4D v14 R2 - アップグレード

新しい4Dのプロダクトリリースサイクルの最初のバージョンとなる、4D v14 R2へようこそ。このマニュアルではこのリリースに追加された全ての新機能や実装について説明しています。

- ランゲージリファレンス
- SQL
- ODBC Driver
- 4D Internet Commands

ランゲージリファレンス

- Application version(新しいRリリースのサポート)
- FORM GET OBJECTS(新しい引数)
- FORM GET PROPERTIES (formName が任意に)
- TRANSFORM PICTURE (透過度演算子)
- WA GET/SET PREFERENCE
- WEB Is server running

Application version(新しいRリリースのサポート)

Application version {(buildNum {; *})} -> 戻り値

説明

4Dの新しい"R"リリースをサポートするため、**Application version** コマンド("**4D環境**" テーマ)で返される値が更新されました。

原則として、"R"リリースの場合、"R"番号がリリース番号となり、リビジョン番号は常に"0"となります。この原理は長いバージョン番号でも短いバージョン番号でも適用されます。

短いバージョン番号での例です:

```
value:=Application version // 短いバージョン番号
```

バージョン	返り値	
4D v14 R2	"1420"	
4D v14 R3	"1430"	
4D v14.1	"1401"	4D v14の最初のバグフィックスリリース
4D v14.2	"1402"	4D v14の2つ目のバグフィックスリリース

長いバージョン番号での例です:

```
value:=Application version(*) // 長いバージョン番号
```

バージョン	返り値
4D v14 beta R2	"B0011420"
4D v14 final R3	"F0011430"
4D v14.1 beta	"B0011401"

例題

コマンドから返されたアプリケーションの短いバージョン番号の値を使用して4Dアプリケーションのリリース名を表示したい場合を考えます。以下の様に書くことが出来ます:

```
C_LONGINT($Lon_build)
C_TEXT($Txt_info;$Txt_major;$Txt_minor;$Txt_release;$Txt_version)

$Txt_version:=Application version($Lon_build)

$Txt_major:=$Txt_version[[1]]+$Txt_version[[2]] //バージョン番号、この場合は14
$Txt_release:=$Txt_version[[3]] //Rx
$Txt_minor:=$Txt_version[[4]] //.x

$Txt_info:="4D v"+$Txt_major
If($Txt_release="0") //4D v14.x
   $Txt_info:=$Txt_info+Choose($Txt_minor#"0";"."+$Txt_minor;"")

Else //4D v14 Rx
   $Txt_info:=$Txt_info+" R"+$Txt_release
End if
```

FORM GET OBJECTS(新しい引数)

FORM GET OBJECTS (objectsArray {; variablesArray {; pagesArray}} {; * | *formPageOption*})

引数	型		詳細
objectsArray	文字列配列	<-	フォームオブジェクト名
variablesArray	ポインター配列	<-	オブジェクトに関連付けられた変数またはフィールドへのポインター
pagesArray	整数配列	<-	各オブジェクトのページ番号
* | *formPageOption*	演算子 | 倍長整数	->	* 指定時=カレントページまで減らす、または *1=Form current page, 2=Form all pages, 4=Form inherited*

説明

FORM GET OBJECTS コマンド("**フォーム**" テーマ)は新しい *formPageOption* 引数を受け取るようになりました。これにより、オブジェクトをどこから取得したいのかというフォームの部分を指定することができるようになります。

デフォルトでは、*formPageOption* 引数(と* 演算子)が省略された場合継承されたオブジェクトを含むすべてのページからのオブジェクトが返されます(以前の4Dのリリースと同じ挙動です)。

コマンドのスコープを限定するためには*formPageOption* に値を渡します。以下の値のうちどれか一つ(またはそれらの組み合わせ)を使用することが出来ます。これらの値は"**Form Objects (Access)**" テーマ内にあります:

定数	型	値	詳細
Form current page	倍長整数	1	カレントページの全てのオブジェクトを返します。0ページ目も含めますが、継承されたオブジェクトは除きます。
Form all pages	倍長整数	2	全てのページの全てのオブジェクトを返しますが、継承されたオブジェクトは除きます。
Form inherited	倍長整数	4	継承されたオブジェクトのみを返します。

注: * 演算子を渡す事はForm current page+Form inherited の組み合わせを渡す事と等価です。* 演算子を使用したシンタックスは、今後は使用されるべきではありません。

