is Alpha Field</u> (0)	
<code>[Table]Text_Field</code>	<u>Is Text</u> (2)	

Get database parameter コマンドを使用すると、プログラミングを通してカレントのデータベース実行モードを確認できます ([ページ 331](#)、"[SET DATA0 BASE PARAMETER](#), [Get database parameter](#)" の節を参照)。

注 Unicode ではセマフォ名がコード化されており、そのサイズは最大 255 文字に制限されています (4D の前バージョンでは最大 30 文字に制限されていました)。

文字列テーマのコマンド

Unicode モードでは:

Char コマンドは、ASCII コード (0 ~ 255) ではなく、UTF-16 の値 (1 ~ 65535 に含まれる) を要求します。

- Ascii コマンドの名前は [Character code](#) に変更されました。これは、以前のように ASCII コードではなく、Unicode 文字の先頭の UTF-16 値を返します。
- 文字列を処理するコマンド (Length, Replace string, Position, Delete string, Substring, Change string) は、以前のように文字列のバイトではなく、UTF-16 文字に基づいて動作します。
- シンタックス `mystring[x]` を使用した文字列の x 番目の文字へのアクセスは、以前のように x 番目のバイトではなく、x 番目の UTF-16 文字を返します。
- 文字列変換コマンド (Mac to Win, Win to Mac, Mac to ISO、および ISO to Mac) は機能しなくなりました。これらのコマンドは他の文字セットを使用するテキストを表記するために使われていました。しかし、Unicode モードでは、すべてのテキスト変数が Unicode 文字セットで表記されており、その変更はできません。結果として、これらのコマンドは渡した文字列と同一の文字列を返します。文字列の変換には新しいコマンドを使用できます (下記を参照)。

- 様々な文字セット間での文字列変換を可能にするために 2 つの新しいコマンド、即ち [CONVERT FROM TEXT](#) および [Convert to text](#) が追加されました。これらのコマンドについては、[ページ 347](#)、"[文字列](#)" の節で説明されています。

これらの新しいメカニズムは、文字が 1 バイトにコード化されたランゲージ (西欧のランゲージ) にとって特定の変更は生じさせるものではありませんが、文字が 2 バイトにコード化されたランゲージ (例えば、日本語) の処理を単純化することになります。

Unicode では、次の文字コードは予約されており、テキストに挿入してはいけません。

0
65534 (FFFE)
65535 (FFFF)

BLOB テーマのコマンド

互換性の目的から、BLOB to text コマンドおよび TEXT TO BLOB コマンドは、従来のように ASCII (Mac Roman) で表記されたテキストを受け入れますが、これらは Unicode モードでは Unicode 表記のテキストもサポートします。両方の使用を可能にするために新しい定数が追加され、既存の定数は名前が変更されました。詳細は、[ページ 356](#)、"[BLOB](#)" の節を参照してください。

通信テーマのコマンド

デフォルトで、Unicode モードではこのテーマのコマンドは UTF-8 文字セットで機能します。[USE CHARACTER SET](#) コマンドは IANA 名をキャラクター・マップとして受け取るようになりました。

注 [互換モード](#)では、テキストは従来のまま 32 KB に制限されています。

外部データソース

4D v11 はネイティブの SQL エンジを搭載しています ([ページ 199](#)、"[4D SQL エンジンの使用](#)" の章を参照)。
バージョン 2004 の 4D で追加された "外部データソース" テーマには、SQL を使用して ODBC 互換性のデータソースにアクセスするためのハイレベルなコマンドが備わっています。バージョン 11 では、4D 内部 SQL エンジンに対してこれらのコマンドが使えるよう、いくつかのコマンドが変更され、このために新しい定数 `.SQL_INTERNAL` が追加されました。さらに、新しいコマンドにより、マシンにある外部 SQL データソースに直接アクセスし、メソッドエディタからクエリを実行できるようになりました。[ページ 251](#)、"[SQL データソースへの直接接続](#)" の節を参照してください。

ODBC LOGIN

ODBC LOGIN {(dataEntry;user;password)}

ODBC LOGIN コマンドを使用して、4D 内部 SQL エンジンとの接続を開けるようになります。

このタイプの接続を開くには、SQL_INTERNAL 定数を dataEntry 引数に渡します。

Begin SQL/End SQL タグ (ページ 127、"メソッドエディタへの SQL コード入力" の節を参照)、または QUERY BY SQL コマンドを使って 4D SQL 環境にアクセスしようとする場合、このコマンドを使って接続を開く必要はありません。一方、4D の他の ODBC コマンドを使うためには、接続を初期化しなければなりません。

接続を終了するには、ODBC LOGOUT コマンドを使ってそれを閉じます。

- ▼ 次は 4D 内部 SQL エンジンとの接続を初期化する例です。

```
ODBC LOGIN(SQL_INTERNAL;$user;$password)
```

ODBC SET OPTION

ODBC SET OPTION (option; value)

4D 内部 SQL エンジンを使用する場合、ODBC Asynchronous オプションは不要です。実際、このタイプの接続は常に同期です。

ODBC EXECUTE

ODBC EXECUTE (sqlStatement{; boundObj{; boundObj2;...;boundObjN})

4D 内部 SQL エンジンに対する ODBC EXECUTE コマンドは、外部エンジンの場合と同じ動作をします。しかしながら、次の詳細には留意してください。

4D 式を参照する

有効な 4D 式 (変数、フィールド、配列、式など) は、どのタイプでも sqlStatement 引数で参照できます。SQL クエリ内で 4D 参照を指定するには、下記のいずれかの記述を使用します。

- 参照を 2つの大なり記号と 2つの小なり記号("<<" と ">>") の間に挿入します。または、

- 参照の前に ":" (コロン) を付けます。

例えば、

```
C_ALPHA(80;vName)
vName:=Request("Name:")
ODBC EXECUTE("SELECT age FROM PEOPLE WHERE
name=<<vName>>")
```

上と下の記述は全く同等の意味を表します。

```
C_ALPHA(80;vName)
vName:=Request("Name:")
ODBC EXECUTE("SELECT age FROM PEOPLE WHERE name = :vName")
```

この記述は最適化されています: 4D ループ内でリクエストを実行すると、参照のみがサーバの各グループで計算され、リクエストは初回のみ評価されます。

4D で値を取得する

4D では、SQL クエリの結果生じる値は、次の 2 つの方法で取得できません。

- ODBC EXECUTE コマンドの追加の引数を使う方法 (4D が推奨するソリューション)。

ODBC EXECUTE("SELECT ename FROM emp";[Employees]Name)

- SQL クエリそのものの中で INTO 節を使う方法 (特定ケースの場合の代替ソリューション)。

ODBC EXECUTE("SELECT ename FROM emp INTO ;[Employees]Name")

4D SQL サーバとの相互作用については、[ページ 200](#)、"[4D と SQL エンジン間のデータ送信](#)" の節を参照してください。

ODBC IMPORT、ODBC EXPORT

4D 内部 SQL エンジンに接続している場合、これらのコマンドは使用できません。

SQL

この新しいテーマは、4D v11 の新しい SQL エンジンを開始、使用するための特定のコマンドを 1 つにまとめグループ化しました。

SQL データソースへの直接接続

4D v11 では、ランゲージを通して外部 ODBC データソースに直接接続し、Begin SQL/End SQL タグ構造の中で SQL クエリを実行できます。これは次のように機能します: 新しい [GET DATA SOURCE LIST](#) コマンドを使って、マシンに存在するデータソースのリストを取得できます。次に [USE EXTERNAL DATABASE](#) コマンドを実行して、SQL パススルークエリに使用するソースを指定できます。それから、Begin SQL/End SQL タグ構造を使ってカレントソースに SQL クエリを実行します。メソッドエディタの SQL コマンドについては、[ページ 127](#)、"[メソッドエディタへの SQL コード入力](#)" の節を参照してください。

GET DATA SOURCE LIST

GET DATA SOURCE LIST(sourceType; sourceNamesArray; driversArray)

引数	タイプ	説明
sourceType	倍長整数	→ ソースのタイプ: ユーザまたはシステム
sourceNamesArray	テキスト配列	← データソース名の配列
driversArray	テキスト配列	← ソース用ドライバの配列

[GET DATA SOURCE LIST](#) コマンドは、sourceNamesArray 配列と driver0sArray 配列に、オペレーティングシステムの ODBC マネージャで定義さ

れた sourceType タイプ・データソースの名前とドライバを返します。sourceType では、取得するデータソースのタイプを渡します。"External Data Source" テーマにある次の定数のいずれかを使うことができます。:

定数	タイプ	値
User Data Source	倍長整数	1
System Data Source	倍長整数	2

注 このコマンドはファイルタイプのデータソースを考慮しません。

コマンドは sourceNamesArray 配列と driversArray 配列に対応する結果を返します。配列のサイズは調整されます。コマンドが正しく実行されると、OK システム変数は 1 に設定されます。そうでない場合は 0 に設定され、エラーが生成されます。

▼ ユーザデータソースを使用した例:

```
ARRAY TEXT(arrDSN;0)
ARRAY TEXT(arrDSNDrivers;0)
GET DATA SOURCE LIST(User Data Source;arrDSN;arrDSNDrivers)
```

参照: [USE EXTERNAL DATABASE](#)

USE EXTERNAL DATABASE

USE EXTERNAL DATABASE (sourceName{;user{;password{}}

引数	タイプ	説明
sourceName	文字列	→ 接続する ODBC データソースの名前
user	文字列	→ ユーザ名
password	文字列	→ ユーザパスワード

[USE EXTERNAL DATABASE](#) コマンドは、4D アプリケーションと、sourceName 引数で指定したデータソースの間の接続を開きます。

注 [GET DATA SOURCE LIST](#) コマンドを使って、マシンに存在する有効なデータソースのリストを取得できます。

接続が開かれると、カレントのプロセスの Begin SQL/End SQL の構造内で実行されたすべての SQL ステートメントは、[USE INTERNAL DATABASE](#) コマンドまたは他の [USE EXTERNAL DATABASE](#) ステートメントが実行されるまで、この外部ソース (SQL パススルー) に送信されます。データソースが要求する任意の ID 情報を user 引数および password 引数に渡します。コマンドが正しく実行されると、OK システム変数は 1 に設定されます。そうでない場合は 0 に設定され、エラーが生成されます。

▼ 例

```
C_TEXT(sourceNamesArray;sourceDriversArray;0)
```

```

GET DATA SOURCE LIST(1;sourceNamesArray;sourceDriversArray) `
ユーザデータソース
If (Find in array(sourceNamesArray;"emp")#-1)
    `emp ソースが存在すれば
    USE EXTERNAL DATABASE("emp";"tiger";"scott")
    Begin SQL
    ... `SQL 文
    End SQL
End if

```

参照：[GET DATA SOURCE LIST](#), [USE INTERNAL DATABASE](#)

USE INTERNAL DATABASE

USE INTERNAL DATABASE

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません。

[USE INTERNAL DATABASE](#) コマンドを使用して、[USE EXTERNAL DATABASE](#) コマンドを使って開いた任意の外部接続を閉じ、アプリケーションをローカル 4D データベースに再接続することができます。その場合、Begin SQL/End SQL の構造内で実行された SQL ステートメントは、どれもローカル 4D データベースに送られます。

[USE EXTERNAL DATABASE](#) コマンドを使って開いた外部接続がない場合、このコマンドは何もしません。

コマンドが正しく実行されると、OK システム変数は 1 に設定されます。そうでない場合は 0 に設定され、エラーが生成されます。

参照：[USE EXTERNAL DATABASE](#)

Get current data source

Get current data source → String

引数	タイプ	説明
返り値	文字列	← 使用中のカレントデータソース名

[Get current data source](#) コマンドは、アプリケーションのカレントデータソース名を返します。カレントデータソースは、Begin SQL/End SQL の構造内で実行された SQL クエリを受け取ります。

カレント・データソースがローカル 4D データベースの場合、コマンドは SQL_INTERNAL 定数の値に対応する文字列 ";DB4D_SQL_LOCAL;" を返します。

このコマンドによって、例えば SQL クエリの実行前などに、カレントデータソースを確認できます。

参照：[USE EXTERNAL DATABASE](#), [USE INTERNAL DATABASE](#), [GET DATA SOURCE LIST](#)

QUERY BY SQL

QUERY BY SQL({table}; {sqlFormula})

引数	タイプ	説明
table	テーブル	→ レコードのセレクションを返すテーブル、またはこの引数が省略された場合デフォルトテーブル
sqlFormula	文字列	→ SELECT クエリの WHERE 節を示す有効な SQL 検索フォーミュラ

4D v11 には、ネイティブの SQL エンジンが搭載されています (ページ 199、"4D SQL エンジンの使用" の章を参照)。新しい **QUERY BY SQL** コマンドは、このエンジンの性能を活用する方法の 1 つです。

他の 2 つの方法は、ODBC コマンド (ページ 249、"外部データソース" の節を参照) およびメソッドエディタでの Begin SQL/End SQL タグの使用 (ページ 127、"メソッドエディタへの SQL コード入力" の節を参照) です。

QUERY BY SQL コマンドを使って、次のような単純な SELECT クエリを実行できます。

```
SELECT *
FROM table
WHERE <sqlFormula>
```

table は最初の引数に渡されたテーブル名で、sqlFormula は WHERE 節のクエリ文字列です。

例えば、次のステートメントは、

```
QUERY BY SQL([Employees];"name='smith'")
```

次の SQL クエリに相当します。

```
SELECT * FROM Employees WHERE "name='smith'"
```

QUERY BY SQL は **QUERY BY FORMULA** コマンドにとっても良く似ています。**QUERY BY SQL** は特定のテーブルでレコードを検索します。これは、カレントプロセスのカレントセレクションを変更し、新しいセレクションの最初のレコードをカレントレコードにします。

QUERY BY SQL は、sqlFormula をテーブルセレクションの各レコードに適用します。sqlFormula は、必ず **TRUE** または **FALSE** のいずれかに評価されるブール式です。ご存知の通り、SQL-2 の基準では、検索条件は **TRUE**、**FALSE** または **NULL** の結果をもたらします。新しいカレントセレクションでは、検索条件が **TRUE** の結果をもたらすレコード (行) のみがクエリ結果として含まれます。

sqlFormula 式は、例えばフィールド (カラム) と値を比較する時は単純で、計算を行う時は複雑になります。**QUERY BY FORMULA** のように、**QUERY BY SQL** はリレートされたテーブルで情報を評価できます (例 4 を参照)。sqlFormula は、SQL-2 の基準および現在の 4D SQL 実装における制限に適合する有効な SQL ステートメントであれば、何でも構いません。4D SQL 実装の詳細は、ページ 199、"4D SQL エンジンの使用" の章を参

照してください。

sqlFormula 引数は 4D 式の参照を使用できます。使用するシンタックスは ODBC コマンドまたは Begin SQL/End SQL タグの場合と同じです。

<<MyVar>> または :MyVar

詳細は、[ページ 200](#)、"[4D と SQL エンジン間のデータ送信](#)" の節を参照してください。

検索条件がうまくフォーマットされると、OK システム変数は 1 に設定されます。そうでない場合は 0 に設定され、コマンドの結果は空のセレクションとなり、ON ERR CALL コマンドによりインストールされたエラー処理メソッドでトラップ可能なエラーが返されます。

注 このコマンドは、SET QUERY LIMIT コマンドおよび SET QUERY DESTINATION コマンドと互換性があります。

リレーションについて

QUERY BY SQL は、4D ストラクチャエディタで定義されたテーブル間のリレーションを使用しません。この目的では、join 節を追加する必要があります。

[PEOPLE] City から [CITIES]Name の N 対 1 リレーションを持つ次のようなストラクチャがあると仮定します。

```
[PEOPLE]
  Name
  City
[CITIES]
  Name
  Population
```

QUERY BY FORMULA を使用する場合、次のように記述できます。

```
QUERY BY FORMULA([PEOPLE]; [CITIES]Population>1000)
```

QUERY BY SQL を使用する場合は、リレーションの有無を問わず、次のステートメントを使う必要があります。

```
QUERY BY SQL([PEOPLE];"people.city=cities.name AND
cities.population>1000")
```

注 QUERY BY SQL は、1 対 N リレーションと N 対 1 リレーションを QUERY BY FORMULA と異なる方法で処理します。

- ▼ この例は、売上が 100 を超えるオフィスを表します。SQL クエリは次の通りです。

```
SELECT *
FROM OFFICES
WHERE SALES > 100
```

QUERY BY SQL コマンドを使用する場合は次の通りです。

```
C_STRING(30;$queryFormula)
$queryFormula:="SALES > 100"
QUERY BY SQL([OFFICES];$queryFormula)
```

- ▼ この例は、3000～4000の範囲のオーダーを表します。
SQL クエリは次の通りです。

```
SELECT *
  FROM ORDERS
  WHERE AMOUNT BETWEEN 3000 AND 4000
QUERY BY SQL コマンドを使用する場合は次の通りです。
```

```
C_STRING(40;$queryFormula)
$queryFormula="AMOUNT BETWEEN 3000 AND 4000"
QUERY BY SQL([ORDERS];$queryFormula)
```

- ▼ 次の例は、特定の基準で並び替えられたクエリ結果を得る方法を表しています。
SQL クエリは次の通りです。

```
SELECT *
  FROM PEOPLE
  WHERE CITY =PParis*
  ORDER BY NAME
QUERY BY SQL コマンドを使用する場合は次の通りです。
```

```
C_STRING(40;$queryFormula)
$queryFormula="CITY =PParis*ORDER BY NAME"
QUERY BY SQL([PEOPLE];$queryFormula)
```

- ▼ この例は、4Dのリレートされたテーブルを使用したクエリを表しています。SQLでは、この4DリレーションをシミュレートするためにJOINを使う必要があります。

4Dに次の2つのテーブルがあると仮定します。

[INVOICES]

ID_INV: Long integer
DATE_INV: Date
AMOUNT: Real

[LINES_INVOICES]

ID_LIN: Long integer
ID_INV: Long integer
CODE: Alpha (10)

[LINES_INVOICES]ID_INV から [INVOICES]ID_INV の N 対 1 リレーションがあります。

QUERY BY FORMULA コマンドを使う場合は、次のように記述できます。

```
QUERY BY FORMULA([LINES_INVOICES];([LINES_INVOICES]Code=
"FX-200") & (Month of([INVOICES]DATE_INV)=4))
SQL クエリは次の通りです。
```

```
SELECT ID_LIN
  FROM LINES_INVOICES, INVOICES
  WHERE LINES_INVOICES.ID_INV=INVOICES.ID_INV
  AND LINES_INVOICES.CODE="FX-200"
```

```
AND MONTH(INVOICES.DATE_INV) = 4
```

QUERY BY SQL コマンドを使用する場合は次の通りです。

```
C_STRING(30;$queryFormula)
$queryFormula="LINES_INVOICES.ID_INV=INVOICES.ID_INV
AND LINES_INVOICES.CODE='FX-200'
AND MONTH(INVOICES.DATE_INV)=4"
QUERY BY SQL([LINES_INVOICES];$queryFormula)
```

Is field value Null

Is field value Null (field) → Boolean

引数	タイプ	説明
field	フィールド	→ 評価されるフィールド
返り値	ブール	← TRUE = フィールドは NULL FALSE = フィールドは NULL で ない

Is field value Null コマンドは、field 引数に指定されたフィールドが NULL 値である場合は TRUE を返し、そうでない場合は FALSE を返します。

NULL 値は 4D の SQL エンジンで使用されます。詳細は、[ページ 211](#)、["4D での NULL 値"](#) の節を参照してください。

参照：[SET FIELD VALUE NULL](#)

SET FIELD VALUE NULL

SET FIELD VALUE NULL (field)

引数	タイプ	説明
field	フィールド	→ NULL 値を設定するフィールド

SET FIELD VALUE NULL コマンドは、NULL 値を field 引数に指定されたフィールドに割り当てます。

NULL 値は 4D の SQL エンジンで使用されます。詳細は、[ページ 211](#)、["4D での NULL 値"](#) の節を参照してください。

4D フィールドの NULL 値をストラクチャエディタのレベルで許可しない可能性もある点に注意してください ([ページ 212](#)、["NULL 値の入力を拒否"](#) の節を参照)。

参照：[Is field value Null](#)

GET LAST SQL ERROR

GET LAST SQL ERROR(codesArray; intCompArray; textArray)

引数	タイプ	説明
codesArray	数値配列	← エラー番号
intCompArray	文字列配列	← 内部コンポーネント・コード

引数	タイプ	説明
textArray	文字列配列	← エラーのテキスト

GET LAST SQL ERROR コマンドは、SQL カーネルの使用に関するエラーのコールスタックについて情報を返します。

このコマンドは、ON ERR CALL コマンドによりインストールされたエラー処理メソッドから呼び出されなければなりません。

情報は、次の3つの同期配列に返されます。

- codesArray: この配列は、生成されたエラーコードのリストを受け取ります。
- intCompArray: この配列は、各エラーと関連付けられた内部コンポーネントのコードを含みます。
- textArray: この配列は、各エラーのテキストを含みます。

エラーコードのリストおよびそれらのテキストは、[ページ 231](#)、"[SQL エラーコード](#)"の節で提供しています。

START SQL SERVER

START SQL SERVER

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません。

START SQL SERVER コマンドは、それが実行された 4D アプリケーションで統合 SQL サーバを起動します。SQL サーバは、起動すると、環境設定アプリケーションに設定された TCP ポートで受信した SQL クエリに対応できます。

SQL サーバが正確に起動すると、OK システム変数は 1 に設定されます。そうでない場合は 0 に設定されます。

注 このコマンドは 4D SQL エンジンの内部機能に影響しません。SQL エンジンは内部クエリには常に対応します。

参照：[STOP SQL SERVER](#).

STOP SQL SERVER

STOP SQL SERVER

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません。

STOP SQL SERVER コマンドは、それが実行された 4D アプリケーションで統合 SQL サーバを停止します。

SQL サーバを停止すると、すべての SQL 接続が中断され、サーバは外部の SQL クエリを一切受け入れなくなります。SQL サーバが起動していない場合、コマンドは何もしません。

注 このコマンドは 4D SQL エンジンの内部機能に影響しません。SQL エンジンは内部クエリには常に対応します。

参照：[START SQL SERVER](#)

クエリ

新しい SET QUERY AND LOCK コマンドを使用して、検索したレコードをロックできます。

QUERY BY FORMULA コマンドおよび QUERY SELECTION BY FORMULA コマンドのシンタックスと機能は若干修正されました。

QUERY WITH ARRAY コマンドおよび Find index key コマンドは、インデックスの付かないフィールドでも動作するようになりました。さらに、Find index key コマンドは名前が Find in field に変更されました。

さらに、4D v11 ではキーワード検索を実行できます。この新しい特徴は、[ページ 261](#)、"[キーワード検索](#)"の節で説明されています。

SET QUERY AND LOCK

SET QUERY AND LOCK(lock)

引数	タイプ	説明
lock	ブール	→ TRUE = 検索結果レコードをロックする。 FALSE = 検索結果レコードをロックしない。

新しい SET QUERY AND LOCK コマンドを使用して、カレントトランザクションでこのコマンド呼び出しに続くすべてのクエリで検索されたレコードの自動ロックをリクエストできます。この結果検索されたレコードは、クエリと結果処理の間、他のプロセスでは修正できません。

デフォルトでは、検索結果レコードはロックされません。ロックを有効にするには、lock 引数に TRUE(真)を渡します。

このコマンドはトランザクション内で使用されることが肝要です。トランザクションの外でこのコマンドが呼び出された場合、エラーが生成されます。これによりレコードロックのより良い管理が可能になります。見つかったレコードは、トランザクションが(受け入れられたかまたはキャンセルされたかにかかわらず)終了されない限りはロックされたままです。トランザクションの完了後に、全レコードのロックは解除されます。

レコードは、カレント・トランザクションの全テーブルに対してロックされます。このメカニズムを後に無効にするには、SET QUERY AND LOCK(False)を呼び出します。

- ▼ この例では、QUERY から DELETE SELECTION の間で他のプロセスは Category が C に属するレコードを更新したり、削除したりすることはできません。

```
START TRANSACTION
```

```
SET QUERY AND LOCK(TRUE)
```

```
QUERY([Clients];[Clients]Category="C")
```

```
  `この時点で、検索されたレコードは
```

```
  `他のプロセスに対し自動的にロックされます。
```

```
DELETE SELECTION([Clients])
```

SET QUERY AND LOCK(FALSE)
VALIDATE TRANSACTION

QUERY BY FORMULA, QUERY SELECTION BY FORMULA

QUERY BY FORMULA (table; queryFormula)

QUERY SELECTION BY FORMULA(table; queryFormula)

4D v11 のこれら 2 つのコマンドに重要な変更が行われました。

- table 引数は必須になりました。
- これらのコマンドは著しく最適化され、特に、インデックスを活用できるようになりました。
クエリのタイプが許せば、これらのコマンドは QUERY コマンドに相当するクエリを実行します。例えば、ステートメント QUERY BY FORMULA([mytable]; [mytable]myfield=value) は、QUERY([mytable]; [mytable]myfield=value) と全く同等に実行されます。つまり、このステートメントはインデックスを使用できます。

4D はまた、最初にクエリの最適化可能部分を実行して、その結果と残りの部分を結合することにより、"最適化不可" 部分を含むクエリも最適化することができます。

例えば、QUERY BY FORMULA([mytable]; Length(myfield)=value) ステートメントは最適化されません。一方、QUERY BY FORMULA([mytable]; Length(myfield)=value1 | myfield=value2) は部分的に最適化されます。

- これらのコマンドは、クライアントサーバではサーバで実行されるようになりました。変数が直接フォーミュラから呼び出された場合、クエリはクライアントマシンにある変数の値で計算される点に注意してください。
例えば、ステートメント QUERY BY FORMULA([mytable]; [mytable]myfield=myvariable) はサーバで実行されますが、クライアント・マシンにある myvariable 変数の内容で計算されます。

QUERY WITH ARRAY

QUERY WITH ARRAY (targetField; array)

引数	タイプ	説明
targetField	フィールド	→ 値の比較に使用されるフィールド
array	配列	→ 検索値の配列

QUERY WITH ARRAY はインデックスが設定されていないフィールドも使用できるようになりました。

Find in field

Find in field (targetField; value) → Longint

引数	タイプ	説明
targetField	フィールド	→ 検索を実行するフィールド
value	フィールド; 変数	→ 検索する値
		← 見つかった値
返り値	倍長整数	← 見つかったレコード数、またはレコードが見つからない場合は -1

注 Field in field コマンドは、4D の前バージョンでは Find index key という名前でした。

Field in field コマンドはインデックスの付かないフィールドとも動作するようになりました。

キーワード検索

4D v11 では文字フィールドおよびテキストフィールドでの キーワード検索をサポートします。キーワード検索はテキスト中の単語を検索します。また拡張機能により、テキストまたは文字フィールドが特定の単語を含むレコードも検索します。検索は完全な単語に適用されますが、一連の文字、数字または記号には適用されません。

インデックスの有無

この機能は、テキストまたは文字フィールドにインデックスが付いているかどうかに関わらず、有効です。ただしインデックスがフィールドに付いている場合 (特に "キーワードインデックス" タイプ)、検索速度はかなり速くなります。このタイプのインデックスについては、[ページ 35](#)、"[インデックスの管理](#)" の節を参照してください。

"キーワードを含む" 比較演算子

% 記号で表された新しい "キーワードを含む" 演算子は、キーワード検索専用で作成されました。この演算子を使用すると、テキストが "単語" を含むかどうかをテストしてブール値 (TRUE または FALSE) を返します。この演算子は QUERY、QUERY BY FORMULA、QUERY SELECTION、そして QUERY SELECTION BY FORMULA コマンドに使用できます。

注 標準の「クエリ」ダイアログボックスで定義された場合、キーワード検索は新しい「キーワードを含む」比較演算子を使用します。詳細は、[ページ 103](#)、"[レコードのクエリ](#)" の節を参照してください。

- ▼ 次は、Description に "easy" という単語が含まれる全製品のレコードを検索する例です。

```
QUERY([Products];[Products]Description%"easy")
```

キーワード検索のルール

下記は、キーワード検索に適用するルールです。

- "単語"だけが%演算子によって認識されます。
単語は、単語ごとかつ単語の全体で認識されます。ユーザが複数の単語を検索しようとしたり、単語の一部(即ち、単語の音節など)を検出しようとする、演算子は常にFALSEを返します。
単語は、"単語の途切れ"、つまりスペースや符号で区切られた文字列と定義されます。
数字も文字列として認識されれば検索対象になりますが、千単位の区切りや小数点(「,」カンマや「.」ピリオド)、および他の記号(通貨、温度などの記号)は無視されます。
スペースなどで区切られない日本語の文章などは、検索できないことに注意してください。他方、日本語であっても、半角スペースで区切れば検索対象となります。
例:
"Alpha Bravo Charlie" % "Alpha" は TRUE
ですが、
"Alpha Bravo Charlie" % "Bravo Charlie" は FALSE
また、
"Alpha Bravo Charlie" % "ravo" は FALSE です。
- %演算子は大文字と小文字を区別しません。また、発音区別符号を無視します。
例:
"ALPHA BRAVO" % "alpha" は TRUE、
そして
"Alpha Bravo" % "ALPHA" は TRUE、
そして
"First Noël" % "noel" は TRUE になります。
- %演算子は検索中の単語の末尾にあるワイルドカードを許可します。
例:
"Software and Computers" % "Comput@" は TRUE になります。

プロジェクトフォーム

4D v11 では、プロジェクトフォームを作成、使用できるようになりました。テーブルには一切属さないこれらのフォームは、コンポーネントの開発およびユーザインタフェースの構造にとって貴重なツールの要素を構成します。

これらのフォームの管理については、[ページ 135](#)、"[プロジェクトフォーム](#)"の節で詳細を説明しています。

この項では、4D ランゲージコマンドによるこの新しい特性のサポートについて説明しています。

コマンドおよびイベントとの互換性

フォームに関する 4D ランゲージ・コマンドの多くは、プロジェクトフォームとも使用できます。ただし、このコンテキストで使用できない

特定のコマンドがあります。

- 次のコマンドは、制限なしにプロジェクトフォームと使用できます。
 - INPUT FORM および OUTPUT FORM(下記参照)を除く、"フォーム"テーマのすべてのコマンド。
 - Print form および PAGE SETUP("プリント"テーマ)。
 - Open form window("ウィンドウ"テーマ)。
 - DIALOG("データ入力"テーマ)。
- 構造上の理由から、INPUT FORM コマンドおよび OUTPUT FORM コマンド("フォーム"テーマ)とプロジェクトフォームは互換性がありません。いずれかのコマンドにプロジェクトフォームを渡すと、無視されます。
- Current form table コマンド("テーブル"テーマ)は、プロジェクトフォームのコンテキストで実行されると Nil を返します。
- ユーザフォーム: ユーザフォームのメカニズムとプロジェクトフォームは互換性がありません。従って、"ユーザフォーム"テーマのコマンドをプロジェクトフォームに対して使用することはできません。

フォームイベント

出力フォームに特有な次のイベントを除き、すべてのフォームイベントはプロジェクトフォームで有効化されます。

- On Display Detail
- On Open Detail
- On Close Detail
- On Load Record
- On Header
- On Printing Detail
- On Printing Break
- On Printing Footer

プロジェクトフォームの呼び出し

フォームに関連するコマンド(ユーザフォームを除く)の多くは、オプションの table 引数を最初の引数として受け入れます。例えば、GET FORM PARAMETER コマンド、Open form window コマンドおよび DIALOG コマンドなどの場合がそうです。

プロジェクトフォームとテーブルフォームは同じ名前を持つことができるため、この引数により使用するフォームを決定できます: テーブルフォームをターゲットにする場合は table 引数を渡し、プロジェクトフォームの場合はそれを省略します。

- ▼ "TheForm" という名前のプロジェクトフォーム、および同じ名前を持つ [Table1] テーブルのテーブルフォームを含むデータベースでは:

DIALOG([Table1];"TheForm") `4D はテーブルフォームを使用します。
DIALOG("TheForm") `4D はプロジェクトフォームを使用します。

- ▼ "TheForm" という名前のプロジェクトフォームは含むが、[Table1] テーブルの "TheForm" という名前のテーブルフォームは含まないデータベースでは:

DEFAULT TABLE([Table1])
DIALOG("TheForm") `4D はプロジェクトフォームを使用します。
しかし、データベースに名前の同じプロジェクトフォームとテーブルフォームがある時に DEFAULT TABLE コマンドが実行された場合、この原則は無効になります。実際、この場合 4D は table 引数が渡されないとしても、デフォルトによりテーブルフォームを使用します。プロジェクトフォームの使用を保証するためには、新しい NO DEFAULT TABLE コマンド (下記で説明) を使用します。

NO DEFAULT TABLE

NO DEFAULT TABLE

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません。

NO DEFAULT TABLE コマンドを使用して、DEFAULT TABLE コマンドの効力を無効にできます。このコマンドの実行後は、プロセスのために定義されたデフォルトテーブルは一切存在しません。

事前に DEFAULT TABLE コマンドが呼び出されていない場合、このコマンドは何の効力も持ちません。

- ▼ "TheForm" という名前のプロジェクトフォーム、および同じ名前を持つ [Table1] テーブルのテーブルフォームを含むデータベースでは:

DEFAULT TABLE([Table1])
DIALOG("TheForm") `4D はテーブルフォームを使用します。

NO DEFAULT TABLE
DIALOG("TheForm") `4D はプロジェクトフォームを使用します。
テーマ: テーブル

ドラッグ & ドロップ

4D v11 のドラッグ & ドロップは今も既存のメカニズムと互換性がありますが、そのサポートが拡張されました。

自動ドラッグ & ドロップが新しく設定されました。新しいイベントおよびコマンドにより、特にデスクトップから 4D フォームまたはその逆にドキュメントをドラッグ & ドロップするようなインタフェースを設定できます。

新しいドラッグ & ドロップ機能の多くは、"ペーストボード" テーマを通してサポートされています。このテーマのコマンドは修正され、新しい

コマンドが追加されました。これらの新しい特性については、[ページ 269](#)、"[ペーストボード](#)"の節で詳細を説明しています。

- 注 新しく統合された機能を使用すると、2つの4D データベース間をドラッグ & ドロップまたはコピー / ペーストすることによってデザインモードでのオブジェクト(メニュー、リストなど)の移動を処理できます。この点については、[ページ 79](#)、"[デザインモードでのオブジェクトの移動](#)"の節で説明しています。

新機能の概要

4D v11 は、ドラッグ & ドロップの使用によるインタフェースの動作領域を拡大しています。

- 新しいオプションにより、ネイティブなテキストエリアでの自動ドラッグ & ドロップ操作を処理できます。
- 4D フォームと他のアプリケーションやオペレーティングシステムのデスクトップの間、またはその逆においてオブジェクトをドラッグ & ドロップできるようになりました。

例えば、GIF ピクチャファイルを4D ピクチャフィールドにドラッグ & ドロップできます。また、ワープロアプリケーションからテキストを選択して、それを4Dのテキスト変数にドロップすることもできます。

- 必ずしも最前面のフォームでなくても、オブジェクトを直接4Dアプリケーション上にドロップできます。この場合、新しい On Drop データベースメソッドを使用してドラッグ & ドロップ操作を処理できます。例えば、4D Write ドキュメントを4D アプリケーション・アイコン上にドロップすると、それを開くことができます。

