DataEase Custom Functions

CDFS2 Library (continued)

Custom Functions: What is a CDF?

DataEase contains more than fifty functions which have been built into it by the developers – functions such as “JoinText”, “SpellDate”, and “Random”. But each user application is unique, and has its own special requirements. So DataEase allows the application developer to “plug-in” extra commands as and when they’re needed. These plug-in commands are called **Custom Defined Functions**.

Your DataEase package contains more than two hundred of these functions, ready for immediate use. Many more CDFs have been produced by third-party developers, and can be purchased from them.

How are CDFs Used?

Before you can use a CDF, you must **register** it with your DataEase Application – registration ‘plugs in’ the function.

Once registered, Custom Functions work in exactly the same way as the built-in DataEase functions. So they can be used in Field **Validations and Derivations**, **Button and Picture Actions**, **Filters**, Procedures and **Custom Menu Bars**. They can **not** be used on a Menu Document.

Parameters and CDFs

Manyt CDFs require one or more **parameters**. A parameter is simply a value which the function needs to work properly – for example, the DataEase function **Jointext** requires two parameters – Jointext(“value1”, “value2”) – where “value1” and “value2” are two parameters being passed to the function so that they can be joined.

When you pass a character string to a function, the function must pick up the address of the string as the parameter (by reference). The character string is not passed by a value. You receive a copy of the string; the DataEase value being passed does not change …which is a long-winded way of saying that the parameters themselves are NOT affected by the function. Instead the function returns a new value.

If you pass a parameter and the DataEase data type of the value does not match the specified parameter type, DataEase converts the data into the correct type before passing the data to the function. (Numbers and character strings, for example, will be automatically converted, if this is required).

Passing Values to a CDF from the Current Record

The parameter type(s) that you specify on your Custom Functions record for the function dictates how the data values are passed to the function.

For example, if you want to pass the values of the NAME and SALARY fields to a CDF (called "Compute") which accepts parameters defines as String and Double, you would enter the following function call;

NEWSAL:= COMPUTE (name, salary)

When processing reaches this line in the query, DataEase assigns the NEWSALvariable to the value returned by the Compute Function. The two parameters that DataEase passes to the function are the address of the copy of the string for the NAME field (STRING parameter type) and the value in the SALARY field (FLOAT parameter type) in the current record.

Executing CDFs in Different DLL files

If two or more CDFs are referenced in the same document, when DataEase encounters the second CDF, it checks to see if that function's DLL file is already dynamically linked.

If the DLL is linked, DataEase executes the function. If the DLL file is not linked, DataEase must call Windows to link the function.

How are CDFs Created?

