4D 12.1 追加・修正情報

- 新しい日付フォーマット (Stringコマンド)
- リッチテキストエリア
- 外部PHPインタプリターの使用
- データベース設定
- SQLを使用した書き出し
- 4D HTMLタグの変更
- 64 bitバージョンの4D Server
- サブテーブルからの移行を補助するコマンド
- 4D Internet CommandsでのSSLサポート

新しい日付フォーマット (Stringコマンド)

4D v12.1では新しい日付/時間フォーマットをサポートすることで、タイムスタンプの生成がより容易になりました。これらのフォーマットは特にXMLやWebの処理で使用されます。

これらのフォーマットは既存の**String**コマンドで利用できます。日付と時間を処理するために、このコマンドはオプションの第三引数*combTime*を受け入れるようになりました。この時間タイプの引数は*expression*引数が日付の場合にのみ使用されます (後述)。

時間付きのISO Dateフォーマット

ISO8601フォーマットの日付を返すISO Date日付フォーマットで、第三引数を使用することにより、日付と時間を合成した結果を取得できるようになりました:

```
$mydate:=String(Current date;ISO Date) // 戻り値例: 2010-09-13T00:00:00
$mydate:=String(Current date;ISO Date;Current time) // 戻り値例: 2010-09-13T18:11:53
```

時間付きの新しいISO Date GMTフォーマット

新しいISO Date GMT日付フォーマットはISO Dateフォーマットと同じですが、時間帯を考慮に入れる点が異なります (ISO Dateフォーマットはローカルの日付と時刻を表します)。 そのためローカルの時間帯により日付が前や後ろにずれることがあります。

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

最後の文字"Z"はUTCで表記していることを表します。

日付のみを渡すと、コマンドはその日のローカル時間00:00:00をUTCで表現した値を返します:

```
$mydate:=String(Current date;ISO Date GMT) // 日本では2010-09-12T15:00:00Zとなる
```

新しいDate RFC 1123フォーマット

RFC 822およびRFC 1123に定義された標準に基づく日付と時刻フォーマットを得るために、新しいDate RFC 1123日付フォーマットを使用できます。このフォーマットはたとえばHTTPヘッダーにCookieの有効期限を設定するために使用されます。

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

表現される時間は時間帯が考慮されます (GMT)。日付のみを渡すと、コマンドはローカルタイムの00:00:00をGMTに変換して返します。

返される値は時間帯により日付が前後に移動することがあります:

```
$mydate:=String(Current date;Date RFC 1123) // 日本では例えばThu, 09 Sep 2010 15:00:00 GMTが返される
```

リッチテキストエリア

4D v12.1では複数の新しい機能が追加され、開発者はリッチテキストエリア (マルチスタイルエリア) をより制御できるようになりました。**OBJECT SET PLAIN TEXT**コマンドが追加され、いくつかの既存コマンドの機能が変更されました。
また本テーマのいくつかのコマンドはOK変数を設定するようになりました。

新しいOBJECT SET PLAIN TEXTコマンド

新しい**OBJECT SET PLAIN TEXT**コマンドは例えば"<"や">"、"&"などの文字を含むプレーンテキストを挿入するために使用します。このコマンドは**OBJECT SET STYLED TEXT**や**OBJECT Get plain text**コマンドを補完します。

変更されたコマンド

以下のコマンドの動作が4D v12.1で変更されました:

- **FONT LIST**: Mac OSでこのコマンドは**フォントファミリー**に基づく*font*配列を生成するようになりました。以前のバージョンではMac OSにおいてフォント名の配列が返されました。
- **OBJECT SET STYLED TEXT ATTRIBUTES**: このコマンドが*attribName*引数にAttribute font name定数を渡されて使用された場合、*attribValue*にはフォントファミリーを渡さなければなりません。例えば:

```
OBJECT SET STYLED TEXT ATTRIBUTES(*;"MyText";0;MAXLONG;Attribut font name;"ARIAL") // 有効
OBJECT SET STYLED TEXT ATTRIBUTES(*;"MyText";0;MAXLONG;Attribut font name;"TIMES BOLD ITALIC") // 無効 (コマンドは何も行いません)
```