- 注 4D v11 では、標準ドラッグ & ドロップ処理の図形イメージ(ドラッグされているオブジェクトの周りの囲み点線)は以前と異なりサポートされていません。

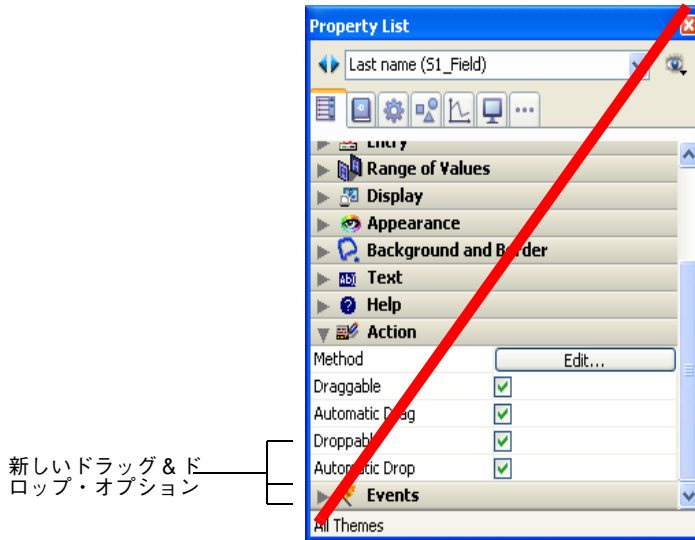
ネイティブテキスト エリアでのドラッグ & ドロップ・オプ ション

4D v11 では、実装されたメカニズムにより大きい制御を持たせるため、自動ドラッグ & ドロップ(2004 バージョンでの名前は"システム"ドラッグ & ドロップ)の管理に修正が加えられました。

自動ドラッグ & ドロップを使用して、2つのフォームエリア間で直接テキストまたはピクチャをコピーできます。このタイプのドラッグ & ドロップは、ネイティブ・コントロール、つまりフィールド、変数およびコンボボックスでのみ使用できます。

これらのオブジェクトにドラッグ & ドロップ操作を処理する新しいオプションが追加されました。これにはフォームエディタで、各オブジェク

トのプロパティリストの "アクション" テーマからアクセスできます。:



新しいドラッグ & ドロップ・オプション

ドラッグ可

このオプションは以前のバージョンに既に存在しました。このオプションにチェックが入れていると、自動ドラッグが有効である場合を除き(下記参照)、オブジェクトがドラッグされた時、新しい On Begin Drag Over イベントが生成されます(ページ 268、"新しい On Begin Drag Over フォームイベント"の節)。ピクチャ変数やフィールドの場合、ピクチャとその参照両方がドラッグされます。

標準のドラッグを有効にするには、Alt (Windows) や Option (Mac OS) キーを押しながらドラッグします。ピクチャの場合、ピクチャや変数参照のみがドラッグされます。

自動ドラッグ

このオプションにチェックが入れていると、テキストエリアの自動ドラッグモードが有効になります(前のバージョンと同じ動作)。ドラッグ可オプションがチェックされていても、このモードがより優先されます。このケースで標準のドラッグを "強制" するにはドラッグの間 Alt (Windows) や Option (Mac OS) キーを押します。

注 自動ドラッグは選択されたテキストを移動します。テキストをコピーするには、ドラッグの間 Ctrl (Windows) や Command (Mac OS) キーを押します。

前のバージョンから変換されたデータベースでは、デフォルトでこのオプションにチェックが入っています。

ドロップ可能

このオプションは 4D の前バージョンに既にあったものです。これは、ドラッグされているデータのタイプに関係なく、オブジェクトがドラッグされたデータを受け入れ、On Drag Over フォームイベントおよび On Drop フォームイベントを呼び出すことを示しています。

ただし、環境設定の互換性ページで、新しい"外部からのドラッグアンドドロップを拒否"オプションにチェックが入っていて、4Dの外部オブジェクトがドラッグされた場合、ドラッグは拒否され、イベントは生成されません。詳細は、[ページ 267](#)、"[互換性の新しい環境設定](#)"の節を参照してください。

自動ドロップ

このオプションは、自動ドロップモードを有効化する時に使用します。4Dは、可能な場合は、オブジェクト上にドロップされたテキストタイプまたはピクチャタイプのドラッグされたデータの挿入を自動的に管理します(データはオブジェクトの中にペーストされます)。この場合、On Drag Over フォームイベントおよび On Drop フォームイベントは生成されません。

テキストまたはピクチャ以外のデータ(他の4Dオブジェクト、ファイルなど)、または複合データがドロップされる場合、アプリケーションはドロップ可能オプションの値を参照します:それにチェックが入っている場合は On Drag Over フォームイベントと On Drop フォームイベントが作成されます;そうでない場合、ドロップは拒否されます。また、これは"外部からのドラッグアンドドロップを拒否"オプションの値によっても変わります。

Drop position

Drop position {(columnNumber)} → Number

引数	タイプ	説明
columnNumber	倍長整数	← リストボックスのカラム数、または最後のカラムを越えてドロップが発生した場合は -1
返り値	数値	← <ul style="list-style-type: none"> ・ 数 (配列 / リストボックス)、または ・ 位置 (リスト)、または ・ 行き先項目の文字列 (テキスト / コンボボックス) での位置、または ・ 最後の配列要素または最後のリスト項目を越えてドロップが発生した場合は -1

Drop position コマンドは、テキストタイプやコンボボックスの変数およびフィールドと連動するようになりました。これに関連して、コマンドは文字列中でドロップが発生した文字の位置を返します。

互換性の新しい環境設定

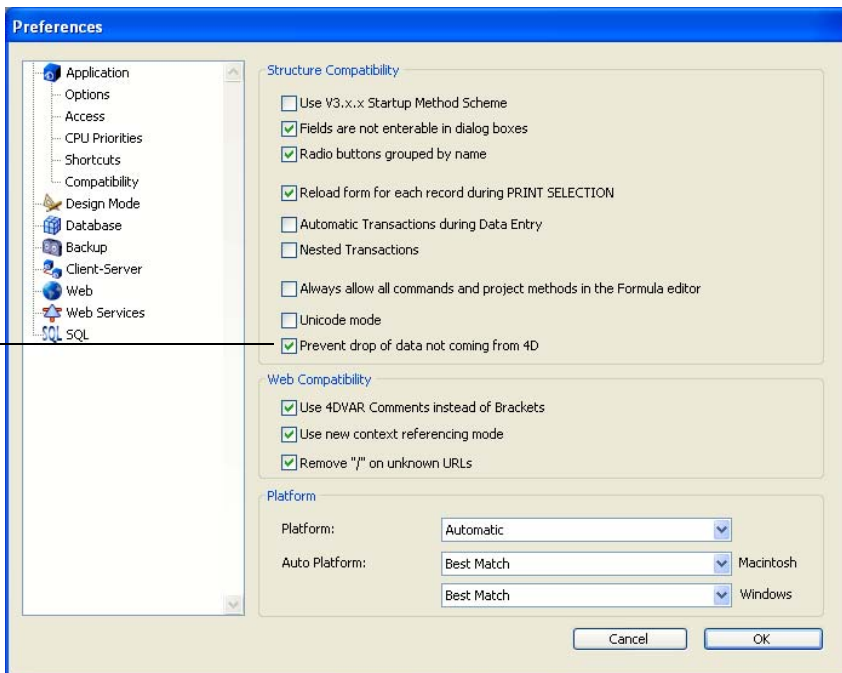
4D v11 はセレクションやオブジェクト、また例えばピクチャファイルのような4Dの外部ファイルのドラッグ & ドロップを許可します。この動作は、データベースコードによってサポートされなければなりません。4Dの前バージョンから変換されたデータベースの場合、既存のコードが然るべく適合されていないと、この動作が不具合につながることもあり

ます。

この理由から、環境設定の新しいオプション、外部からのドラッグアンドドロップを拒否はデータベースへの外部オブジェクトのドロップを防ぐために使用されます。

このオプションは「アプリケーション / 互換性」ページにあります。

外部からのドラッグ & ドロップを処理するオプション



このオプションにチェックが入っている場合、4D フォームへの外部オブジェクトのドロップは拒否されます。ただし、アプリケーションがドロップされるデータ (テキストまたはピクチャ) を解釈できれば、自動ドロップオプションを持つオブジェクトの中に外部オブジェクトを挿入できる点に留意してください。詳細は、[ページ 267](#)、"[自動ドロップ](#)" の節を参照してください。

このオプションは、変換されたデータベースのデフォルトではチェックが入っており、バージョン 11 で作成されたデータベースのデフォルトではチェックが入っていません。

新しい On Begin Drag Over フォームイベント

4D v11 には新しい On Begin Drag Over フォームイベントが追加されました。このイベントは、ドラッグ可能な任意のフォームオブジェクトに対して選択できます。これは、オブジェクトがドラッグ可能プロパティを持つ場合は常に生成されます。

既存の On Drag Over フォームイベントと違い、On Begin Drag Over フォームイベントはソースオブジェクトのメソッドまたはソースオブジェクトのフォームメソッドから呼び出されます。

プロセスをまたがるドラッグ & ドロップの場合、On Begin Drag Over

フォームイベントは On Drag Over フォームイベントと同様にソースオブジェクトのコンテキストで生成されます。

この新しいイベントはドラッグ操作の高度処理に役立ちます。これは、次のような目的で使用できます。

- (GET PASTEBOARD DATA コマンドを使って)、ペーストボードにあるデータおよびシグネチャを取得する時。
- (APPEND DATA TO PASTEBOARD コマンドを使って)、ペーストボードにデータおよびシグネチャを追加する時。
- ドラッグされたオブジェクトのメソッドの \$0 によって、ドラッグを許可または拒否する時。ドラッグを許可するには \$0=0、拒否するには \$0=-1 を渡します。

4D データは、イベントを呼び出す前にペーストボードに配置されます。例えば、自動ドラッグオプションなしのドラッグの場合、ドラッグされたテキストはイベントが呼び出された時には既にペーストボードにあります。

新しい On Drop データベースメソッド

4D v11 には新しい On Drop データベースメソッドが追加されました。これは、4D シングルユーザ・アプリケーションおよび 4D Client で使用できます。

このデータベースメソッドは、4D アプリケーションのフォームまたはウィンドウ以外の部分にオブジェクトがドロップされる場合、自動的に実行されます。たとえば、

- MDI ウィンドウの空白エリア上 (Windows)。
- Dock (Mac OS) やデスクトップ上の 4D アイコン。

デスクトップの 4D アプリケーションアイコンにドロップが発生した場合、On Drop データベースメソッドは、アプリケーションが既に立ち上がっている場合のみ呼び出されます。ただし、4D Desktop にマージされたアプリケーションは例外で、この場合はアプリケーションが立ち上がってない時でもデータベースメソッドは呼び出されます。これは、カスタムドキュメントのシグネチャを定義できることを意味します。

- ▼ 次の例は、任意のフォームの外部にドロップされた 4D Write ドキュメントを開く場合に使用できます。

```

`On Drop database method
droppedFile:=Get file from pasteboard(1)
If(Position(".4W7";droppedFile)=Length(droppedFile)-3)
  externalArea:=Open external window(100;100;500;500;0;
droppedFile;"_4D Write")
  WR OPEN DOCUMENT(externalArea;droppedFile)
End if

```

ペーストボード

"ペーストボード" テーマは 4D v11 における "クリップボード" テーマの

新しい名前です。このテーマのコマンドに使われるドラッグ & ドロップ操作に関して、その新しい処理能力を反映させるために名前が変更されました。

実際、4D v11 ではこのテーマのコマンドがドラッグ & ドロップに関する新機能をサポートしています。新しいドラッグ & ドロップの可能性については、[ページ 264](#)、"[ドラッグ & ドロップ](#)"の節で詳細を説明しています。

"ペーストボード"テーマのコマンドも名前が変更され、新しいコマンドが追加されました。

コピー/ペーストとドラッグ&ドロップのペーストボード

"ペーストボード"テーマのコマンドは、コピー/ペースト操作に加えてドラッグ & ドロップ操作にも使えるようになりました。

これを行うため、4D v11 は2つのデータペーストボードを使います。1つはコピー(またはカット)されたデータ用で、これは4Dの前バージョンから既に存在する実質的クリップボードになります。もう1つはドラッグ & ドロップされるデータ用です。

これら2つのペーストボードは同じコマンドを使って処理されます。どちらにアクセスするかは、次の通り、コンテキストによって変わります。

- ドラッグ & ドロップのペーストボードには、On Begin Drag Over、On Drag Over、On Drop フォームイベント、および On Drop データベースメソッドの範囲内でのみアクセスできます。

これらのコンテキストの外では、ドラッグ & ドロップのペーストボードは利用できません。

- コピー/ペーストのペーストボードには、上記以外のすべての場合にアクセスできます。ドラッグ & ドロップのペーストボードと違い、中に配置されたデータは削除または再使用されない限り、セッションを通してそのデータを保存されます。

コマンド名

ドラッグ & ドロップに関するコマンドの新しい処理能力を反映させるため、"ペーストボード"テーマの全コマンドの名前が変更されました。コマンドの多くは、その処理能力が拡張されました。

次のテーブルでは、コマンド名の変更をまとめています。

以前の名前 (4D 2004.x)	新しい名前 (4D v11.x)
Test clipboard	Pasteboard data size
GET CLIPBOARD	GET PASTEBOARD DATA
CLEAR CLIPBOARD	CLEAR PASTEBOARD
APPEND TO CLIPBOARD	APPEND DATA TO PASTEBOARD
SET PICTURE TO CLIPBOARD	SET PICTURE TO PASTEBOARD

以前の名前 (4D 2004.x)	新しい名前 (4D v11.x)
GET PICTURE FROM CLIPBOARD SET TEXT TO CLIPBOARD Get text from clipboard	GET PICTURE FROM PASTEBOARD SET TEXT TO PASTEBOARD Get text from pasteboard

定数名

一貫性を保つため、関連する以下の定数に変更が加えられました。

- "Clipboard" テーマは "Pasteboard" に変更されました。
- No such data in clipboard 定数は No such data in pasteboard に名称が変更されました。

データのタイプ

ドラッグ & ドロップ操作の間は、ペーストボードにタイプの違うデータを配置したり、そこから読み取ったりできます。データタイプへは、次のような方法でアクセスできます。

- 4D シグネチャを使用: 4D シグネチャは、4D に参照されたデータタイプを示す文字列です。4D シグネチャは Mac OS でも Windows でも同じため、その使用によりマルチプラットフォーム・アプリケーションの開発が促進されます。

4D シグネチャのリストは次の節を参照してください。

- UTI (Uniform Type Identifier の略、Mac OS のみ) を使用: Apple 社が指定する UTI 標準は、文字列とネイティブ・オブジェクトの各タイプを関連付けます。例えば、GIF ピクチャは UTI タイプ "com.apple.gif" を持ちます。UTI タイプは、Apple 社の文書や関連エディタで公表されています。
- 番号またはフォーマット名を使用 (Windows のみ): Windows では、各ネイティブ・データタイプはその番号 ("3"、"12" など) および名前 ("Rich Text Edit" など) によって参照されます。デフォルトにより、Microsoft 社は標準データフォーマットと呼ばれる、いくつかのネイティブタイプを指定しています。加えて、サードパーティーエディタは独自のフォーマット名をシステムに "登録" できます。その場合、新しいデータタイプの番号が割り当てられます。この件およびネイティブタイプに関する詳細は、Microsoft 社の開発ドキュメントを参照してください。

<http://msdn2.microsoft.com/en-us/library/ms649013.aspx>

注 4D コマンドでは、Windows のフォーマット番号はテキストとして処理されます。

ペーストボードテーマのコマンドはすべて、これらの各データタイプと連動します。GET PASTEBOARD DATA TYPE コマンドを使って、各フォーマットのペーストボードにどのデータタイプが存在するかを確認できま

す。

注 以前のバージョンの 4D との互換性のため、4 文字のタイプ (TEXT、PICT またはカスタムタイプ) も引き続きサポートされています。

4D シグネチャ

次の表は、標準の 4D シグネチャおよびその説明をまとめたリストです。

シグネチャ	説明
"com.4d.private.text.native"	ネイティブ文字セットのテキスト
"com.4d.private.text.utf16"	Unicode 文字セットのテキスト
"com.4d.private.text.rtf"	リッチテキスト
"com.4d.private.picture.pict"	PICT ピクチャフォーマット
"com.4d.private.picture.pgn"	PGN ピクチャフォーマット
"com.4d.private.picture.gif"	GIF ピクチャフォーマット
"com.4d.private.picture.jfif"	JPEG ピクチャフォーマット
"com.4d.private.picture.emf"	EMF ピクチャフォーマット
"com.4d.private.picture.bitmap"	BITMAP ピクチャフォーマット
"com.4d.private.picture.tiff"	TIFF ピクチャフォーマット
"com.4d.private.picture.pdf"	PDF ドキュメント
"com.4d.private.file.url"	ファイルパス名

変更されたコマンド

ドラッグ & ドロップ操作によるデータを処理するため、「クリップボード」テーマの既存コマンドのほとんどが変更されました。

GET PASTEBOARD DATA

GET PASTEBOARD DATA(dataType; data)

引数	タイプ	説明
dataType	文字列	→ ペーストボード・データタイプ、またはドラッグ & ドロップ用の 4D シグネチャ
data	BLOB	← ペーストボードから抽出されたデータ

注 4D の前バージョンでは、このコマンドの名前は GET CLIPBOARD でした。

GET PASTEBOARD DATA コマンドは、ドラッグ & ドロップ操作によるデータを返すようになりました。

ドラッグ & ドロップの場合、[ページ 271](#)、「データのタイプ」の節で定義されたタイプの 1 つを dataType に渡すことができます。

このコマンドでファイルタイプのデータは取得できません。[Get file from pasteboard](#) コマンドを使う必要があります。

APPEND DATA TO PASTEBOARD

APPEND DATA TO PASTEBOARD (dataType; data)

引数	タイプ	説明
dataType	文字列	→ 4文字ペーストボード・データタイプ文字列、またはドラッグ&ドロップ用の4Dシグネチャ
data	BLOB	→ ペーストボードに追加するデータ

注 4Dの前バージョンでは、このコマンドの名前は APPEND TO CLIPBOARD でした。

APPEND DATA TO PASTEBOARD コマンドは、ドラッグ & ドロップ操作のコンテキストにデータを追加できるようになりました。ドラッグ & ドロップの場合、[ページ 271](#)、"[データのタイプ](#)"の節で定義されたタイプの1つを dataType で渡すことができます。

注 ペーストボードは、On Drag Over フォームイベントの間は読込みのみモードになります。従って、このコマンドはそのコンテキストでは使用できません。

Get text from pasteboard

Get text from pasteboard → String

引数	タイプ	説明
返り値	文字列	← ペーストボードに存在するテキスト(もし、あれば)を返します。

注 4Dの前バージョンでは、このコマンドの名前は Get text from clipboard でした。

Get text from pasteboard コマンドは、ドラッグ & ドロップ操作のコンテキストにあるペーストボードからテキストデータを取得する時に使用します。

注 4D v11 では、テキストタイプの最大データサイズが 2GB になりました。

ペーストボードのデータが(例えば、RTF フォーマットにある)リッチテキストを含む場合、ドロップエリアとの互換性があれば、テキストはその属性も一緒にドロップされます。

SET TEXT TO PASTEBOARD

SET TEXT TO PASTEBOARD (text)

引数	タイプ	説明
text	文字列	→ ペーストボードに配置されるテキスト

注 4D の前バージョンでは、このコマンドの名前は SET TEXT TO CLIPBOARD でした。

SET TEXT TO PASTEBOARD コマンドは、ドラッグ & ドロップ操作のコンテキストにあるペーストボードにテキストを配置する時に使用します。

注 ペーストボードは、On Drag Over フォームイベントの間は読込のみモードになります。従って、このコマンドはそのコンテキストでは使用できません。

GET PICTURE FROM PASTEBOARD

GET PICTURE FROM PASTEBOARD (picture)

引数	タイプ	説明
picture	ピクチャ	← ペーストボードに存在するピクチャ

注 4D の前バージョンでは、このコマンドの名前は GET PICTURE FROM CLIPBOARD でした。

GET PICTURE FROM PASTEBOARD コマンドは、ドラッグ & ドロップ操作のコンテキストにあるペーストボードから、そこにあるピクチャを取得する時に使用します。

注 4D v11 では、ピクチャはネイティブ・フォーマット (JPEG、TIF、PNG など) で保存されます。

ドロップエリアがネイティブ・フォーマットを受け入れる場合、ピクチャはそのフォーマットを保持します; そうでない場合、ピクチャは PICT フォーマットに変換されます。

SET PICTURE TO PASTEBOARD

SET PICTURE TO PASTEBOARD (picture)

引数	タイプ	説明
picture	ピクチャ	→ ペーストボードに配置されるピクチャ

注 4D の前バージョンでは、このコマンドの名前は SET PICTURE TO CLIPBOARD でした。

SET PICTURE TO PASTEBOARD コマンドは、ドラッグ & ドロップ操作のコンテキストにあるペーストボードにピクチャを配置する時に使用します。

注 ペーストボードは、On Drag Over フォームイベントの間は読込のみモードになります。従って、このコマンドはそのコンテキストでは使用できません。

Pasteboard data size

Pasteboard data size (dataType) →Number

引数	タイプ	説明
dataType	文字列	→ 4文字ペーストボード・データタイプ文字列、またはドラッグ & ドロップ用の4D シグネチャ
返り値	数値	← ペーストボードに保存されたデータのサイズ(バイト)、またはエラーコードの結果

注 4Dの前バージョンでは、このコマンドの名前は Test clipboard でした。

Pasteboard data size コマンドは、ドラッグ & ドロップ操作のコンテキストにデータのサイズを返せるようになりました。
ドラッグ & ドロップの場合、[ページ 271](#)、"[データのタイプ](#)"の節で定義されたタイプの1つを dataType に渡すことができます。

CLEAR PASTEBOARD

CLEAR PASTEBOARD

引数	タイプ	説明
		このコマンドは引数を必要としません。

注 4Dの前バージョンでは、このコマンドの名前は CLEAR CLIPBOARD でした。

CLEAR PASTEBOARD コマンドは、ドラッグ & ドロップ操作のコンテキストにあるペーストボードの内容を消去します。

新しいコマンド

ドラッグ & ドロップ操作のサポートに関するニーズを満たすため、"ペーストボード" テーマに新しいコマンドが追加されました。

SET FILE TO PASTEBOARD

SET FILE TO PASTEBOARD (filePath)

引数	タイプ	説明
filePath	テキスト	→ ファイルの完全なパス名

SET FILE TO PASTEBOARD コマンドは、filePath 引数に渡された完全なパス名をペーストボードに追加します。
このコマンドを使って、例えば4Dオブジェクトをデスクトップ上ファイルにドラッグ & ドロップするインタフェースを設定できます。

注 ペーストボードは、On Drag Over フォームイベントの間は読込のみモードになります。従って、このコマンドはそのコンテキストでは使用できません。

Get file from pasteboard

Get file from pasteboard (xIndex) → Text

引数	タイプ	説明
xIndex	数値	→ ドラッグ操作に含まれる X 番目のファイル
返り値	テキスト	← ペーストボードから抽出されたファイルのパス名

Get file from pasteboard コマンドは、ドラッグ & ドロップ操作に含まれるファイルの絶対パス名を返します。xIndex 引数は、選択されたファイルのセットからファイルを指定する時に使用します。ペーストボードに X 番目のファイルがない場合、コマンドは空の文字列を返します。

- ▼ 次の例は、ドラッグ & ドロップ操作に含まれるファイルのパス名すべてを配列で検索する時に使用できます。

```

ARRAY TEXT($filesArray;0)
C_TEXT($vfileArray)
C_INTEGER($n)
$n:=1
Repeat
  $vfileArray:=Get file from pasteboard($n)
  If($vfileArray# "")
    APPEND TO ARRAY($filesArray;$vfileArray)
  $n:=$n+1
End if
Until($vfileArray="")

```

GET PASTEBOARD DATA TYPE

GET PASTEBOARD DATA TYPE(4Dsignatures; nativeTypes; formatNames)

引数	タイプ	説明
4Dsignatures	テキスト配列	← データタイプの 4D シグネチャ
nativeTypes	テキスト配列	← ネイティブ・データタイプ
formatNames	テキスト配列	← フォーマット名 (Windows のみ)、Mac OS の場合は空の文字列

GET PASTEBOARD DATA TYPE コマンドを使って、ペーストボードに存在するデータタイプのリストを取得できます。一般に、このコマンドは移動先オブジェクトの On Drop フォームイベントまたは On Drag Over フォームイベントの枠組みの中で使用すべきものです。特に、このコマンドにより、ペーストボードに特定タイプのデータが存在するかテストできます。このコマンドは 2 つ (または 3 つ) の配列を用いて、次のようにそれぞれ

違うデータタイプを返します。

- 4Dsignatures 配列は、内部の 4D シグネチャを使って記述されたデータタイプ (例えば、"com.4d.picture.gif") を含みます。
見つかったデータタイプが 4D に認識されない場合、空の文字列 ("") が配列で返されます。
- nativeTypes 配列は、ネイティブタイプを使って記述されたデータタイプを含みます。Mac OS と Windows では、ネイティブタイプのフォーマットが異なります。
 - Mac OS の場合、ネイティブタイプは UTIs (Uniform Type Identifier) で記述されます。
 - Windows の場合、ネイティブタイプはフォーマット名に割り当てられた番号で記述されます。nativeTypes 配列は、文字列の形での番号 ("3"、"2" など) を含みます。より明確なラベルを使用するには、Windows のネイティブタイプのフォーマット名を含む、オプションの formatNames 配列 (下記参照) を使うことをお勧めします。

nativeTypes 配列により、ペーストボードにあるデータはどのようなタイプでもサポートされます。これには、4D に参照されないタイプのデータも含まれます。

- Windows の場合、ペーストボードにあるデータタイプの名前を受け取る formatNames 配列も渡すことができます。この配列で返される値は、例えば、フォーマット・セレクションのポップアップメニューを構築する時に使用できます。
Mac OS の場合、formatNames 配列は空の文字列を返します。

注 サポートされるデータタイプの詳細は、[ページ 271](#)、"[データのタイプ](#)" の節を参照してください。

メニュー

4D v11 ではメニューの管理が完全に変更されました。この変更は、次のような 2 つの目的を満たすために行われたものです。

- 階層メニューの作成と処理を可能にする。
- メニューとメニュー項目により高度な動的コントロールを与える。
これを実現するため、メニューエディタが変更されました ([ページ 167](#)、"[メニューエディタ](#)" の節を参照)。さらに、"メニュー" テーマは、新しいコマンドの追加や既存コマンドの修正により充実しました。

概要

4D v11 を使用して、メニューやメニューバーを動的に作成できます。この場合、これらのオブジェクトが予めメニューエディタで定義されてい

る必要はありません。また、プログラミングで作成されたメニューやメニューバーをすべて動的に削除することもできます。

4D v11 では、メニューとメニューバーに性質の違いがなくなっています(これらのオブジェクトがプログラミングによって作成された場合)。実際、両方とも項目タイトルのリストであり、使用方法が違うだけです。メニューバーの場合、各項目はメニューに対応し、それ自体は項目タイトルで構成されています。これは、階層メニューの背後にある原則と同じです。各項目タイトルはそれ自身がメニューになることもできます。メニューが新しい **Create menu** コマンドを使ってプログラミングにより作成された場合、セッション中このメニューに行われた変更はすべて、データベースの全プロセスにおける、このメニューの他のインスタンスに即座に渡されます。

メニュー参照

階層リスト同様、プログラミングによって作成されたメニューは、セッションを通してメニューが参照できる参照 ID を受け取ります。このマニュアルで MenuRef と呼ばれるこの ID は、16 文字の英数字です。"メニュー" テーマのコマンドはすべて、この ID またはメニュー番号(デザインモードで作成されたメニューを指定する標準操作)を受け入れます。

デザインモードで作成されたメニューとの互換性

両タイプのメニュー(1つはデザインモードで作成されたメニューおよび"前世代のメニューバー、もう1つは4D v11のプログラミングにより作成されたメニュー)には互換性があり、同時に使用できます。

両タイプのオブジェクトを同じメニューバーに組み入れることができます。ユーザの視点から見ると、両タイプの外観および操作は全く同じです。

"メニュー" テーマの既存コマンドは、プログラミングで作成されたメニューをサポートするために変更されました。特に、既存コマンドは参照 ID を最初の引数として受け入れます。

反対に、すべての新しいメカニズムは"前世代"メニューには適用されません。例えば、新しい **RELEASE MENU** コマンドはメニューエディタで設計されたメニューを削除しません。

メニュー項目 = -1

メニュー項目の処理を容易にするため、4D はメニューに最後に追加された項目を特定できるショートカットを提供しています: これを行うには、menuItem 引数に -1 を渡します。この原則は、メニュー項目と連動する"メニュー" テーマのコマンドすべてに適用できます。

新しいコマンド

4D v11 では、メニューを管理する新しいコマンドが提供されます。これらのコマンドは、プログラミング(MenuRef タイプの引数、16 文字の文字列)またはメニューエディタ(倍長整数タイプの引数)で作成されたメニューとメニューバーをサポートします。

Create menu

Create menu {(menu)} → MenuRef

引数	タイプ	説明
menu	MenuRef 倍長整数 文字	→ メニュー参照、またはメニューバーの番号または名前
返り値	MenuRef	→ メニュー参照

Create menu コマンドはメモリ上に新しいメニューを作成する時に使用します。このメニューはメモリ上にのみ存在し、デザインモードのメニューエディタには追加されません。このメニューに行われた修正はすべて、データベースの全プロセスにおける、このメニューの他のインスタンスに即座に反映されます。

このコマンドは新しいメニューに MenuRef タイプの ID を返します。

- オプションの menu 引数を渡さない場合、メニューは空白で作成されます。これを構築して処理するには、[APPEND MENU ITEM](#) コマンドや [SET MENU ITEM](#) コマンドなどを使います。
- menu 引数を渡すと、作成されるメニューはこの引数が指定したソースメニューの正確なコピーになります。ソースメニューのすべてのプロパティは、関連サブメニューも含み、新しいメニューに適用されます。新しい MenuRef 参照はソースメニューとソースメニューに関連付けられた既存のサブメニューに対して作成されることに注意してください。

menu 引数には、有効なメニュー参照、もしくはデザインモードで定義されたメニューバーの番号や名前のいずれかを渡すことができます。後者の場合、新しいメニューはソースメニューバーのメニューおよびサブメニューで構成されます。

このコマンドで作成されたメニューは、[SET MENU BAR](#) コマンドを使うとメニューバーとして使用できます。

RELEASE MENU

RELEASE MENU(menu)

引数	タイプ	説明
menu	メニュー参照	→ メニュー参照

RELEASE MENU コマンドは、menu で指定されたメニューをメモリから取り除きます。これは、メニューの [Create menu](#) コマンドで作成されたメニューでなければなりません。

このコマンドは、すべてのメニューおよびすべてのプロセスから、menu メニューのインスタンスを取り除きます。

使用中のメニューバーに属すメニューは、メニューの機能は果たしますが修正はできなくなります。このメニューは、それが表示される最後のメニューバーが使用中でなくなった時に、はじめて本当にメモリから取り除かれます。

このコマンドはメニューバーとして使用されるメニューと連動します。menu に使用されるサブメニューはどれも取り除くことができません。メニューとそのサブメニューを取り除くには、サブメニューのそれぞれを個別に取り除かなければなりません。

GET MENU ITEMS

GET MENU ITEMS{menu; menuTitlesArray; menuRefsArray}

引数	タイプ	説明
menu	メニュー参照 ; 倍長整数	→ メニュー参照またはメニュー番号
menuTitlesArray	文字配列 (32)	← メニュータイトルの配列
menuRefsArray	文字配列 (16)	→ メニュー参照の配列

GET MENU ITEMS コマンドは、menuTitlesArray 配列と menuRefsArray 配列に、menu 引数に指定されたメニューまたはメニューバーの項目のタイトルと ID を返します。

menu 引数には、メニュー参照 (MenuRef)、メニューバー番号、または [Get menu bar reference](#) コマンドを使って取得したメニューバー参照を渡すことができます。

項目に割り当てられたメニュー参照がない場合、空の文字列が対応する配列要素に返されます。

- ▼ 次は、カレントプロセスのメニューバーの内容を確認する例です。

```
ARRAY STRING(32;menuTitlesArray;0)
ARRAY STRING(16;menuRefsArray;0)
MenuBarRef:=Get menu bar reference(Frontmost process)
GET MENU ITEMS(MenuBarRef;menuTitlesArray;menuRefsArray)
```

Get menu bar reference

Get menu bar reference {(process)} → MenuRef

引数	タイプ	説明
process	数値	→ プロセス参照番号
返り値	メニュー参照	← メニューバー ID

Get menu bar reference コマンドは、カレントメニューバーまたは特定プロセスのメニューバーの ID を返します。

メニューバーが Create menu コマンドで作成された場合、この ID は作成されたメニューの参照 ID に対応します。そうでない場合、コマンドは特定の内部 ID を返します。すべての場合において、この ID はテーマの他の全コマンドでメニューバーを参照する時に使用できます。

process 引数は、カレントメニューバー ID を取得したいプロセスを指定する時に使用できます。この引数を省略すると、コマンドはカレントプ

プロセスのメニューバー ID を返します。

GET MENU ITEM ICON

GET MENU ITEM ICON(menu; menuItem; iconRef{; process})

引数	タイプ	説明
menu	メニュー参照 ! 数値	→ メニュー参照、または メニュー番号
menuItem	数値	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
iconRef	テキスト変数 ! 倍長整数変数	→ メニュー項目に割り当てられたアイコン のライブラリ・ピクチャの名前または 番号
process	数値	→ プロセス参照番号

GET MENU ITEM ICON コマンドは、menu 引数と menuItem 引数が指定したメニュー項目に割り当てられたアイコンの参照を iconRef 変数に返します。この参照は、ピクチャライブラリにあるピクチャの名前または番号になります。

メニュー項目に割り当てられたアイコンは、アプリケーションのツールバーに追加されます。

menu に最後に追加された項目を特定するには、menuItem に -1 を渡します。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、process 引数は不要なので渡しても無視されます。メニュー番号を渡すと、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数にその番号を渡します。

iconRef 引数に渡された変数のタイプによって、コマンドはこの引数にピクチャの名前または番号のいずれかを返します。iconRef 変数に特定のタイプを割り当てないと、デフォルトによりピクチャの名前 (テキストタイプ) が返されます。

メニュー項目に割り当てられたアイコンがない場合、コマンドは空白のピクチャを返します。

参照: [SET MENU ITEM ICON](#)

SET MENU ITEM ICON

SET MENU ITEM ICON(menu; menuItem; iconRef{; process})

引数	タイプ	説明
menu	メニュー参照 ! 数値	→ メニュー参照、または メニュー番号

引数	タイプ	説明
menuItemem	数値	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
iconRef	テキスト； 倍長整数	→ メニュー項目に割り当てられるライブラリ・ピクチャの名前または番号
process	数値	→ プロセス参照番号

SET MENU ITEM ICON コマンドは、menu 引数と menuItemem 引数が指定したメニュー項目に割り当てられたアイコンを変更する時に使用します。

menu に最後に追加された項目を特定するには、menuItemem に -1 を渡します。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、コマンドは全プロセスにおけるメニューのインスタンスすべてに適用されます。この場合、process 引数は渡しても無視されます。

メニュー番号を渡すと、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数にその番号を渡します。

メニュー項目に割り当てられたアイコンは、アプリケーションのツールバーに追加されます。ピクチャは 20 x 20 ピクセルのフレームで表示されます。

iconRef には、アイコンとして使用するライブラリ・ピクチャの名前または番号のいずれかを渡すことができます。ピクチャの番号は名前と違いユニーク ID なので、一般的には名前より番号を使う方が望ましいでしょう。

参照：[GET MENU ITEM ICON](#)

Get menu item method

Get menu item method (menu; menuItemem; process) → String

引数	タイプ	説明
menu	メニュー参照；数値	→ メニュー参照、またはメニュー番号
menuItemem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
process	倍長整数	→ プロセス参照番号
返り値	文字列	← メソッド名

Get menu item method コマンドは、menu 引数と menuItemem 引数が指定したメニュー項目に割り当てられた 4D プロジェクトメソッドの名前を返します。

menu に最後に追加された項目を特定するには、menuItemem に -1 を渡します。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、process 引数は不要なので渡しても無視されます。メニュー番号を渡すと、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数にその番号を渡します。コマンドは 4D メソッドの名前を文字列 (式) として返します。メニュー項目に割り当てられたメソッドがない場合、コマンドは空の文字列を返します。

参照：[SET MENU ITEM METHOD](#)

SET MENU ITEM METHOD

SET MENU ITEM METHOD(menu; menuItem; method{; process})