例題 1

継承されたフォームのオブジェクトも含めて(もしあれば)、全てのページの情報を取得したい場合:

```
  //開いているフォーム
FORM GET OBJECTS(objectsArray;variablesArray;pagesArray)
```

または:

```
  //ロードしたフォーム
FORM LOAD([Table1];"MyForm")
FORM GET OBJECTS(objectsArray;variablesArray;pagesArray;Form all pages+Form inherited)
```

例題 2

カレントページに関する情報だけを取得し、ロードされたフォームのページ0と継承されたフォームオブジェクトも(もしあれば)含めたい場合:

```
FORM LOAD("MyForm")
```

```
FORM GOTO PAGE(2)
FORM GET OBJECTS(objectsArray;variablesArray;pagesArray;Form current page+Form inherited)
```

例題 3

継承されたフォーム内の全てのオブジェクトの情報が(もしあれば)取得したい場合(ただい、もし継承されたフォームがない場合には空の配列が返されます):

```
FORM LOAD("MyForm")
FORM GET OBJECTS(objectsArray;variablesArray;pagesArray;Form inherited)
```

例題 4

0ページ目のオブジェクトも含め、4ページ目のオブジェクトの情報を取得し、継承されたフォームオブジェクトに関しては(もしあれば)除外したい場合:

```
FORM LOAD([Table1];"MyForm")
FORM GOTO PAGE(4)
FORM GET OBJECTS(objectsArray;variablesArray;pagesArray;Form current page)
```

例題 5

全てのページのオブジェクトの情報を取得し、継承されたフォームオブジェクトに関しては(もしあれば)除外したい場合:

```
FORM LOAD([Table1];"MyForm")
FORM GET OBJECTS(objectsArray;variablesArray;pagesArray;Form all pages)
```

FORM GET PROPERTIES (formName が任意に)

FORM GET PROPERTIES ({{aTable*{;formName ;}* width ; height {; numPages {; fixedWidth {; fixedHeight {; title}}}})

---

formName parameter引数が任意となりました。この引数が省略された場合、コマンドは**FORM LOAD** コマンドを使用してロードされたカレントフォームへと適用されます。
参照*:* **FORM GET PROPERTIES**

TRANSFORM PICTURE (透過度演算子)

TRANSFORM PICTURE (picture ; operator {; param1 {; param2 {; param3 {; param4}}}})

新しい演算子

TRANSFORM PICTURE コマンド("**ピクチャ**"テーマ)は*operator* 引数において新しいTransparency 定数を受け取るようになりました。これにより、カスタムの透過度を変換されたピクチャに適用することが出来るようになりました。

この機能は特に、廃止されたPICTフォーマットのピクチャから変換されたピクチャの透過度を操作するために設定されたものですが、どんな種類のピクチャに対しても使用することが出来ます。

新しい定数は"Picture Transformation" テーマに追加されています。これを渡すと、使用できる引数は*param1* のみになります:

定数(値)	param1	param2	param3	param4	値
Transparency (102)	RGB color	-	-	-	16進数

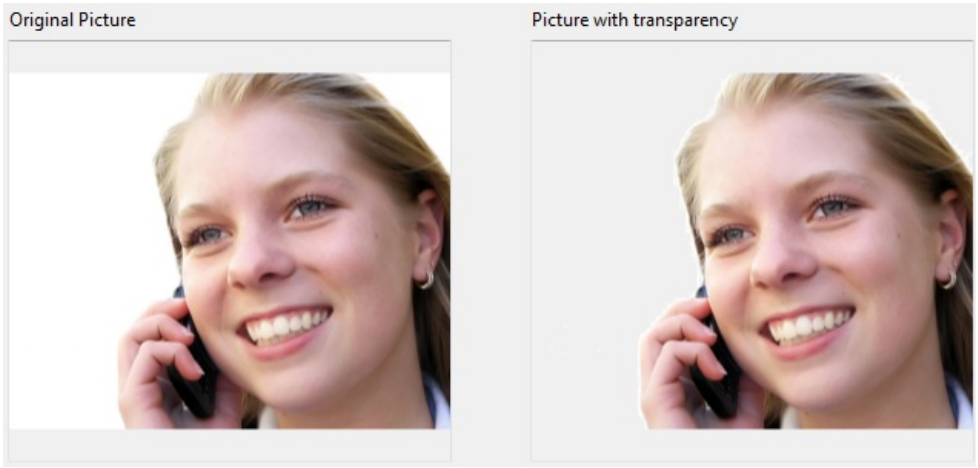
- Transparency: *param1* に渡された色に基づいて透過マスクがピクチャに適用されます。たとえば、0x00FFFFFF (白)を*param1*に渡した 場合、オリジナルのピクチャ内の白いピクセルは、変換されたピクチャの中では透明になります。この操作はビットマップピクチャにもヴェクター画像にも適用することが出来ます。デフォルトでは、*param1* 引数が省略されている場合は、白(0x00FFFFFF) がターゲットカラーとして設定されます。