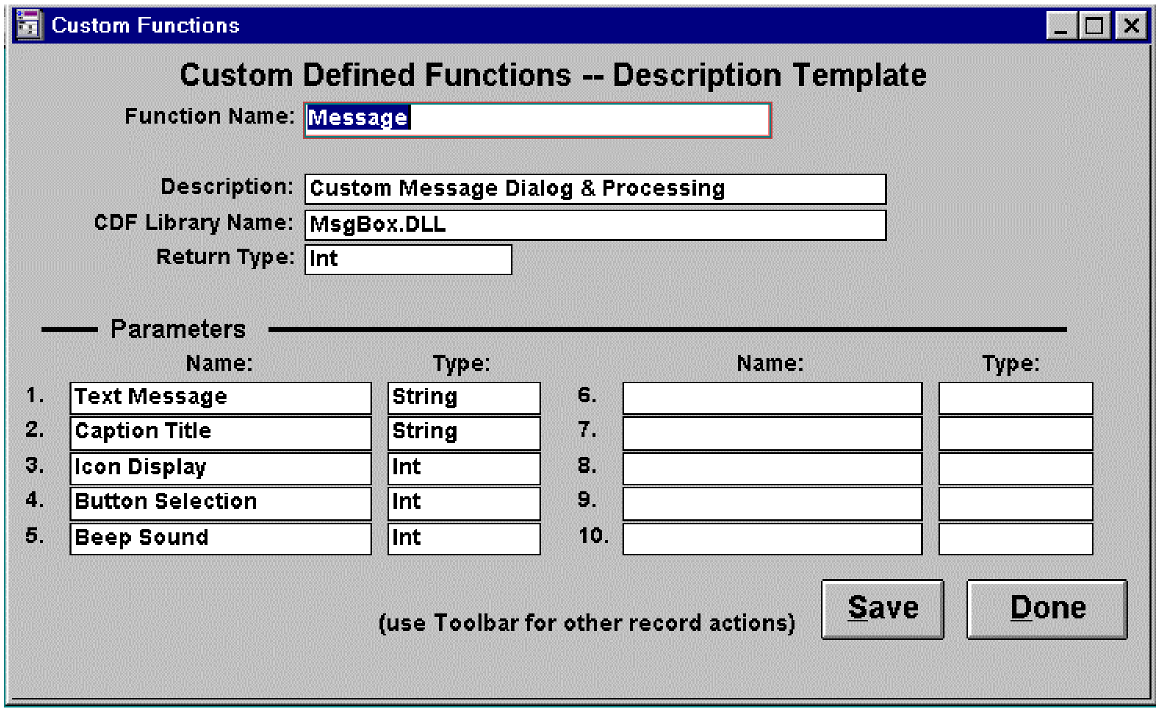
A Custom-Defined Function (CDF) is a program that has been written outside of DataEase, using a language such as C, Assembler, or another supported programming language. Many CDFs have been created specifically for DataEase. Other functions – such as **ShellExecuteA** and **SwapMouseButton** - are part of the **Windows API**.

If you're a programmer, you can create your own CDFs. Guidelines for the **Creation of CDFs** can be found in the Designer Guide.

Registering a CDF

Before you can use a CDF with DataEase, you must register the CDF in the Custom Functions Form of the application that will access it. You must enter a Custom Function record for **each** CDF function, even for those in the same DLL library file.

Use the Function Parameters fields to tell DataEase the name and type of value(s) you want passed to the application. The example below shows the “Message” function, part of the MsgBox CDF Library, being registered.



Custom Defined Functions -- Description Template

Function Name: Message

Description: Custom Message Dialog & Processing

CDF Library Name: MsgBox.DLL

Return Type: Int

Parameters

	Name:	Type:		Name:	Type:
1.	Text Message	String	6.		
2.	Caption Title	String	7.		
3.	Icon Display	Int	8.		
4.	Button Selection	Int	9.		
5.	Beep Sound	Int	10.		

(use Toolbar for other record actions)

Once the CDF has been registered, it is available for use within your DataEase application. You use a CDF function in exactly the same way as you would use any of the in-built DataEase functions, such as ‘jointext’, ‘spelldate’, and so on. Thus you can include the assigned function name and parameters in the Derivation Formula or Validation Formula for a field, or in the expression for a Button or Picture action.

CDF Installation

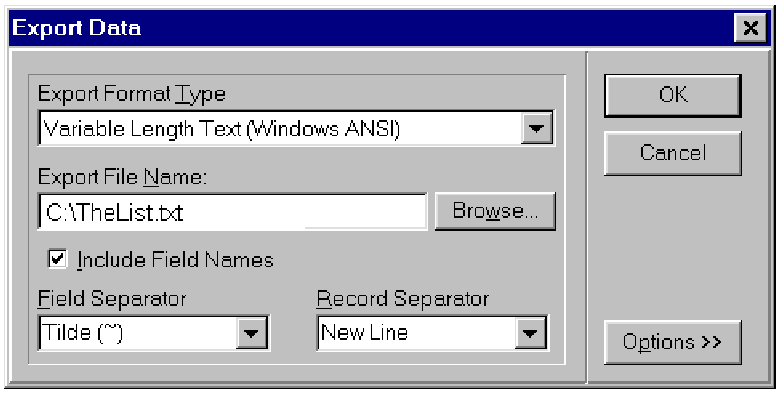
There are a number of ways you can register CDFs with a DataEase Application. If you only need to register one or two CDF’s, then the easiest way is to use Copy and Paste, as described below:

Open the sample CDF application - called **CDF Library** - which comes with DataEase. (The default path is DE6\Samples\CDFLibs\CDF Library). Open the Main Menu, press the button on the bottom of the screen labeled **'CDFs SYSTEM FORM'**. Press F3 until you find the function you want to use in your own application. From the menu bar select **Edit>>Copy Record**. Exit the Sample CDF application and access your own application. Open the CDFs System Form, **Paste** the record, and save it.