引数	タイプ	説明
menu	メニュー参照 ! 倍長整数	→ メニュー参照、またはメニュー番号
menuItem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
method	文字列	→ メソッド名
process	倍長整数	→ プロセス参照番号

SET MENU ITEM METHOD コマンドは、menu 引数と menuItem 引数が指定したメニュー項目に割り当てられた 4D プロジェクトメソッドを変更する時に使用します。

menu に最後に追加された項目を特定するには、menuItem に -1 を渡します。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、コマンドは全プロセスにおけるメニューのインスタンスすべてに適用されます。この場合、process 引数は渡しても無視されます。

メニュー番号を渡すと、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数にその番号を渡します。

メソッドでは、4D メソッドの名前を文字列 (式) として渡します。

注 メニュー項目が階層サブメニューの親項目の場合、メニュー項目が選択されてもメソッドは呼び出されません。

参照：[Get menu item method](#)

GET MENU ITEM PROPERTY

GET MENU ITEM PROPERTY (menu; menuItem; property; value{; process}) → Longint

引数	タイプ	説明
menu	メニュー参照 倍長整数	→ メニュー参照、または メニュー番号
menuItem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
property	文字列	→ プロパティ・タイプ
value	式	← プロパティ値
process	倍長整数	→ プロセス参照番号

GET MENU ITEM PROPERTY コマンドは、menu 引数と menuItem 引数が指定したメニュー項目のプロパティのカレント値を value 引数に返します。

menu に最後に追加された項目を特定するには、menuItem に -1 を渡します。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、process 引数は不要なので渡しても無視されます。メニュー番号を渡すと、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数でその番号を渡します。

property 引数には、値を求めたいプロパティを渡します。"メニュー項目プロパティ" テーマの定数の 1 つ、またはカスタムプロパティに対応する文字列を使用できます。メニュープロパティおよびその値の詳細は、[SET MENU ITEM PROPERTY](#) コマンドの説明を参照してください。

参照：[SET MENU ITEM PROPERTY](#)

SET MENU ITEM PROPERTY

SET MENU ITEM PROPERTY (menu; menuItem; property; value{; process})

引数	タイプ	説明
menu	メニュー参照 倍長整数	→ メニュー参照、または メニュー番号
menuItem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
property	文字列	→ プロパティ・タイプ
value	式	→ プロパティ値
process	倍長整数	→ プロセス参照番号

SET MENU ITEM PROPERTY コマンドは、menu 引数と menuItem 引数が指定したメニュー項目にプロパティの値を設定する時に使用します。

menu に最後に追加された項目を特定するには、menuItem に -1 を渡しま

す。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、コマンドは全プロセスにおけるメニューのインスタンスすべてに適用されます。この場合、process 引数は渡しても無視されます。

メニュー番号を渡すと、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数でその番号を渡します。

property 引数には値を修正したいプロパティを渡し、新しい値を value に渡します。

property 引数には、"メニュー項目プロパティ" テーマの定数の 1 つ、または任意のカスタム値を使用できます。

- 標準プロパティ："メニュー項目プロパティ" テーマの定数、およびその取り得る値については下記で説明しています。

Associated Standard Action プロパティの場合、"Value for Associated Standard Action" テーマの定数の 1 つを value 引数に渡せる点に注意してください。:

プロパティ 定数 (値)	値 取り得る値
<u>Associated Standard Action</u> メニュー項目に標準アクションを割り当てる場合に使用します。	0 = No Action 1 = Cancel Action 2 = Accept Action 3 = Next record Action 4 = Previous record Action 5 = First record Action 6 = Last record Action 7 = Delete record Action 8 = Next page Action 9 = Previous page Action 10 = First page Action 11 = Last page Action 12 = Edit subrecord Action 13 = Delete subrecord Action 14 = Add subrecord Action 17 = Undo Action 18 = Cut Action 19 = Copy Action 20 = Paste Action 21 = Clear Action 22 = Select all Action 23 = Show Clipboard Action

Start a New Process 新規プロセス開始オプションを設定するために使用します。	26 = Test Application Action 27 = Quit Action 31 = Redo Action 32 = Preferences Action 35 = Return to Design mode 36 = MSC Action 0 = Yes 1 = No
Access Privileges グループをアクセス権に割り当てるために使用します。	0 = All Groups
	>0 = Group ID

標準メニュー項目プロパティについての詳細は、「デザインリファレンス」マニュアルの「カスタムメニューの作成」の章を参照してください。

- カスタムプロパティ: property には、どのような独自のテキストでも渡すことができ、それにテキスト、数値またはブールタイプの value を割り当てることができます。この value は項目に保存され、[GET MENU ITEM PROPERTY](#) コマンドを使って検索できます。

property 引数には、どのような独自の文字列でも使用できますが、4D が使用するタイトルは使用しないよう注意する必要があります(慣例により、4D が設定するプロパティは "4D_" で始まります)。

- 注 メニュー項目が階層サブメニューの親項目の場合、メニュー項目が選択されても標準動作は呼び出されません。

参照: [GET MENU ITEM PROPERTY](#)

Get menu item modifiers

Get menu item modifiers(menu; menuitem[; process]) → Number

引数	タイプ	説明
menu	メニュー参照 数値	→ メニュー参照、またはメニュー番号
menuitem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
process	倍長整数	→ プロセス参照番号
戻り値	数値 r	← メニュー項目に割り当てられた修正キー

Get menu item modifiers コマンドは、menu 引数と menuitem 引数が指定したメニュー項目の標準ショートカットに割り当てられた追加のモディファイアを返します。

標準ショートカットは、Command キー + カスタムキー (Mac OS) または Ctrl キー + カスタムキー (Windows) から成っています。標準ショートカットは、[SET MENU ITEM SHORTCUT](#) コマンドおよび [Get menu item key](#) コマンドを使って処理されます。

追加のモディファイアは、Shift キー + Option キー (Mac OS) または Alt キー (Windows) です。モディファイアは、標準ショートカットが予め指定された時のみ使用できます。

コマンドによって返された数値は追加のモディファイアキーのコードに対応します。キーコードは次の通りです。

■ Shift キー = 512

■ Option キー (Mac OS) または Alt キー (Windows) = 2048

両方のキーが使われている場合、これらの値は合算されます。

注 "Event(Modifier)" テーマの **Shift key mask** 定数または **Option key mask** 定数を使用して返された値を評価できます。

メニュー項目に割り当てられたモディファイアキーがない場合、コマンドは 0 を返します。

menu に最後に追加された項目を特定するには、menuItem に -1 を渡します。

menu には、メニュー参照 (MenuRef) またはメニュー番号を渡すことができます。

メニュー参照を渡すと、process 引数は無意味なので渡しても無視されます。

メニュー番号を渡すは、コマンドはカレントプロセスのメインメニューバーに対応するメニューを考慮します。他のプロセスを指定するには、オプションの process 引数にその番号を渡します。

▼ [Get menu item key](#) コマンドの例を参照してください。

参照：[SET MENU ITEM SHORTCUT](#), [Get menu item key](#).

Dynamic pop up menu

Dynamic pop up menu (menu{; default{; xCoord; yCoord}}) → ItemRef

引数	タイプ	説明
menu	メニュー参照	→ メニュー参照
default	項目参照	→ デフォルトで選択されている項目の参照
xCoord	数値	→ 左上隅の X 座標
yCoord	数値	→ 左上隅の Y 座標
返り値	項目参照	← 選択されたメニュー項目参照

ダイナミック・ポップアップメニューを使用して、階層ポップアップメニューをマウス位置、またはオプションの xCoord 引数と yCoord 引数が

設定する位置に表示させることができます。

使用する階層メニューは、新しい [Create menu](#) コマンドを使って作成する必要があります。また、[Create menu](#) によって返された参照は、menu 引数に渡さなければなりません。

標準のインタフェースルールに基づいて、通常このコマンドは右クリックの時、またはボタンが一定の時間押し続けられた時 (例えば、コンテキストメニュー) に対応して呼び出されます。

メニューが表示される時はいつでも、オプションの default 引数を使用してポップアップメニューの項目をデフォルト選択に設定できます。この引数にはメニュー項目参照を渡します。この参照は、新しい [SET MENU ITEM REFERENCE](#) コマンドを使って予め設定されていなければなりません。

この引数を渡さないと、デフォルトでメニューの最初の項目が選択されます。

オプションの xCoord 引数と yCoord 引数を使用して、表示させるポップアップメニューの位置を指定できます。xCoord 引数と yCoord 引数には、メニュー左上隅のそれぞれ水平座標と垂直座標を渡します。これらの座標は、カレントフォームのローカルな座標によるピクセルで表記されなければなりません。両引数は一緒に渡す必要があります; いずれか1つが渡されたとしても、それは無視されます。

3D ボタンと関連付けられたポップアップメニューを表示する時は、オプションの xCoord 引数と yCoord 引数は渡してはいけません。この場合、4D はカレントのプラットフォームのインタフェース標準に従ってボタンに関するメニューの位置を自動的に計算します。

メニュー項目が選択されている場合、コマンドは (例えば、新しい [SET MENU ITEM REFERENCE](#) コマンドを使って定義されたような) その参照を返します。そうでない場合、コマンドは空の文字列を返します。

注 既存の Pop up menu コマンド ("ユーザインタフェース" テーマ) を使ってテキストに基づいたポップアップメニューを作成できます。これは 4D v11 では変更されていません。

参照: [SET MENU ITEM REFERENCE](#)

SET MENU ITEM REFERENCE

SET MENU ITEM REFERENCE (menu; menuItem; itemRef)

引数	タイプ	説明
menu	メニュー参照 倍長整数	→ メニュー参照、または メニュー番号
menuItem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
itemRef	文字列	→ 参照として割り当てる文字列

[SET MENU ITEM REFERENCE](#) コマンドは、menu 引数と menuItem 引数が指

定したメニュー項目に独自の参照を割り当てる時に使用します。
この参照は、主に [Dynamic pop up menu](#) コマンドによって使用されます。

注 メニューエディタで定義されたメニュー項目に参照を割り当てることもできます ([ページ 175](#)、" [メニュー項目参照](#) " の節参照)。

参照 : [Dynamic pop up menu](#)

Get menu item reference

Get menu item reference (menu; menuItem) → ItemRef

引数	タイプ	説明
menu	メニュー参照 数値	→ メニュー参照、またはメニュー番号
menuItem	倍長整数	→ メニュー項目の数、または menu に最後に追加された項目の場合は -1
返り値	項目参照	← メニュー項目参照

[Get menu item reference](#) コマンドは、menu 引数と menuItem 引数が指定したメニュー項目参照を返します。この参照は、[SET MENU ITEM REFERENCE](#) コマンドを使って予め設定されていなければなりません。

参照 : [SET MENU ITEM REFERENCE](#)

Get selected menu item reference

Get selected menu item reference → ItemRef

引数	タイプ	説明
		このコマンドは引数を必要としません。
返り値	項目参照	← メニュー項目参照

[Get selected menu item reference](#) コマンドは、選択されたメニュー項目参照を返します。この参照は、[SET MENU ITEM REFERENCE](#) コマンドを使って予め設定されていなければなりません。

メニュー項目が選択されていない場合、コマンドは空の文字列 "" を返します。

参照 : [SET MENU ITEM REFERENCE](#), [Menu selected](#)

修正されたコマンド

4D v11 のメニュー管理原則に従って、"メニュー" テーマの既存コマンドはすべて、menu 引数の参照として [Create menu](#) コマンドで返された 16 文字の文字列を受け入れるようになりました。この場合、コマンドは全プロセスにおけるメニューのインスタンスすべてに適用されます。

もちろん、メニューやメニューバーは以前と同様にそれぞれの番号を使って参照できます。しかし、このシンタックスは階層サブメニューをサポートしません。この場合も、コマンドはカレントプロセスの主要メ

メニューバーまたは process 引数が (適用されれば) 指定したプロセスだけに適用されます。
この節では、追加修正が行われた "メニュー" テーマのコマンドについてのみ詳細を説明しています。

コマンド名の変更

一貫性と明確性を高めるため、"メニュー" テーマの特定の既存コマンドは、次のように名前変更または移動しました。:

以前の名前 (4D 2004.x)	新しい名前 (4D v11.x)
HIDE MENU BAR	"ユーザインタフェース" テーマに移動
MENU BAR	SET MENU BAR
SET ABOUT	"ユーザインタフェース" テーマに移動
SET MENU ITEM KEY	SET MENU ITEM SHORTCUT
SHOW MENU BAR	"ユーザインタフェース" テーマに移動

SET MENU BAR

SET MENU BAR (menuBar{;process}{;*)

引数	タイプ	説明
menuBar	数値 Alpha メニュー参照	→ メニューバーの番号または名前または、メニュー参照
process	倍長整数	→ プロセス参照番号
*	*	→ Save menu bar state

注 4D の前バージョンでは、このコマンドは MENU BAR という名前でした。

SET MENU BAR コマンドは、menuBar 引数のメニュー参照 (MenuRef タイプ、16 文字の文字列) を受け入れるようになりました。4D v11 では、メニューをメニューバーとして使用することができ、その逆も同様です ([ページ 277](#)、"概要" の節を参照)。menuBar に MenuRef タイプの引数を渡すと、process 引数は不要なので無視される点に注意してください。
[Create menu](#) コマンドを使ってメニュー参照を生成できます。

Menu selected

Menu selected {(submenu)} → Longint

引数	タイプ	説明
subMenu	メニュー参照	← 選択された項目を含むメニュー参照

引数	タイプ	説明
返り値	倍長整数 ←	選択されたメニューコマンド 上位バイトにメニュー番号 下位バイトにメニュー項目番号

Menu selected コマンドは階層サブメニューと連動して使えるようになりました。第1レベルを超える階層メニュー項目を選択する場合、コマンドはオプションの subMenu 引数に、選択されたメニュー項目が属すサブメニューの参照 (MenuRef タイプ、16 文字の文字列) を返します。この引数を使用して階層サブメニューを管理できます。メニュー項目に階層サブメニューが含まれない場合、この引数は空の文字列を返します。

参照：[Get selected menu item reference](#)

SET MENU ITEM

SET MENU ITEM (menu; menuItem; itemText{; process})

引数	タイプ	説明
menu	数値； メニュー参照 →	メニュー番号 または、メニュー参照
menuItem	倍長整数 →	メニュー項目番号、または menu に最後に追加された項目の場合は 04
itemText	文字列 →	メニュー項目用の新しいテキスト
process	倍長整数 →	プロセス参照番号

SET MENU ITEM コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。menu に MenuRef タイプの引数を渡すと、process 引数は不要なので無視される点に注意してください。

menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を特定できます。

区切り線としてのメニュー項目を設定するには、itemText 引数に "-" の記号を渡します。

APPEND MENU ITEM

APPEND MENU ITEM (menu; itemText{; subMenu{; process})

引数	タイプ	説明
menu	数値； メニュー参照 →	メニュー番号 または、メニュー参照
itemText	文字列 →	新しいメニュー項目用のテキスト
subMenu	メニュー参照 →	項目に割り当てられたサブメニューの参照
process	倍長整数 →	プロセス参照番号

APPEND MENU ITEM コマンドは、menu 引数に MenuRef タイプの文字列を受け入れ、新しいオプションの subMenu 引数を含むようになりました。menu に

MenuRef タイプの引数を渡すと、process 引数は不要なので無視される点に注意してください。

subMenu 引数には、[Create menu](#) コマンドを使って作成されたメニューを指定する MenuRef タイプの文字列を渡すことができます。この場合、メニューはサブメニューとして追加されたメニュー項目に割り当てられます。

このコマンドはまた、[Create menu](#) コマンドを使って作成し、[SET MENU BAR](#) コマンドでインストールしたメニューバーとも連動します。

INSERT MENU ITEM

INSERT MENU ITEM (menu; afterItem; itemText{; subMenu{; process{;})

引数	タイプ	説明
menu	数値！ メニュー参照	→ メニュー番号 または、メニュー参照
afterItem	倍長整数	→ メニュー項目番号、または menu に最後に追加された項目の場合は -1
itemText	文字列	→ 挿入されるメニュー項目用のテキスト
subMenu	メニュー参照	→ 項目に割り当てられたサブメニューの参照
process	倍長整数	→ プロセス参照番号

INSERT MENU ITEM コマンドは、menu 引数に MenuRef タイプの文字列を受け入れ、新しいオプションの subMenu 引数を含むようになりました。menu に MenuRef タイプの引数を渡すと、process 引数は不要なので無視される点に注意してください。

[Create menu](#) コマンドを使って作成されたメニューを指定する subMenu 引数には、MenuRef タイプの文字列を渡すことができます。この場合、メニューはサブメニューとして挿入されたメニュー項目に割り当てられます。item0 Text 引数が複数の項目を含む場合、メニューは最初に挿入された項目に割り当てられます。

このコマンドはまた、[Create menu](#) コマンドを使って作成し、[SET MENU BAR](#) コマンドでインストールしたメニューバーとも連動します。

- ▼ 次は、メソッドが指定される 2 つのメニュー項目から成るメニューを作成する例です。

```
menuRef:=Create menu
APPEND MENU ITEM(menuRef;"Characters")
SET MENU ITEM METHOD(menuRef;1;"CharDialMgmt")
INSERT MENU ITEM(menuRef;1;"Paragraphs")
SET MENU ITEM METHOD(menuRef;2;"ParDialMgmt")
```

DELETE MENU ITEM

DELETE MENU ITEM(menu; menuitem; process)

引数	タイプ	説明
menu	数値 メニュー参照	→ メニュー番号 または、メニュー参照
menuitem	倍長整数	→ メニュー項目番号
process	倍長整数	→ プロセス参照番号

DELETE MENU ITEM コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。このタイプの引数を使用すると、process 引数は不要なので無視される点に注意してください。

menu 引数と menuItem 引数が指定したメニュー項目そのものが **Create menu** コマンドを使って作成したメニューの場合、DELETE MENU ITEM は menu にある menuItem のインスタンスだけを削除します。menuItem によって参照されたサブメニューはまだメモリに存在します。メモリから作成されたメニューを取り除くには、新しい **RELEASE MENU** コマンドを使う必要があります。

このコマンドはまた、**Create menu** コマンドを使って作成し、**SET MENU BAR** コマンドでインストールしたメニューバーとも連動します。

- ▼ 次は、カレントプロセスのメニューバーから 2 番目のメニューを取り除く例です。

```
DELETE MENU ITEM(Get menu bar reference;2)
```

- ▼ 次のメニュー構造を仮定します。

- メニューバー : File/Edit/Sales/Contacts

- Sales メニュー :

- Customers

- Orders

- Deliveries

- Sales メニューの各項目では、階層サブメニューは 2 つのコマンド、即ち「追加」と「修正」を提供します。

Sales メニューの Customers サブメニューを Contacts メニューに移動するには、次のように記述します。

```
ARRAY STRING(16;$MenuRef;0)
ARRAY STRING(32;$MenuTitles;0)
C_LONGINT($SalesMenuRef;$ContactsMenuRef;$DelMenuPos;
$DelMenuRef)
GET MENU ITEMS($MenuTitles;$MenuRef)
$SalesMenuRef:=$MenuRef{Find in array($MenuTitles;"Sales")}
$ContactsMenuRef:=$MenuRef{Find in array($MenuTitles;"Contacts")}
GET MENU ITEMS($SalesMenuRef;$MenuTitles;$MenuRef)
$DelMenuPos:=Find in array($MenuTitles;"Deliveries")
$DelMenuRef:=$MenuRef{$DelMenuPos}
```

```
APPEND MENU ITEM($ContactsMenuRef;"Customers";$DelMenuRef)
DELETE MENU ITEM($SalesMenuRef;$DelMenuPos)
```

ENABLE MENU ITEM

```
ENABLE MENU ITEM(menu; menuItem; process}}
```

引数	タイプ	説明
menu	数値！ メニュー参照	→ メニュー番号 または、メニュー参照
menuItem	倍長整数	→ メニュー項目番号、または menu に最後に追加された項目の場合は 04
process	倍長整数	→ プロセス参照番号

ENABLE MENU ITEM コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。このタイプの引数を使用すると、process 引数は不要なので無視される点に注意してください。

menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を特定できます。

menuItem 引数そのものがメニューの場合、このメニューの全項目と既存サブメニューのすべてが使用可能になります。

このコマンドはまた、[Create menu](#) コマンドを使って作成し、[SET MENU BAR](#) コマンドでインストールしたメニューバーとも連動します。

DISABLE MENU ITEM

```
DISABLE MENU ITEM (menu; menuItem; process}}
```

引数	タイプ	説明
menu	数値！ メニュー参照	→ メニュー番号 または、メニュー参照
menuItem	倍長整数	→ メニュー項目番号、または p hqx に最後に追加された項目の場合は 04
process	倍長整数	→ プロセス参照番号

DISABLE MENU ITEM コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。このタイプの引数を使用すると、process 引数は不要なので無視される点に注意してください。

menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を特定できます。

menuItem 引数そのものがメニューの場合、このメニューの全項目と既存サブメニューのすべてが使用不可になります。

このコマンドはまた、[Create menu](#) コマンドを使って作成し、[SET MENU BAR](#) コマンドでインストールしたメニューバーとも連動します。

SET MENU ITEM MARK

SET MENU ITEM MARK(menu; menuItem; mark{; process})

引数	タイプ	説明	
menu	数値	→	メニュー番号
	メニュー参照		または、メニュー参照
menuItem	倍長整数	→	メニュー項目番号、または p hqxに最後に追加された項目の場合は 04
mark	文字列	→	新しいメニュー項目マーク
process	倍長整数	→	プロセス参照番号

SET MENU ITEM MARK コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。このタイプの引数を使用すると、process 引数は不要なので無視される点に注意してください。

menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を特定できます。

menuItem 引数そのものがメニューの場合、このコマンドは何の効力も持ちません(サブメニューにマークを割り当てることはできません)。

Get menu item mark

Get menu item mark(menu; menuItem; process) → String

引数	タイプ	説明	
menu	数値	→	メニュー番号
	メニュー参照		または、メニュー参照
menuItem	倍長整数	→	メニュー項目番号、または p hqxに最後に追加された項目の場合は 04
process	倍長整数	→	プロセス参照番号
返り値	文字列	←	カレントメニュー項目マーク

Get menu item mark コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。このタイプの引数を使用すると、process 引数は不要なので無視される点に注意してください。

menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を特定できます。menuItem 引数そのものがメニューの場合、このコマンドは空の文字列を返します。

Get menu item key

Get menu item key (menu; menuItem; process) → Number

引数	タイプ	説明	
menu	数値	→	メニュー番号
	メニュー参照		または、メニュー参照

引数	タイプ	説明
menuItem	倍長整数	→ メニュー項目番号、または p hqxに最後に追加された項目の場合は 0 4
process 返り値	倍長整数 数値	→ プロセス参照番号 ← メニュー項目に割り当てられた標準 ショートカットキーの文字コード

Get menu item key コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。このタイプの引数を使用すると、process 引数は無意味なので無視される点に注意してください。

menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を特定できます。menuItem 引数そのものがメニューの場合、このコマンドは 0 を返します。

- ▼ 4D v11 では、次のタイプのプログラミング・ストラクチャを実装するとメニュー項目に割り当てられたショートカットを取得する時に役立ちます。

```
If(Get menu item key(mymenu;1) # 0)
  $modifiers:=Get menu item modifiers(mymenu;1)
  Case of
  : ($modifiers=Option key mask)
  ...
  : ($modifiers=Shift key mask)
  ...
  : ($modifiers=Option key mask + Shift key mask)
  ...
  End case
End if
```

参照：[SET MENU ITEM SHORTCUT](#), [Get menu item modifiers](#)

SET MENU ITEM SHORTCUT

SET MENU ITEM SHORTCUT(menu; menuItem; keyCode | keyName; modifiers | ; process|)

引数	タイプ	説明
menu	数値 メニュー参照	→ メニュー番号 または、メニュー参照
menuItem	倍長整数	→ メニュー項目番号、または p hqxに最後に追加された項目の場合は 0
keyCode keyName	数値 テキスト	→ キーボードショートカットの ASCII コード

引数	タイプ	説明
modifiers	倍長整数	→ ショートカットに割り当てられたモディファイア (keyCode が渡されると無視されます)
process	倍長整数	→ プロセス参照番号

注 4D の前バージョンでは、このコマンドの名前は SET MENU ITEM KEY でした。

SET MENU ITEM SHORTCUT コマンドは、menu 引数に MenuRef タイプの文字列を受け入れるようになりました。menu に menuRef 引数を渡すと、process 引数は無意味なので無視される点に注意してください。menuItem 引数の新機能に基づき、-1 を渡すと menu に最後に追加された項目を指定できます。

さらに、SET MENU ITEM SHORTCUT コマンドは、モディファイアキーを特定するための引数 (keyName) にテキストを受け入れるようになりました。この新しいシンタックスを使用すると、標準ショートカットに1つまたは両方の追加モディファイアを割り当てる時にオプションの modifiers 引数を使用できます。

例えば、Ctrl キー +U (Windows) または Command キー +U (Mac OS) ショートカットを指定する場合の "U" のように、keyName 引数にキー名を文字列で直接渡すことができます。

このシンタックスを使用すると、ショートカットに追加モディファイアを割り当てる時にオプションの modifiers 引数も渡すことができます。このように、Ctrl キー +Alt キー +Shift キー +Z (Windows) または Cmd キー +Option キー +Shift キー +Z (Mac OS) タイプのショートカットを指定できます。

これを行うには、次の値を modifiers に渡します。

- Shift キーには、512
- Option キー (Mac OS) または Alt キー (Windows) には、2048
- 両方のキーを割り当てるには、それぞれの値を合算します。

Ctrl キー (Windows) および Command キー (Mac OS) は、4D によって自動的にキーボードショートカットに追加される点に留意してください。

注 "Event(Modifier)" テーマの Shift key mask 定数および Option key mask 定数を使用して、渡す値を指定できます。

modifiers 引数は、モディファイアキーがその ASCII コード (以前のシンタックス) で指定された時に渡すと考慮されません。

注 以前のシンタックス (キーの ASCII コードに基づく) は、互換性の目的だけで保持されています。特に追加のモディファイアを指定する時に使える点から、新しいシンタックスはの使用を強くお勧めします。

- ▼ 次は、"Underline" メニュー項目のショートカット、Ctrl キー +Shift キー +U (Windows) および Cmd キー +Shift キー +U (Mac OS) を定義します。

```
SET MENU ITEM(menuRef;1;"Underline")
SET MENU ITEM SHORTCUT(menuRef;1;"U";Shift key mask)
```

参照：[Get menu item key](#), [Get menu item modifiers](#)

階層リスト

階層リストは、よりパワフルかつ柔軟に使用できるよう、またその処理が 4D の他のフォームオブジェクトにより適合するよう、4D v11 では構造的に変更されました。

4D v11 における階層リストの操作はほとんど以前のバージョンと同じです。しかし、以前の制限は取り除かれました。

- 同じフォームに複数の階層リストを作成できるようになりました。
- 修正されたリストを更新するために REDRAW LIST コマンドを実行する必要がなくなりました；この更新は自動的に行われるようになりました。
- リスト項目の長さの最大が 255 文字から 20 億文字以上 (テキストフィールドの最大サイズ) に拡張されました。
- リストに "透過" の属性を与えられるようになりました。
- 水平のスクロールバーを使えるようになりました。

さらに、次のような新しい機能が提供されています。

- 同じリストの、1 つまたはそれ以上のフォームによる複数表示。
- 既存のオブジェクト管理コマンドを使ってフォームのコンテキストで特定のプロパティ (セクション、フォント、スクロールバーなど) を修正できるデフォルト設定。
- リスト項目のフォントを変更できる機能。
- リスト内の検索機能。
- 項目アイコンに変数を使用できる機能。

これらの新しい機能については、下記で説明しています。

複数表示

4D v11 は、ListRef で参照される 1 ランゲージオブジェクトとしての階層リストと、オブジェクト名で参照される フォームオブジェクトとしての階層リストとを区別します。

ランゲージオブジェクトのメモリ上のインスタンスは 1 つだけで、そのインスタントに行われた変更はすべて、それが使用されている他の場所すべてに即座に反映されます。

一方、同じ階層リストを同じフォームまたは別のフォームで、複数表示することができます。リストの各表示はそれぞれ、独自の特性を持つと共に、他のすべての表示と共有する共通の特性を持っています。

次にあげる特性は、リストの各表示に特有なものです。

- 選択された項目
- リスト項目の展開された / 折りたたまれた状態
- スクロールカーソルの位置

他の特性 (フォント、フォントサイズ、スタイル、入力制御、カラー、リスト内容、アイコンなど) はすべての表示に共通で、個別に変更できません。

位置関連コマンドの新しいシンタックス

複数表示により、項目の位置に関連するコマンド (例えば、SELECT LIST ITEMS BY POSITION) やカレント項目に関するコマンドの使い方が変更されました。実際、同じリストが複数表示されると、各表示にはそれぞれの「展開された / 折りたたまれた項目」の構造とそれぞれのカレント項目ができる場合があります。例えば (* が省略された時の) Count list items のように、特定のコマンドは正にこの構造に基づいているため、明確な方法で、対象の階層リストを指定できるということが重要です。

このため、表示に関する "階層リスト" テーマのコマンドは、ListRef 参照 (倍長整数) の代わりに、リストのフォームオブジェクト名 (文字列) を最初の引数として受け入れるようになりました。新しいシンタックスは、フォームオブジェクトに適用されるコマンドのシンタックスと一致します。

COMMAND(*;"ObjectName"; parameters...)

例えば:

SELECT LIST ITEMS BY POSITION (*;"mylist1";5;\$posArray)

フォームの複数の階層リストで作業していて、その内の 1 つの表示上でのみ操作や情報収集を行う場合には、このシンタックスを使用すべきです。この場合、ListRef に基づいたシンタックスは正確にリストを指定できないため避ける必要があります。

下記の階層リスト関連コマンドは、この代替シンタックスを受け入れません。

SET LIST ITEM

GET LIST ITEM

DELETE FROM LIST (see Note)

SELECT LIST ITEMS BY POSITION

Count list items

Selected list items

INSERT IN LIST (see Note)

List item parent

List item position

SET LIST ITEM PROPERTIES

GET LIST ITEM PROPERTIES

[SET LIST ITEM FONT](#) (4D v11 の新しいコマンド)

[Get list item font](#) (4D v11 の新しいコマンド)

[Find in list](#) (4D v11 の新しいコマンド)

[SET LIST ITEM ICON](#) (4D v11 の新しいコマンド)

[GET LIST ITEM ICON](#) (4D v11 の新しいコマンド)

SET LIST ITEM PARAMETER (4D v11 の新しいコマンド)

GET LIST ITEM PARAMETER (4D v11 の新しいコマンド)

- 注 DELETED FROM LIST コマンドと INSERT IN LIST コマンドは、それぞれ DELETE LIST ITEM コマンドと INSERT LIST ITEM コマンドから変更された新しいコマンド名です。これについては、[ページ 387](#)、"[名称の変更](#)" の節を参照してください。

プロパティを設定するコマンドの場合、オブジェクト名に基づいたシンタックスというのは、指定されたフォームオブジェクトだけがコマンドで修正されるのではなく、コマンドの動作はこのオブジェクトの状態に基づくという意味であることに留意してください。階層リストに共通する特性は常にすべての表示で変更されます ([ページ 298](#)、"[複数表示](#)" の節を参照)。例えば、ステートメント `SET LIST ITEM FONT(*;"mylist1";*;thefont)` を実行すると、これは mylist1 フォームオブジェクトに割り当てられた階層リストにある項目のフォントを変更したいことを示します。コマンドは項目の変更を設定するため mylist1 オブジェクトのカレント項目を考慮しますが、この変更は全プロセスにおいてリストの表示すべてで実行されます。

@ のサポート

他のオブジェクトプロパティ管理関連のコマンド同様、ObjectName 引数に "@" の記号を使用できます。原則として、このシンタックスはフォーム内のオブジェクトのセットを指定する時に使用します。ただし、階層リスト関連コマンドのコンテキストでは、これがすべての場合に適用される訳ではありません。下記の通り、このシンタックスはコマンドのタイプによって 2 つの違った効力を持ちます。

- プロパティを設定するコマンドに対して、このシンタックスはオブジェクト名が対応するすべてのオブジェクトを指定します (標準動作)。例えば、引数 "LH@" は名前が "LH" で始まる階層リストタイプのすべてのオブジェクトを指定します。

該当するコマンドは次の通りです。

```
DELETE FROM LIST
INSERT IN LIST
SELECT LIST ITEMS BY POSITION
SET LIST ITEM
SET LIST ITEM FONT
SET LIST ITEM ICON
SET LIST ITEM PARAMETER
SET LIST ITEM PROPERTIES
```

- プロパティを取得するコマンドに対して、このシンタックスはオブジェクト名が対応する最初のオブジェクトを指定します。該当するコマンドは次の通りです。

Count list items
[Find in list](#)
 GET LIST ITEM
[Get list item font](#)
 GET LIST ITEM ICON
 GET LIST ITEM PARAMETER
 GET LIST ITEM PROPERTIES
 List item parent
 List item position
 Selected list items

プロパティ関連コマンドの優先順位

階層リストの特定のプロパティ (例えば、"入力可能" 属性やカラー) は、次の3つの方法で設定できます: デザインモードのプロパティリストを使用する、"オブジェクトプロパティ" テーマのコマンドを使用する、または "階層リスト" テーマのコマンドを使用する。

リストのプロパティ設定にこれら3つの方法が用いられる場合、次の優先順位が適用されます。

1. "階層リスト" テーマのコマンド
2. "オブジェクトプロパティ" テーマのコマンド
3. プロパティリスト関連の引数。

この原則はコマンドが呼び出される順番に関係なく適用されます。ある項目のプロパティを階層リストコマンドで単独に変更した場合、同等のオブジェクトプロパティコマンドは、後で呼び出されてもこの項目に対して効力を持ちません。例えば、SET LIST ITEM PROPERTIES コマンドを使って項目の色を変更した場合、SET COLOR コマンドはこの項目に対して効力を持ちません。

階層リストに使用可能な既存のコマンド

4D v11 では、フォームにおける階層リストインタフェースの管理が改善されています。一般的な 4D コマンドを使ってフォームの階層リスト表示の外観を変更できるようになりました。これらのコマンドには、階層リストオブジェクトの名前 (* 引数を使用) またはその変数名 (標準シンタックス) のいずれかを渡さなければなりません。

注 階層リストの場合、フォーム変数は ListRef の値を含みます。階層リストに割り当てられた変数を渡して属性を変更するコマンドを実行することは、つまり ListRef を渡すことと同じです。オブジェクト名によってのみ、表示を個別に指定できます。

FONT, FONT STYLE, FONT SIZE

FONT({*; }object; font)
 FONT STYLE({*; }object; styles)
 FONT SIZE({*; }object; size)

FONT コマンド、FONT STYLE コマンド、および FONT SIZE コマンドを使って、フォームの階層リスト表示の (それぞれ) フォント、フォントスタイル、

およびフォントサイズを変更できるようになりました。リストとそのサブリストすべてが変更されます。

これらのコマンドは、対応する、フォームレベルで階層リストに設定されたフォントプロパティを置き換えます。一方、これらは "階層リスト" テーマのコマンドで変更された項目に対して効力を持ちません。

テーマ: オブジェクトプロパティ

SET SCROLLBAR VISIBLE

SET SCROLLBAR VISIBLE(*; |object; horizontal; vertical)

SET SCROLLBAR VISIBLE コマンドを使って、フォームの階層リストの水平 / 垂直スクロールバーを表示または隠すことができるようになりました。

テーマ: オブジェクトプロパティ

SCROLL LINES

SCROLL LINES(*; |object; position; *)

SCROLL LINES コマンドにより、フォームの階層リスト表示のラインをスクロールして最初に選択した項目や特定の項目を表示できるようになりました。オブジェクトのシンタックスを使う場合は、対応するフォームオブジェクトだけが変更されます (スクロールバーカーソルの位置はリストの各表示によって異なる特性です)。

ラインの数を指定するために position 引数を使用する場合、コマンドは項目の展開された / 折りたたまれた状態を考慮します。

テーマ: ユーザーインターフェース

注 4D の前バージョン同様、SET COLOR コマンド、SET RGB COLORS コマンド、SET FILTER コマンド、および SET ENTERABLE コマンドは、前述の優先順位ルールに従って階層リストに作用します。