例題

ピクチャの白い部分を透過にしたい場合を考えます。このためには、以下のコードを使用します:

```
TRANSFORM PICTURE(Pict1;Transparency;0x00FFFFFF) //0x00FFFFFF は白のカラーコード
```

結果は以下の様になります:



WA GET/SET PREFERENCE

新しいデフォルトの挙動

セキュリティ上の理由から、4D v14 R2以降、ファイルまたはURLをWebエリアにドロップすることによってエリアのコンテンツを変えることは、デフォルトでは不可となりました。ユーザーがファイルまたはURLをエリアにドロップしようとした場合には禁止アイコン  が表示されます。

以前のリリースでは、こういったアクションを禁止するためには特定のフィルターを、例えば **WA SET URL FILTERS**などを使用してインストールする必要がありました。

4D v14 R2でこういったドロップを許可するためには、新しいWebエリア設定、wa enable url drop を設定する必要があります。

の新しいセレクター

WA SET PREFERENCEコマンドの *selector* 引数において、WebエリアへのURLやファイルのドロップを許可するための新しい定数が追加されました:

定数	型	値	詳細
WA enable URL drop	倍長整数	101	WebエリアへのURLやファイルのドロップを有効化(デフォルト = False)

WebエリアへのURLドロップを有効化したい場合、URLをロードする前(On load フォームイベント中等が良いでしょう)にこの設定がなされている必要があります。

例題

'myarea' というWebエリア内でURLドロップを有効化したい場合:

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

WEB Is server running

WEB Is server running -> 戻り値

引数	型		説明
戻り値	ブール		Webサーバーが実行中が作動中であればTrue、それ以外はFalse

説明

新しい**WEB Is server running** コマンド("**Webサーバ**"テーマ)は、4Dがビルトインされ ているWebサーバーが動作中である場合にはTrueを、Webサーバーがオフである場合にはFalseを返します。

このコマンドは、それが実行されたWebサーバーの動作状況を返します:

実行されたコンテキスト	どこの状況を返すか
4D スタンドアローンアプリケーション	ローカルの4D Web サーバー
4D Server	4D Server Web サーバー
4D リモートモード(ローカルプロセス)	ローカルの4D Web サーバー
4D リモートモード(4D Server ストアドプロシージャー)	4D Server Web サーバー
4D リモートモード(他の4D のリモートストアドプロシージャー)	リモート4D Web サーバー

例題

Webサーバーが実行中かどうかをチェックしたい場合:

```
If(WEB Is server running)
   ... //実行する処理
End if
```

SQL

_USER_INDEXESシステムテーブルの新しいフィールドKEYWORD

_USER_INDEXESシステムテーブルの新しいフィールドKEYWORD

新しいKEYWORDというブール型のフィールドが_USER_INDEXESのシステムテーブルに追加されました。これにより、通常のインデックスとキーワードに基づいたインデックスを識別することが出来るようになりました。

_USER_INDEXES		**データベースのユーザーインデックスを記述します**
KEYWORD	BOOLEAN	インデックスがキーワードに基づいていた場合にはTrue、そうでない場合にはFalse

キーワードインデックスはクラスター、またはBTree型のものが使用できます。

ODBC Driver

OS XでのODBC Driverのインストール

OS XでのODBC Driverのインストール

4D v14 R2では、OS X用の64-bit版の4D ODBC Driverが用意されています。ドライバーは以下の手順に従って手動でインストールする必要があります。

ODBCの使用を有効化する前に、必ず最新のODBCフレームワークを以下のリンクからダウンロードして下さい:

http://www.iodbc.org/dataspace/iodc/wiki/iODBC/Downloads#Mac OS X

ODBC Driverは4DのWebサイトからダウンロード可能です。フォルダーには32-bit版と64-bit版とが両方入っています:

- 4D ODBC x32.bundle
- 4D ODBC x64.bundle

ドライバーは、4Dや4D Serverのバージョンではなく、ODBCクライアントのバージョンに合わせて選択して下さい。例えば、ODBCクライアントとして64-bit版のPythonを使用している場合、使用している4D Serverが32-bit版だったとしても必要になるのは64-bit版のODBC Driverです。ドライバーは、同じマシンに両方のバージョンをインストールすることが可能です。

64-bit ドライバーをMac OS Xにインストールする

ODBCドライバーをインストールするためには以下の手順に従って下さい:

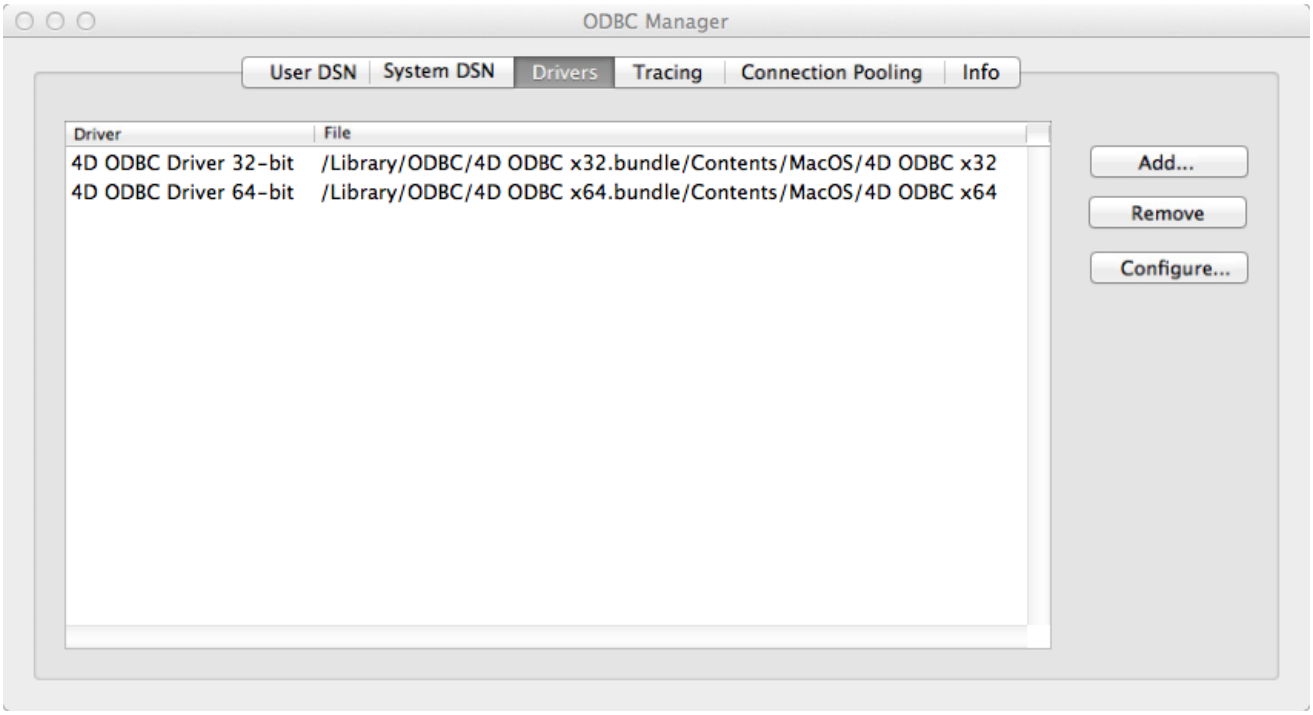
1. **4D ODBC x64.バンドルまたは4D ODBC x32.バンドル(もしくはその両方)を{Library}/ODBC/ フォルダへとコピーします**
2. **/Library/ODBC/ フォルダ内にある*odbcinst.ini*というテキストファイルをテキストエディタで開き、以下の様に書き換えて下さい:**

```
[ODBC Drivers]
4D ODBC Driver 64-bit = Installed
4D ODBC Driver 32-bit = Installed

[4D ODBC Driver 64-bit]
Driver = /Library/ODBC/4D ODBC x64.bundle/Contents/MacOS/4D ODBC x64
Setup = /Library/ODBC/4D ODBC x64.bundle/Contents/MacOS/4D ODBC x64
APILevel = 2
ConnectFunctions = YYN
DriverODBCVer = 3.52
FileUsage = 0
SQLLevel = 3

[4D ODBC Driver 32-bit]
Driver = /Library/ODBC/4D ODBC x32.bundle/Contents/MacOS/4D ODBC x32
Setup = /Library/ODBC/4D ODBC x32.bundle/Contents/MacOS/4D ODBC x32
APILevel = 2
ConnectFunctions = YYN
DriverODBCVer = 3.52
FileUsage = 0
SQLLevel = 3
```