  結果Mac OSにおいて、**FONT LIST**コマンドから返された値を**OBJECT SET STYLED TEXT ATTRIBUTES**コマンドで使用したい場合、**FONT LIST**コマンドに * 引数を渡さないでください。

- **OBJECT SET STYLED TEXT**: *newText*引数には (スタイル付き) リッチテキストを渡さなければなりません。"<"や">"、"&"文字を含むプレーンテキストを渡すと、エラーが生成されます。これらの文字を含むプレーンテキストを挿入したい場合、新しい**OBJECT SET PLAIN TEXT**コマンドを使用してください。

OK変数の更新

4Dバージョン12.1より、以下のコマンドはOKシステム変数を更新するようになりました:

OBJECT SET STYLED TEXT ATTRIBUTES
OBJECT SET STYLED TEXT
OBJECT GET STYLED TEXT ATTRIBUTES
OBJECT Get plain text
OBJECT Get styled text

これらのコマンド実行後、エラーがなければOK変数が1に、エラーがあれば0に設定されます。特にスタイルタグが正しく評価できない場合に発生します (正しくない、あるいはタグが足りない等)。

エラーが発生した場合、元のテキスト変数は変更されません。テキストが評価されたときに変数上でエラーが発生した場合、4Dはテキストを生のテキストに変換します。結果"<"や">"、"&"文字はHTML実体参照に変換されます。

外部PHPインタープリターの使用

外部PHPインタープリターやカスタムモジュールを使用する際の手順が4D v12.1で変更されました。この変更はプログラムが格納されたフォルダーへのアクセス権が制限されている場合の設定をサポートするために行われました。

今バージョンより、4Dが起動するFastCGI-php初期化ファイル (*php.ini*) はデータベースのResourcesフォルダーに配置しなければなりません。最初の呼び出し時にこのファイルが見つからないと、4Dは適切な設定オプションを使用してそれを作成します。

4Dの組み込みインタープリターではカスタム*php.ini*ファイルを使用することはできません。提供されたデフォルト設定とは異なるPHP設定を使用したい場合、外部FastCGI-phpインタープリターを管理しなければなりません (この点については**4DでPHPスクリプトを実行する**で説明しています)。

注: 外部インタープリターのphp.iniファイルは、4D_Execute_PHP.phpユーティリティスクリプトの完全パスを提供する"auto_prepend_file"エントリーを含んでいなければなりません。このスクリプトは[4Dapplication]Resources/php/Windows または /Macにあります。このエントリーが書かれていない場合、完全なスクリプトのみが実行可能です。スクリプト中でのルーチンの呼び出しは動作しません。

PHP Executeコマンドの変更

PHPから送信されたデータの処理を容易にするため、**PHP Execute**と**PHP GET FULL RESPONSE**コマンドの動作が標準化 (受信した値は常にPHP開発者が返した値と同じ) されました。データ解釈の原則は**PHP Execute**コマンドの説明に記載されています。

データベース設定

4D v12.1の2つの新機能はデータベース設定 (**デザイン/データベース設定...** メニュー) で利用できるパラメターに関連します。

キャッシュの書き出し間隔

データベースキャッシュをディスクへ自動で書き込む時間の間隔を秒単位で設定できるようになりました。このオプションはデータベース設定のデータベース/メモリーページにあります:

4D v12.1では20秒がデフォルト値となります。アプリケーション要求に基づき、この値を変更できます。

日本語データの検索

データベース言語で日本語が選択されている場合、新しい**旧バージョン互換の文字列比較を使用する**オプションを使用して検索の動作を変更できます。このオプションを設定すると、検索の動作がバージョン2004以前と同じになります:

このオプションはデータベース設定の**データベース/データストレージ**ページにあります。このオプションは現在のデータ言語が日本語の時にのみ表示されます。このオプションは、4D v12.1で新しく作成された日本語データベースでは、デフォルトで選択されています。

他方、v12.0以前のバージョンで作成されたアプリケーションをv12.1で開いた場合、このオプションは選択されていません。旧バージョン互換の文字列比較を使用するためには明示的にこのオプションを設定し、アプリケーションを再起動してインデックスを再生成しなければなりません (4D v11.8ですでに設定済みの場合を除く)。