変更されたコマンド

4D v11 では、"階層リスト" テーマの次のコマンドが変更されました。

REDRAW LIST

REDRAW LIST(list)

REDRAW LIST コマンドは 4D v11 では必要ありません。階層リストの表示すべては自動的に再描画されるようになりました。このコマンドは呼び出されても何もしなくなっています。

新しいコマンド

"階層リスト" テーマに、階層リスト (ListRef) を管理する新しいコマンドがいくつか追加されました。

SET LIST ITEM FONT

SET LIST ITEM FONT({*; }list; itemRef !*; font)

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名(文字列) 省略した場合、リストはリスト参照番号
list	リスト参照 ; 文字列	→ リストタイプのオブジェクト名(*を指定した場合) リスト参照番号(*を省略した場合)
itemRef !*	倍長整数 !*	→ 項目参照番号、または リストに最後に追加された項目の場合 は0、または 現在選択されているリスト項目の場合 は*
font	文字列 ; 数値	→ フォント名または番号

SET LIST ITEM FONT コマンドは、参照番号またはオブジェクト名が渡されたリストの itemRef 引数が指定する項目の文字フォントを変更します。

最初のオプションの * 引数を渡すということは、list 引数はフォームのリスト表示に対応するオブジェクト名(文字列)であることを示します。この引数を渡さないということは、list 引数は階層リスト参照(ListRef)であることを示します。

リストの1つの表示だけを使う場合、または structural 項目で作業する場合(2番目の*を省略)、いずれかのシンタックスを使用できます。

反対に、同じリストの複数の表示を使う場合、またはカレント項目で作業する場合(2番目の*を渡す)、各表示はそれぞれのカレント項目を持つため、オブジェクト名に基づいたシンタックスが要求されます。

itemRef に参照番号を渡すことができます。この番号がリストのどの項目にも対応しない場合、コマンドは何もしません。

(APPEND TO LIST を使って) リストに最後に追加された項目の変更を要求するために、itemRef に0を渡すこともできます。

最後に、itemRef に*を渡すことができます: この場合、コマンドはリストのカレント項目に適用されます。複数の項目が手動で選択された場合、カレント項目は最後に選択された項目になります。どの項目も選択されない場合、コマンドは何もしません。

font 引数には、使用するフォントの名前または番号を渡します。階層リストのデフォルトフォントを再度適用するには、font に空の文字列を渡します。

- ▼ 次は、リストのカレント項目にフォントの Times を適用する例です。

```
SET LIST ITEM FONT(*,"Mylist";*,"Times")
```

参照: [Get list item font](#)

Get list item font Get list item font({*; }list; itemRef | *) → String

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名(文字列) 省略した場合、リストはリスト参照番号
list	リスト参照 文字列	→ リストタイプのオブジェクト名(*を指定した場合) リスト参照番号(*を省略した場合)
itemRef *	倍長整数 *	→ 項目参照番号、または0の場合、リストに最後に追加された項目、または*の場合、現在選択されているリスト項目
返り値	文字列	← フォント名

Get list item font コマンドは、list 引数で指定されたリスト中、itemRef で指定された項目の、現在のフォント名を返します。

第一引数にオプションの * を渡した場合、list 引数にはフォーム上のリストに与えられたオブジェクト名(文字列)を渡します。この引数を省略した場合、list 引数にはリスト参照を渡します。

リストを一つしか表示していない場合、または2番目の * を使用しない場合、list 引数にはいずれの記述も使用できます。

対して、同一のリストをフォーム上で複数使用していて、2番目の * 引数を使用する場合は、オブジェクト名を使用してリストを指定しなければなりません。それぞれのリストが異なるカレント項目をもつことができるからです。

itemRef には参照番号を渡すことができます。この番号がいずれの項目にも対応しない場合、コマンドは何も行いません。

(APPEND TO LIST) を使用してリストに最後に追加された項目のフォントを取得するには、itemRef に 0 を渡します。

さらに itemRef に * を渡すことができます。この場合、コマンドは現在選択されている項目のフォントを返します。複数の項目が選択されている場合、最後に選択された項目が対象となります。項目が選択されていない場合、コマンドは何も行いません。

参照：[SET LIST ITEM FONT](#)

Find in list

Find in list({*; }list; value; scope; itemsArray){; *}) → Longint

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名(文字列) 省略した場合、リストはリスト参照番号
list	リスト参照; 文字列	→ リストタイプのオブジェクト名(*を指定した場合) リスト参照番号(*を省略した場合)
value	文字列	→ 検索される値
scope	整数	→ 0= 主要リスト、1= サブリスト
itemsArray	倍長整数配列	← ・ 2 番目の * を省略した場合: 見つかった項目のポジションの配列 ・ 2 番目の * を渡した場合: 見つかった項目の参照番号の配列
*	*	→ ・ 省略した場合: 項目のポジションを使用 ・ 渡した場合: 項目の参照番号を使用
返り値	倍長整数	← ・ 2 番目の * を省略した場合: 見つかった項目のポジション ・ 2 番目の * を渡した場合: 見つかった項目の参照番号

Find in list コマンドは、value に渡された文字列に最初に一致する項目の位置または参照を返します。複数の項目が一致する場合、itemsArray に検索されたすべて項目の位置または参照が返されます。

第一引数にオプションの * を渡した場合、list 引数にはフォーム上のリストに与えられたオブジェクト名(文字列)を渡します。この引数を省略した場合、list 引数にはリスト参照を渡します。

リストを一つしか表示していない場合、または 2 番目の * を使用しない場合、list 引数にはいずれの記述も使用できます。

対して、同一のリストをフォーム上で複数使用していて、2 番目の * 引数を使用する場合は、オブジェクト名を使用してリストを指定しなければなりません。それぞれのリストが異なるカレント項目をもつことができるからです。

2 番目の * 引数を省略した場合、項目の位置が返されます。渡した場合は項目参照番号が返されます。

value には、検索する文字列を渡します。検索は完全一致で行われます。たとえば "wood" を検索したばあい、"wooden" 検索されません。しかしワイルドカード "@" を使用すれば前方一致、後方一致、あるいは含む検索を行うことができます。

scope 引数にはリストの第一レベルを検索対象とするか、あるいはサブリ

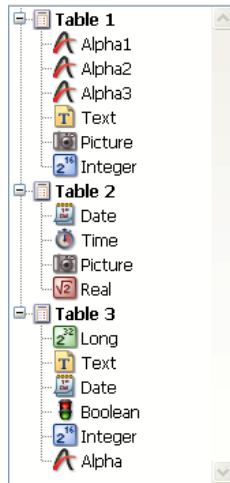
ストも対象とするかを指定できます。0を渡すと、第一レベルのみが検索されます。1を渡すとすべてのサブリストが検索されます。

value に対応するすべての項目を検索したい場合は、オプションの引数 itemsArray に倍長整数配列を渡します。必要に応じて配列が作成され、サイズが調整されます。2番目の * 引数を渡していれば配列には位置が返され、省略した場合参照番号が返されます。

位置はメインリストの最初の項目からの相対位置であらわれます。つまり、リストやサブリストの現在の拡張 / 折りたたみ状態が考慮されません。

項目が検索されなかった場合、コマンドは 0 を返し、itemsArray は空になります。

▼ 以下の階層リストがある時、:



```
$vItemPos:=Find in list(hList;"P@";1; $arrPos)
  `vItemPos @ 6
  `sarrPos{1} @ 6, `sarrPos{2} @ 11
```

```
$vItemRef:=Find in list(hList;"P@";1;$arrRefs;*)
  `vItemRef @ 7
  `sarrRefs{1} @ 7, `sarrRefs{2} @ 18
```

```
$vItemPos:=Find in list(hList;"Date";1;$arrPos)
  `vItemPos @ 9
  `sarrPos{1} @ 9, `sarrPos{2} @ 16
```

```
$vItemRef:=Find in list(hList;"Date";1;$arrRefs;*)
  `vItemRef @ 11
  `sarrRefs{1} @ 11, `sarrRefs{2} @ 23
```

```
$vItemPos:=Find in list(hList;"Date";0)
```

```
`$vlItemPos @ 0
```

SET LIST ITEM ICON

```
SET LIST ITEM ICON({*; }list; itemRef |*; icon)
```

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名(文字列) 省略した場合、リストはリスト参照番号
list	リスト参照 文字列	→ リストタイプのオブジェクト名(*を指定した場合) リスト参照番号(*を省略した場合)
itemRef *	倍長整数 *	→ 項目参照番号、または 0の場合、リストに最後に追加された項目、または *の場合、現在選択されているリスト項目
icon	ピクチャ	→ 項目に割り当てられたアイコン

SET LIST ITEM ICON コマンドは、list で指定されたリスト中、itemRef で指定された項目に割り当てるアイコンを変更するために使用します。

注 SET LIST ITEM PROPERTIES コマンドを使用して、すでに項目に割り当てるアイコンを変更することが可能でした。しかしこの新しいコマンドと異なり、SET LIST ITEM PROPERTIES コマンドは(リソース参照やピクチャライブラリのピクチャなど)スタティックなピクチャリファレンスのみを受け入れます。

第一引数にオプションの*を渡した場合、list 引数にはフォーム上のリストに与えられたオブジェクト名(文字列)を渡します。この引数を省略した場合、list 引数にはリスト参照を渡します。

リストを一つしか表示していない場合、または2番目の*を使用しない場合、list 引数にはいずれの記述も使用できます。

対して、同一のリストをフォーム上で複数使用していて、2番目の*引数を使用する場合は、オブジェクト名を使用してリストを指定しなければなりません。それぞれのリストが異なるカレント項目をもつことができるからです。

itemRef には参照番号を渡すことができます。この番号がリスト中の項目に対応しない場合、コマンドは何も行いません。(APPEND TO LIST を使用して、)リストに最後に追加された項目を指定するには、0を渡します。さらに itemRef に*を渡して、現在選択されている項目を指定することができます。複数の項目が選択されている場合、最後に選択された項目が対象となります。項目が選択されていない場合、コマンドは何も行いません。

icon 引数には有効な 4D のピクチャ参照(フィールド、変数、ポインタ等

)を渡します。ピクチャは項目の左に表示されます。
階層リストの最適化のため、ポインタの利用を推奨します。この場合、異なる項目に同じアイコンが使用されても、メモリ上にひとつのインスタンスのみが作成されます。

注 逆に、GET ICON RESOURCE や GET PICTURE RESOURCE コマンドで生成された変数を直接使用した場合、アイコンは項目ごとメモリ上に複製されるため、お勧めしません。

▼ 以下のコードはポインタを使用しているので最適化されています。

```
vIcon:=[Params]Icon
SET LIST ITEM ICON(mylist;ref1;->vIcon)
SET LIST ITEM ICON(mylist;ref2;->vIcon)
```

参照：GET LIST ITEM ICON

GET LIST ITEM ICON

GET LIST ITEM ICON({*; }list; itemRef !*; icon)

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名(文字列) 省略した場合、リストはリスト参照番号
list	リスト参照 ! 文字列	→ リストタイプのオブジェクト名(*を指定した場合) リスト参照番号(*を省略した場合)
itemRef !*	倍長整数 !*	→ 項目参照番号、または 0の場合、リストに最後に追加された項目、または *の場合、現在選択されているリスト項目
icon	ピクチャ変数	← 項目に割り当てられたアイコン

GET LIST ITEM ICON コマンドは、list 引数で指定されたリスト中、item0 Ref に参照番号を渡して指定された項目のアイコンを返します。

第一引数にオプションの * を渡した場合、list 引数にはフォーム上のリストに与えられたオブジェクト名(文字列)を渡します。この引数を省略した場合、list 引数にはリスト参照を渡します。

リストを一つしか表示していない場合、または2番目の * を使用しない場合、list 引数にはいずれの記述も使用できます。

対して、同一のリストをフォーム上で複数使用していて、2番目の * 引数を使用する場合は、オブジェクト名を使用してリストを指定しなければなりません。それぞれのリストが異なるカレント項目をもつことができます。

itemRef には参照番号を渡すことができます。この番号がリスト中の項目に対応しない場合、コマンドは何も行いません。(APPEND TO LIST を使用して、) リストに最後に追加された項目を指定するには、0 を渡します。さらに itemRef に * を渡して、現在選択されている項目を指定することができます。複数の項目が選択されている場合、最後に選択された項目が対象となります。項目が選択されていない場合、コマンドは何も行いません。

icon 引数にはピクチャ変数を渡します。コマンドが実行されると、アイコンのソース (スタティックピクチャ、リソースまたはピクチャ表現) にかかわらず、項目に割り当てられたアイコンが返されます。

項目にアイコンが割り当てられていない場合、ピクチャ変数は空になります。

参照：[SET LIST ITEM ICON](#)

SET LIST ITEM PARAMETER

SET LIST ITEM PARAMETER({*; }list; itemRef !*; selector; value)

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名 (文字列) 省略した場合、リストはリスト参照番号
list	リスト参照 ; 文字列	→ リストタイプのオブジェクト名 (* を指定した場合) リスト参照番号 (* を省略した場合)
itemRef !*	倍長整数 !*	→ 項目参照番号、または 0 の場合、リストに最後に追加された項目、または * の場合、現在選択されているリスト項目
selector	文字列	→ パラメタ定数
value	文字列 ; ブール ! 数値	→ パラメタ値

SET LIST ITEM PARAMETER コマンドは、list で指定されたリスト中、item0 Ref で指定した項目の、selector に対応するパラメタを変更するために使用します。

第一引数にオプションの * を渡した場合、list 引数にはフォーム上のリストに与えられたオブジェクト名 (文字列) を渡します。この引数を省略した場合、list 引数にはリスト参照を渡します。

リストを一つしか表示していない場合、または 2 番目の * を使用しない場合、list 引数にはいずれの記述も使用できます。

対して、同一のリストをフォーム上で複数使用していて、2 番目の * 引数を使用する場合は、オブジェクト名を使用してリストを指定しなければなりません。それぞれのリストが異なるカレント項目をもつことができ

るからです。

itemRefには参照番号を渡すことができます。この番号がリスト中の項目に対応しない場合、コマンドは何も行いません。

(APPEND TO LIST を使用して、) リストに最後に追加された項目を指定するには、0 を渡します。

さらに itemRef に * を渡して、現在選択されている項目を指定することができます。複数の項目が選択されている場合、最後に選択された項目が対象となります。項目が選択されていない場合、コマンドは何も行いません。

selector には、Additional text 定数 ("Hierarchical Lists" テーマ) またはカスタム値を渡せます。

- | 定数 | タイプ | 値 |
|-----------------|-----|--------------------|
| Additional text | 文字 | 4D_additional_text |
- Additional text: この定数は、itemRef で指定した項目テキストの右側に、テキストを追加するために使用します。この追加のテキストは、ユーザが水平スクロールバーを移動しても、常にリストの右側に表示されます。この定数を指定する場合、value にはテキストを渡します。
 - カスタム値: さらに、selector にカスタムテキストを渡して、それをテキスト、数値またはブール型のパラメタに結びつけることができます。この値はリスト項目とともに格納され、[GET LIST ITEM PARAMETER](#) コマンドを使用して取り出すことができます。
- この機能を使用して、階層リストに結び付けられたインターフェースを作成することができます。たとえば顧客名のリストがあって、それぞれの顧客ごとに年齢を格納し、項目が選択されたらそれを表示させることができます。

参照: [GET LIST ITEM PARAMETER](#)

GET LIST ITEM PARAMETER

GET LIST ITEM PARAMETER({*; }list; itemRef | *; selector; value)

引数	タイプ	説明
*	*	→ 指定した場合、リストはオブジェクト名 (文字列) 省略した場合、リストはリスト参照番号
list	リスト参照 文字列	→ リストタイプのオブジェクト名 (* を指定した場合) リスト参照番号 (* を省略した場合)

引数	タイプ	説明
itemRef ! *	倍長整数 ! *	→ 項目参照番号、または 0 の場合、リストに最後に追加され た項目、または * の場合、現在選択されているリスト 項目
selector	文字列	→ パラメタ定数
value	文字列 ! ブール ! 数値	← パラメタ値

GET LIST ITEM PARAMETER コマンドは、list 引数で指定された階層リスト
中、itemRef で特定されたリスト項目の、selector に対応するパラメタ値を取得す
るために使用します。

第一引数にオプションの * を渡した場合、list 引数にはフォーム上のリス
トに与えられたオブジェクト名 (文字列) を渡します。この引数を省略し
た場合、list 引数にはリスト参照を渡します。

リストを一つしか表示していない場合、または 2 番目の * を使用しない場
合、list 引数にはいずれの記述も使用できます。

対して、同一のリストをフォーム上で複数使用していて、2 番目の * 引数
を使用する場合は、オブジェクト名を使用してリストを指定しなければ
なりません。それぞれのリストが異なるカレント項目をもつことができ
るからです。

selector には、Additional text 定数 ("Hierarchical Lists" テーマ) またはカ
スタム値を渡せます。

selector に関する詳細は、[ページ 309](#)、[コマンド SET LIST ITEM PARAMO
ETER](#) の説明を参照してください。

参照：[SET LIST ITEM PARAMETER](#)

LIST OF CHOICE LISTS

LIST OF CHOICE LISTS(numsArray;namesArray)

引数	タイプ	説明
numsArray	倍長整数配列	← 選択リストの数
namesArray	テキスト配列	← 選択リストの名前

LIST OF CHOICE LISTS コマンドは、デザインモードのリストエディタ
で作成された選択リストの番号と名前を numsArray と namesArray に返
します。

選択リストの番号は、作成順に付けられます。(リストエディタでは、選
択リストは名前で並び替えて表示されます。)

リストボックス

4D v11 では、フィールドや計算式をリストボックスの列に割り当てるこ
とが可能です。この新しい機能は[ページ 142](#)、"[フィールドまたは式とリ
ストボックスとの関連付け](#)" の節で説明されています。

それに伴い、"リストボックス"テーマのコマンドが更新され、また二つの新しいコマンドが追加されました。

なお、リストボックスに関連する他のコマンドも 4D v11 に追加されたり更新されていることに留意してください。

- 新しい **Choose** コマンドが"ツール"テーマに追加されました。式の挿入にとっても便利です。
- **Displayed line number** コマンドはリストボックスでも動作するようになりました。
- **表示フォーマット定数** は計算列をサポートするようになりました。

変更されたコマンド

INSERT LISTBOX COLUMN

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

引数	タイプ	説明
*		→ 指定した場合、オブジェクトはオブジェクト名(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	→ オブジェクト名(*を指定した場合)、または変数(*を省略した場合)
colPosition	数値	→ 挿入するカラムの場所
colName	文字列	→ カラムオブジェクトの名前
colVariable	配列 フィールド 変数	→ カラム配列名、またはフィールド、または変数
headerName	文字列	→ カラムヘッダオブジェクトの名前
headerVar	整数の変数	→ カラムヘッダ変数

colVariable 引数にフィールドや引数を渡すことができるようになりました。この場合、列の内容はフィールドや変数の値となり、リストボックスに結び付けられたレコードごとに値が決定されます。この指定は、リストボックスの"データソース"プロパティにカレントセレクションまたは命名セレクションが指定されているときに利用可能です。

注 一つのリストボックス中で、カラムに配列(配列データソース)と、フィールドや変数(セレクションデータソース)を組み合わせることはできません。

文字、テキスト、数値、日付、時間、ピクチャ、ブール型のフィールドや変数を指定できます。

セレクションに基づくリストボックスのコンテキストでは、INSERT LISTBOX COLUMN を使用して単純な要素 (フィールドや変数) を挿入することができます。より複雑な式 (フォーミュラやメソッド) を扱う場合は、[ページ 314](#)、[コマンド INSERT LISTBOX COLUMN FORMULA](#) を使用しなければなりません。

- ▼ リストボックスの一番右に列を追加し、[Transmitted]Fees フィールドを割り当てる例:

```
$last:=Get number of listbox columns("ListBox1")+1
INSERT LISTBOX COLUMN("ListBox1";$last;"FieldCol";
[Transmitted]Fees;"HeaderName";HeaderVar)
```

INSERT LISTBOX ROW

INSERT LISTBOX ROW({*; }object; position)

このコマンドは、配列内容を表示するリストボックスに対してのみ有効です。セレクションを表示するリストボックスにこのコマンドを実行しても、効果はなく、OK システム変数には 0 が設定されます。

DELETE LISTBOX ROW

DELETE LISTBOX ROW({*; }object; position)

このコマンドは、配列内容を表示するリストボックスに対してのみ有効です。セレクションを表示するリストボックスにこのコマンドを実行しても、効果はなく、OK システム変数には 0 が設定されます。

GET LISTBOX ARRAYS

GET LISTBOX ARRAYS({*; }object; arrColNames; arrHeaderNames; arrColVars; arrHeaderVars; arrVisible; arrStyles)

引数	タイプ	説明
*		→ 指定した場合、オブジェクトはオブジェクト名 (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	→ オブジェクト名 (* を指定した場合)、または変数 (* を省略した場合)
arrColNames	文字列配列	← カラムオブジェクト名
arrHeaderNames	文字列配列	← ヘッダオブジェクト名
arrColVars	ポインタ配列	← 列変数へのポインタ、または列フィールドへのポインタ、または NIL
arrHeaderVars	ポインタ配列	← ヘッダ変数へのポインタ
arrVisible	ブール配列	← 列ごとの表示 / 非表示設定
arrStyles	ポインタ配列	← 配列へのポインタまたはスタイルやカラー変数または Nil

リストボックスのデータソースが "セクション" である場合、arrCol0 Vars 配列の内容はそれぞれの列に割り当てられたソースのタイプによって異なります。

- フィールドが割り当てられた列の場合、arrColVars には割り当てられたフィールドへのポインタが返されます。
- 変数が割り当てられた列の場合、arrColVars には割り当てられた変数へのポインタが返されます。
- 式が割り当てられた列の場合、arrColVarsにはNilポインタが返されます。セクションに基づくリストボックスの場合、arrStyles 引数にはスタイル設定に使用されるソースタイプにより3つのポインタが返されます。
- 変数で設定されている場合、arrStyles には変数へのポインタが返されます。
- 式で設定されている場合、arrStyles には Nil ポインタが返されます。

新しいコマンド

INSERT LISTBOX COLUMN FORMULA

INSERT LISTBOX COLUMN FORMULA ({*; }object; colPosition; col0 Name; formula; dataType; headerName; headerVariable)

引数	タイプ	説明
*		→ 指定した場合、オブジェクトはオブジェクト名(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	→ オブジェクト名(*を指定した場合)、または変数(*を省略した場合)
colPosition	数値	→ 列の挿入場所
colName	文字	→ 列オブジェクト名
formula	文字	→ 列に割り当てる 4D フォーミュラ
dataType	倍長整数	→ フォーミュラの結果の型
headerName	文字	→ 列ヘッダオブジェクト名
headerVariable	整数変数	→ 列ヘッダ変数

新しい INSERT LISTBOX COLUMN FORMULA コマンドは [INSERT LISTBOX COLUMN](#) コマンドと同じですが、列の内容としてフォーミュラを設定できる点が異なります。

このタイプの内容は、リストボックスの "データソース" プロパティがカレントセクションまたは命名セクションの場合に利用できます。

formula 引数には有効な式を指定できます。

- 命令
- フォーマエディタを使用して生成されたフォーミュラ
- 4D コマンドの呼び出し
- プロジェクトメソッドの呼び出し

このコマンドが呼び出されると、フォーミュラが解析され、実行されます。

注 アプリケーションランゲージに依存しないフォーミュラを定義するには、Command name コマンドを使用します (フォーミュラで 4D コマンドを使用する場合)。

dataType 引数には、フォーミュラ実行結果のデータの型を定義するために使用します。この引数には "Field and Variable Types" テーマの定数を渡します。

定数	型	値
Is Real	倍長整数	1
Is Text	倍長整数	2
Is Picture	倍長整数	3
Is Date	倍長整数	4
Is Boolean	倍長整数	6
Is Time	倍長整数	11

フォーミュラの結果が期待されるデータ型に一致しない場合、エラーが生成されます。

他の引数は [INSERT LISTBOX COLUMN](#) コマンドと同じです。

- ▼ リストボックスの一番右に列を追加し、従業員の年齢を計算するフォーミュラを割り当てる例:

```
vAge:="Current Date-[Employees]BirthDate)\365"
$last:=Get number of columns(*;"ListBox1")+1
INSERT LISTBOX COLUMN FORMULA(*;"ListBox1";$last;
"ColFormula";Is_Real;vAge;"Age";HeaderVar)
```

参照: [INSERT LISTBOX COLUMN](#)

SET LISTBOX TABLE SOURCE

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

引数	タイプ	説明
*		→ 指定した場合、オブジェクトはオブジェクト名 (文字列) 省略した場合、オブジェクトは変数

引数	タイプ	説明
object	フォームオブジェクト	→ オブジェクト名 (* を指定した場合)、または変数 (* を省略した場合)
tableNum ; name	倍長整数 ; 文字	→ Number of table whose current selection is to be used or Named selection to be used

新しい SET LISTBOX TABLE SOURCE コマンドは、* と object 引数で指定されたリストボックスに表示される、データのソースを変更するために使用します。

tableNum 引数にテーブル番号を渡すと、リストボックスにはテーブルのカレントセレクションのレコードデータが表示されます。

name 引数に命名セレクションを渡すと、リストボックスには命名セレクションに属するレコードデータが表示されます。

このコマンドは、配列が割り当てられたリストボックスに対して実行しても効果はありません。

リストボックスにすでに列が含まれている場合、コマンド実行後に内容が更新されます。

参照：[GET LISTBOX TABLE SOURCE](#)

GET LISTBOX TABLE SOURCE

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

引数	タイプ	説明
*		→ 指定した場合、オブジェクトはオブジェクト名 (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	→ オブジェクト名 (* を指定した場合)、または変数 (* を省略した場合)
tableNum	倍長整数	← セレクションのテーブル番号
name	文字	← 命名セレクション名、またはカレントセレクションの場合 ""

新しい GET LISTBOX TABLE SOURCE コマンドは、* と object 引数で指定されたリストボックスに表示されたデータのソースを取得するために使われます。

tableNum 引数にはリストボックスに割り当てられたテーブルのテーブル番号が、オプションの name 引数には命名セレクション名が返されます。リストボックスの行がテーブルのカレントセレクションにリンクされている場合、name 引数が渡されていると空の文字が返されます。

リストボックスの行が命名セレクションにリンクされている場合、name 引数には命名セレクション名が返されます。

リストボックスに配列が割り当てられている場合、tableNum には -1 が、name が渡されていれば空の文字が返されます。

参照：[SET LISTBOX TABLE SOURCE](#)

ウィンドウ

Open window コマンドは Mac OS 上でメタルルックのウィンドウを生成するための新しい定数を受け入れます。

Open window

4D v11 では、Mac OS のウィンドウにメタルルックを適用することができるようになりました ([ページ 167](#)、"[メタルルック](#)" の節を参照)。これを行うために、"Open window" テーマの定数に新しいウィンドウタイプの定数が追加されました。

Metal Look (value = 2048).

この定数を使用して、Open window コマンドで生成されたウィンドウにメタルルックを適用することができます。この定数を type 引数に指定されたウィンドウタイプに加えます。たとえば：

```
$win:=Open window(10;80;-1;-1;Plain window+Metal Look;")
```

注 Windows では、この定数の効果はありません。

このルックは Mac OS のほとんどのウィンドウタイプに適用できます。
テーマ：ウィンドウ

プリント

"プリント" テーマに、プリントジョブをコントロールするための新しい [OPEN PRINTING JOB](#) と [CLOSE PRINTING JOB](#) コマンドが追加されました。またいくつかのコマンドが変更されました。

新しいコマンド

OPEN PRINTING JOB

OPEN PRINTING JOB

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません

[OPEN PRINTING JOB](#) コマンドは、プリントジョブを開いて、[CLOSE PRINTING JOB](#) コマンドがコマンド呼び出されるまで、続くすべてのプリント命令をスタックします。このコマンドはプリントジョブのコントロールを可能とし、特に印刷中に他のプリントジョブが予期せず挿入されることを防止します。

[OPEN PRINTING JOB](#) コマンドは、クイックレポートコマンド、4D Write や 4D View のプリントコマンドを含む、すべての 4D プリントコマンドで使用できます。

他方、このコマンドは 4D Chart や 4D Draw、サードパーティ製のプラグインとは互換性がありません。このコマンドでプリントジョブが開かれると、プリンタはプリントが実際に開かれるまで "ビジー" モードになり

ます。この途中で、非互換のプラグインがプリントジョブを起動すると、"プリンタ使用中"エラーが返されます。

プリントジョブを終了し、プリンタにドキュメントを送信するために、CLOSE PRINTING JOB コマンドを呼び出さなくてはなりません。このコマンドを呼び出さないと、プリントドキュメントはスタックに残り、4D アプリケーションを終了するまでプリンタが使用できなくなります。

プリントジョブはプロセスに対してローカルです。プロセスの数だけプリントジョブを開くことができます。ただしこの場合、プリントジョブの数だけプリンタが必要となります。

OPEN PRINTING JOB はカレントのプリント設定を使用します (デフォルト設定、または PAGE SETUP や SET PRINT OPTION を使用した設定)。プリント設定を変更するコマンドは OPEN PRINTING JOB の呼び出し前に実行されてい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

参照：[CLOSE PRINTING JOB](#)

CLOSE PRINTING JOB

CLOSE PRINTING JOB

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません

CLOSE PRINTING JOB は、事前に OPEN PRINTING JOB コマンドで開かれたプリントジョブを閉じ、カレントプリンタ用に準備されたプリントドキュメントを送信するために使用します。

このコマンドが実行されると、プリンタは他のプリントジョブを受け付けることができるようになります。

参照：[OPEN PRINTING JOB](#)

変更されたコマンド

PRINT SETTINGS

PRINT SETTINGS{(dialType)}

引数	タイプ	説明
dialType	倍長整数	→ 表示するダイアログボックス: 0 (または省略された場合) = すべて 1 = 用紙設定 2 = プリント

PRINT SETTINGS コマンドは、オプションの dialType 引数を受け入れるようになり、表示するプリントダイアログを指定できるようになりました。

- dialType に 0 を渡すか省略すると、用紙設定とプリント両方のダイアログが表示されます。

- dialType に1を渡すと、用紙設定ダイアログボックスのみが表示され、カレントのプリント設定が使用されます。
- dialType に2を渡すと、プリントダイアログボックスのみが表示され、カレントの用紙設定が使用されます。

このコマンドは Print form や [OPEN PRINTING JOB](#) コマンドより前に呼び出さなければなりません。

参照：[OPEN PRINTING JOB](#)

ACCUMULATE, BREAK LEVEL

4D v11 では、コンパイル済みまたはインタプリタにかかわらず、ブレーク処理を必要とするすべてのレポート生成の前に、BREAK LEVEL と ACCUMULATE コマンドを実行しなければなりません。以前のバージョンでは、これらのコマンドの実行はコンパイル済みモードでのみ必須でした。

4D 環境

このテーマはいくつかの新しいコマンドにより拡張され、プログラムにより 4D データファイルの検証、修復、圧縮の操作ができるようになりました。これらの操作は [Maintenance & Security Center](#) から可能です。

注 以前の 4D では、これらの機能は 4D Tools で提供されていました。4D v11 ではこのツールの機能は 4D に統合されました。

このテーマには新しい [Get current database localization](#) コマンドも提供され、アプリケーションで使用されるランゲージを取得できるようになりました。[COMPONENT LIST](#) を使用して、インストールされているコンポーネントのリストも取得することができます。さらにいくつかの既存のコマンドが変更されました。

VERIFY DATA FILE

VERIFY DATA FILE (structurePath; dataPath; objects; options; method ; tablesArray; fieldsArray)

引数	タイプ	説明
structurePath	テキスト	→ 検証する 4D ストラクチャファイルのパス名
dataPath	テキスト	→ 検証する 4D データファイルのパス名
objects	数値	→ 検証するオブジェクト
options	数値	→ 検証オプション
method	テキスト	→ 4D コールバックメソッドの名前
tablesArray	数値配列	→ 検証するテーブル番号
fieldsArray	2D 数値配列	→ 検証するインデックスの番号

VERIFY DATA FILE コマンドは、structurePath と dataPath で指定さ

れた 4D データファイルに含まれるオブジェクトの構造的な検証を行います。

- 注** データの検証については、[ページ 185](#)、"[「検証」ページ](#)"の節を参照してください。
- `structurePath`は検証するデータファイルに対応するストラクチャファイル (コンパイル済みまたはインタプリタ) を指定します。開かれたストラクチャや他のストラクチャを指定できます。OS に対応した完全パス名を指定しなければなりません。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、ユーザが、使用するストラクチャファイルを指定できます。
 - `dataPath` は 4D データファイル (4DD) を指定します。データファイルは `structurePath` 引数で指定されたストラクチャファイルに対応していなければなりません。カレントのストラクチャファイルを指定することができますが、データファイルは開かれてはいけないうことに注意してください。開かれたデータファイルを検証するためには、[VERIFY CURRENT DATA FILE](#) コマンドを使用します。
[VERIFY DATA FILE](#) コマンドでカレントのデータファイルを検証しようとすると、エラーが生成されます。
指定されたデータファイルは読み込みのみで開かれます。他のアプリケーションが書き込み可能でこのファイルにアクセスしないようにしなければなりません。そうでなければ検証結果は正しくないものになります。
`dataPath` 引数に空の文字列、ファイル名、または OS に対応した完全パス名を渡すことができます。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、ユーザが検証するファイルを選択することができます。カレントのデータファイルを選択できないことに注意してください。データファイル名のみを渡した場合、4D は指定されたストラクチャファイルと同階層にあるデータファイルを探します。
 - `objects` 引数は検証するオブジェクトを指定するために使用します。二つのタイプのオブジェクト、レコードとインデックスを検証できます。"Data file maintenance" テーマの以下の定数を使用できます。
 - `Verify Records (4)`
 - `Verify Indexes (8)`
 - `Verify All No Callback (16)`レコードとインデックス両方を検証するには、`Verify Records+Verify Indexes`、または `0` を渡します。
`Verify All No Callback` オプションは特別な場合に使用します。このオプションを指定すると、レコードとインデックスに対する内部的な検証は完全

に行われますが、内部的な理由により、コールバックメソッドを許可しません。しかしながら、ログの作成とは互換性があります。

- options 引数は検証オプションを設定するために使用します。現時点では、"Data file maintenance" テーマの一つのオプションのみを指定できます。

■ Do not create log file (16384)

通常、VERIFY DATA FILE コマンドは XML フォーマットのログファイルを作成します (このコマンドの最後の説明を参照してください)。このオプションを指定して、ログの作成をキャンセルすることができます。

ログファイルを作成するには、0 を渡します。

- method 引数には、検証中定期的に呼び出されるコールバックメソッドを設定するために使用します。空の文字列を渡すと、メソッドはコールされません。渡されたメソッドが存在しない場合、検証は行われず、エラーが生成され、OK システム変数に 0 が設定されます。

コールバックメソッドは、呼び出されるときに、イベントタイプにより最大 5 つの引数が渡されます。コールバックメソッドではこれらの引数を宣言しなければなりません。

\$1Longint メッセージタイプ (下の表参照)

\$2Longint オブジェクトタイプ

\$3Text メッセージ

\$4Longint テーブル番号

\$5Longint 予約

以下の表は、イベントタイプごとの引数の内容を示しています:

イベント	\$1(倍長整数)	\$2(倍長整数)	\$3(テキスト)	\$4(倍長整数)	\$5(倍長整数)
Message	1	0	進行状況 メッセージ	処理率 (0-100%)	予約
Verification OK	2	オブジェクトタイプ	OK メッセージテキスト	テーブル / インデックス番号	予約
Error	3	オブジェクトタイプ	エラーメッセージテキスト	テーブル / インデックス番号	予約
End of execution	4	0	終了	0	予約
Warning	5	オブジェクトタイプ	エラーメッセージテキスト	テーブル / インデックス番号	予約