Note: You may need to change the contents of the CDF LIBRARY NAME field to make sure DataEase can find the DLL file.

Importing CDFs

If you have a larger number of CDFs to register, then the best way is to create an Export File, and then Import this file into the target application. To create the Export file, enter Custom Functions and select **File>>Export**. Fill in the dialog as shown below (Note…the file name should be whatever suits you…):


Export Data

Export Format Type: Variable Length Text (Windows ANSI)

Export File Name: C:\TheList.txt

☑ Include Field Names

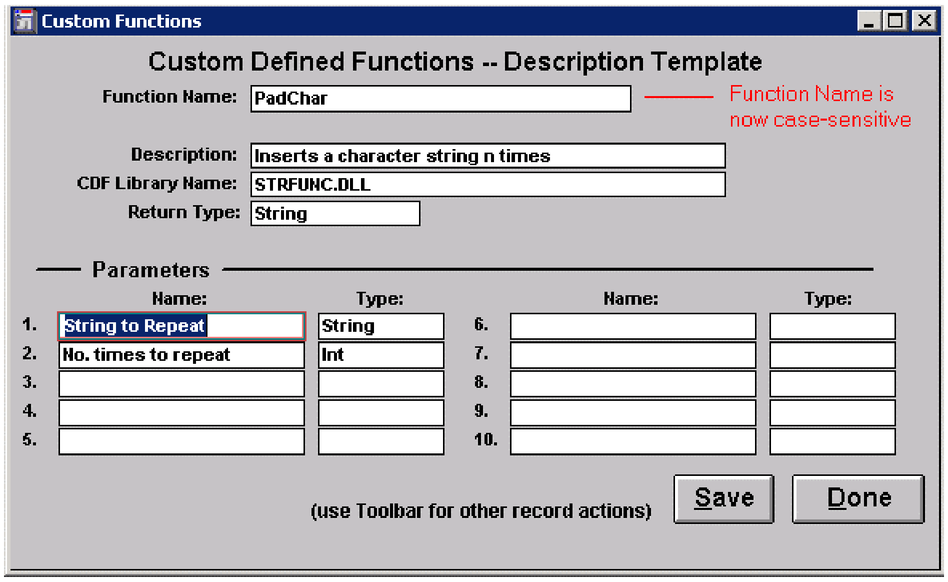
Field Separator: Tilde (~)

Record Separator: New Line

To Import the CDF’s, you must first go into **Application>>Preferences**, and tick the **Show System Tables** option. You can then define and run an Import file, similar to the one shown below.

Case Sensitivity

In previous versions of DataEase, the Custom Function Registration Dialog Box was case-insensitive. In DataEase 6 the Function Name field **IS** case sensitive.


Custom Functions

Custom Defined Functions -- Description Template

Function Name: PadChar

Function Name is now case-sensitive

Description: Inserts a character string n times

CDF Library Name: STRFUNC.DLL

Return Type: String

Parameters

	Name:	Type:		Name:	Type:
1.	String to Repeat	String	6.		
2.	No. times to repeat	Int	7.		
3.			8.		
4.			9.		
5.			10.		

(use Toolbar for other record actions)

Be sure to spell the Function Name correctly in this registration form!