SQLを使用した書き出し

4D v12.1ではSQLを使用したデータベースからのデータ書き出しメカニズムが拡張され、開発者は書き出すデータのタイプに基づき、より詳細な設定を行うことができるようになりました:

- **SQL EXPORT DATABASE**と**SQL EXPORT SELECTION**コマンドは書き出しファイルにバイナリーデータを組み込むよう指示する、追加の引数を受け入れるようになりました。
- 4D SQLエンジンは16進フォーマットで格納されたデータをサポートするようになりました。

SQL EXPORT SELECTIONとSQL EXPORT DATABASEコマンドの新しい引数

SQL EXPORT DATABASEと**SQL EXPORT SELECTION**コマンドは追加のオプション引数*fieldLimitSize*を受け入れます。この引数はデータベースがBlob、ピクチャー、およびテキスト型のフィールドを含む場合にのみ効果があります。

この引数にはBlob、ピクチャー、およびテキスト (レコードの外に格納されたテキスト) 型のフィールドデータを書き出す際、データをBLOBサブフォルダーではなくメインの"Export.SQL"ファイル内に組み込んで書き出すよう指示するための、サイズの敷値をバイト単位で渡します。

この引数を使用して、書き出し中に生成される外部ファイルの量を制御し、処理の時間を短縮することができます。これは特に、データベースに大量のBlob、ピクチャー、およびテキスト型のフィールドがあり、そこにはある一定サイズのデータしか格納されていない場合に効果があります。

データが"Export.SQL"ファイルに書き出されると、バイナリーデータ (BLOBやピクチャー型) は16進フォーマットで格納されます。このデータを解釈するために、4D SQLエンジンはこの記法をサポートするようになりました (後述)。

16進のリテラル値

4D SQLエンジンは16進記法のリテラル値 (**literal**) をサポートするようになりました。16進値は数字 (0 - 9) や文字 (A - F) で構成され、すべてのデータタイプをバイトとして表現できます。1バイトは常に2つの16進値で決定されます。

SQLコマンドではこの記法を使用することを示すために、標準のSQLシンタックスを使用します:

```
X'<16進値>'
```

例えば10進値が*15*の場合、**X'0f'**と書きます。空の値 (ゼロバイト) は**X''**と書きます。

注: この動作は**SQL EXPORT DATABASE**と**SQL EXPORT SELECTION**コマンドの新しい引数をサポートするために追加されました。書き出しファイルに組み込まれたバイナリーデータは16進記法で格納されます。

4D HTMLタグの変更

警告: 技術的な理由により、ここで説明されている変更点は4D v12.1に組み込まれていません。この変更は4D v12.2 (およびパートナーの皆様には4D v12.1 Hotfix 1に) 組み込まれる予定です。

4D v12.1で4D HTMLタグの一部が変更されました:

- 4DHTMLVARの代わりに新しい**4DHTML**タグを使用しなければなりません。動作はまったく同じです。
- 4DVARは廃止予定となりました。このタグの動作は以下のいずれかのタグにより置き換えられます:
  - **4DHTML** (以前の4DHTMLVAR): 4D変数や式に格納されたHTMLソースを挿入するために使用します。
  - **4DTEXT**: 4D変数や式に格納されたテキストを挿入するために使用する新しいタグです。4DVARと異なり、挿入する値の先頭にChar(1)を置くことでHTMLソースを挿入できるという機能はサポートされていません。

新しい動作をまとめると以下のようになります:

myvarの値	12.1より前の4Dタグ	新しい4D 12.1タグ	Webページに挿入される値
myvar:="<B>"	*<!--#4DVAR myvar-->*	*<!--#4DTEXT myvar-->*	<B>
myvar:=Char(1)+"<B>"	*<!--#4DVAR myvar-->*	*<!--#4DTEXT myvar-->*	<B>
myvar:="<B>"	*<!--#4DHTMLVAR myvar-->*	*<!--#4DHTML myvar-->*	<B>
myvar:=Char(1)+"<B>"	*<!--#4DVAR myvar-->*	*<!--#4DHTML myvar-->*	<B>

互換性に関する注意: 4DVARと4DHTMLVARタグは4D v12.1でもサポートされており、以前のバージョンと同様に動作します。しかし可能な限り早く、新しいタグを、既存のアプリケーションを含むすべてのソリューションでご利用になることを強くお勧めします。