オブジェクトタイプ: オブジェクトが検証されると、OK メッセージ

(\$1=2)、エラー (\$1=3)、警告 (\$1=5) が送信されます。\$2 に返されるオブジェクトタイプは以下のうちのいずれかになります。

- 0 = 不明
- 4 = レコード
- 8 = インデックス
- 16 = ストラクチャオブジェクト (データファイルの予備検証)

特別なケース: \$1 が 2、3、または 5 のとき、\$4 が 0 ならば、それはメッセージがテーブルやインデックスについてではなく、データファイル全体に関するものであることを示します。

コールバックメソッドは \$0 に倍長整数値を返さなくてはなりません。これは処理の実行をチェックするために使用されます。

- \$0 = 0 の場合、処理は通常通り続行されます。
- \$0 = -128 の場合、処理は停止されますが、エラーは生成されません。
- \$0 が他の値の場合、処理が停止され、\$0 に返された値をエラー番号としてエラーを生成します。このエラーはエラーハンドラメソッドでとらえることができます。

データベースに、指定したコールバックメソッドが存在しない場合、エラーが生成され、OK システム変数に 0 が設定されます。

2つのオプションの配列をこのコマンドで利用できます。

- tablesArray 配列にはテーブル番号が含まれ、検証するテーブルを指定するために使用します。この引数は検証するテーブルを制限するために使用します。この引数を渡さないか配列が空の場合で、objects 引数に Verify Records が指定されている場合、すべてのテーブルが検証されます。
- fieldsArray配列には検証対象とするインデックス付きフィールドの番号を渡します。

この引数が渡されないか配列が空の場合で、objects 引数に Verify Indexes が指定されている場合、すべてのインデックスが検証されます。

コマンドはインデックスが作成されていないフィールドは無視します。

フィールドに複数のインデックスが含まれる場合、すべてが検証されます。フィールドが複合インデックスの一部である場合、インデックス全体が検証されます。

fieldsArray には二次元配列を渡します。配列の内容は以下の通りです。

- 要素 {0} にはテーブル番号が含まれます。
- 他の要素 {1...x} にはフィールド番号が含まれます。

デフォルトで、VERIFY DATA FILE コマンドは、(options 引数に Do not create log file オプションが指定されていなければ) XML フォーマットのログファイルを作成します。ログファイルの名前はデータファイル名に基

づきつけられ、データファイルと同階層に作成されます。たとえば、"data.4dd" データファイルを検証すると、"data_verify_log.xml" が作成されます。

▼ データとインデックスの検証

```
VERIFY DATA FILE($StructName;$DataName;Verify Indexes+
Verify Records; Do not create log file;")
```

▼ 完全な検証を行い、ログを作成する :

```
VERIFY DATA FILE($StructName;$DataName;Verify All No Callback;0;")
```

▼ レコードのみの検証 :

```
VERIFY DATA FILE($StructName;$DataName;Verify Records;0;")
```

▼ テーブル 3 と 7 のみを検証 :

```
ARRAY LONGINT($arrTableNums;2)
ARRAY LONGINT($arrIndex;0) `not used but mandatory
$arrTableNums|1|:=3
$arrTableNums|2|:=7
VERIFY DATA FILE($StructName;$DataName;Verify Records;0;
"FollowScan";$arrTableNums;$tindex)
```

▼ 特定のインデックスを検証 ([table2]field1、[table5]field2 と field3):

```
ARRAY LONGINT($arrTableNums;0) `使用されませんが、必須です
ARRAY LONGINT($arrIndex;2;0) ` (カラムは後で追加)
```

```
$arrIndex|1|0|:=4 ` 要素 0 にテーブル番号
APPEND TO ARRAY($arrIndex|1|;1) ` field1 を検証
```

```
$arrIndex|2|0|:=5 ` 要素 0 にテーブル番号
APPEND TO ARRAY($arrIndex|2|;2) ` field2 を検証
APPEND TO ARRAY($arrIndex|2|;3) ` field3 を検証
```

```
VERIFY DATA FILE($StructName;$DataName;Verify Indexes;0;
"FollowScan";$arrTableNums;$arrIndex)
```

参照 : [Compact data file](#), [VERIFY CURRENT DATA FILE](#)

VERIFY CURRENT DATA FILE

```
VERIFY CURRENT DATA FILE{(objects; options; method; {tablesArray;
fieldsArray}}
```

引数	タイプ	説明
objects	数値	→ 検証するオブジェクト
options	数値	→ 検証オプション

引数	タイプ	説明
method	テキスト	→ 4D コールバックメソッド名
tablesArray	数値配列	→ 検証するテーブル番号
fieldsArray	二次元数値配列	→ 検証するインデックス番号

VERIFY CURRENT DATA FILE コマンドは、現在 4D が開いているデータファイル中のオブジェクトの、構造的な検証を行います。

このコマンドは [VERIFY DATA FILE](#) コマンドと同じ動作を行いますが、データベースにより開かれているカレントのデータファイルに適用されます。そのため、ストラクチャとデータファイルを指定する引数は必要ありません。

引数についての説明は、[VERIFY DATA FILE](#) コマンドを参照してください。

VERIFY CURRENT DATA FILE を引数なしで実行すると、検証はデフォルト設定を使用して行われます。

- objects = Verify Records+Verify indexes (= 0)
- options = 0 (ログファイルが作成される)
- method = ""
- tablesArray と fieldsArray は省略される。

このコマンドが実行されるとデータキャッシュがフラッシュされ、検証中はデータにアクセスするすべての操作がブロックされます。

注 インデックスを作成または更新中にこのコマンドを実行してはいけません。これを行うと、検証結果は正しくなくなります。

参照：[VERIFY DATA FILE](#)

Compact data file

Compact data file (structurePath; dataPath; {archiveFolder}; options; method{||})→ Text

引数	タイプ	説明
structurePath	テキスト	→ ストラクチャファイルのパス名
dataPath	テキスト	→ 圧縮するデータファイルのパス名
archiveFolder	テキスト	→ 元のファイルを配置するフォルダのパス名
options	数値	→ 圧縮オプション
method	テキスト	→ 4d コールバックメソッド名
返回值	テキスト	← 元のデータファイルが置かれたフォルダへの完全パス名

Compact data file コマンドは、structurePath ストラクチャファイルに対

応する、dataPath 引数で指定されたデータファイルを圧縮します。圧縮についての詳細はページ 190、"ファイルを圧縮する理由"の節を参照してください。

データベース操作を継続して行えるようにするために、新しい圧縮済みデータファイルは自動で元のファイルと置き換えられます。安全のため、元のファイルは変更されずに特別なフォルダ "Replaced files (compacting) YYYY-MM-DD HH-MM-SS" に移動されます。YYYY-MM-DD HH-MM-SS には、バックアップの時間が反映されます。例："Replaced files (compacting) 2007-09-27 15-20-35"

このコマンドからは、元のデータファイルを格納するために作成された実際のフォルダへの完全パス名が返されます。

このコマンドは 4D Developer または 4D Server (ストアドプロシージャ) からのみ呼び出せます。

圧縮するデータファイルは、structurePath で指定するストラクチャファイルに対応したものでなければなりません。さらに、コマンドが実行される時、データファイルが開かれてはなりません。そうでなければエラーが生成されます。

圧縮処理中にエラーが発生した場合、オリジナルのファイルは元の位置から移動されません。

インデックスファイル (4DIndx file) がデータファイルに結び付けられている場合、それも圧縮されます。データファイルと同様、オリジナルのファイルはバックアップされ、圧縮されたファイルと置き換えられます。

- structurePath 引数には、圧縮するデータファイルに対応するストラクチャファイルのパス名を渡します。この情報は圧縮処理のために必要です。パス名は OS の書式に対応するものでなければなりません。

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、指定するストラクチャファイルを選択できます。

- dataPath 引数には、空の文字列、ファイル名、または OS に対応する完全パス名を渡すことができます。空の文字列を渡すと標準のファイルを開くダイアログが表示され、圧縮するデータファイルを選択できます。このファイルは structurePath 引数で指定されたストラクチャファイルに対応するものでなければなりません。データファイル名のみを渡した場合、4D はストラクチャファイルと同階層のデータファイルを探します。
- オプションの archiveFolder 引数を使用して、元のデータファイルとインデックスファイルのバックアップコピーを格納する "Replaced files (compacting) DateTime" フォルダの場所を指定することができます。コマンドからは実際に作成されたフォルダの完全パス名が返されます。
- この引数を省略すると、元のファイルは自動でデータファイルと同階層に作成される "Replaced files (compacting) DateTime" に格納されます。

- 空の文字列を渡すと、標準のフォルダを選択ダイアログが表示され、フォルダを作成する場所を選択できます。
- OS に対応する書式でパス名を渡すと、"Replaced files (compacting) DateTime" フォルダは指定された場所に作成されます。
- オプションの options 引数を使用して、さまざまな圧縮オプションを設定できます。以下で説明する "Data file maintenance" テーマの定数が利用できます。加算することで、複数の値を指定できます。
 - Do not create log file (16384)
通常 Compact data file は XML フォーマットのログファイルを作成します (このコマンドの説明の最後を参照)。ファイル名はデータファイル名に基づきつけられ、データファイルと同階層に置かれます (デフォルトの場所)。たとえばメ data.4dd モ データファイルの場合、ログファイル名はメ data_compact_log.xml モ となります。このオプションを指定すると、ログファイルは作成されません。
 - Create log file (16384)
このオプションは、XML フォーマットのログファイルを作成します。データファイルに対応する名前が付けられ、元のデータファイルと同階層に作成されます。たとえばデータファイル "data.4dd" の場合、ログファイル名は "Struct_Data_Compact_Log.xml" となります。
 - Create process (32768)
このオプションが指定されると、
 - 圧縮は非同期で行われ、後述するコールバックメソッドを使用して結果を管理しなければなりません。
 - 4D は進捗状況を表示しません (コールバックメソッドを使用して表示させることができます)。
 - 処理が正しく起動されれば OK システム変数に 1 が設定され、他の場合は 0 となります。このオプションが指定されない場合、圧縮が正しく行われれば OK システム変数に 1 が設定され、他の場合は 0 となります。

- method引数には使用するコールバックメソッドを渡します。Create process オプションが選択されていれば、定期的にコールバックメソッドが呼び出されます。オプションが指定されていなければ、コールバックメソッドが呼び出されることはありません。コールバックメソッドに関する詳細は [ページ 319](#)、[コマンド VERIFY DATA FILE](#) の説明を参照してください。データベースに指定されたコールバックメソッドが存在しない場合、エラーが生成され、OK システム変数に 0 が設定されます。

圧縮処理が正しく行われると、OK システム変数に 1 が設定され、他の場合は 0 となります。

デフォルトで、Compact data file コマンドは XML フォーマットのログファイルを作成します (Do not create log file オプションを指定していない場合)。ログファイル名はデータファイル名に基づき決定され、データファイルと同階層に作成されます。たとえばデータファイルメ data.4dd の場合、ログファイル名はメ data_compact_log.xml になります。

- ▼ 以下の例 (Windows) では、データファイルの圧縮を行い、ログファイルを作成します。

```
$structFile:=Structure file
$dataFile:="C:\Databases\Invoices\January\Invoices.4dd"
$origFile:="C:\Databases\Invoices\Archives\January\"
$searchFolder:=Compact data file($structFile;$dataFile;$origFile)
```

参照：[VERIFY DATA FILE](#)

OPEN SECURITY CENTER

OPEN SECURITY CENTER

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません

OPEN SECURITY CENTER コマンドは Maintenance and Security Center (MSC) ウィンドウを表示します。

カレントユーザのアクセス権により、ウィンドウ中のいくつかの機能は無効にされることがあります。

MSC についての詳細は [ページ 177](#)、["Maintenance & Security Center"](#) の章を参照してください。

Get current database localization

Get current database localization → String

引数	タイプ	説明
----	-----	----

このコマンドは引数を必要としません

返り値	文字	← データベースのカレントランゲージ
-----	----	--------------------

Get current database localization コマンドはデータベースのカレントのランゲージ

ジを、RFC 3066 標準の形式で返します。たとえば日本語では "ja"、英語では "en" が返されます。この標準と、返される値のより詳しい説明は [ページ 101](#)、"[.lproj フォルダ名](#)" の節を参照してください。

データベースのカレントランゲージは、プログラムがローカライズされたリソースを取り出すために使用する .lproj フォルダを設定するために使用されます。4D は起動時に、Resources フォルダの内容とシステム環境に基づいて、カレントランゲージを決定します。4D は以下の優先順位に基づき、参照するランゲージの .lproj フォルダをロードします。

1. システムランゲージ (Mac OS では環境設定でいくつかのランゲージの順番を設定できます。4D はこの設定を使用します)
2. 4D アプリケーションのランゲージ
3. English
4. Resources フォルダで最初に見つかった .lproj フォルダ

注 データベースに .lproj フォルダが存在しない場合、4D は以下の優先順位を使用します：1. システムランゲージ、2. English (システムランゲージを決定できない場合)

COMPONENT LIST

COMPONENT LIST(componentsArray)

引数	タイプ	説明
componentsArray	文字配列	← コンポーネントの名前

データベースが開かれると、4D はストラクチャファイルと同階層の Components フォルダにある有効なコンポーネントをロードします。COMPONENT LIST コマンドは componentsArray 配列のサイズを調整し、カレントのホストデータベース用に 4D アプリケーションがロードしたコンポーネントの名前を返します。

このコマンドはホストデータベースまたはコンポーネントから呼び出すことができます。データベースがコンポーネントを利用しない場合、componentsArray は空になります。

コンポーネントの名前は、マトリクスデータベースのストラクチャファイルの名前です。(4db、4dc または 4dbase)

このコマンドは、コンポーネントがインストールされているかいないかで、アーキテクチャやモジュールのインターフェースを構築するような場合に使用できます。

注 4D v11 のコンポーネントに関する詳細は [ページ 45](#)、"[新しいコンポーネントアーキテクチャ](#)" の節を参照してください。

変更されたコマンド "4D 環境" テーマのいくつかのコマンドは機能が拡張されました。

Structure file

Structure file {*} → String

引数	タイプ	説明
*		→ ホストデータベースのストラクチャファイル
戻り値	文字	← ストラクチャファイルのロング名

Structure file コマンドは、オプションの * 引数を受け入れるようになりしました。この引数はコンポーネントを使用しているときに便利です。

- コマンドがコンポーネントから呼び出されたとき：
 - * 引数が渡されていると、コマンドはホストデータベースストラクチャファイルのロング名を返します。
 - * が渡されていなければ、コマンドはコンポーネントストラクチャファイルのロング名を返します。

コンポーネントのストラクチャファイルは、データベースの "Resources" フォルダにあるコンポーネントの .4db や .4dc ファイルに対応します。しかしコンポーネントをエイリアス / ショートカット、または .4dbase フォルダ / パッケージでインストールすることも可能です。

- エイリアス / ショートカットでインストールされている場合、コマンドは元の 4db や .4dc ファイルのパス名を返します (エイリアスやショートカットは解決されます)。
- 4dbase フォルダ / パッケージとしてインストールされている場合、コマンドはフォルダ / パッケージ内の .4db または .4dc ファイルのパスを返します。
- コマンドがホストデータベースのメソッドから呼び出された場合、常にホストデータベースストラクチャファイルのロング名が返されます。
- ▼ 以下の例は、メソッドがコンポーネントから呼ばれているかどうかをテストします。

```
C_BOOLEAN($0)
$0=(Structure file#Structure file(*))
`$0=TRUE はメソッドがコンポーネントから呼ばれている
```

Get 4D folder

Get 4D folder{(folder)}{*} → String

引数	タイプ	説明
folder	倍長整数	→ フォルダタイプ
*		→ ホストデータベースのフォルダ
戻り値	文字	← 指定したフォルダのパス名

Get 4D folder コマンドは、オプションの * 引数と新しい Current Resources

folder 定数を受け入れるようになりました。

定数	タイプ	値
Current Resources folder	倍長整数	6

これらの新しい機能はコンポーネント利用時に便利です。

- コマンドがコンポーネントから呼び出されたとき:
 - * 引数が渡され、folder 引数に 4、5、または 6 が渡されていると、コマンドはホストデータベースフォルダのパス名を返します。
 - * 引数が渡されず、folder 引数に 4、5、または 6 が渡されていると、コマンドはコンポーネントフォルダのパス名を返します。
データベースフォルダ (4 と 5) はコンポーネントアーキテクチャにより異なる値を返します。
 - .Adbase フォルダ / パッケージの場合、コマンドは .Adbase フォルダ / パッケージのパス名を返します。
 - .Adb や .Adc ファイルの場合、コマンドは "Components" フォルダのパス名を返します。
 - エイリアスやショートカットの場合、オリジナルのマトリクスデータベースが格納されたフォルダのパス名を返します。この結果も、上記で説明したデータベースのフォーマット (.Adbase フォルダ / パッケージまたは .Adb / .Adc ファイル) により異なります。
- コマンドがホストデータベースから呼ばれた場合で、4、5、6 が渡されている場合、常にホストデータベースフォルダのパス名が返されます。
新しい Current Resources folder 定数は、ホストデータベースやコンポーネントの Resources フォルダのパス名を得るために使用します。

Is compiled mode Is compiled mode(*) → Boolean

引数	タイプ	説明
*		→ ホストデータベースの情報を返す
返り値	Boolean	← コンパイル (True)、インタプリタ (False)

注 以前のバージョンでは、このコマンドは Compiled application という名前でした。明確にするため、4D v11 で名称が変更されました。

Is compiled mode 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。この引数はコンポーネント利用時に使用できます。

- コマンドがコンポーネントから呼び出されると:
 - 引数が渡されていると、ホストデータベースの実行モードが返されます。

- *引数が渡されていない場合、コンポーネントの実行モードが返されます。
- コマンドがホストデータベースで呼び出された場合、常にホストデータベースの実行モードが返されます。

SET DATABASE PARAMETER, Get database parameter

SET DATABASE PARAMETER({table; }selector; value)
Get database parameter({table; }selector) → Longint

注 このコマンドは以前 "Structure Access" テーマに分類されていました。

新しいセレクト

Selector = 41 (Unicode mode)

- 値: 0 (互換モード) または 1 (Unicode モード)
- 説明: 文字セットに関連する現在のデータベースの実行モード。
4D v11 は Unicode 文字セットをサポートしますが、Mac ASCII 文字セットに基づく "互換" モードでの動作もサポートしています。
デフォルトで、返還されたデータベースは互換モード (0) で動作し、v11 以降で作成されたデータベースは Unicode モードで動作します。実行モードは環境設定で変更できます ([ページ 61](#)、"[4D データベースにおける Unicode の互換性](#)" の節を参照)。このセレクトを使用して、設定を取得したり、(テストの目的で) 変更したりできます。
設定を変更した場合、それ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。コンポーネント内部ではこの値を読むことはできませんが、変更はできないことに留意してください。

注 日本語環境においては Unicode モードの利用を強くお勧めします。

- ▼ 以下のコードは現在のモードを変更し、データベースを再起動します。:

```
Current_Unicode_mode:=Get database parameter(Unicode mode)
SET DATABASE PARAMETER(Unicode mode;1-Current_Unicode_mode)
OPEN DATA FILE(Data file)
Selector = 42 (Temporary memory size)
■ 値: 倍長整数の正値 > 50,000,000
```

- 説明: バイト単位の一時的なメモリのサイズ。デフォルトのサイズは 50,000,000 (50 MB)。

4D v11 はインデックスや並び替え操作に、特別な一時的メモリを使用します。このメモリの目的は、大量の操作を行う際に、"標準の" キャッシュメモリを保護することにあります。一時メモリの最大サイズを使い切るような (危機的な) 状況では、4D は自動でリクエストされた最後の操作を中断し、他の実行中の処理に影響が出ることを避けます。

ほとんどの場合、一時メモリに対するデフォルトの設定で問題はありませぬ。しかし並び替えやインデックスが集中して行われるような特殊なアプリケーションでは、この値を増やすことでパフォーマンスの向上を図ることができるかもしれません。アプリケーションの特定の機能に対する理想的な値をテストから求める必要があります。

Selector = 43 (SQL Autocommit)

- 値: 0 (無効) または 1 (有効)

- 説明: SQL の自動コミットを有効または無効にします。デフォルトで、値は 0 (無効) です。

自動コミットモードは、データベースの参照整合性を強化するために使用します。このモードが有効であれば、すべての SELECT, INSERT, UPDATE そして DELETE (SIUD) クエリは自動でアドホックなトランザクションに含まれます (まだトランザクションに含まれていなければ)。このモードはデータベースの環境設定でも設定できます。詳細は [ページ 214](#)、"自動コミット" の節を参照してください。

変更されたセレクト

Selector = 17 (Character set)

アプリケーションが Unicode モードで実行されているとき、value に渡す、あるいは返される値は、文字セット識別子 (IANA により定義された MIBEnum。 <http://www.iana.org/assignments/character-sets> を参照) となります。以下は 4D Web サーバでサポートされる文字セットに対応する識別子です。

- 4 = ISO-8859-1
- 12 = ISO-8859-9
- 13 = ISO-8859-10
- 17 = Shift_JIS
- 2026 = Big5
- 38 = euc-kr
- 106 = UTF-8

- 2250 = Windows-1250
- 2251 = Windows-1251
- 2253 = Windows-1253
- 2255 = Windows-1255
- 2256 = Windows-1256

また Character set セレクタのフレームワークにおいて、Get database parameter コマンドは、オプションの stringValue に、文字セットの IANA 名を返します。

Selector = 29 (Web Log Recording)

Selector = 30 (Client Web Log Recording)

4D v11 では、新しい Web ログファイルのフォーマットがサポートされました ([ページ 239](#)、"[Web ログの設定 \(logweb.txt\)](#)" の節)。それに伴いセレクタ 29 と 30 で使用できる新しい値が加えられました。これらの新しい値 (2、3 および 4) は以下のフォーマットを意味します。

- 利用可能な値: 0、1、2、3 または 4
 - 0 = ログを記録しない (デフォルト)
 - 1 = CLF フォーマット
 - 2 = DLF フォーマット
 - 3 = ELF フォーマット
 - 4 = WLF フォーマット

これらのフォーマットの説明は [ページ 239](#)、"[ログフォーマット](#)" の節を参照してください。

警告: フォーマット 3 と 4 はカスタムフォーマットであり、ログの内容は環境設定の "Web/Log フォーマット" ページで定義しなければなりません。ログに記録するフィールドが定義されていない場合、ログファイルは生成されません。

削除されたセレクタ

以下のセレクタは 4D v11 で保証されない設定や最適化に関連するので、無効となっています。

Selector = 1 (Seq Order Ratio)

Selector = 2 (Seq Access Optimization)

Selector = 3 (Seq Distinct Values Ratio)

Selector = 4 (Index Compacting)

Selector = 5 (Seq Query Select Ratio)

Selector = 26 (Cache Writing Mode)

それに伴い、これらのセレクタは、SET DATABASE PARAMETER では無視され、Get database parameter に渡すと 0 が返されます。

OPEN 4D PREFERENCES

OPEN 4D PREFERENCES (selector)

引数	タイプ	説明
selector	文字	→ 環境設定のテーマやページ、パラメタグループを指定するキー

4D の環境設定に加えられた変更を反映するために、selector 引数に指定可能なキーのリストが変更されました。

変更されたキー：

- ・ /Application/Compatibility/Design Compatibility
(/Application/Compatibility/Structure Compatibility より)
- ・ /Design Mode/Method Editor/Syntax Styles
(/Design Mode/Method Editor/Styles for Syntax Elements より)
- ・ /Database/International (/Database/Script Manager より)
- ・ /Client-Server/Publishing/Allow-Deny Configuration Table
(/Client-Server/Publishing/Allow-Deny Table Configuration より)

新しいキー：

- ・ /Application/Access/General Settings
- ・ /Design Mode/Structure/Automatic Form Creation
- ・ /Moving
- ・ /Moving/Default Actions during the Copy if Dependent Objects
- ・ /Moving/Moving Dialog
- ・ /Database/International/Right-to-left Languages
- ・ /Database/International/Numeric Display Format
- ・ /Web/Options/Options
- ・ /Web/Log Format
- ・ /Web/Log Format/Web Log Type
- ・ /Web/Log Format/Web Log Token Selection
- ・ /Web/Log Scheduler
- ・ /Web/Log Scheduler/Backup Frequency for Web Log File
- ・ /SQL
- ・ /SQL/Configuration
- ・ /SQL/Configuration/SQL Server Access

DATA SEGMENT LIST, ADD DATA SEGMENT

4 D v11 では、データセグメントは分割されなくなりました。返還されたデータベースではすべてのセグメントが統合されます ([ページ 24](#)、"[複数のセグメントを持つデータベースの変換](#)" の節を参照)。それにともない、これらのコマンドは動作しなくなりました。

システム環境

Select RGB Color

Select RGB Color({defaultColor}; message)}) → Longint

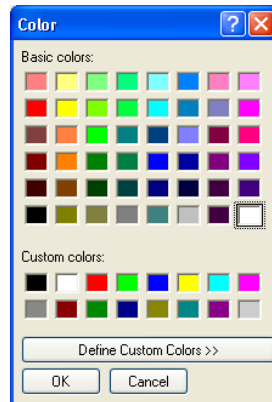
引数	タイプ	説明
defaultColor	倍長整数	→ あらかじめ選択するカラー

引数	タイプ	説明
message	文字 255	→ ウィンドウタイトル
返り値	倍長整数	← RGB カラー

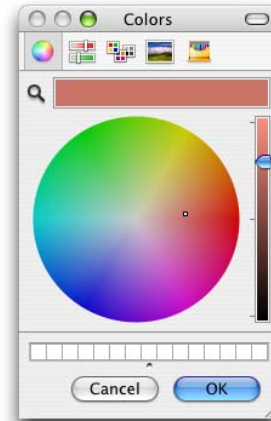
新しい Select RGB Color コマンドは、システムカラー選択ウィンドウを表示し、ユーザが選択したカラーの RGB 値が返されます。

システムのカラー選択ウィンドウは以下のように表示されます：

Windows



Mac OS



オプションの defaultColor 引数は、ウィンドウでカラーをあらかじめ選択させるために使用できます。たとえばユーザが以前に設定したカラーをデフォルトで選択させることができます。

この引数には、RGB フォーマットの値を渡します (詳細は SET RGB COLORS コマンドの説明を参照)。「SET RGB COLORS」テーマの定数を使用できます。

defaultColor 引数が省略されているか、0 が渡された場合、黒が選択されます。

オプションの message 引数を使用して、システムウィンドウのタイトルを指定できます。デフォルトで、この引数が省略されると、タイトルに「カラー」が表示されます。

ユーザがダイアログボックスを受け入れると、コマンドからは選択されたカラーが RGB フォーマットで返され、OK システム変数には 1 が設定されます。ダイアログがキャンセルされると、コマンドからは -1 が返され、OK システム変数には 0 が設定されます。

注 このコマンドを 4D Server や Web プロセスで実行してはいけません。

GET SYSTEM FORMAT

GET SYSTEM FORMAT(format; value)

引数	タイプ	説明
format	倍長整数	→ 取り出すシステムフォーマット

引数	タイプ	説明
value	文字	← システムに設定されたフォーマット

GET SYSTEM FORMAT コマンドは、OS に設定された、地域パラメタの現在値を返します。このコマンドは、システムの環境設定に基づく、"自動" カスタムフォーマットを構築するために使用できます。

format 引数には、取得したいパラメタのタイプを指定します。結果はシステムから value 引数に直接文字列として返されます。

format には、"System format" テーマの以下の定数を渡さなければなりません。

定数 (値)	返される値
Decimal separator (0)	小数点 (例 : ".")
Thousand separator (1)	千単位区切り (例 : ",")
Currency symbol (2)	通貨記号 (例 : "\$")
System time short pattern (3)	"HH:MM:SS" 形式の時間表示
System time medium pattern (4)	フォーマットに対応
System time long pattern (5)	
System date short pattern (6)	"dddd d MMMM yyyy" 形式の日
System date medium pattern (7)	付表示フォーマットに対応
System date long pattern (8)	
Date separator (13)	日付フォーマットの区切り文字 (例 : "/")
Time separator (14)	時間フォーマットの区切り文字 (例 : ":")
Short date day position (15)	日付短表記の年月日の位置 :
Short date month position (16)	"1" = 左
Short date year position (17)	"2" = 中央 "3" = 右
System time AM label (18)	12 時間表記時、午前を表すラ ベル (例 : "AM")
System time PM label (19)	12 時間表記時、午後を表すラ ベル (例 : "PM")

- ▼ 機械でタイプされた小切手では、不正を防ぐため、一般的に金額は左が "*" で埋められます。金額のシステム表示フォーマットが "\$ 5,422.33" であるとき、小切手用のフォーマットは "\$***5432.33" のようになります。千単位以降にはカンマがなく、通貨単位と最初の数字の間にスペースはありません。String 関数で使用されるフォーマットは "\$*****.*)" になります。

プログラムでこれを構築するには、通貨単位と小数点文字を知る必要があります。

GET SYSTEM FORMAT(Currency symbol,\$currency)

```
GET SYSTEM FORMAT(Decimal separator;$decimal)
$MyFormat:=$currency+"*****"+$decimal+"**"
$Result:=String(amount,$MyFormat)
```

Windows における 複数スクリーン

4D v11 では、Count screens、SCREEN COORDINATES、SCREEN DEPTH
そして SET SCREEN DEPTH コマンドは、Windows でマルチスクリーン
をサポートします。

変更されたコマンド

PLATFORM PROPERTIES

PLATFORM PROPERTIES(platform; system; machine; language;))

引数	タイプ	説明
platform	数値	← 2=Mac OS, 3=Windows
system	数値	← バージョンに基づく (変更なし)
machine	数値	← プロセッサファミリ
language	数値	← バージョンに基づく (変更なし)

PLATFORM PROPERTIES から返される情報は、開発者のニーズにより
適合させるため、またハードウェアの発展に伴い単純化されました。
また定数の名前も変更されました。

■ platform

今バージョンより、platform プロパティは使用されている OS のみを示す
ようになりました。以下の定数に対応する二つの値のみが返されます。

定数	タイプ	値
Mac OS	Longint	2
Windows	Longint	3

■ machine

machine 引数には、マイクロプロセッサの "ファミリ" が返されます。以下
の定数に対応する二つの値が返されます。

定数	タイプ	値
Power PC	Longint	406
Intel Compatible	Longint	586

使用されているマシンを明確にするために、二つの値を使用できます。
たとえば "MacIntel" マシンの場合、platform=Mac OS で machine=Intel
Compatible となります。

定数

"Platform Properties" テーマの一部の定数はコマンド拡張のため変更され
ました。以下にリストするなかで、有効な定数は名称が変更され、無効

な定数は先頭に `_O_` がつけられました。

4D v11 定数	値	以前の名称
Intel Compatible	586	Pentium
Mac OS	2	Power Macintosh
Windows	3	Windows (変更なし)
Power PC	406	Other G3 and above
<code>_O_INTEL 386</code>	386	INTEL 386
<code>_O_INTEL 486</code>	468	INTEL 486
<code>_O_Macintosh 68K</code>	1	Macintosh 68K
<code>_O_PowerPC 601</code>	601	PowerPC 601
<code>_O_PowerPC 603</code>	603	PowerPC 603
<code>_O_PowerPC 604</code>	604	PowerPC 604
<code>_O_PowerPC G3</code>	510	PowerPC G3

ユーザインターフェース

新しい `Tool bar height` コマンドが "ユーザインターフェース" テーマに追加され、`Focus object` コマンドはリストボックスで使用できるよう変更されました。

また以前他のテーマに属していたいくつかのコマンドが、このテーマに統合されました (ページ 388、"テーマの変更" の節参照)。

Tool bar height

Tool bar height → Longint

引数	タイプ	説明
		このメソッドは引数を必要としません。
戻り値	Longint	← ツールバーの高さ (ピクセル単位) ツールバーが表示されていなければ 0

`Tool bar height` コマンドはツールバーの高さをピクセル単位で返します。ツールバーが表示されていなければ 0 が返されます。

注 このコマンドを使用して、以前のバージョンの 4D Pack のコマンド、`AP Toolbar installed` を置き換えることができます。

参照 : `Menu bar height`, `HIDE TOOL BAR`, `SHOW TOOL BAR`

Focus object

リストボックスのコンテキストでこのコマンドが使用されると、このコマンドは以下を返します。

- フィールドが割り当てられた列では、そのフィールドへのポインタ。

- 変数が割り当てられた列では、その変数へのポインタ。
- 表現式が割り当てられた列では、リストボックス変数へのポインタ。
4D v11 のリストボックス新機能に関する詳細は、[ページ 311](#)、"[リストボックス](#)" の節を参照してください。

バーチャルストラクチャへのアクセスの原則

バーチャルストラクチャとは、SET TABLE TITLES や SET FIELD TITLES コマンドで名称変更されたテーブル名やフィールド名を指します。これらの名称は 4D エディタ (クエリ、クイックレポートなど) やユーザに提供されるストラクチャ情報として使用されます。

4D エディタにおけるバーチャルストラクチャへのアクセス原則は、4D v11 で変更され、テーブルやフィールドが呼び出されるコンテキストにより異なります。エディタがデザインモードで直接呼び出された時は、バーチャルストラクチャは表示されません。対して、エディタがランゲージコマンド (QUERY や ORDER BY など) から呼び出された時は表示されます。プラグインは常にバーチャルストラクチャにアクセスし、"非表示" プロパティが優先されることに注意してください。

以下の表でこの原則をまとめます：

要素の表示設定	バーチャルストラクチャ	実ストラクチャ	非表示テーブル / フィールド
デザイン (エディタ)		○	○
プラグイン	○		
アプリケーションモード	○		

システムドキュメント

Select folder コマンドは第二引数を受け入れ、Open document、Create document そして Append document コマンドの type 引数は変更されました。

Select folder

Select folder{(message{;defaultPath})} → String

引数	タイプ	説明
message	String	→ ウィンドウタイトル
defaultPath	String Longint	→ <ul style="list-style-type: none"> ・ デフォルトのパスまたは ・ 空の文字列でユーザフォルダ (Windows は "My documents"、Mac OS は "Documents" または ・ 記憶するパスの数
返り値	String	← 選択されたフォルダのアクセスパス

新しい defaultPath 引数を使用して、フォルダ選択ダイアログに最初に表示されるフォルダの場所を指定できるようになりました。

この引数には 3 つのタイプを渡すことができます。

- プラットフォームに対応する既存のフォルダのパス名
- 空の文字を渡すと、システムのユーザフォルダ (Windows は "My documents"、Mac OS は "Documents")
- 関連するフォルダを表示するために記憶するパス名の数 (1 ~ 32,000)
この場合、ユーザが選択ボタンをクリックして開いたフォルダ、言い換えればユーザが選択したフォルダのパス名がメモリに格納されます。ランダムな数値をコールすると、コマンドはシステムのユーザフォルダを表示します (空の文字列を渡したのとおなじ)。そののち、ユーザはハードディスクのフォルダをブラウズします。ユーザが選択ボタンをクリックすると、パス名が記憶され、前述のランダムな数値と結び付けられます。もう一度同じ数値を使用してコマンドを呼び出すと、デフォルトで記憶されたパスが表示されます。新しい場所が選択されると、その数値に対尾ぶするパス名は更新されます。

このメカニズムを使用して 32,000 までのパス名を記憶させることができます。Windows では、パスはセッション中でのみ記憶されます。Mac OS では、セッションを超えてパス名が記憶されます。

注 この機構は Select document コマンドで使用されているものと同じです。記憶されたパス名は両コマンドで共有されます。

パス名が正しくない場合、defaultPath 引数は無視されます。

Open document, Create document, Append document

Open document (document{; fileType{; mode{}}) → DocRef

Create document (document{; fileType{; mode{}}) → DocRef

Append document (document{; fileType{; mode{}}) → DocRef

これら 3 つのコマンドの fileType 引数に変更されました。セミコロンで区切られたドキュメントタイプのリストを渡すことができます。これにより、開くまたは新規ダイアログボックスで、開発者が有効なファイルのタイプを制限することができます (document に空の文字列を渡した場合)。