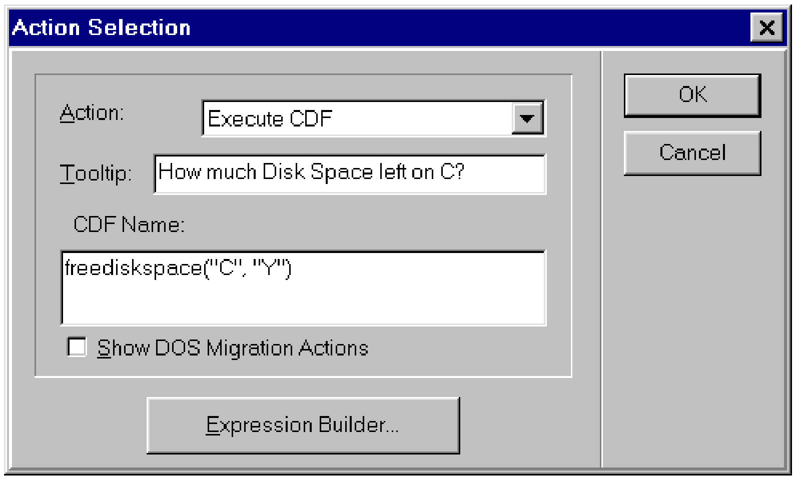
Note: DataEase does NOT regard a Function Name as being case-sensitive, when called from a button or script. The Registration Dialog box shown above is the **only** place in which the Function Name is case-sensitive.

Therefore you must not have two Functions which share the same name, but contain letters in a different case.

For example, having two functions called "CallTime" and "CALLTime" would lead to confusion…so don't duplicate Function Names.

CDF Examples: Using CDFs in Buttons and Pictures

The graphic below shows a CDF being run from a button action. It could be run from a Picture Action in exactly the same way. This particular CDF – **FreeDiskSpace** – will display the amount of disk space left on the specified drive ("C", here), in a Message Box.



Action: Execute CDF

Tooltip: How much Disk Space left on C?

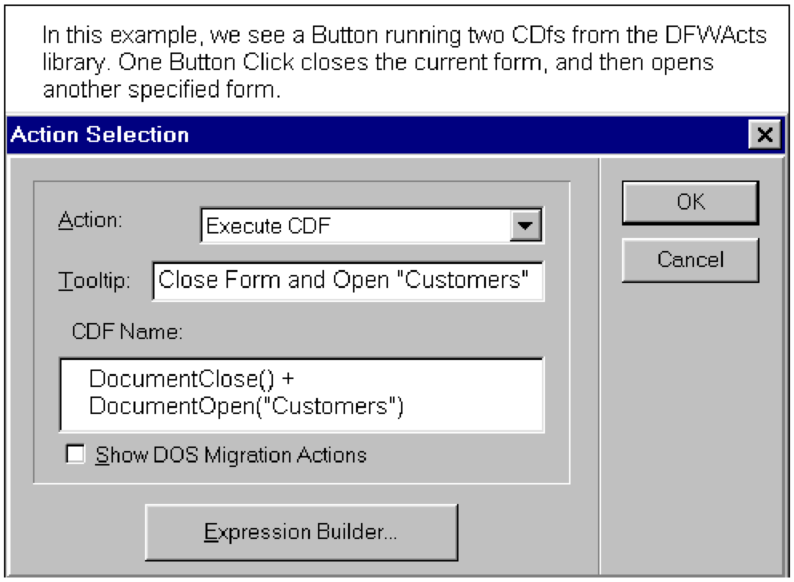
CDF Name:

freediskspace("C", "Y")

☐ Show DOS Migration Actions

CDFs are very powerful when run from buttons and pictures, because they allow you to:

1) Run multiple DataEase Actions on a button/picture.
2) Use conditional statements.
3) Pass derived parameters to the Action.
4) Run any combination of the above on a single button/picture.


In this example, we see a Button running two CDFs from the DFWActs library. One Button Click closes the current form, and then opens another specified form.