64 bitバージョンの4D Server

バージョン12より、4D ServerはWindows 64-bit OSをサポートしています。64-bitテクノロジーの主な利点はより多くのRAMがアドレス可能になることです。

ここでは4D Server v12.1の64-bitバージョンについて説明します。

必要となる最低Windowsバージョン

64-bitの4D Serverは最低以下のバージョンのWindows 64-bit OS上で動作します:

- Windows Vista 64-bit
- Windows 2003 Server 64-bit

アーキテクチャー

64-bit用の4D Server.exeアプリケーションは特別なバージョンであり、64-bit環境でのみ動作します。 32-bitシステム上では動作しません。

他方、標準の4DServer.exe (32 bits) をWindows 64-bitシステム上で起動した場合、動作しますが、それはエミュレーションモードになります。
クライアント側では、すべての4D v12マシーン (Mac OS および Windows) から64-bitバージョンの4D Server v12に接続できます。使用される4Dアプリケーションは標準の32-bitバージョンです (下図参照)。

互換性

インタープリターモードでは、同じ4Dデータベースを64-bit 4D Serverあるいは32-bit 4D Serverで実行できます。どちらのアプリケーションを使用しても、開発手順は同じです。

コンパイルモードでは、64-bit 4D Serverで実行させるために、64-bitプロセッサー用にコンパイルしなければなりません ("64-bit用にコンパイル"参照)。
32-bitのみ用にコンパイルされ、インタープリターコードを含まないデータベースは64-bit 4D Serverで実行できません。

実行時には、以下の相違点に留意してください:

- 64-bit 4D Serverには64-bitモード用にコンパイルされたプラグインのみがロードされます。64-bitプラグインはフォルダーとしてビルドされ、サーバーのPluginsフォルダーに配置されなければなりません (Win4DXフォルダーに配置される.4DXと.RSRファイルに基づく以前のアーキテクチャーはもうサポートされません)。32-bitプラグインは64-bit 4D Serverにロードされませんが、サーバー上のPluginsフォルダーに32-bitプラグインを配置し、リモートマシンに配布することはできます。この場合、サーバーを呼び出すメカニズムは動作しません (例えばサーバー上のテンプレートを読み書きする4D Writeの**WR SET AREA PROPERTY**コマンドなど)。
  32-bitの.4DXファイルはプラグイン内の/Contents/Windowsフォルダー内に置き、64-bit用のファイルは/Contents/Windows64フォルダーに置きます。
- 64-bit 4D Serverで使用するコンパイル済みの4Dコンポーネントは、64-bit用にコンパイルしなければなりません。
- アプリケーションがロードするBlobに使用されるメモリー量は依然2GBに制限されます。
- QuickTimeは64-bit 4D Serverでサポートされません。
  4D社ではPICTフォーマットのピクチャーの利用は推奨しません。ピクチャーが100% Quickdrawであれば64-bitバージョンの4D Serverで扱うことができますが、ピクチャーにQuicktimeが含まれている場合、ロードすることができません。

4D Internet Commands

64-bit 4D Serverで4D Internet Commandsを利用できるようにするためには、バージョン12.1以降の4D Internet Commandsプラグインをインストールしなければなりません。

64-bit用にコンパイル

4D v12アプリケーションは32-bitと64-bitプロセッサー用にコンパイルできるようになりました。これを行うために、新しい**64-bitプロセッサー用にもコンパイルする**オプションがデータベース設定の"コンパイラー"ページに追加されました:

このオプションが選択されていると、コンパイラーは.4DCと.4DBファイルに64-bitコードと32-bitコードを含めます。結果これらのファイルを32-bitあるいは64-bitの4D Serverいずれでも実行できるようになります。デフォルトでこのオプションは選択されていません。

注: データベースを64-bitバージョンでコンパイルするにはUnicodeモードで動作しなければなりません。そうでなければコンパイル時にエラーが生成されます。

キャッシュメモリーのサイズ

64-bitアーキテクチャーでは1 TB (1000 GB) までのRAMメモリーをアドレス可能になるので、64-bit 4D Serverに割り当てることのできるキャッシュメモリーは事実上無制限となります。