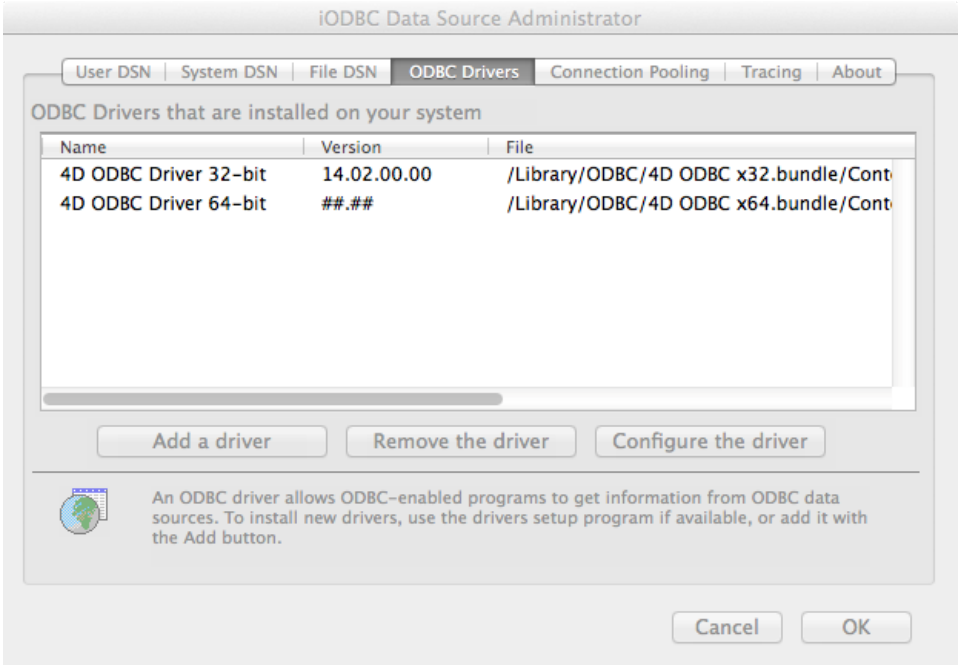
"Applications/Utilities/"フォルダー内にあるODBC Managerを起動し、Data Source Name (DSN)を作成することが出来ます。または、iODBCフレームワークと提供されているマネージャーを使用することもできます。

注: Mac OS X 10.6 (Snow Leopard)以降、ODBC Administration Toolは標準では提供されていません。しかしながら、以下からダウンロードすることが出来ます:
http://support.apple.com/downloads/ODBC_Administrator_Tool_for_Mac_OS_X

"Drivers"タブをクリックすることによって、4D ODBC Driverが正常にインストールされているかどうかを確認することが出来ます:



iODBC Administratorに関しても同様です:



64-bit DSNの作成

64-bitドライバーにはDSNがありませんが、以下の二つの方法を用いて64-bitDSNを作成することが出来ます:

- 32-bitドライバーを使用して作成
- ODBC Administratorを使用して作成

32-bit ドライバーを使用して作成

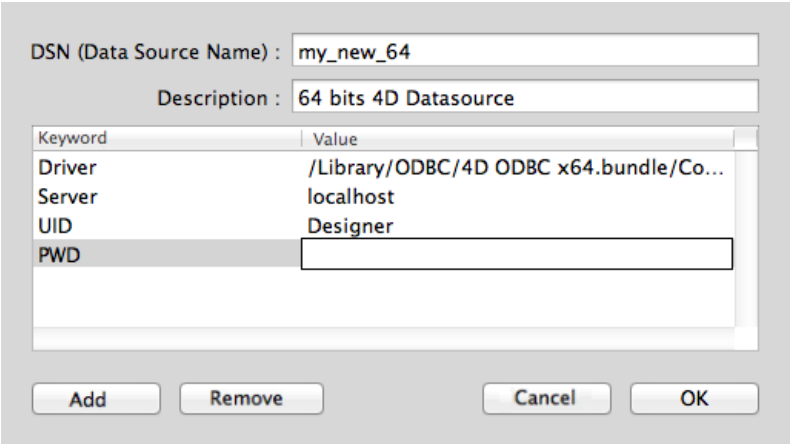
32-bit ODBCドライバーを使用して32-bit DSNを作成し、その後 /Library/ODBC フォルダから直接DSNを修正することが出来ます。/Library/ODBC/odbc.ini のテキストファイルをテキストエディターで開き、以下の手順に従って編集をしてください:

1. [ODBC Data Sources] セクション内にて、"4D ODBC Driver **32-bit**" を "4D ODBC Driver **64-bit**"に変更します
2. [{your data source name}] セクションにて、
   Driver=/Library/ODBC/4D ODBC **x32**.bundle/Contents/MacOS/4D ODBC **x32**
   を、
   Driver=/Library/ODBC/4D ODBC **x64**.bundle/Contents/MacOS/4D ODBC **x64**
   に変更します。

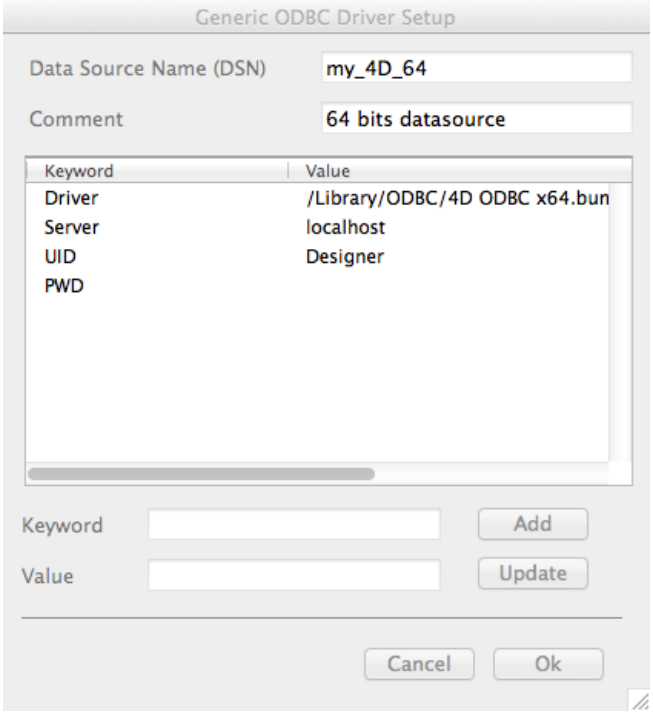
ODBC Administrator または iODBC Administrator を使用して作成

1. **System DSNタブをクリックします**
2. **Add ボタンをクリックして64-bit 4D ODBC Driverを設定済みのドライバの一覧から選択します。通常のDSNジェネレーターが表示されます。**
3. **以下のキーワード/値のペアをダイアログに入力して下さい**:

キーワード	**値**
Driver	/Library/ODBC/4D ODBC x64.bundle/Contents/MacOS/4D ODBC x64
Server	<4D ServerのIPアドレス>
UID	<ユーザー名(Designer可)>
PWD	<ユーザーのパスワード(空欄可)>



iODBCの場合は以下の様に表示されます:



4. **OKをクリック**
   新しいDSNがODBC AdministratorのSystem DSNタブ内で選択できるようになります。

4D Internet Commands

SMTP_Attachment (重要)

SMTP_Attachment (重要)

互換性

4D Internet Commands プラグインの新機能の実装を進めつつありますが、検証が充分とは言えないため、新機能としての提供は今回のバージョンでは見合わせることになりました。

その影響で、**SMTP_Attachment** コマンドに対して6つめの引数の受け渡しが必須となっています:

- **SMTP_Attachment** コマンドを使用している場合、6つめのパラメータに空の文字列を受け渡してください。
- 4D Internet Commands を使用しているコンポーネントがある場合、それらを 4D v14 R2 で再コンパイルしてください。

上記の対策が難しい場合には、引き続き 4D Internet Commands v14.1 を利用してください。他の変更点はなく、完全互換です。