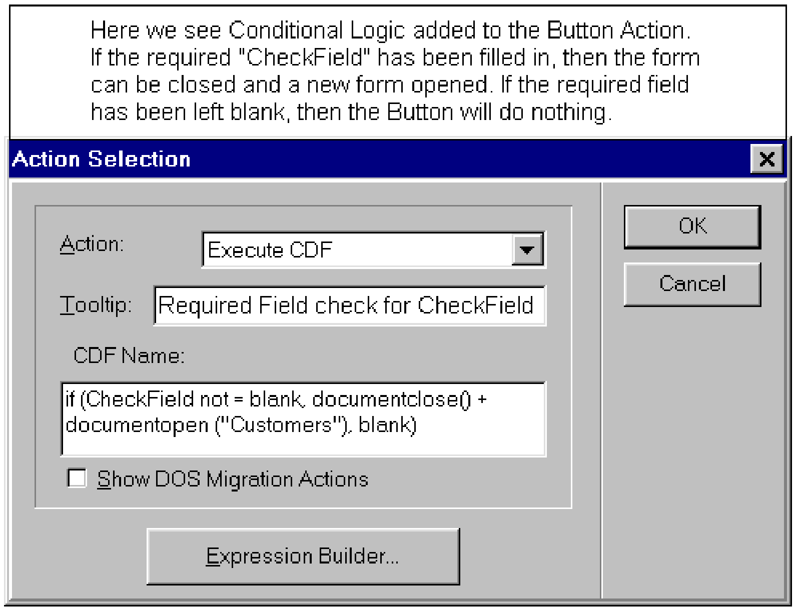
Action: Execute CDF

Tooltip: Close Form and Open "Customers"

CDF Name:

DocumentClose() +
DocumentOpen("Customers")

☐ Show DOS Migration Actions



Here we see Conditional Logic added to the Button Action. If the required "CheckField" has been filled in, then the form can be closed and a new form opened. If the required field has been left blank, then the Button will do nothing.

Action: Execute CDF

Tooltip: Required Field check for CheckField

CDF Name:

```
if (CheckField not = blank, documentclose() +
documentopen ("Customers"), blank)
```

Important Notes

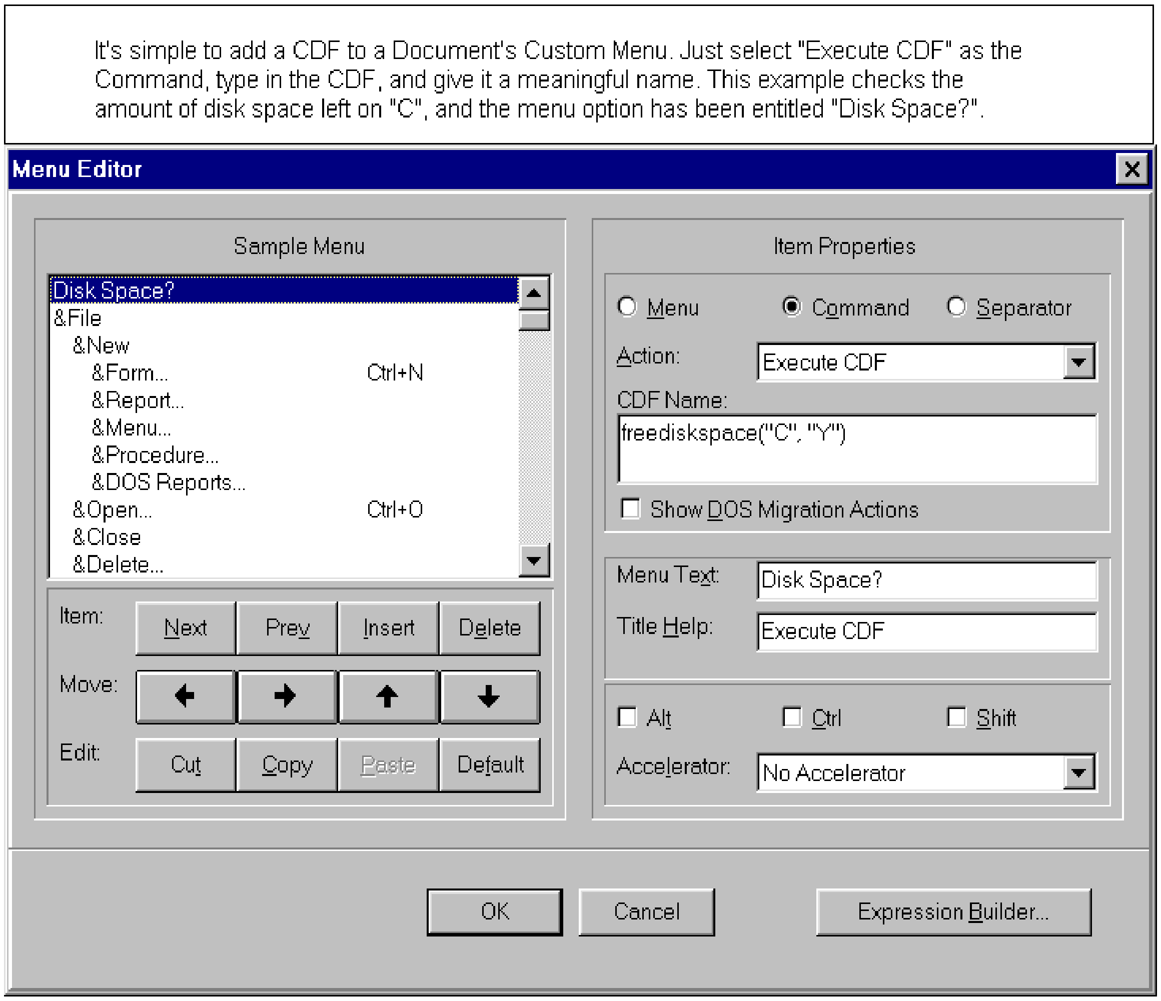
Inside the button/image “CDF Name” box, you can join Multiple Functions together with a “+” between them, as shown above.