ピクチャ

ピクチャ管理は 4D v11 で変更され、異なるフォーマットがネイティブでサポートされるよう拡張されました。この変更に関する詳細は [ページ 155](#)、"[ピクチャフィールドおよび変数の最適化](#)" の節を参照してください。さらにネイティブピクチャを処理するための新しいコマンドが追加されました。

以前のバージョンの 4D Pack で提供されていたピクチャ管理コマンドのいくつかは互換性がなくなったり、廃止されることとなりました。利用

できなくなった 4D Pack のコマンドもこの節で説明します。

PICTURE CODEC LIST

PICTURE CODEC LIST (codecArray; namesArray)

引数	タイプ	説明
codecArray	String array ←	利用可能なピクチャ CODEC の ID
namesArray	String array ←	ピクチャ CODEC の名前

PICTURE CODEC LIST コマンドは、codecArray 配列にマシン上で利用可能なピクチャ CODEC ID を返します。このリストには、4D v11 でネイティブに管理されるピクチャフォーマットの CODEC ID と、マシンにインストールされた追加の QuickTime コード ID が含まれます。

これらの ID はピクチャの書き出しに使用する WRITE PICTURE FILE や PICTURE TO BLOB コマンドの引数として渡すことができます。

codecArray に返される CODEC ID は 3 つの異なるフォーマットで返されます。

- 拡張子 (例 ".gif")
- Mime type (例 "image/jpeg")
- 4 文字の QuickTime コード (例 "PNTG")

返される形式は、OS レベルにどのように CODEC が記録されているかで決定されます。

オプションの引数 namesArray 配列を使用して、それぞれの CODEC 名を取得することができます。名前は ID よりも明確なので、有効な CODEC を表示するメニューを構築するためなどに使用できます。

ネイティブピクチャフォーマット

4D v11 複数のピクチャフォーマットのネイティブな管理を統合しました。これらのフォーマットはどのような場合にも利用可能で、OS やマシンの設定にかかわらず常に [PICTURE CODEC LIST](#) コマンドから返されます。このフォーマットには以下のようなものがあります：

- JPEG
- PNG
- BMP
- GIF
- TIF
- EMF (Windows のみ)
- PICT
- PDF (Mac OS のみ)

注 PICTURE TYPE LIST コマンドは互換性のために残されています。しかしこのコマンドには QuickTime が必要であり、4D がネイティブにサポートするフォー

マットにはアクセスしません。そのため [PICTURE CODEC LIST](#) に置き換えることをお勧めします。

参照：[CONVERT PICTURE](#)

TRANSFORM PICTURE

TRANSFORM PICTURE(*picture*; *operator*{; *param1*{; *param2*{; *param3*{; *param4*}}}})

引数	タイプ	説明	
<i>picture</i>	Picture	→	変形するソースピクチャ
		←	変形結果のピクチャ
<i>operator</i>	Longint	→	変形のタイプ
<i>param1...4</i>	Number	→	変形パラメタ

新しい TRANSFORM PICTURE コマンドは *picture* 引数に渡されたピクチャに、*operator* タイプの変形を提供して返します。

注 このコマンドはピクチャ演算子 (+/, etc.) で提供される機能を拡張します。これらの演算子は 4D v11 でも利用可能です。

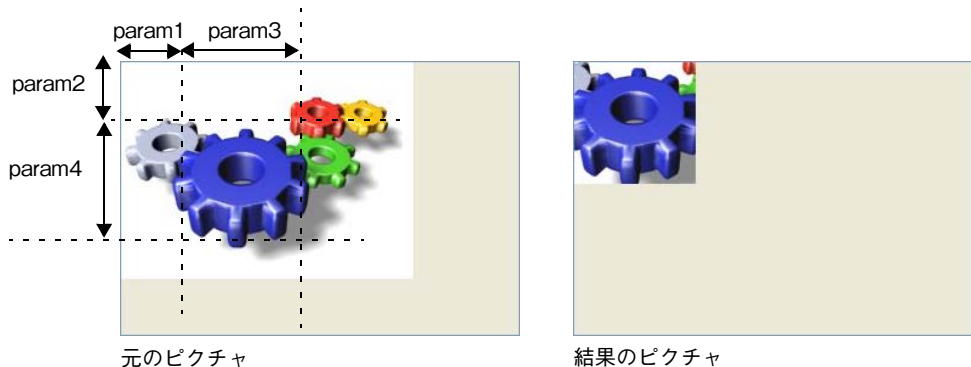
このコマンドが実行されるとソースピクチャが直接変形されます。"Scale" と "Fade to grey scale" を除き、変形は可逆であり、反対の操作や "Reset" 処理で元に戻すことができます。たとえば 1% に縮小されたピクチャや、トリミングされたピクチャを元に戻すことはできません。変形は元のピクチャタイプを変更しません。たとえばベクタピクチャは変形後もベクタピクチャのままです。

operator には実行する操作の番号を渡し、*param* にはその操作に必要なパラメタを渡します (パラメタ数は操作により異なります)。operator には新しい "Picture Transformation" テーマの定数を渡します。以下の表で操作とそのパラメタについて説明します。

operator (値)	param1	param2	param3	param4	値
Reset (0)	-	-	-	-	
Scale (1)	幅	高さ	-	-	係数
Translate (2)	X 軸	Y 軸	-	-	ピクセル
Flip horizontally (3)	-	-	-	-	
Flip vertically (4)	-	-	-	-	
Crop (100)	X 開始位置	Y 開始位置	幅	高さ	ピクセル
Fade to grey scale (101)	-	-	-	-	

- Reset: ピクチャに対して実行されたすべてのマトリクス操作 (scale, flip, など) は元に戻すことができません。
- Scale: ピクチャは、param1 と param2 に渡された値に基づき、水平および垂直方向にリサイズされます。値は係数で指定します。たとえば幅を50% 広げるにはparam1に1.5を、高さを50%縮めるにはparam2に0.5を渡します。
- Translate: ピクチャは param1 ピクセルだけ水平方向に、param2 ピクセルだけ垂直方向に移動されます。右や下方向に移動するには正数を、左や上方向に移動するには負数を渡します。
- Flip horizontally と Flip vertically: 元のピクチャを反転します。事前に行われた移動は考慮されません。
- Crop: ピクチャはparam1とparam2で指定された位置を起点として、param3 と param4 で指定された幅と高さでトリミングされます。
この変形は元に戻せません。
- Fade to grey scale: ピクチャをグレースケールにします。パラメタは必要ありません。この変形は元に戻せません。
- ▼ 以下はトリミングの例です。ピクチャは"トランケート (中央合わせなし)" で表示されています。

TRANSFORM PICTURE(\$vpGears;Crop;50;50;100;100)



参照: [COMBINE PICTURES](#)

COMBINE PICTURES

COMBINE PICTURES(resultingPict; pict1; operator; pict2; horOffset; vertOffset)

引数	タイプ	説明
resultingPict	Picture	← 合成結果のピクチャ
pict1	Picture	→ 合成する一番目のピクチャ
operator	Longint	→ 合成タイプ

引数	タイプ	説明
pict2	Picture	→ 合成する二番目のピクチャ
horOffset	Longint	→ 水平オフセット
vertOffset	Longint	→ 垂直オフセット

新しい COMBINE PICTURES コマンドは、pict1 と pict2 ピクチャを operator モードで合成し、resultingPict ピクチャを生成します。結果のピクチャは合成タイプで、ソースピクチャの特性をすべて保持します。

注 このコマンドはピクチャ演算子 (+/, etc.) で提供される機能を拡張します。これらの演算子は 4D v11 でも利用可能です。

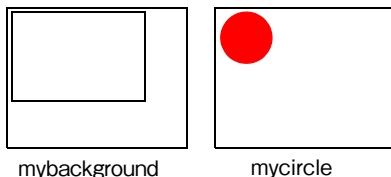
operator には、適用する合成のタイプを渡します。

"Picture Transformation" テーマの定数でアクセス可能な 3 つの合成タイプが提供されます。

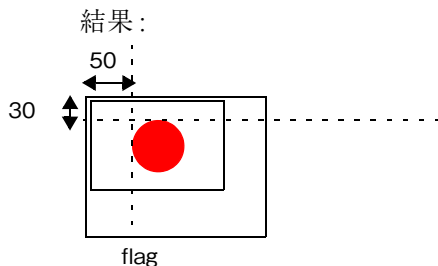
- Horizontal concatenation (1): pict2 が pict1 に追加されます。pict2 の左上隅が、pict1 の右上隅に置かれます。
- Vertical concatenation (2): pict2 が pict1 に追加されます。pict2 の左上隅が、pict1 の左下隅に置かれます。
- Superimposition (3): pict2 が pict1 の上に置かれます。pict2 の左上隅が、pict1 の左上隅に置かれます。オプションの引数 horOffset と vertOffset を使用すると、合成の前に、pict2 に対して移動が適用されます。horOffset と vertOffset に渡す値はピクセル単位です。右や下に移動するには正数を、左や上に移動するには負数を渡します。

注 COMBINE PICTURES コマンドで実行される重ね合わせは、& や | 演算子で提供される重ね合わせと異なります。COMBINE PICTURES コマンドはソースピクチャの特性を保持するのに対し、& や | 演算子はピクセルごとに処理を行いビットマップピクチャを生成しました。これらの演算子とはもともと白黒のピクチャのために用意されたものです。

▼ 以下のソースピクチャがあります



COMBINE PICTURES(flag;mybackground;Superimposition;mycircle;50;30)



参照：[TRANSFORM PICTURE](#)

CONVERT PICTURE

CONVERT PICTURE(*picture*; *codec*)

引数	タイプ	説明
<i>picture</i>	Picture	→ 変換するピクチャ
		← 変換されたピクチャ
<i>codec</i>	String	→ ピクチャ CODEC ID

CONVERT PICTURE コマンドはピクチャを新しいタイプに変換します。*codec* 引数には新たに生成するピクチャのタイプを指定します。CODEC は拡張子 (例 ".gif")、mime type (例 "image/jpeg") または 4 文字の QuickTime コード (例 "PNTG") を使用できます。利用可能な CODEC のリストは [PICTURE CODEC LIST](#) コマンドで取得できます。

ピクチャフィールドや変数が合成タイプの場合 (たとえばコピー / ペーストアクションの結果のピクチャ)、CODEC タイプに対応する情報のみが結果のピクチャに保持されます。

- ▼ vpPhoto ピクチャを jpeg フォーマットに変換する:

CONVERT PICTURE(vpPhoto;"jpg")

参照：[PICTURE CODEC LIST](#)

4D Pack で廃止されたコマンド

4D Pack で提供されていたほとんどのピクチャ管理コマンドは廃止されました。既存のデータベースを変換した際には、これらのコマンドを置き換える必要があります。詳細は [ページ 404](#)、[annexe C](#)、["4D Pack v11"](#) を参照してください。

グラフ

4D v11 SQL には SVG 描画エンジンが含まれます。SVG (Scalable Vector Graphics) はグラフィックファイルフォーマットです (.svg 拡張子)。このフォーマットに関する詳細は、以下のアドレスを参照してください:

<http://www.w3.org/Graphics/SVG/>

このエンジンを有効に活用するために、[GRAPH](#) と [GRAPH SETTINGS](#) コマンドは *area* 引数にピクチャ変数を受け入れるようになりました。この

場合、グラフィックの表示は SVG エンジンにより処理されます。

注 ページ 378、コマンド `DOM Find XML element by ID` を使用して 4D SVG 描画エンジンを活用することも可能です。

GRAPH

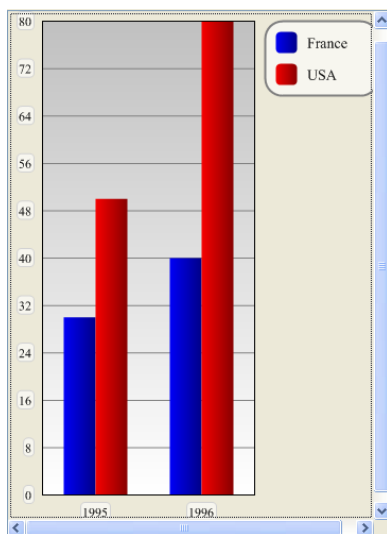
GRAPH(graphArea; graphNumber; xLabels; yElements{; yElements2;...; yElementsN})

引数	タイプ	説明
graphArea	Graph variable ; Picture variable	→ フォーム上のグラフエリア
graphNumber	Number	→ グラフタイプ番号
xLabels	Array	→ x 軸のラベル
yElements	Array	→ グラフデータ

GRAPH コマンドは graphArea 引数にピクチャ変数を受け入れるようになりました。この場合グラフは標準のグラフィックエンジンではなく、4D に統合された SVG 描画エンジンによって生成されます。

この新しい描画にはいくつかの利点があります。モダンなアピアランス、スクロールバーの追加が可能、アプリケーションモードでのコンテキストモードの利用 (ページ 159、" コンテキストメニュー " の節) など。他のコマンドシンタックスに変更はありません。

▼ これは以下のコードで SVG エンジンを使用して作成されたグラフの例です。



```
C_PICTURE(vGraphPict) `ピクチャ変数の宣言
ARRAY STRING(4;X;2) `X 軸配列を作成
X{1}:="1995" `一番目の X ラベル
```

```

X{2}:="1996" `二番目の X ラベル
ARRAY REAL(A;2) ` Y 軸配列を作成
A{1}:=30
A{2}:=40
ARRAY REAL(B;2) ` ふたつめの Y 軸配列を作成
B{1}:=50
B{2}:=80
GRAPH(vGraphPict;l;X;A;B) ` タイプ 1 のグラフを描画
` グラフ凡例の定義
GRAPH SETTINGS(vGraphPict;0;0;0;False;False;True;"France";"USA")

```

GRAPH SETTINGS

```

GRAPH SETTINGS(graph; xmin; xmax; ymin; ymax; xprop; xgrid; ygrid;
title; title2;...;titleN)

```

Parameter	Type	Description
graph	Graph variable ; Picture variable	→ フォーム上のグラフエリア
xmin	Num, Date or Time	→ プロポーショナルグラフの X 軸最小値
xmax	Num, Date or Time	→ プロポーショナルグラフの X 軸最大値
ymin	Num	→ Y 軸最小値
ymax	Num	→ Y 軸最大値
xprop	Boolean	→ True= プロポーショナル X 軸 False= 通常の X 軸
xgrid	Boolean	→ True=X 軸グリッド False=X 軸グリッドなし
ygrid	Boolean	→ True=Y 軸グリッド False=Y 軸グリッドなし
title	String	→ 凡例タイトル

GRAPH SETTINGS コマンドは graph 引数にピクチャ変数を受け入れるようになりました。つまり [GRAPH](#) コマンドを使用して、4D に統合された SVG レンダリングエンジンにより描画されたグラフを、このコマンドで設定できます。詳細は [GRAPH](#) コマンドを参照してください。

他のシンタックスに変更はありません。

文字列

Unicode モードでの文字列変換をサポートするため、2つの新しいコマンド ([CONVERT FROM TEXT](#), [Convert to text](#)) が追加され、また [Ascii](#) コマンドは [Character code](#) に名称変更されました。

新しい [Get localized string](#) コマンドは XLIFF アーキテクチャを利用でき、[Match regex](#) コマンドは正規表現をサポートします。

Num コマンドは、小数点を定義するための新しい引数を受け入れ、String コマンドは新しいタイプの表現式を受け入れます。Uppercase と Lower0 case コマンドはアクセント付き文字を処理するための新しい引数を受け入れます。

CONVERT FROM TEXT

CONVERT FROM TEXT(4Dtext; charSet; convertedBLOB)

引数	タイプ	説明
4Dtext	String	→ 4D のカレント文字セットで表現されたテキスト
charSet	String ; Longint	→ 文字セット番号または名前
convertedBLOB	BLOB	← 変換後テキストを格納した BLOB

CONVERT FROM TEXT コマンドは、4D のカレント文字セットで表現されたテキストを、他の文字セットに変換します。

4Dtext 引数には変換する文字列を渡します。この文字列は 4D の文字セットが使用されていなければなりません。v11 では、4D は Unicode 文字セットを使用しています。

charSet には、変換に使用する文字セットを渡します。文字セットを表す文字列 (例 "ISO-8859-1" や "UTF-8")、または MIBEnum 識別子を渡します。

識別子	IANA 名
1	UTF-16BE
2	UTF-16LE
7	UTF-8
8	UTF-7
9	US-ASCII
10	ebcdic-cp-us
100	x-mac-roman
101	windows-1252
102	x-mac-ce
103	windows-1250
104	x-mac-cyrillic
105	windows-1251
106	x-mac-greek
107	windows-1253
108	x-mac-turkish
109	windows-1254
110	x-mac-arabic

111	windows-1256
112	x-mac-hebrew
113	windows-1255
114	x-mac-ce
115	windows-1257
1000	Shift_JIS
1001	ISO-2022-JP
1002	Big5
1003	EUC-KR
1004	KOI8-R
1005	ISO-8859-1
1006	ISO-8859-2
1007	ISO-8859-3
1008	ISO-8859-4
1009	ISO-8859-5
1010	ISO-8859-6
1011	ISO-8859-7
1012	ISO-8859-8
1013	ISO-8859-9

IANA により、標準的な文字セット名が定義されています。コマンドには、文字セットの主たる名前のほか、エイリアス名も渡すことができます (例 "IO-8859-1" は "CP819"、"csISOLatin1"、"latin1" または "l1" が指定可能です)。

文字セット名については以下の Web サイトを参照してください。

<http://www.iana.org/assignments/character-sets>

コマンド実行後、変換されたテキストは convertedBLOB BLOB に返されます。この BLOB は [Convert to text](#) コマンドで読むことができます。

注 このコマンドは 4D が Unicode モードで動作しているときのみ利用できます。(変換されたデータベースでは、環境設定で Unicode オプションにチェックされていない必要があります。[ページ 61](#)、"[4D データベースにおける Unicode の互換性](#)" の節を参照してください)。互換モード (非 Unicode) で使用されると、convertedBLOB は空になり、OK システム変数に 0 が設定されます。

詳細は [ページ 247](#)、"[Unicode 関連の変更](#)" の節を参照してください。

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

参照：[Convert to text](#)

Convert to text

Convert to text (blob; charSet) → Text

引数	タイプ	説明
blob	BLOB	→ 指定された文字セットで表現されたテキストを含む BLOB
charSet	String ; Longint	→ BLOB に格納された文字の文字セット番号または名前
返り値	Text	← 4D 文字セットで表現されたテキスト

Convert to text コマンドは blob 引数に含まれるテキストを変換し、4D 文字セットで表現されたテキストを返します。4D v11 では Unicode 文字セットが使用されます。

charSet には、blob に含まれるテキストが使用している文字セットを渡します。文字セットの標準名 (例 "ISO-8859-1" や "UTF-8")、または "Character Set" テーマで定義された定数を渡すことができます。詳細は [ページ 348](#)、[コマンド CONVERT FROM TEXT](#) を参照してください。

注 このコマンドは 4D が Unicode モードで動作しているときにのみ利用できます。(変換されたデータベースでは、環境設定で Unicode オプションにチェックされていないならばなりません。[ページ 61](#)、"[4D データベースにおける Unicode の互換性](#)" の節を参照してください)。互換モード (非 Unicode) で使用されると、convertedBLOB は空になり、OK システム変数に 0 が設定されます。

詳細は [ページ 247](#)、"[Unicode 関連の変更](#)" の節を参照してください。

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

参照：[CONVERT FROM TEXT](#)

Character code

Character code(character) → Number

引数	タイプ	説明
character	Alpha	→ コードを取得する文字
返り値	Number	← 文字コード

注 以前のバージョンでは、このコマンドは Ascii という名前でした。Unicode をサポートするため、名称が変更されました。

Character code コマンドは character 引数に渡された文字の文字コードを返します。データベースの実行モード (Unicode モードまたは互換モード) により、コマンドは Unicode コードまたは ASCII 文字コードを返します。

Get localized string

Get localized string (resName) → String

引数	タイプ	説明
resName	String	→ resname 属性値
返り値	String	← カレントランゲージの、resname に対応する文字列

このコマンドは XLIFF アーキテクチャで動作します。XLIFF については [ページ 98](#)、"[XLIFF 標準のサポート](#)" の節を参照してください。

Get localized string コマンドは、カレントランゲージの、resName 属性で指定された文字を返します。

resName には、取得したい翻訳済み文字列に対応するリソース名を渡します。

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。resName が見つからない場合、空の文字列が返され、OK システム変数に 0 が設定されます。

- ▼ XLIFF ファイルが以下であるとき：

```
<file source-language="en-US" target-language="ja">
[...]
```

```
  <trans-unit resname="Show on disk">
    <source>Show on disk</source>
    <target> デスクトップに表示 </target>
  </trans-unit>
```

以下を実行すると：

```
$JValue:=Get localized string("Show on Disk")
```

カレントランゲージが日本語であれば、\$JValue に " デスクトップに表示 " が返されます。

Match regex

Match regex(pattern; aString[, start; pos_found; length_found; *]) → Boolean

引数	タイプ	説明
pattern	Text	→ 正規表現
aString	Text	→ 検索対象文字列
start	Number	→ 検索開始位置
pos_found	Longint Var ; Longint Array	← オカレンスの位置
length_found	Longint Var ; Longint Array	← オカレンスの長さ
*	*	→ 指定された場合：開始位置のみの検索
返り値	Boolean	← TRUE = オカレンスが検索された；そうでなければ FALSE

新しい Match regex コマンドは、正規表現で示されたルールに適合する文字列が存在するかどうかを検証するために使用します。patter には検索パターンの正規表現式を渡します。これは、特別な文字を使用して、文字の並びを定義する文字列で構成されます。

aString には、検索対象の文字列を渡します。

start には、theString 中で検索を開始する文字位置を渡します。

pos_found と length_found が渡されていれば、コマンドはこれらの変数に、検索されたオカレンスの位置と長さを返します。配列を渡すと、コマンドはオカレンスの位置と長さを配列の要素 0 に返し、続く要素に正規表現によってキャプチャされたグループの位置と長さを返します。

オプションの * 引数が渡されると、検索は start 位置から開始され、それ以降は検索されません。

コマンドはオカレンスが見つかりると TRUE を返します。

このコマンドは複数の方法で利用できます：

- 完全に一致する文字列の検索：

```
vfound:=Match regex(pattern;mytext)
```

例：

```
QUERY BY FORMULA([Employees];Match regex("*.smith.*"; [Employees]name))
```

- 位置を指定した検索：

```
vfound:=Match regex(pattern;mytext; start; pos_found; length_found)
```

\$1 のタグをすべて表示する例：

```
start:=1
```

```
Repeat
```

```
  vfound:=Match regex("<.*>";$1;start;pos_found;length_found)
```

```
  If(vfound)
```

```
    ALERT(Substring($1;pos_found;length_found))
```

```
    start:=pos_found+length_found
```

```
  End if
```

```
Until(Not(vfound))
```

- "グループキャプチャ" をサポートした検索：

```
vfound:=Match regex(pattern;mytext; start; arr_pos_found; arr_length_found)
```

例：

```
ARRAY LONGINT(arr_pos_found;0)
```

```
ARRAY LONGINT(arr_length_found;0)
```

```
vfound:=Match regex("(.)stuff(.)";$1;1;arr_pos_found; arr_length_found)
```

```
If (yfound)
```

```
  $group1:=Substring($1;arr_pos_found{1};arr_length_found{1})
```

```
  $group2:=Substring($1;arr_pos_found{2};arr_length_found{2})
```

```
End if
```

- 指定した位置に限定して検索を行う例：

例：

```
vfound:=Match regex("a.b";"---a-b--";1,$pos_found,$length_found)
`returns TRUE
```

```
vfound:=Match regex("a.b";"---a-b--";1,$pos_found,$length_found;*)
`returns FALSE
```

```
vfound:=Match regex("a.b";"---a-b--";4,$pos_found,$length_found;*)
`returns TRUE
```

注 位置と長さは、Unicode モードまたはテキストが 7-bit ASCII タイプである場合にのみ意味を持ちます。

エラーが発生すると、コマンドはエラーを生成します。このエラーは ON ERR CALL コマンドでインストールされたエラー処理メソッドで割り込みできます。

- 正規表現についての詳細は以下の Web サイトを参照してください:

http://en.wikipedia.org/wiki/Regular_expression

- 正規表現のパターンについては以下の Web サイトを参照してください:

<http://www.icu-project.org/userguide/regexp.html>

Num

Num (expression{; separator}) → Number

引数	タイプ	説明
expression	String Boolean Num	→ 数値形式に変換する文字列、ブール、または数値表現式
separator	String	→ 小数点区切り文字
戻り値	String	← 表現の数値形式

- Num コマンドは数値タイプの表現式を受け入れます。この場合、コマンドは expression 引数に渡された値を返します。この機能は特にポインタを使用した汎用プログラミングで有用です。

- さらに、Num コマンドは新しい引数で、小数点区切り文字を受け入れます。デフォルトで、コマンドは OS に設定された区切り文字を使用します。OS に設定された区切り文字と、文字列で使用されているそれとが異なる場合、コマンドは正しくない値を返します。

新しい separator 引数を使用して、このような場合で正しい値を返すようにすることができます。この引数が渡されると、コマンドはシステムの区切り文字を使用しません。一つ以上の文字を渡すことができます。

注 新しい [GET SYSTEM FORMAT](#) コマンドを使用して、地域に対応したシステムの小数点区切り文字を取得できます。

- ▼ この例は、カレントのセパレータにより返される値を比較しています：

```
$theString:="33,333.33"
$thenum:=Num($theString)
  ` デフォルトでフランスシステムでは $thenum が 33,333.33 となります
$thenum:=Num($theString;".")
  ` システムの設定にかかわらず、正しく評価されます；
  ` たとえばフランス語のシステム上でも 33 333,33 となります
```

String

String (expression|; format|) → String

引数	タイプ	説明
expression		→ 文字列に変換する表現式
format	String Number	→ 表示フォーマット
返り値	String	← 変換後の文字列

String コマンドは二つの表現式タイプ、文字列とブールを新たに受け入れるようになりました。

- 文字型の表現式： この場合、コマンドは引数として渡された値と同じ値を返します。この機能は特にポインタを使用した汎用プログラミングで有用です。
- ブール型表現式： この場合、コマンドは "True" または "False" を返します。両方のケースで、format 引数は、それが渡されていても、無視されます。

Uppercase

Uppercase (string|; *) → String

引数	タイプ	説明
string	String	→ 大文字に変換する文字列
*	*	→ 渡されていればアクセントを保持する
返り値	String	← 大文字の文字列

Uppercase 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。この引数が渡されていると、アクセントキャラクタがそのまま大文字に変換されます。

デフォルトで、この引数が省略されると、アクセント文字はそのアクセントを失います。

- ▼ この例は * のあるなしによって返される値を比較します。

```
$thestring:=Uppercase("hélène") ` $thestring = "HELENE"
$thestring:=Uppercase("hélène";*) ` $thestring = "HÉLÈNE"
```

参照：[Lowercase](#)

Lowercase

Lowercase (string; *) → String

引数	タイプ	説明
string	String	→ 小文字に変換する文字列
*	*	→ 渡されていればアクセントを保持する
返り値	String	← 小文字の文字列

Lowercase コマンドの機能はアクセント記号に関する Uppercase のそれと揃えられました。またオプションの * 引数を受け入れるようになりました。

- デフォルトで、この引数が省略されると、Lowercase コマンドはすべての文字を、アクセント記号を取り除いて小文字にします。

注 以前のバージョンでは、アクセント付き大文字は小文字に変換されませんでした。

- オプションの * 引数が渡されると、アクセントはそのまま保持されます。
- ▼ この例は * のあるなしによって返される値を比較します。

```
$thestring:=Lowercase("DÉJÀ VU") ` $thestring is "deja vu"
$thestring:=Lowercase("DÉJÀ VU";*) ` $thestring is "déjà vu"
```

参照：[Uppercase](#)

Position

Position (find; aString; start; lengthFound; *) → Number

Parameter	Type	Description
find	String	→ 検索したい文字列
aString	String	→ 検索対象文字列
start	Number	→ 検索開始位置
lengthFound	Longint	← 検索された文字列の長さ
*	*	→ 渡されていれば：付加記号を区別する検索
返り値	Number	← 最初のオカレンスの位置

Position コマンドは 3 つのオプション引数を受け入れます。

- start 引数は aString 中の検索開始文字を指定するために使用します。
- lengthFound 引数は、検索の結果見つかった文字列の実際の長さを返します。この引数は、一つ以上の文字を含むかもしれない文字を管理するために必要です (例: æ と ae、ß や ss など)。
- * 引数が渡されている場合 (以下参照)、これらの文字は等しいとはみなされません (æ # ae)。

- * 引数が渡されていると、検索は付加記号を区別して行われます。言い換えれば、検索は大文字小文字やアクセント文字を区別して行われます (a#A, a#à...).
- ▼ 以下新しい引数がどのように動作するか示します。:


```
$posFound:=Position("day";"Today is the first day";1) `3 を返す
$posFound:=Position("day";"Today is the first day";4) `20 を返す
$posFound:=Position("DAY";"Today is the first day";1;*) `0 を返す

$posFound:=Position("oe";"Boeuf";1;$stringlength)
    $posFound には 2 が返され、$stringlength には 1 が返されます。
```
- ▼ 以下の例では、lengthFound 引数を使用して、それがどのように書かれているかにかかわらず ("ægis" or "aegis")、テキスト中のすべての "ægis" を検索します。


```
$start:=1
Repeat
    $pos:=Position("ægis";$thetext;$start;$stringlength)
    $start:=$start+$stringlength
Until($pos=0)
```

制御文字に関する互換性 メモ

4D v11 では、Unicode 標準により、文字列中の Char(1) など制御文字の使用を禁止されています。以前のバージョンでは、Position コマンドはこれらの文字に対して使用できました。それゆえ、この動作を適用したい場合もあるかもしれません。たとえば以下の例で:

```
$s1:=Char(1)
$s2:="a"+Char(1)+"b"
$pos:=Position($s1,$s2;1)
```

- 4D 200x では \$pos が 2 になります。
- 4D v11 では \$pos に 1 が返されます。Char(1) が無視され、空文字を検索することになります。

Unicode で禁止されている文字については、以下のアドレスを参照してください: http://www.unicode.org/charts/collation/chart_Null.html
Tip: * 引数を使用して、v11 で以前の動作を保持することができます。先の例でいえば、\$pos:=Position(\$s1,\$s2;1;*) とすると、\$pos に 2 が返ります。

BLOB

Unicode モードで動作するアプリケーションに対応するため、いくつかのコマンドと定数に変更されました ([ページ 247](#)、"Unicode 関連の変更" の節を参照)。

TEXT TO BLOB

TEXT TO BLOB(text; blob; textFormat[; offset | *])

引数	タイプ	説明
text	String	→ BLOB に書き込むテキスト
blob	BLOB	← テキストを受け取る BLOB
textFormat	Number	→ テキストのフォーマットと文字セット
offset *	Variable *	→ BLOB 中のオフセット (byte 単位) または * の場合値の追加 ← * でない場合、新しいオフセット

4D v11 は Unicode 文字セット (UTF8) を使用するため、このコマンドで生成される BLOB は、以前のバージョンのプログラムで生成されるもの (ASCII Mac 文字セット) とは異なります。

互換性のため、Mac Roman 文字セットを強制してこのコマンドを使用することができます。文字セットの選択は textFormat 引数を通して行います。

このため、"BLOB" テーマの既存の定数は名称が変更され、新しい定数が追加されました。

4D v11 "BLOB" テーマ定数	値	以前のバージョンの名称
Mac C String	0	C string
Mac Pascal string	1	Pascal string
Mac text with length	2	Text with length
Mac text without length	3	Text without length
UTF8 C string	4	-
UTF8 text with length	5	-
UTF8 text without length	6	-

注 ・ "UTF8" 定数は、アプリケーションが Unicode モードで動作しているときにのみ使用できます。

・ Mac text with length 定数は、32KB を超えるテキストを処理できません。

以下の表でそれぞれのフォーマットを説明します。

テキストフォーマット	説明	Mac	UTF8
C string	テキストは NULL 文字で終了する	"" → \$00 "Café" → \$43 61 66 8E 21 00	"" → \$00 "Café" → \$43 61 66 C3 A9 21 00

Pascal string	文字長を格納した 1 バイトが文字の前に付加される	"" → \$00 "Café" → \$05 43 61 66 8E 21	-
Text with length	文字長を格納した 2 バイト (Mac) または 4 バイト (UTF8) が文字の前に付加される	"" → \$00 00 "Café" → \$00 05 43 61 66 8E 21	"" → \$00 00 00 "Café" → \$00 00 00 06 43 61 66 C3 A9 21
Text without length	文字のみでテキストが構成される	"" → 値なし "Café" → \$43 61 66 8E 21	"" → 値なし "Café" → \$43 61 66 C3 A9 21

UTF-8 以外文字セットを使用したい場合は [CONVERT FROM TEXT](#) コマンドを使用します。

BLOB to text

BLOB to text(blob; textFormat; offset; textLength)) → Text

引数	タイプ	説明
blob	BLOB	→ テキストを取り出す BLOB
textFormat	Number	→ テキストのフォーマットと文字セット
offset	Variable	→ BLOB 中のオフセット (byte 単位) ← 新しいオフセット
textLength	Number	→ 読み込む文字数
返り値	String	← テキスト

4D v11 は Unicode 文字セット (UTF8) を使用するため、このコマンドで生成される BLOB は、以前のバージョンのプログラムで生成されるもの (ASCII Mac 文字セット) とは異なります。

互換性のため、Mac Roman 文字セットを強制してこのコマンドを使用することができます。文字セットの選択は textFormat 引数を通して行います。このため、"BLOB" テーマの既存の定数は名称が変更され、新しい定数が追加されました。

4D v11 定数	値	以前のバージョンの名称
Mac C String	0	C string
Mac Pascal string	1	Pascal string
Mac text with length	2	Text with length
Mac text without length	3	Text without length
UTF8 C string	4	-
UTF8 text with length	5	-
UTF8 text without length	6	-

注 ・ "UTF8" 定数は、アプリケーションが Unicode モードで動作しているときにのみ使用できます。

・ Mac text with length 定数は、32KB を超えるテキストを処理できません。

これらのフォーマットについての詳細は [ページ 348](#)、[コマンド CONVERT FROM TEXT](#) を参照してください。

UTF-8 以外の文字セットを使用したい場合は [Convert to text](#) コマンドを使用します。

ランゲージ

EXECUTE METHOD コマンドが 4D v11 の "ランゲージ" テーマに追加されました。明確にするために、既存の EXECUTE コマンドは [EXECUTE FORMULA](#) に名称が変更され、"フォーミュラ" テーマに移動されました。

EXECUTE METHOD

```
EXECUTE METHOD(methodName; result ! * {; param1{;
param2;...;paramN{}}
```

引数	タイプ	説明
methodName	String	→ 実行するプロジェクトメソッド名
result ! *	Variable ! *	← 戻り値を受け取る変数、またはメソッドが値を返さない場合 *
param1...paramN	Expression	→ メソッドの引数

新しい EXECUTE METHOD コマンドは、param1...paramN 引数を使用して、methodName プロジェクトメソッドを実行します。データベースで実行可能なすべてのメソッド、またはコマンドが実行されているコンポーネントのメソッドを実行できます。

result には、methodName メソッドから返される (methodName 内部で \$0 に置かれる) 値を受け取る変数を渡します。メソッドが値を返さない場合、* を第二引数に渡します。

実行のコンテキストは呼ばれたメソッドに保持されます。つまりカレントフォームやカレントフォームイベントは引き続き定義されています。このコマンドがコンポーネントから呼ばれ、methodName にホストデータベースのメソッド名が渡される場合、あるいはその逆の場合も、メソッドはコンポーネントとホストデータベースで共有されていなければなりません ("コンポーネントとホストデータベースで共有する" オプション。 [ページ 54](#)、"[プロジェクトメソッドの共有](#)" の節を参照)。