注: 比較すると、32-bitアーキテクチャーにおいては4 GBのRAMに制限されます (OSレベルで)。

データベース設定の"データベース/メモリー"ページで指定したキャッシュ量を確保できない場合、4D Serverは確保可能な最大サイズを割り当て、アプリケーションの起動時にユーザーに知らせます。ユーザーは終了するか、そのままのサイズで続行するかを選択できます。

インターフェース

実行中、64-bit 4D Serverアプリケーションか標準の4D Serverアプリケーションかは、サーバー管理ウィンドウのモニターページに表示されるロゴで見分けることができます:

注: このロゴは4D Serverについてウィンドウにも表示されます。

ランゲージ

変更されたコマンド

以下の4Dランゲージコマンドは新しい64-bit 4D Serverをサポートするために変更されました: **Version type**、**SET DATABASE PARAMETER**、そして**Get database parameter**。詳細はそれぞれのコマンドの説明を参照してください。

プロセススタックのサイズ

64-bitバージョンの4D Server上で走るプロセスのスタックは、32-bitバージョンよりも多くのメモリーを必要とします (約2倍)。**Execute on server**や**New process**コマンドを使用して64-bitバージョンの4D Server上でプロセスを作成する場合、最低128,000 byteを*stack*引数に渡すことを、呼び出し連鎖が大きくなる場合やスタックが足りないというエラーが発生する場合にはさらにそれを増やすよう推奨します。コードが64-bit 4D Server上で実行されるためのものである場合、この引数をチェックするようにしてください。

64-bitサーバーで利用できないコマンド

4D Serverで使用できない標準コマンド (**ストアドプロシージャで行わないこと**参照) に加え、以下のコマンドは64-bitサーバーで実行できません。これらのコマンドを例えばストアドプロシージャーで呼び出すと、警告ダイアログが表示され、エラー67が返されます。このエラーは**ON ERR CALL**コマンドでインストールされるエラー処理メソッドでとらえることができます。

4D, **クイックレポート**テーマ	**QR New offscreen area**
4D Chart, **CTエリア**テーマ	**CT New offscreen area**

サブテーブルからの移行を補助するコマンド

4Dバージョン11より、サブテーブルはもうサポートされていません。変換されたデータベースにおいては互換性メカニズムにより、以前のサブテーブルはまだ動作します。しかしサブテーブルは標準のリレートしたテーブルに置き換えることを強く推奨します。

これを容易に行うために新しいコマンド**Get subrecord key**が4D v12.1に追加されました。このコマンドを使用すれば、変換されたデータベースに作成された特別なリレーションを保持したまま、順次コードを移行することが可能です。後でこのリレーションは削除することができます。アプリケーションに変換されたサブテーブルが含まれる場合、新しいコマンドを使用しつつコードを変更し、サブテーブルを使用しないように書き換えてください。

4D Internet CommandsでのSSLサポート

バージョン12.1より、4D Internet Commandsプラグインでメールサーバーに接続し、メールを送受信する際に、SSLプロトコルを使用できるようになりました。この新機能によりデータ交換の際のセキュリティが強化されます。

注: 4DIC 12.1のSSL実装では陰解法 (implicit method) が使用されています。

4DICでSSLプロトコルを使用するに当たって特段の設定をする必要はありません。以下のコマンドに追加の引数を渡すことでSSLを有効にできます。(サーバー側でSSLの設定が行われている必要はあります。)

- **IMAP_Login**コマンドは追加の引数*sessionParam*を受け入れるようになりました。この引数に1を渡すと、IMAPサーバーへの接続がSSLで行われます。
- **SMTP_Send**コマンドは追加の引数*sessionParam*を受け入れるようになりました。この引数に1を渡すと、メッセージ送信がSSLで行われます。
- **SMTP_QuickSend**コマンドは追加の引数*sessionParam*を受け入れるようになりました。この引数に1を渡すと、メッセージ送信がSSLで行われます。
- **POP3_Login**コマンドは追加の引数*sessionParam*を受け入れるようになりました。この引数に1を渡すと、POP3サーバーへの接続がSSLで行われます。
- **IT_SetPort**コマンドは*protocol*引数に3つの新しい値を受け入れるようになりました:
  - 12 = SMTP SSL
  - 13 = POP3 SSL
  - 14 = IMAP SSL