You are allowed a maximum of 254 characters inside the button/image “CDF name” box. If you type more than 254 characters, these characters will be truncated when you save.

The button/image expression builder (the pick-lists of available commands) does NOT list CDFs, nor does it list field names in Subforms – so you have to have a good memory, or better yet a written list of the commands and SubForm field names you may want to use.

CDF Examples: Using CDFs in Custom Menus

Custom Functions can add functionality to a Document's Custom Menu, as shown below.



It's simple to add a CDF to a Document's Custom Menu. Just select "Execute CDF" as the Command, type in the CDF, and give it a meaningful name. This example checks the amount of disk space left on "C", and the menu option has been entitled "Disk Space?".

Menu Editor

Sample Menu

```
Disk Space?
&File
  &New
    &Form...            Ctrl+N
    &Report...
    &Menu...
    &Procedure...
    &DOS Reports...
  &Open...              Ctrl+O
  &Close
  &Delete...
```

Item Properties

Menu [✓ Command] Separator

Action: Execute CDF

CDF Name: freediskspace("C", "Y")

☐ Show DOS Migration Actions

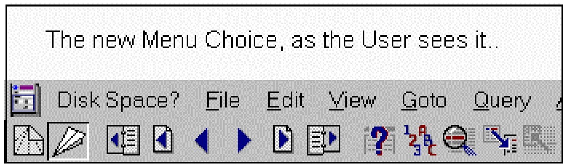
Menu Text: Disk Space?

Title Help: Execute CDF

☐ Alt ☐ Ctrl ☐ Shift

Accelerator: No Accelerator

Item: Next Prev Insert Delete
Move:
Edit: Cut Copy Paste Default
OK Cancel Expression Builder...