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

EXECUTE FORMULA

EXECUTE コマンドは、4D v11 で EXECUTE FORMULA に名称が変更されました。ただし、機能は変更されていません。

この変更は新しいコマンド [EXECUTE METHOD](#) との混乱を避け、より機能を明確にするために行われました。

Self

Self コマンドは、オブジェクトメソッドから直接または間接に呼び出されるプロジェクトメソッドでも利用可能になりました。

さらに、このコマンドがリストボックスのコンテキストで呼び出されると、Self コマンドは以下を返します：

- フィールドが関連付けられた列の場合、そのフィールドへのポインタ。
- 変数が関連付けられた列の場合、その変数へのポインタ。
- 式が関連付けられた列の場合、Nil ポインタ。

v11 におけるリストボックスの新しい機能については [ページ 311](#)、"[リストボックス](#)" の節を参照してください。

プロセス変数へのポインタ

v11 で、プロセス変数へのポインタの機能が変更されました。以前のバージョンでは、異なるプロセスでポインタのポイント先を参照する際に、インタプリタモードとコンパイルモードで動作が異なりました。たとえばプロセス 1 で：

```
C_TEXT(vVar)
vVar:="hello"
<>pointer_vVar:=->vVar
プロセス 2 で：
```

```
ALERT(<>pointer_vVar->)
コンパイルモードでは、警告ダイアログに "hello." が表示され、以前のバージョンのインタプリタモードではこの場合空の文字 (プロセス 2 の vVar) が表示されていました。
今バージョンから、インタプリタモードでの動作はコンパイルモードのそれに揃えられ、両方のケースで "hello" が表示されるようになります。
```

リソース

"リソース" テーマの二つのコマンド [Get indexed string](#) と [STRING LIST TO ARRAY](#) が v11 で変更され、XLIFF アーキテクチャで使用できるようになりました。

注 XLIFF について詳細は [ページ 98](#)、"[XLIFF 標準のサポート](#)" の節を参照してください。

Get indexed string

Get indexed string (resID; strID; resFile) → String

Get indexed string コマンドは、4D v11 で XLIFF アーキテクチャと互換になりました。コマンドは (resFile が省略されていれば) まず resID と strID に対応するリソースを、開かれたすべての XLIFF ファイルから探します。この場合、resID は group 要素の id 属性値に、strID は trans-

unit の id 属性値に対応します。
 値が見つからなければ、コマンドは開かれたリソースファイルを検索します。

STRING LIST TO ARRAY

STRING LIST TO ARRAY (resID; strings; resFile)

STRING LIST TO ARRAY コマンドは、4D v11 で XLIFF アーキテクチャと互換になりました。コマンドは (resFile が省略されていれば) まず resID と strID に対応するリソースを、開かれたすべての XLIFF ファイルから探します。この場合、resID は group 要素の id 属性値に、strID は trans-unit の id 属性値に対応します。
 値が見つからなければ、コマンドは開かれたリソースファイルを検索します。

通信

[SEND PACKET](#) と [RECEIVE PACKET](#) コマンドの機能が拡張され、[USE CHARACTER SET](#) コマンドは Unicode モードで異なる動作をします。

SEND PACKET

SEND PACKET({docRef; }packet)

引数	タイプ	説明
docRef	DocRef	→ ドキュメント参照またはクライアントチャンネル
packet	String BLOB	→ 送信するテキストまたは BLOB

このコマンドは BLOB を送信できるようになりました。これにより、テキストモードでの文字セットの制限から解放されます。

注 BLOB タイプの引数をこのコマンドに渡すと、[USE CHARACTER SET](#) で指定された文字セットは考慮されません。BLOB は変更されずに送信されます。

- ▼ この例は、BLOB を使用してドキュメントに拡張文字を送信また受け取ります。

```
C_BLOB($send_blob)
```

```
C_BLOB($receive_blob)
```

```
TEXT TO BLOB("ázértý";$send_blob;Text without length)
```

```
SET BLOB SIZE($send_blob;16;255)
```

```
$send_blob|6|:=0
```

```
$send_blob|7|:=1
```

```
$send_blob|8|:=2
```

```
$send_blob|9|:=3
```

```
$send_blob|10|:=0
```

```

$vlDocRef:=Create document("blob.test")
If (OK=1)
  SEND PACKET($vlDocRef;$send_blob)
  CLOSE DOCUMENT($vlDocRef)
End if

```

```

$vlDocRef:=Open document(document)
If (OK=1)
  RECEIVE PACKET($vlDocRef;$receive_blob;65536)
  CLOSE DOCUMENT($vlDocRef)
End if

```

参照：[RECEIVE PACKET](#)

RECEIVE PACKET

RECEIVE PACKET({docRef; }receiveVar; stopChar ; numChars)

引数	タイプ	説明
docRef	docRef	→ ドキュメント参照またはカレントチャンネル
receiveVar	String variable ; BLOB variable	→ データを受け取る変数
stopChar ; numChars	String ; Num	→ 受信を停止する文字、または読み込む文字数

このコマンドは、[SEND PACKET](#) コマンドで送信された BLOB パケットを受信することができるようになりました。この場合、receiveVar 変数は BLOB でなければなりません。

- ▼ [SEND PACKET](#) コマンドの例を参照してください。

注 このコマンドが BLOB タイプのパケットを受信すると、[USE CHARACTER SET](#) で指定された文字セットは考慮されません。BLOB は変更されません。

参照：[SEND PACKET](#)

USE CHARACTER SET

USE CHARACTER SET(map ; *; mapInOut)

注 USE CHARACTER SET は前のバージョンで USE ASCII MAP という名称でした。

4D アプリケーションが Unicode モードで実行されているとき、map 引数

が渡される場合、この引数は使用する文字セットの "IANA" 名やそのエイリアスに対応していなければなりません。たとえば "iso-8859-1" や "utf-8"、エイリアス "latin1" などが有効です。IANA 名に関する詳細は以下の Web サイトを参照してください。

<http://www.iana.org/assignments/character-sets>

* 引数が渡されると、デフォルトの文字セットが設定されます。4D v11 では、UTF-8 です。

注 4D アプリケーションが非 Unicode モード (互換モード) で実行されている場合、このコマンドの動作は以前のバージョンと同じです。

ストラクチャアクセス

"ストラクチャアクセス" テーマのいくつかのコマンドが変更され、テーブルやフィールドの削除に対応できるようになりました。さらに、[Field, Field name](#)、そして [SET INDEX](#) コマンドが変更されました。

テーブルとフィールドのカウン

4D v11 では、テーブルとフィールドを削除できるようになりました ([ページ 30](#)、"[テーブルとフィールドの削除](#)" の節を参照)。

この結果、テーブルやフィールドの数を数えるアルゴリズムに変更を行う必要が生じました。実際、Count tables や Count fields コマンドは削除を考慮に入れておらず、これらのオブジェクトのカウンとの間に差異が生じます。たとえばテーブル番号 1、2、3、5 の 4 つのテーブルを持つデータベース (テーブル 4 が削除された) で、Count tables は 5 を、すなわち最も数の大きいテーブル番号返します。

この理由により、Count tables と Count fields コマンドはそれぞれ Get last table number と Get last field number に名称が変更されました。さらに Is table number valid と Is field number valid コマンドが追加され、データベースのテーブル数とフィールド数を正しくカウントできるようにしました。

いかにそのアルゴリズムを示します：

```
For($thetable;1;Get last table number)
  If (Is table number valid($thetable))
    For($thefield;1;Get last field number($thetable))
      If(Is field number valid($thetable;$thefield))
        `フィールドは存在し有効
      End if
    End for
  End if
End for
```

Get last table number

Get last table number → Longint

引数	タイプ	説明
返り値	Longint	← データベース中の最大テーブル番号

注 このコマンドは名称が変更されました。以前のバージョンでは Count tables という名称でした。

Get last table number コマンドは、データベースのテーブル番号のうち最大のものを返します。

テーブルは作成順に番号がふられます。データベースでテーブルが削除されていないければ、このコマンドはデータベースに存在するテーブルの数を返します。データベースのテーブルを反復してループする場合、Is table number valid コマンドを使用して、テーブルが削除されていないかテストしなければなりません。

- ▼ 以下の例は tablesArray 配列を初期化します。この配列はフォームにデータベースのテーブルを表示するために使用できます。

```
ARRAY STRING (31;tablesArray;Get last table number)
For ($vTables;Size of array(tablesArray);1;-1)
  If(Is table number valid($vTables))
    tablesArray {$vTables;}:=Table name ($vTables)
  Else
    DELETE FROM ARRAY(tablesArray;$vTables)
  End if
End for
```

参照：[Get last field number](#), [Is table number valid](#)

Get last field number

Get last field number (tableNum | tablePtr) → Longint

引数	タイプ	説明
tableNum tablePtr	Number Pointer	→ Table number or Pointer to table
返り値	Longint	← Highest field number in table

注 このコマンドは名称が変更されました。以前のバージョンでは Count fields という名称でした。

Get last field number コマンドは、テーブル番号またはポインタが table0 Num または tablePtr 引数に渡されたテーブルのフィールド番号のうち最大のものを返します。

フィールドは作成順に番号がふられます。テーブルからフィールドが削除されていないければ、このコマンドはテーブルに存在するフィールドの数を返します。テーブルのフィールドを反復してループする場合、Is field

number valid コマンドを使用して、フィールドが削除されていないかテストしなければなりません。

- ▼ 以下のプロジェクトメソッドは、引数に渡されたテーブルの名前を使用して、fieldArray 配列を作成します。

```

$vlTable:=Table($1)
ARRAY STRING(31;fieldsArray;Get last field number($vlTable))
For ($vlField;Size of array(fieldsArray);1;-1)
  If(Is field number valid($vlTable;$vlField))
    fieldsArray{$vlField}:=Field name($vlTable;$vlField)
  Else
    DELETE FROM ARRAY(fieldsArray; $vlField)
  End if
End for

```

参照：[Get last table number](#), [Is field number valid](#)

Is table number valid

Is table number valid (tableNum) → Boolean

引数	タイプ	説明
tableNum	Longint	→ テーブル番号
返り値	Boolean	← TRUE = テーブルが存在する FALSE = テーブルは存在しない

新しい Is table number valid コマンドは、tableNum に渡されたテーブル番号のテーブルが存在すれば TRUE を、存在しなければ FALSE を返します。テーブルがエクスプローラのゴミ箱にある場合、FALSE が返されることに注意してください。

このコマンドはテーブルの削除をテストするために使用できます。

参照：[Get last table number](#), [Is field number valid](#)

Is field number valid

Is field number valid (tableNum | tablePtr; fieldNum) → Boolean

引数	タイプ	説明
tableNum tablePtr	Number Pointer	→ テーブル番号またはテーブルへのポインタ
fieldNum	Longint	→ フィールド番号
返り値	Boolean	← TRUE = フィールドが存在する FALSE = フィールドは存在しない

新しい Is field number valid コマンドは、tableNum または tablePtr で指定されたテーブルに、fieldNum で指定したフィールドが存在すれば TRUE を、存在しなければ FALSE を返します。フィールドを含むテーブルがエクスプローラのゴミ箱にある場合、FALSE が返されることに注意してください。

このコマンドはフィールドの削除をテストするために使用できます。

参照：[Get last field number](#), [Is table number valid](#)

SET INDEX

SET INDEX(field; index{; mode}; *)

引数	タイプ	説明
field	Field	→ インデックスを作成または削除するフィールド
index	Boolean Integer	→ インデックスを作成 (TRUE) またはインデックスを削除 (FALSE) またはインデックス作成: -1 = キーワード, 0 = デフォルト, 1 = 標準 B-Tree, 3 = クラスタ B-Tree
mode	Longint	→ インデックス作成モード (%)
*		→ 渡されていれば = 非同期

SET INDEX コマンドは、新しいタイプのインデックスをサポートするために変更されました。

index 引数は整数値を受け入れるようになりました。この場合、このコマンドは指定されたタイプのインデックスを作成します。"Index Type" テーマの以下の定数を渡すことができます。

- Keywords Index (-1): フィールド内容の単語ごとにインデックスを作成するキーワードインデックスを作成します。このタイプのインデックスはテキストや文字タイプにのみ作成できます。
単語は英語などでのスペース等で区切られたものです。日本語もスペースで区切られていればキーワードインデックスを作成することはできます。
- Default Index Type (0): この場合、フィールドの内容に基づき 4D がもっとも最適なインデックスを (キーワードインデックスを除く) 決定します。
- Standard BTree Index (1): 標準 B-Tree タイプのインデックス。この汎用目的のインデックスは以前の 4D で使用されてきました。
- Cluster BTree Index (3): クラスタを使用した B-Tree タイプインデックス。このタイプのインデックスは、インデックスに格納されるキー数が少ない時、たとえばデータに同じ値が繰り返し現れる場合に最適化されています。

新しいタイプのインデックスについては[ページ 35](#)、"[インデックスの管理](#)" の節を参照してください。

注 このコマンドで作成されたインデックスには名前がありません。そのため **DELETE INDEX** コマンドで名前を指定して削除することはできません。

index 引数に整数を渡した場合、mode 引数は必要ありません。コマンドは index 引数に引き続きブールタイプを受け入れます (以前のシンタックス)。TRUE を渡すと、コマンドはデフォルトタイプのインデックスを作成します。FALSE を渡すとコマンドはフィールドに関連付けられた複合インデックス以外のインデックスをすべて削除します。

注 ・このコマンドは複合インデックスを作成したり削除したりしません。
 ・このコマンドは **CREATE INDEX** コマンドで作成されたキーワードインデックスを削除することはできません。

▼ 以下の例はこのコマンドの二つのシンタックスを示します。

```

  `キーワードインデックスを作成
SET INDEX([Books]Summary; -1)
  `Summary フィールドの複合インデックス以外すべてのインデックスを
削除
SET INDEX([Books]Summary; FALSE)

```

CREATE INDEX

CREATE INDEX(table; fieldsArray; indexType; indexName{*}; *)

引数	タイプ	説明
table	Table	→ インデックスを作成するテーブル
fieldsArray	Array pointer	→ インデックスを作成するフィールドへのポインタ
indexType	Integer	→ 作成するインデックスのタイプ: -1 = キーワード, 0 = デフォルト, 1 = 標準 B-Tree, 3 = クラスタ B-Tree
indexName	Text	→ インデックスの名称
*		→ 渡された場合 = 非同期

CREATE INDEX コマンドは以下のインデックスを作成するために使用します。

- ひとつ、またはそれ以上 (複合インデックス) のフィールドの標準インデックス
- フィールドのキーワードインデックス

fieldsArray 配列で指定された一つ以上のフィールドに対しインデックスを作成します。単純なインデックスを作成する場合、配列には一つの要素が含まれ、複合インデックスを作成する場合は二つ以上の要素が含まれます (キーワードインデックスの場合を除く)。複合インデックスを作成する場合、配列要素の順番は重要です。

indexType 引数で作成するインデックスのタイプを指定します。"Index

Type" テーマの以下の定数を渡すことができます。

- Keywords Index (-1): キーワードインデックス。キーワードは複合インデックスにできないことに留意してください。fieldsArray 配列には一つのフィールドのみが渡せます。
- Default Index Type (0): この場合、4D はフィールドの内容に応じて (キーワードを除く) 最適なインデックスタイプを決定し作成します。
- Standard BTree Index (1): 標準 B-Tree タイプのインデックス。この汎用目的のインデックスは以前の 4D で使用されていました。
- Cluster BTree Index (3): クラスタを使用した B-Tree タイプインデックス。このタイプのインデックスは、インデックスに少数のキーしか含まれない場合に最適です。例えばデータに繰り返し同じデータが現れる場合です。

インデックスタイプについての詳細は、[ページ 35](#)、"[インデックスの管理](#)" の節を参照してください。

indexName 引数には、作成するインデックスの名前を渡します。この名前は必須です。空の文字列を渡すとエラーが生成されます。indexName インデックスが既に存在する場合、コマンドは何も行いません。

オプションの * 引数が渡されると、インデックス作成は非同期で行われます。このモードではこのコマンドを実行したメソッドは、インデックス作成の終了を待たずに実行が継続されます。

CREATE INDEX はロックされたレコードを見つけると、そのレコードのロックが外れるまで待ちます。

コマンド実行中に (インデックスできないフィールド、複数フィールドのキーワードインデックス作成など) 問題が発生すると、エラーが生成されます。このエラーはエラー処理メソッドでとらえることができます。

- ▼ [Customers] テーブルの "Last Name" と "Telephone" フィールドに標準のインデックスを作成します。

```
ARRAY POINTER(fieldPtrArray;1)
fieldPtrArray{1};=->[Customers]LastName
CREATE INDEX([Customers];fieldPtrArray;Standard BTree Index;
"CustLNameIdx")
fieldPtrArray{1};=->[Customers]Telephone
CREATE INDEX([Customers];fieldPtrArray;Standard BTree Index;
"CustTelIdx")
```

- ▼ [Customers] テーブルの "Observations" フィールドにキーワードインデックスを作成します。

```
ARRAY POINTER(fieldPtrArray;1)
fieldPtrArray{1};=->[Customers]Observations
CREATE INDEX([Customers];fieldPtrArray;Keywords Index;"CustObsIdx")
```

- ▼ [Customers] テーブルの "City" と "Zipcode" フィールドに複合インデックスを作成します。

```
ARRAY POINTER(fieldPtrArray;2)
fieldPtrArray|1|:=->[Customers]City
fieldPtrArray|2|:=->[Customers]Zipcode
CREATE INDEX([Customers];fieldPtrArray;Standard BTree Index;"CityZip")
```

参照：[DELETE INDEX](#), [SET INDEX](#)

DELETE INDEX

```
DELETE INDEX(fieldPtr | indexName | ; *)
```

引数	タイプ	説明
fieldPtr indexName	Pointer Text	→ インデックスを削除するフィールドへのポインタまたは削除するインデックスの名前
*		→ 渡されていれば = 非同期

DELETE INDEX コマンドはデータベースから既存のインデックスを削除するために使用します。

フィールドへのポインタまたはインデックスの名称を指定できます。

- フィールドへのポインタを渡した場合 (fieldPtr)、フィールドに関連付けられたキーワードや標準すべてのインデックスが削除されます。フィールドが複合インデックスに含まれる場合、それらも削除されます。
- インデックス名を渡した場合 (indexName)、指定されたキーワードや標準などのインデックスのみが削除されます。フィールドが他のインデックスを含んでいたり、他の複合インデックスに属している場合、それらは削除されません。

オプションの * 引数が渡されると、インデックス削除は非同期で行われます。このモードではこのコマンドを実行したメソッドは、インデックス削除の終了を待たずに実行が継続されます。

fieldPtr や indexName, に対応するインデックスがない場合、コマンドは何も行いません。

- ▼ 以下の例はコマンドの両シンタックスを示します。

```
LastName フィールドに関連するすべてのインデックスを削除
DELETE INDEX(->[Customers]LastName)
"CityZip" インデックスを削除
DELETE INDEX("CityZip")
```

参照：[CREATE INDEX](#), [SET INDEX](#)

Field, Field name

サブフィールドへのポインタを渡すと、これらのコマンドはソースフィールドではなく、サブフィールドの番号や名前を返すようになります。

した。

Web サーバ

"Web サーバ" テーマに、Digest Web 認証モードで利用できる新しい `Validate Digest Web Password` コマンドが追加されました。`SET HTML ROOT` コマンドは変更されました。

Validate Digest Web Password

`Validate Digest Web Password(userName; password) → Boolean`

引数	タイプ	説明
<code>userName</code>	Text	→ ユーザ名
<code>password</code>	Text	→ パスワード
返り値	Boolean	← TRUE= 認証 OK FALSE= 認証失敗

`Validate Digest Web Password` コマンドは、Web サーバに接続したユーザより提供された名前とパスワードによる、識別情報の検証に使用できます。このコマンドは Digest モードの Web 認証コンテキストの On Web Authentication データベースメソッドで使用されなければなりません ([ページ 235](#)、"[Digest モードによる認証](#)" の節を参照)。

`userName` と `password` 引数には、ローカルに格納されたユーザの識別情報を渡します。コマンドはこの情報を使用して、Web ブラウザから送信された情報と比較する値を生成します。

値が同じであれば、コマンドからは TRUE が返され、そうでなければ FALSE が返されます。

このメカニズムを使用して、プログラミングにより、セキュアなアクセスシステムを保守、管理することができます。Digest 認証は 4D のパスワードシステムともには利用できないことに留意してください。

注 ブラウザが Digest 認証をサポートしていない場合、認証エラーが返されます。

▼ Digest モードの On Web Authentication データベースメソッドの例

```
` On Web Authentication Database Method
C_TEXT($1;$2;$3;$4;$5;$6)
C_TEXT($user)
C_BOOLEAN($0)
```

```
$0:=FALSE(ãU)
```

```
$user:=$5
```

```
`For security reasons, refuse names containing @
```

```
If (WithWildcard($user))
```

```
  $0:=FALSE(ãU)
```

```
  `The WithWildcard method is described in the 4D documentation
```

```

Else
  `Check whether it is a 4D user
  QUERY([WebUsers];[WebUsers]User=$user)
  $0:=Validate Digest Web Password($user:[WebUsers]password)
Else
  $0:=FALSE(ãU) `User does not exist
End if
End if

```

SET HTML ROOT SET HTML ROOT(rootFolder)

引数	タイプ	説明
rootFolder	String	→ Web サーバルートフォルダのパス名

SET HTML ROOT コマンドは、4D v11 で機能が拡張されました。

- このコマンドは非コンテキストモードでも制限なしに機能するようになりました。このコマンドはカレントセッションのすべての Web プロセスのルートフォルダを変更します。
 - カレントプラットフォームのシンタックスを使用して、絶対パスによるルートフォルダの位置指定ができるようになりました。よって、rootFolder 引数は以下のタイプの文字列を受け入れます。
 - (Mac OS) Disk:Applications:myserv:folder
 - (Windows) C:\Applications\myserv\folder
- もちろん以前の相対 "URL" タイプのシンタックスも有効です。

注 以前のバージョン (コンテキストモード) では、環境設定に入力されたパスが考慮され、値が追加されました。今バージョンからはこのコマンドがルートフォルダを完全に定義し、環境設定は考慮されなくなります。

ツール

"ツール" テーマに新しい [Choose](#) コマンドおよび、マルチコンポーネント環境のメソッドエディタでマクロの使用を可能にする新しいユーティリティコマンド [GET MACRO PARAMETER](#) と [SET MACRO PARAMETER](#) が追加されました。

さらに、既存の LAUNCH EXTERNAL PROCESS と SET ENVIRONMENT VARIABLE コマンドが変更されました。

注 4D v11 の "ツール" テーマには、他のテーマから移動されたコマンドも含まれます ([ページ 388](#)、"[テーマの変更](#)" の節を参照)。

Choose

Choose (criterion; value1 {; value2;...; valueN}) → Value

引数	タイプ	説明
criterion	Boolean Integer	→ テストする値
value1...N	Expression	→ とりうる値
返り値	Expression	← テスト結果に基づく value1, value2 または valueN

Choose コマンドは criterion 引数の値に基づき、value1 または value2 または valueN を返します。

criterion 引数にはブールまたは数値タイプを渡せます。

- criterion がブールの場合、Choose コマンドはブールが TRUE のとき value1 を、FALSE のとき value2 を返します。

この場合、このコマンドは3つの引数、criterion、value1 そして value2 を要求します。

- criterion が整数の場合、Choose は criterion の位置にある value を返します。カウントは0から始まることに注意してください。つまり value1 の位置は0です。この場合、コマンドは少なくとも2つの引数、criterion と value1 を要求します。value 引数には、ピクチャ、ポインタ、BLOB、配列を除くデータタイプを渡せます。しかしすべての value が同じタイプでなければなりません。4D はこの点に関する検証を一切行いません。

criterion に対応する値がない場合、Choose は value 引数に対応する "null" 値を返します (たとえば数値タイプの場合 0 が、文字タイプの場合空の文字が返されます)。

このコマンドは、Case of に代わり、いくつかのテストを行う簡易なコードを書けるようにします。またフォーミュラが実行できるクエリエディタ、クイックレポート、リストボックスの式などで使用できます。

- ▼ これはブールタイプを criterion に使用する定型的な例です。

```
vTitle:=Choose([Person]Masculine;"Mr";"Ms")
```

このコードは以下と同じです。

```
if([Person]Masculine)
  vTitle:="Mr"
Else
  vTitle:="Ms"
End if
```

- ▼ これは数値タイプを criterion に使用する典型的な例です。

```
vStatus:=Choose([Person]Status;"Single";"Married";"Widowed";"Divorced")
```

このコードは以下と同じです。

```

Case of
:([Person]Status=0)
  vStatus:="Single"
:([Person]Status=1)
  vStatus:="Married"
:([Person]Status=2)
  vStatus:="Widowed"
:([Person]Status=3)
  vStatus:="Divorced"
End case

```

GET MACRO PARAMETER

GET MACRO PARAMETER(selector; textParam)

引数	タイプ	説明
selector	Longint	→ 使用するセクション
textParam	Text	← 返されるテキスト

GET MACRO PARAMETER コマンドは、textParam 引数に、マクロを呼び出したメソッド中の、すべてまたは一部の文字列を返します。selector 引数は返される情報のタイプを設定するために使用します。"4D Environment" に追加された以下の定数を指定します。

定数	型	値
Full method text	Longint	1
Highlighted method text	Longint	2

selector に Full method text を渡すと、textParam にメソッドのすべてのテキストが返されます。Highlighted method text を渡すと、textParam にはメソッド中で選択されている文字列のみが返されます。

参照：[SET MACRO PARAMETER](#)

SET MACRO PARAMETER

SET MACRO PARAMETER(selector; textParam)

引数	タイプ	説明
selector	Longint	→ 使用するセクション
textParam	Text	→ 挿入するテキスト

[SET MACRO PARAMETER](#) コマンドは、textParam テキストを、マクロを呼び出したメソッドに挿入します。

メソッド中でテキストが選択されている場合、selector 引数を使用して、textParam テキストでメソッド全体を置き換えるか、選択部分のみを置き換えるか、指定できます。selector には、"4D Environment" テーマに

追加された以下の定数を使用できます。

定数	型	値
Full method text	Longint	1
Highlighted method text	Longint	2

テキストが選択されていなければ、textParam はメソッドに挿入されま
す。

▼ 例:

```
C_TEXT($input_text)
C_TEXT($output_text)
GET MACRO PARAMETER(Highlighted method text;$input_text)
  `選択されているのがテーブルだとする、たとえば "[Customers]"
  `このマクロは、新しいコードを構築し、メソッドに送信する
$output_text:=""
$output_text:=$output_text+Command name(47)+"("+input_text+")"
  `Select all ([Customers])
$output_text:=$output_text+"$i:="+Command name(76)+"("+
$input_text+")"
  ` $i:=Records in selection([Customers])
$output_text:=$output_text+...etc
SET MACRO PARAMETER(Highlighted method text;$output_text)
  `選択文字列と置き換える
```

参照：[GET MACRO PARAMETER](#)

注 [GET MACRO PARAMETER](#) と [SET MACRO PARAMETER](#) コマンドが正しく動作するには、マクロで "version" 属性が正しく宣言されていなければなりません。

```
<macro name="MyMacro" version="2">
-- Text of macro --
</macro>
```

LAUNCH EXTERNAL PROCESS

```
LAUNCH EXTERNAL PROCESS(fileName; inputStream; output0
Stream; errorStream{||})
```

注 以前のバージョンで、このコマンドは "システム環境" テーマに属して
いました ([ページ 388](#)、"[テーマの変更](#)" の節を参照)。

このコマンドを使用して、非同期で外部プロセスを呼び出せるようにな
りました。これを行うには
[_4D_OPTION_BLOCKING_EXTERNAL_PROCESS](#) 環境変数 ([SET ENVIO
RONMENT VARIABLE](#) コマンドを参照) を使用します。以前のバージョンで

は、同期での実行のみが可能でした。

この場合、このコマンドは `outputStream` と `errorStream` 引数に値を返さないことに注意してください。

参照：[SET ENVIRONMENT VARIABLE](#)

SET ENVIRONMENT VARIABLE

SET ENVIRONMENT VARIABLE (varName; varValue)

注 以前のバージョンで、このコマンドは " システム環境 " テーマに属していましたが ([ページ 388](#)、" [テーマの変更](#) " の節を参照)。

このコマンドを [LAUNCH EXTERNAL PROCESS](#) と共に使用する際、新しい環境変数 `_4D_OPTION_BLOCKING_EXTERNAL_PROCESS` を使用できるようになりました。この変数は外部プロセスを非同期で実行するために使用します。言い換えれば他のアプリケーションをブロックしません。同期で実行するには `varValue` に "false" を渡します。

参照：[LAUNCH EXTERNAL PROCESS](#)

データ入力

DIALOG

DIALOG({table; |form; *})

引数	タイプ	説明
table	Table	→ フォームが所属するテーブルまたはプロジェクトフォームの場合省略
form	Form	→ ダイアログとして表示するフォーム
*	*	→ 同じプロセスを使用

DIALOG コマンドは最後の引数として * を受け入れるようになりました。この引数が渡されると、フォームはカレントプロセスで最後に開かれたウィンドウにロードされ表示されます。コマンドは、スクリーン上にフォームを表示したまま、実行を終了します。

このフォームはユーザアクションに対して通常に振る舞い、オブジェクトメソッドやフォームメソッドで CANCEL や ACCEPT コマンドが実行されれば閉じられます。カレントプロセスが終了すると、この方法で作成されたフォームは、CANCEL が実行されたのと同じ方法で自動的に閉じられます。

この方法は、わざわざ他のプロセスを起動する必要がなく、特にフローティングパレットの表示に便利です。

` ツールパレットを表示

```
$palette_window:=Open form window("tools";Palette form window)
DIALOG("tools;*") ` コントロールをすぐにメソッドに返す
```

```
`メインドキュメントウィンドウを表示
$document_window:=Open form window("doc";Standard form window)
DIALOG("doc")
```

セレクション

APPLY TO SELECTION

APPLY TO SELECTION(table; statement)
table 引数は必須となりました。

Displayed line number

このコマンドはリストボックスのコンテキストで動作するようになりました。
v11 におけるリストボックスの新機能については [ページ 311](#)、"[リストボックス](#)" の節を参照してください。

トランザクション

4D v11 はトランザクションのネストをサポートするようになりました ([ページ 29](#)、"[容量の拡張](#)" の節参照)。このメカニズムの最適化された管理を有効にするために、Transaction level コマンドが "トランザクション" テーマに追加されました。

トランザクション中に作成されたレコード

4D v11 でトランザクション中に作成されたレコードの処理方法が変わりました。今バージョンより、作成されたレコードに (18,000,000 から始まる) 一時的な番号は振られません。その代わりに、標準のレコード番号が割り当てられるようになります。この番号はトランザクションが受け入れられると正式にレコードの番号となり、トランザクションがキャンセルされ作成されたレコードが削除されると解放されます。
CANCEL TRANSACTION と VALIDATE TRANSACTION コマンドがカレントセレクションを変更することはなくなります。レコード番号を使用するコマンド (SCAN INDEX, RELATE ONE SELECTION など) は、制限なしにトランザクション中で利用可能です。

Transaction level

Transaction level → Longint

引数	タイプ	説明
		このメソッドは引数を必要としません
返り値	Longint	← 現在のトランザクションレベル (トランザクションが開始されていない場合は 0)

Transaction level コマンドはプロセスの現在のトランザクションレベルを返します。このコマンドは、それが 4D 言語または SQL どちらから開始されたか

にかかわらず、カレントプロセスのすべてのトランザクションを対象とします。

XML

新しいコマンド

DOM EXPORT TO PICTURE

DOM EXPORT TO PICTURE(elementRef; pictVar{; exportType})

引数	タイプ	説明
elementRef	String	→ ルート XML 要素参照
pictVar	Picture	→ XML ツリーを受け取るピクチャ変数 (SVG ピクチャ)
exportType	Longint	→ 0= データソースを格納しない 1= データソースをコピー 2 (デフォルト) = データソースを保有

DOM EXPORT TO PICTURE コマンドは、XML ツリーに含まれる SVG フォーマットのピクチャを、pictVar で指定するピクチャフィールドまたは変数に保存するために使用します。

SVG (Scalable Vector Graphics) は XML でベクター画像 (.svg 拡張子) を定義するために使用されるフォーマットです。これらのファイルは Web ブラウザでネイティブまたはプラグインを使用して表示できます。4D v11 は SVG の描画エンジンを含んでいて、ピクチャフィールドや変数で SVG ファイルを表示させることができます。SVG がもっとも一般的に使用されるのは、統計の表示や地図データです。このフォーマットに関する詳細は以下のアドレスを参照してください。

<http://www.w3.org/Graphics/SVG/>

elementRef に、SVG 画像を含む XML のルート要素参照を渡します。pictVar には SVG を受け取るピクチャフィールドまたは変数を渡します。ピクチャは XML で定義されているネイティブフォーマットのまま書き出され、SVG の描画エンジンにより描画されます。

オプションの exportType を使用して、コマンドが XML データソースを処理する方法を指定できます。"XML" テーマの以下の定数を指定できます。

- Get XML Data Source (0): 4D は XML データソースを読みますが、ソースはピクチャとともに保持されません。この場合コマンドの実行速度が向上しますが、DOM ツリーが保持されないため、ピクチャを格納したり書きだしたりすることはできません。

- Copy XML Data Source (1): 4D は DOM ツリーのコピーをピクチャとともに保持します。この場合、ピクチャをピクチャフィールドに保存したり、再度表示したり、いつでも書きだしたりすることが可能です。
- Own XML Data Source (2): 4D は DOM ツリーをピクチャとともに書き出します。ピクチャを格納したり、書きだしたりできます。しかし elementRef XML 参照を他の 4D コマンドで使用することはできなくなります。これは exportType 引数が省略された時のデフォルトの動作です。

注 SVG 描画エンジンを利用できるようにするために、" グラフ " テーマのコマンドも変更されました。詳細は [ページ 345](#)、" グラフ " の節を参照してください。

- ▼ 以下は "Hello World" を 4D ピクチャに表示する例です。

```
C_PICTURE(vpict)
$svg:=DOM Create XML ref("svg";"http://www.w3.org/2000/svg")
$ref:=DOM Create XML element ($svg;"text";"font-size";26;"fill";"red")
DOM SET XML ELEMENT VALUE($ref;"Hello World")
DOM EXPORT TO PICTURE($svg;vpict;"Copy XML Data Source")
DOM CLOSE XML($svg)
```



DOM Find XML element by ID

DOM Find XML element by ID (elementRef; id) → String

引数	タイプ	説明
elementRef	String	→ XML 要素参照
id	String	→ 検索する要素の ID 属性値
返り値	elementRef	← 検索された要素の参照 (検索されれば)

DOM Find XML element by ID コマンドは ID 属性値に基づき XML 要素を検索するために使用します。検索は elementRef 引数で指定された要素から行われます。

ID 属性値は要素ごとにユニークな識別子を与えるために使用します。ID は有効な XML 名で、ユニークでなければなりません。

id には検索する属性の値を渡します。

このコマンドは result に検索された要素の XML 参照を返します。

変更されたコマンド

DOM Find XML element

DOM Find XML element (elementRef; XPath; elementRefsArray) → String

引数	タイプ	説明
elementRef	String	→ XML 要素参照
xPath	Text BLOB	→ 検索する要素の XPath
elementRefsArray	Alpha 16 array	← 検索された要素参照のリスト
戻り値	elementRef	← 検索された要素参照

DOM Find XML element コマンドは、三番目の引数に文字配列タイプの **elementRefsArray** を受け入れます。この引数はオプションです。**elementRefsArray** 配列が渡されると、見つかった要素参照がこの配列に変えられます。この場合 **elementRefsArray** 配列の最初の要素がこのコマンドから返されます。この引数は、XPath 引数で指定した場所に、同じ名前を持つ要素が複数ある場合に便利です。

- ▼ 以下の XML があるとき：

```
<Root>
  <Elem1>
    <Elem2>aaa</Elem2>
    <Elem2>bbb</Elem2>
    <Elem2>ccc</Elem2>
  </Elem1>
</Root>
```

以下のコードを使用して、arrAfound 配列にすべての elem2 要素参照を取得できます。：

```
ARRAY STRING(16;arrAfound;0)
vFound:=DOM Find XML element(vElemRef;"/Root/Elem1/Elem2";arrAfound)
```

DOM Create XML element, DOM SET XML ATTRIBUTE

DOM Create XML element(elementRef; XPath; attrName; attrValue...; attrNameN; attrValueN) → String

DOM SET XML ATTRIBUTE(elementRef; attrName; attrValue...; attrNameN; attrValueN)

attrValue 引数にテキスト以外の値、ブール、整数、実数、日付、時刻等を渡せるようになりました。4D は以下の原則に従いこれらの値をテキストに変換します。

タイプ	変換例
ブール	" true" または " false"

整数	" 123456"
実数	" 12.34" (小数点は常に ".")
時間	" 5233" (秒数)
日付	" 2006-12-04T00:00:00Z" (RFC 3339)

DOM Parse XML source

DOM Parse XML source (document; validation; dtd | schema) → String

引数	タイプ	説明
document	String	→ ドキュメントパス名
validation	Boolean	→ TRUE = 検証を行う FALSE = 検証を行わない
dtd schema	String	→ DTD の場所、または XML スキーマの場所
返り値	String	← XML 要素参照

DOM Parse XML variable

DOM Parse XML variable (variable; validation; dtd | schema) → String

引数	タイプ	説明
variable	BLOB Text	→ 変数名
validation	Boolean	→ TRUE = 検証を行う FALSE = 検証を行わない
dtd schema	String	→ DTD の場所、または XML スキーマの場所
返り値	String	← XML 要素参照

The DOM Parse XML source と DOM Parse XML variable コマンドは、XML スキーマ、言い換えれば XSD (XML Schema Definition) による検証をサポートします。これを行うには、三番目の引数に ".xsd" 拡張子のファイルまたは URL を渡します。

XML スキーマによる検証は、DTD による検証より自由度があり、よりパワフルであるといわれています。XSD ドキュメントランゲージは XML に基づいています。特に XML スキーマはデータタイプをサポートしています。XML スキーマに関する詳細は以下の URL を参照してください:

<http://www.w3.org/XML/Schema>

validation 引数に TRUE を渡して三番目の引数を省略すると、これらのコマンドはストラクチャ自身の中で見つけることのできる DTD や XSD を使用して、XML ストラクチャを検証しようとします。検証は間接に行うこともできます。ストラクチャに DTD ファイルへの参照が含まれていて、さらにそこに XSD への参照が含まれている場合、コマンドは両方の検証を試みます。

SAX ADD XML ELEMENT VALUE

SAX ADD XML ELEMENT VALUE (document; data; *)

引数	タイプ	説明
document	DocRef	→ ドキュメント参照
data	Text ; Var	→ ドキュメントに挿入するテキストまたは変数
*	*	→ ・ 渡した場合、特別文字をエンコードする ・ 省略した場合エンコードしない

SAX ADD XML ELEMENT VALUE コマンドで、data 引数に含まれる特別文字 (<>"'...) の自動エンコードが可能になりました。これを行うには、オプションの * 引数を渡します。

APPLY XSLT TRANSFORMATION

APPLY XSLT TRANSFORMATION(xmlSource; xslSheet; result; compileSheet)

引数	タイプ	説明
xmlSource	String ; BLOB	→ XML ソースドキュメントの名前やパス、または XML ソースを含む BLOB
xslSheet	String ; BLOB	→ XSL スタイルシートドキュメントの名前やパス、または XSL スタイルシートを含む BLOB
result	String ; BLOB	→ XSLT 変換された結果を受け取るドキュメントの名前やパス、または XSLT 変換された結果を受け取る BLOB
compileSheet	Boolean	→ TRUE: XSLT を最適化 FALSE または省略: 最適化なしコンパイルされた XSL ファイルを取り除く

APPLY XSLT TRANSFORMATION は新しいオプションの引数 compileSheet を受け入れます。この引数は XSLT 変換を最適化するために使用されます。特に同じ XSL シートによる連続した変換に関連します。compileSheet 引数に TRUE が渡された場合、xslSheet XSL ファイルはコマンドの最初の呼び出しで解析、コンパイルされ、メモリに格納されます。同じ XSL シートを使用して引き続き行われる呼び出しでは、コマンドは直接コンパイル済みのファイルを使用します (ファイルが更新されない限り)。これにより処理が高速化されます。

最適化は、(xsl:import により) インポートされたファイルの更新は考慮しません。XSL ファイルから参照されるファイルが更新された場合は、新しい XSL ファイルの再コンパイルを強制しなければなりません。これを

行うには compileSheet 引数に FALSE を渡すあるいは省略してこのコマンドを再度呼び出します。

Web サービス (クライアント)

CALL WEB SERVICE コマンドは v11 で最適化されました。

CALL WEB SERVICE

CALL WEB SERVICE (accessURL; soapAction; methodName; namespace; complexType|; *)

引数	タイプ	説明
accessURL	String	→ Web サービスへのアクセス URL
soapAction	String	→ SOAPAction フィールドの内容
methodName	String	→ メソッド名
namespace	String	→ 名前空間
complexType	Longint	→ 複合タイプの設定 (省略した場合単純型)
*		→ * が渡された場合、接続を閉じない

CALL WEB SERVICE はオプションの引数 (*) を受け入れるようになりました。

この引数が渡されると、コマンドの実行が終了したときも、プロセスで使用されている接続は閉じられません。この場合、次の CALL WEB SERVICE 呼び出しは、* が渡されていれば、同じ接続を再利用します。接続を閉じるには CALL WEB SERVICE コマンドを * なしで呼び出します。このメカニズムにより、同じサーバに対する連続した Web サービスの呼び出し速度が向上します。特にインターネット経由の WAN 接続では顕著です。

この機能は Web サーバの "keep-alive" 設定の影響を受けます。この設定では、同じ接続内での最大リクエスト数を指定したり、リクエストを拒否するよう設定したりします。CALL WEB SERVICE リクエストが最大数を超えたり、あるいは keep-alive 接続が禁止されている場合、4D は接続ごとに新しいリクエストを作成します。

その他の変更

この節では、4D v11 で行われた、コマンドやコマンドテーマに対するさまざまな変更点を説明します。

サブレコード

サブテーブルは 4D v11 ではサポートされていません。このバージョンではサブテーブルを作成することはできません。変換されたデータベースで、サブテーブルは引き続き動作します。しかしサブテーブルは特別なリレーションが張られた通常のテーブルとして表示されます。詳細は [ページ 40](#)、"[サブテーブルの変換](#)" の節を参照してください。

このコンテキストにおいて、"サブレコード" テーマのコマンドは引き続き動作します。

特別なりレーションが削除されると、サブテーブルは通常のテーブルに変換されます。そのあと、再びサブテーブルにすることはできず、"サブレコード" テーマのコマンドをこのテーブルに対して適用することはできなくなります。

CLEAR VARIABLE

CLEAR VARIABLE コマンドはインタプリタモードとコンパイルモードで動作が同じになり、変数の値をデフォルトのタイプに設定します。

以前のバージョンで、このコマンドはインタプリタモードの場合、変数をメモリから消去していました。

テーマ: 変数

ドキュメントシステム変数

Document システム変数は常に、ディスクや一連のフォルダからなる、ドキュメントの完全なパス名を含みます。

以前のバージョンでは Document 変数はドキュメント名のみを含むケースがありました。

たとえば以下の例文の場合、

```
docRef:=Create document("mydoc.txt")
```

- 以前のバージョンでは:

```
Document = "mydoc.txt"
```

- 4D v11 からは:

```
Document = "c:\MyDocuments\mydoc.txt" または
```

```
Document = "MacHD:MyDocuments:mydoc.txt"
```

表示フォーマット定数

4D v11 でシステム表示フォーマットのサポートが強化されました ([ページ 171](#)、"[日付と時間の表示フォーマット](#)" の節と [ページ 335](#)、[コマンド GET SYSTEM FORMAT](#) を参照)。

このフレームワークにおいて、"Date Display Formats" と "Time Display Formats" の定数に変更され、また新しい定数が追加されました。

日付表示フォーマット

明確にするために、日付表示フォーマットの定数の名称が変更されました。"System" と "Internal" 接頭辞は、フォーマットがシステムフォーマットに基づくものか、アプリケーション内部の設定の基づくものかを表します。:

4D v11 定数	値	以前の名称
System date short	1	Short
System date abbreviated	2	Abbreviated
System date long	3	Long
Internal date short special	4	MM DD YYYY

4D v11 定数	値	以前の名称
Internal date long	5	Month Day Year
Internal date abbreviated	6	Abbr Month Day
Internal date short	7	MM DD YYYY Forced
ISO Date	8	ISO Date Time
Blank if null date	100	-

これらのフォーマットについての詳細は、[ページ 173](#)、"[日付フォーマット](#)"の節を参照してください。

時間表示フォーマット

新しい時間表示フォーマット定数が追加され、いくつかの名称が変更されました。

4D v11 定数	値	以前の名称
HH MM SS	1	同じ
HH MM	2	同じ
Hour Min Sec	3	同じ
Hour Min	4	同じ
HH MM AM PM	5	同じ
MM SS	6	-
Min Sec	7	-
ISO Time	8	-
System time short	9	-
System time long abbreviated	10	-
System time long	11	-
Blank if null time	100	-

これらのフォーマットについての詳細は、[ページ 174](#)、"[時間フォーマット](#)"の節を参照してください。

SET FORMAT

4D v11 の SET FORMAT でふたつの変更点があります。

- SET FORMAT で、動的にバーバースョップモードのサーモメータを使用できます。これを行うには、インジケータ (サーモメータとルーラ) の flags 引数に値 128 を渡します。他の引数 (min, max 等) は無視されます。この値は他のフラグと一緒に利用できません。バーバースョップタイプのサーモメータに関する詳細は、[ページ 171](#)、"[バーバースョップサーモメータ](#)"の節を参照してください。

- 3D ボタンに関連付けるピクチャを参照するために、Resources フォルダに置かれたピクチャを参照する URL が使用できます。これがどのように動作するかは [ページ 43](#)、"[ピクチャの自動参照](#)" の節で説明しています。
ピクチャがデータベースの Resources フォルダに置かれていれば、picture や background 引数に URL を渡すことができます ("#folder/|mypicture" や "file:|folder/|mypicture")。
- ▼ サーモメータをバーバーストップモードにする
SET FORMAT(\$mythermo;";;;128")
\$mythermo:=1 `アニメーション開始
- ▼ 3D ボタンのピクチャを変更する。ピクチャは "Resources フォルダのメJA モサブフォルダにある。
SET FORMAT(\$mybutton;"Open;#JA/open.jpg")
テーマ: オブジェクトプロパティ

GOTO AREA

GOTO AREA({*; }object)

このコマンドを、カレントフォームのフォーカスを外すために使用できます。これを行うには、空のオブジェクト名を渡します。

GOTO AREA(*,"") `フォーカスを持つオブジェクトはなくなります
テーマ: 入力制御

IMPORT TEXT, EXPORT TEXT

IMPORT TEXT({aTable; }document)

EXPORT TEXT({aTable; }document)

Unicode モードにおいて、これらの k 万度はデフォルトで UTF-8 文字セットを使用します。文字セットを変更するには [USE CHARACTER SET](#) コマンドを使用してください。

注 4D が非 Unicode モード (ASCII 互換モード) で実行されている場合、これらのコマンドは以前のバージョンと同様 ASCII テーブルを使用します。

テーマ: 読み込みと書き出し

PROCESS PROPERTIES

origin 引数に返されるプロセスタイプに、新しい値が追加されました。これらの値は以下の定数に対応します ("Process Type" テーマ):

定数	タイプ	値
Web server Process	Longint	-13
Execute on Client Process	Longint	-14
4D Server Process	Longint	-15
On Quit Process	Longint	-16

Method editor macro Process	Longint	-17
Internal 4D Server Process	Longint	-18
Backup Process	Longint	-19
Log File Process	Longint	-20
Restore Process	Longint	-21
MSC Process	Longint	-22
Timer Process	Longint	-23
SQL Method Execution Process	Longint	-24
Server Controller Process	Longint	-25
Monitor Process	Longint	-26

4D v11 で使用される用語に統一するため、既存の定数の名称が変更されました。

新しい名称	以前の名称	値
Created from execution dialog	Created from User Mode	3

テーマ : Process Type

二次元配列

- 変数を受け入れる引数に二次元配列を渡すと、4D v11 はエラーを生成しません。

以前のバージョンでは、エラーは生成されませんでした。二番目の次元は無視されていました。

動作は以下の通りでした：

```
myType := Type(my2DArray{1}{1})
```

4D はこの式を以下のように解釈しました：

```
myType := Type(my2DArray{1})
```

- Type コマンドを添え字付きの二次元配列に適用すると、"Array 2D"ではなく、実際の型が返されるようになりました。

以下のコードにおいて：

```
ARRAY TEXT($my2DArray;50;20)
```

```
$thetype:=Type($my2DArray{10})
```

4D v11 では、\$thetype に 18 (Text array) が返されます。以前のバージョンでは \$thetype に 13 (Array 2D) が返されました。

Tip: 汎用コードを記述する場合などでは、配列が通常のものであるか、添え字付きの二次元配列であるかを知る必要があるでしょう。この場合は以下のコードを使用できます。

```
ptrmyArray:=>myArray{6} `myArr{6} が二次元配列か知りたい
```

```
RESOLVE POINTER(ptrmyArray,varName;tableNum;fieldNum)
```

```
If(varName#"")
```

```
  $ptr:=Get pointer(varName)
```

```
$thetype:=Type($ptr->)
`myArr{6} が二次元配列なら、$thetype は 13 になる
```

End if

- 以前のバージョンでは、インタプリタモードで、二次元文字配列の文字長を変更できました。たとえば以下の文が受け入れられていました。

```
ARRAY STRING(80;arrSmyarray;10)
```

...

```
ARRAY STRING(255;arrSmyarray|2|;10)
```

4D v11 では、このコードはインタプリタモードでも許可されず、シンタックスエラーを生成します。

ARRAY TO SELECTION

以前のバージョンでは、異なるサイズの配列をこのコマンドに渡すことができました (一番目の配列のサイズがコピーする要素の数を決定していました)。

4D v11 ではプログラムの制御が強化されたため、これらの配列はすべて同じサイズでなければなりません。そうでなければエラー "配列のサイズが一致しません。" が生成されます。

テーマ: 配列

SELECT LOG FILE

```
SELECT LOG FILE(logFile | *)
```

このコマンドの動作が変更され、既存のログファイルを開くことはできなくなりました。このコマンドは新しいログファイルを作成やカレントのログファイルを閉じることをのみをサポートします。

さらに、ログファイルを作成する場合、このコマンド実行後新しいログファイルはすぐには作成されず、次のバックアップ時に作成されることに注意が必要です (パラメータはデータファイルに保持され、データベースが閉じられたとしても有効です)。BACKUP コマンドを呼び出して、ログファイルを作成させることができます。

これはログファイルを閉じる際には適用されません。ログファイルはすぐに閉じられます。

テーマ: バックアップ

INPUT FORM, OUTPUT FORM

form 引数に渡されたフォームが無効である場合、これらのコマンドはエラー 81: "フォームが見つかりません。" エラーを返すようになりました。

テーマ: フォーム

名称の変更

コマンド名と機能をより明確に対応させるため、"配列" テーマの二つのコマンドおよび "階層リスト" テーマの二つのコマンドの名称が変更されました。GET PLUGIN LIST コマンドも名称とテーマが変更されました。

以前の名称 (4D 2004.x)	新しい名称 (4D v11.x)
INSERT ELEMENT	INSERT IN ARRAY
DELETE ELEMENT	DELETE FROM ARRAY

以前の名称 (4D 2004.x)	新しい名称 (4D v11.x)
INSERT LIST ITEM DELETE LIST ITEM GET PLUGIN LIST	INSERT IN LIST DELETE FROM LIST PLUGIN LIST

テーマの変更

より明確にするために、以下の既存のコマンドのテーマが変更されました (機能に影響はありません)。

コマンド	以前のテーマ	4D v11 での新しいテーマ
SHOW TOOL BAR HIDE TOOL BAR	ツールバー (テーマ削除)	ユーザインターフェース
SHOW MENU BAR HIDE MENU BAR SET ABOUT	メニュー	ユーザインターフェース
SET DATABASE PARAMETER, Get database parameter	ストラクチャアクセス	4D 環境
BUILD APPLICATION	ツール	4D 環境
LAUNCH EXTERNAL PROCESS SET ENVIRONMENT VARIABLE	システム環境	ツール
Is license available	4D 環境	ユーザ & グループ
(GET) PLUGIN LIST	ユーザ & グループ	4D 環境
EXECUTE FORMULA	ランゲージ	フォーミュラ

廃止されたコマンド

以下の廃止されたコマンドは、もはやサポートされず 4D から取り除かれました。

- SEARCH BY INDEX

- SAVE OLD RELATED ONE

SAVE RELATED ONE コマンドを代わりに使用しなければなりません。

- SORT BY INDEX

ORDER BY コマンドを代わりに使用しなければなりません。

以下のコマンドが 4D v11 で廃止されましたが、互換性のため引き続き保持されています。これらのコマンドは "廃止されたコマンド" テーマに移動されました。

- Get component resource ID ("リソース" テーマ): このコマンドは、4D v11 と互換性のない以前の世代のコンポーネントで使用されていました (ページ45、"新しいコンポーネントアーキテクチャ" の節を参照)。このコマンドは動作しません。

A

ランゲージコード

4D v11 がサポートするランゲージコードを以下に示します。詳細は [ページ 98](#)、"[XLIFF 標準のサポート](#)" の節を参照してください。

Languages	ISO639-1	"Legacy"	ISO3166
AFRIKAANS	af	afrikaans	
ALBANIAN	sq	albanian	
ARABIC_SAUDI_ARABIA	ar	arabic	sa
ARABIC_IRAQ	ar	arabic	iq
ARABIC_EGYPT	ar	arabic	eg
ARABIC_LIBYA	ar	arabic	ly
ARABIC_ALGERIA	ar	arabic	dz
ARABIC_MOROCCO	ar	arabic	ma
ARABIC_TUNISIA	ar	arabic	tn
ARABIC_OMAN	ar	arabic	om
ARABIC_YEMEN	ar	arabic	ye
ARABIC_SYRIA	ar	arabic	sy
ARABIC_JORDAN	ar	arabic	jo
ARABIC_LEBANON	ar	arabic	lb
ARABIC_KUWAIT	ar	arabic	kw
ARABIC_UAE	ar	arabic	ae
ARABIC_BAHRAIN	ar	arabic	bh
ARABIC_QATAR	ar	arabic	qa
BASQUE	eu	basque	
BELARUSIAN	be	belarusian	
BULGARIAN	bg	bulgarian	
CATALAN	ca	catalan	
CHINESE_TRADITIONAL	zh	chinese	cht
CHINESE_SIMPLIFIED	zh	chinese	chs
CHINESE_HONGKONG	zh	chinese	hk

CHINESE_SINGAPORE	zh	chinese	sg
CROATIAN	hr	croatian	
CZECH	cs	czech	
DANISH	da	danish	
DUTCH	nl	dutch	
DUTCH_BELGIAN	nl	dutch	be
ENGLISH_US	en	english	us
ENGLISH_UK	en	english	gb
ENGLISH_AUSTRALIA	en	english	au
ENGLISH_CANADA	en	english	ca
ENGLISH_NEWZEALAND	en	english	nz
ENGLISH_EIRE	en	english	ie
ENGLISH_SOUTH_AFRICA	en	english	za
ENGLISH_JAMAICA	en	english	jm
ENGLISH_CARIBBEAN	en	english	cb
ENGLISH_BELIZE	en	english	bz
ENGLISH_TRINIDAD	en	english	tt
ESTONIAN	et	estonian	
FAEROESE	fo	faorese	
FARSI	fa	persian	
FINNISH	fi	finnish	
FRENCH	fr	french	
FRENCH_BELGIAN	fr	french	be
FRENCH_CANADIAN	fr	french	ca
FRENCH_SWISS	fr	french	ch
FRENCH_LUXEMBOURG	fr	french	lu
GERMAN	de	german	
GERMAN_SWISS	de	german	ch
GERMAN_AUSTRIAN	de	german	at
GERMAN_LUXEMBOURG	de	german	lu
GERMAN_LIECHTENSTEIN	de	german	li
GREEK	el	greek	
HEBREW	he	hebrew	
HUNGARIAN	hu	hungarian	
ICELANDIC	is	iceland	
INDONESIAN	id	indonesian	
ITALIAN	it	italian	
ITALIAN_SWISS	it	italian	ch

JAPANESE	ja	japanese	
KOREAN_WANSUNG	ko	korean	
KOREAN_JOHAB	ko	korean	
LATVIAN	lv	latvian	
LITHUANIAN	lt	lithuanian	
NORWEGIAN	no	norwegian	
NORWEGIAN_NYNORSK	nn	nynorsk	no
POLISH	pl	polish	
PORTUGUESE	pt	portuguese	
PORTUGUESE_BRAZILIAN	pt	portuguese	br
ROMANIAN	ro	romanian	
RUSSIAN	ru	russian	
SERBIAN_LATIN	sr	serbian	latn
SERBIAN_CYRILLIC	sr	serbian	cyril
SLOVAK	sk	slovak	
SLOVENIAN	sl	slovenian	
SPANISH_CASTILLAN	es	spanish	
SPANISH_MEXICAN	es	spanish	mx
SPANISH_MODERN	es	spanish	
SPANISH_GUATEMALA	es	spanish	gt
SPANISH_COSTA_RICA	es	spanish	cr
SPANISH_PANAMA	es	spanish	pa
SPANISH_DOMINICAN_REPUBLIC	es	spanish	do
SPANISH_VENEZUELA	es	spanish	ve
SPANISH_COLOMBIA	es	spanish	co
SPANISH_PERU	es	spanish	pe
SPANISH_ARGENTINA	es	spanish	ar
SPANISH_ECUADOR	es	spanish	ec
SPANISH_CHILE	es	spanish	cl
SPANISH_URUGUAY	es	spanish	uy
SPANISH_PARAGUAY	es	spanish	py
SPANISH_BOLIVIA	es	spanish	bo
SPANISH_EL_SALVADOR	es	spanish	sv
SPANISH_HONDURAS	es	spanish	hn
SPANISH_NICARAGUA	es	spanish	ni
SPANISH_PUERTO_RICO	es	spanish	pr
SWEDISH	sv	swedish	

SWEDISH_FINLAND	sv	swedish	fi
THAI	th	thai	
TURKISH	tr	turkish	
UKRAINIAN	uk	ukrainian	
VIETNAMESE	vi	vietnamese	

B

SQL データタイプとの対応表

ここで示す表は、4DのSQLエンジンデータタイプ、4Dの標準データタイプ、および他のSQLデータベースのデータタイプの対応を示します。

- [4D SQL と ORACLE](#)
- [4D SQL と MySQL](#)
- [4D SQL と PostgreSQL](#)
- [4D SQL と Access](#)
- [4D SQL と MS SQL Server](#)
- [4D SQL と Sybase](#)
- [4D SQL と IBM DB2](#)

4D SQL と ORACLE

4D SQL	4D v11 ランゲージ	ORACLE
VARCHAR(n)	Text	VARCHAR2(n [BYTE CHAR])
	-	CLOB
	-	CHAR
	-	NCHAR
	-	NVARCHAR2
REAL	Real (Number)	NUMBER(p,s)
NUMERIC		
FLOAT	Real (Number)	FLOAT[(p)]
SMALLINT	Integer	NUMBER(p,0)
INT	Longint	
BIT	Boolean	-
BOOLEAN		-
BLOB	BLOB	BLOB
	-	RAW(n)
	-	LONG RAW
BIT VARYING	BLOB	BLOB
CLOB	Text	CLOB
	-	NCLOB
TEXT	Text	VARCHAR2
		CLOB
TIMESTAMP	Date and Time processed separately (automatic conversion)	TIMESTAMP
	-	DATE
	-	INTERVAL YEAR TO MONTH
DURATION	Time	INTERVAL DAY TO SECOND
INTERVAL		
PICTURE	Picture	BLOB
-	-	BFILE
-	-	ROWID
-	-	UROWID[(n)]

4D SQL と MySQL

4D SQL	4D v11	MySQL
VARCHAR(n)	Text	LONGTEXT
REAL	Real (Number)	REAL[(p,s)] DOUBLE[(p,s)] DOUBLE PRECISION[(p,s)]
NUMERIC	Real (Number)	NUMERIC[(p[,s])] <ul style="list-style-type: none"> DECIMAL[(p[,s])] DEC[(p[,s])]
	-	BIGINT [UNSIGNED]
FLOAT	Real (Number)	FLOAT[(p[,s])]
	-	DECIMAL[(p[,s])] [UNSIGNED]
SMALLINT	Integer	SMALLINT
	-	TINYINT [UNSIGNED] (BYTE ??)
INT	Longint	INT INTEGER
	-	MEDIUMINT [UNSIGNED]
BIT	Boolean	BOOL
BOOLEAN		TINYINT BIT
BLOB	BLOB	BLOB
BIT VARYING		
BLOB	-	TINYBLOB MEDIUMBLOB LONGBLOB
CLOB	Text	LONGTEXT
TEXT		
CLOB	-	TINYTEXT MEDIUMTEXT LONGTEXT
TIMESTAMP	Date and Time processed separately (automatic conversion)	TIMESTAMP[(p)]
		DATE
DURATION	Time	TIME
INTERVAL		
PICTURE	Picture	BLOB
-	-	[NATIONAL]CHAR[(n)] [BINARY]
-	-	YEAR[(2;4)]
-	-	ENUM('value1','value2',...)

4D SQL	4D v11	MySQL
-	-	SET('value1','value2',...)

4D SQL と PostgreSQL

4D SQL	4D v11	PostgreSQL
VARCHAR(n)	Text	varchar(n) character varying(n)
	-	character(n) char(n)
REAL	Real (Number)	real float4
NUMERIC	Real (Number)	numeric(p,s) decimal(p,s)
	-	bigint int8
FLOAT	Real (Number)	Float8
	-	double precision float8 float
SMALLINT	Integer	smallint Int2
INT	Longint	int integer int4
BIT	Boolean	boolean bool
BOOLEAN	Boolean	boolean bool
BLOB	BLOB	bytea
	-	bit(n)
	-	bit varying (n)
	-	varbit(n)
BIT VARYING	BLOB	bytea
CLOB	Text	text
TEXT	Text	text
TIMESTAMP	Date and Time processed separately (automatic conversion)	timestamp [(p)] [without time zone]
	-	timestamp [(p)] with timezone
	-	date
	-	time [(p)] [without time zone]
DURATION	Time	time
	-	time [(p)] with timezone

4D SQL	4D v11	PostgreSQL
INTERVAL	Time	interval(p)
PICTURE	Picture	bytea
-	-	money
-	-	serial
-	-	bigserial
-	-	box
-	-	line
-	-	lseg
-	-	circle
-	-	path
-	-	point
-	-	polygon
-	-	cidr
-	-	inet
-	-	macadr
-	-	oid
-	-	xid

4D SQL と Access

4D SQL	4D v11	Access
VARCHAR(n)	Text	Text
REAL	Real (Number)	Number (Double)
NUMERIC		
FLOAT		
SMALLINT	Integer	Number (Integer)
INT	Longint	Number (Long Integer)
BIT	Boolean	Number (byte)
BOOLEAN		
BLOB	BLOB	-
BIT VARYING		
CLOB	Text	Memo
TEXT		
TIMESTAMP	Date and Time processed separately (automatic conversion)	Time
DURATION	Time	Time
INTERVAL		
PICTURE	Picture	-
-	-	Currency
-	-	AutoNumber
-	-	OLE Object
-	-	Hyperlink
-	-	Lookup Wizard

4D SQL と MS SQL Server

4D SQL	4D v11	MS SQL Server
VARCHAR(n)	Text	nvarchar[(n)]
	-	char[(n)]
	-	nchar[(n)]
	-	ntext
REAL	Real (Number)	real
NUMERIC	Real (Number)	numeric[(p[,0])]
	-	bigint
FLOAT	Real (Number)	float[(n)]
	-	decimal[(p[,s])]

4D SQL	4D v11	MS SQL Server
SMALLINT	Integer	smallint
	-	tinyint
INT	Longint	int
BIT	Boolean	bit
BOOLEAN		
BLOB	BLOB	Varbinary
	-	binary[(n)]
	-	varbinary[(n)]
BIT VARYING	BLOB	Varbinary
CLOB	Text	ntext
TEXT		
TIMESTAMP	Date and Time processed separately (automatic conversion)	datetime
	-	smalldatetime
DURATION	Time	datetime
INTERVAL		
PICTURE	Picture	image
		money
		smallmoney
		uniqueidentifier

4D SQL と Sybase

4D SQL	4D v11	Sybase
VARCHAR(n)	Text	varchar[(n)]
	-	char[(n)]
	-	nchar[(n)]
	-	nvarchar[(n)]
REAL	Real (Number)	real
NUMERIC	Real (Number)	numeric[(p[,0])]
FLOAT	Real (Number)	float[(precision)]
	-	decimal[(p[,s])]
	-	double precision
SMALLINT	Integer	smallint
	-	tinyint
INT	Longint	int
BIT BOOLEAN	Boolean	bit
BLOB	BLOB	varbinary
	-	binary[(n)]
	-	varbinary[(n)]
BIT VARYING	BLOB	varbinary
CLOB TEXT	Text	text
TIMESTAMP	Date and Time processed separately (automatic conversion)	datetime
	-	smalldatetime
DURATION INTERVAL	Time	datetime
PICTURE	Picture	image
-	-	money
-	-	smallmoney

4D SQL と IBM DB2

4D SQL	4D v11	IBM DB2
VARCHAR(n)	Text	VARCHAR(n) [FOR BIT DATA] CHARACTER VARYING(n) [FOR BIT DATA] CHAR VARYING(n) [FOR BIT DATA]
	-	CHAR[(n)] [FOR BIT DATA] CHARACTER[(n)] [FOR BIT DATA]
REAL	Real (Number)	REAL
NUMERIC	Real (Number)	NUMERIC[(p[,0])] NUM[(p[,s])] DECIMAL[(p[,s])] DEC[(p[,s])]
	-	BIGINT
FLOAT	Real (Number)	FLOAT[(n)]
		DOUBLE [PRECISION]
SMALLINT	Integer	SMALLINT
INT	Longint	INTEGER INT
BIT BOOLEAN	Boolean	-
BLOB BIT VARYING	BLOB	BLOB(n[K;M;G])
CLOB	Text	CLOB(n[K;M;G])
	-	DBCLOB(n[K;M;G])
TEXT	Text	CLOB(n[K;M;G])
	-	LONG VARCHAR [FOR BIT DATA]
TIMESTAMP	Date and Time processed separately (automatic conversion)	TIMESTAMP
		DATE
DURATION INTERVAL	Time	TIME
PICTURE	Picture	BLOB(n[K;M;G])
		GRAPHIC[(n)]
		VARGRAPHIC(n)
		LONG VARGRAPHIC
		DATALINK

C

4D Pack v11

バージョン 11 の 4D Pack は大きく変更されました。新しいコマンドが追加され、いくつかのコマンドの使用は推奨されず、いくつかは削除されました。

この章ではこれらの変更について説明し、削除されたコマンドについては置き換えソリューションを提示します。

注 この章で説明されない 4D Pack のコマンドは 4D Pack v11 でもサポートされています。機能は変更されていません。

新しいコマンド

AP Get file MD5 digest

AP Get file MD5 digest(filePath; digest(; fork)) → Longint

引数	タイプ	説明
filePath	Text	→ ファイルの完全パス名
digest	Text	← ファイルの MD5 Digest
fork	Longint	→ 0= データフォーク、1= リソースフォーク
返り値	Longint	← エラーコード

新しい AP Get file MD5 digest コマンドは、指定されたドキュメントの MD5 digest キーを返します。MD5 (Message Digest 5) はデータを暗号化するために使用されるハッシュ関数です。

Pass the name of the document containing the key in the filePath 引数にキーが含まれたドキュメントの名前を渡します。

fork 引数は Mac OS でのみ有効で、ファイルのパートを指定します。

- 0 = データフォーク
- 1 = リソースフォーク

▼ 使用例 :

```
C_TEXT($thedoc)
C_TEXT(<>digest)
C_LONGINT($resfork)
```

```
$resfork:=0
$thedoc:=Select document
$error:=AP Get file MD5 digest($thedoc; <>digest; , resfork)
```

テーマ : Utilities.

AP Does method exist

AP Does method exist (methodName) → Integer

引数	タイプ	説明
methodName	Alpha	→ テストするメソッドの名称
返り値	Integer	← 0=メソッドは存在しない 1=メソッドは既に存在する

新しい **AP Does method exist** コマンドを使用して、データベースに methodName プロジェクトメソッドが存在するか調べることができます。このコマンドは、インストールされているコンポーネントのプロジェクトメソッドを調べることはできません。

AP Create method コマンドを使用する際、このコマンドを使用して同じ名称のメソッドが存在するかを調べることで、エラーを回避することができます。

テーマ : Utilities

AP Create method

AP Create method(methodName;propertiesArray;methodCode{;folderName}) → Longint

propertiesArray は 6 つの要素を受け入れます。

- コンポーネントとホストデータベースでメソッドを共有する場合、propertiesArray{5} に 1 を代入します。そうでなければ 0 を代入します。
- メソッドを SQL から実行可能にする場合、propertiesArray{6} に 1 を代入します。そうでなければ 0 を代入します。

テーマ : Utilities

廃止されるコマンド

以下の 4D Pack v11 コマンドはいぜん使用することができますが、対応する技術や機能は使われなくなります。これらのコマンドは将来のバージョンで削除されるため、置き換えることを強くお勧めします。

AP Save BMP 8 bits, AP Get picture type

これらのコマンドは 4D の "ピクチャメ" コマンドで置き換えられます。

AP GET PARAM AP SET PARAM

これらのコマンドでアクセス可能なパラメタは、ほとんどが廃止されたか、**SET DATABASE PARAMETER, Get database parameter** コマンドでアクセスできます。

AP AVAILABLE MEMORY

このコマンドは byte ではなく KB 単位で値を返すことに注意してください。

AP ShellExecute

このコマンドは LAUNCH EXTERNAL PROCESS コマンドで置き換えることができます。
このコマンドは Mac OS 上では削除されました。Windows でのみまだ実行できます。

AP FCLOSE, AP fopen, AP FPRINT, AP fread

"ANSI Streams" テーマのコマンドは SET CHANNEL, SEND PACKET, RECEIVE PACKET などのコマンドで置き換えられます。

AP HELP ON KEY, AP HELP INDEX, AP HELP ON HELP, AP CLOSE HELP

"Windows Help Files" テーマのコマンドは Windows 2000 と Windows XP で動作しますが、Windows Vista と互換性がありません。

削除されたコマンド

以下のコマンドはプラグインから削除され、インタプリタはこれらのコマンドを認識しません。アプリケーションをコンパイルするには、これらのコマンドは置き換えなければなりません。

以下の表で、削除されたコマンドと、4D が推奨する置き換えソリューションを説明します。

削除されたコマンド	置き換えソリューション
AP PICT DRAGGER	4D に統合されたドラッグ&ドロップ機能を使用 (クリックタイプイベントの場合、MouseDown, MouseX and MouseY システム変数を使用)

AP PICT UPDATER and %AP Pict displayer	ピクチャフィールドを使用
AP Read Picture BLOB	BLOB TO PICTURE
AP Read Picture File	READ PICTURE FILE
AP Save GIF	CONVERT PICTURE + WRITE PICTURE FILE
AP SET PICT MODE	SET FORMAT
AP Select document	Select document
AP Set palette	Select RGB Color
AP ShellExecute	LAUNCH EXTERNAL PROCESS
AP Sublaunch	LAUNCH EXTERNAL PROCESS
AP PrintDefault	AP BLOB to print settings + AP Print settings to BLOB
AP Text to PrintRec	AP BLOB to print settings + AP Print settings to BLOB
AP PrintRec to text	AP BLOB to print settings + AP Print settings to BLOB
AP PrValidate	AP BLOB to print settings + AP Print settings to BLOB
AP Toolbar installed	Tool bar height (非表示の場合 0)
AP SET WEB FILTERS	SET DATABASE PARAMETER, Get database parameter
AP Add table and fields	4D SQL エンジンを使用
AP Create relation	現在置き換えソリューションはありません。