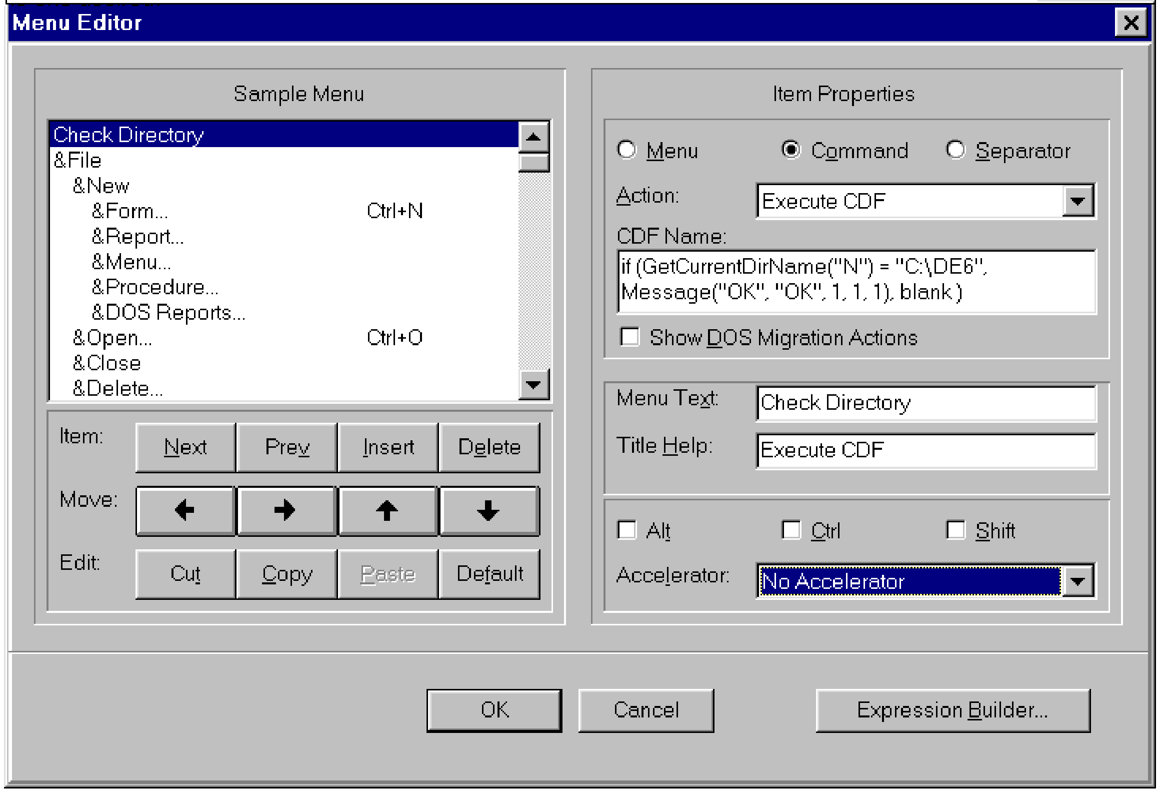


The new Menu Choice, as the User sees it..

Disk Space? File Edit View Goto Query

Just as with Buttons, you can run several CDF's from the same menu item by joining the CDF's with a "+" sign, and you can also use Conditional Logic, as in the example below:

This Menu Item is using conditional logic and two Custom Functions. GetCurrentDirName returns the name of the current directory, while Message sends a message box to the screen. So this command uses one CDF to check the current directory, and then another CDF to send a message to the user, if the current directory is other than the one desired.



( ) Menu [✓ Command] ( ) Separator

Action: Execute CDF

CDF Name:

```
if (GetCurrentDirName("N") = "C:\DE6",
Message("OK", "OK", 1, 1, 1), blank )
```

☐ Show DOS Migration Actions

Menu Text: Check Directory

Title Help: Execute CDF

☐ Alt ☐ Ctrl ☐ Shift

Accelerator: No Accelerator

CDF Examples: Using CDFs in Field Derivations

Here we see a CDF function used in a field derivation. This example uses a function called **Rounding**, which is one of the CDF's included with DataEase. To understand the example, you first need to understand what the **Rounding** function does. As the name suggests, Rounding rounds a number. You pass two parameters to the function, these being:

1. The value to be rounded.
2. The decimal number to round to.

So for example….

```
Rounding(12.716, .5)
```

…would round to 13. While the formula shown below..

```
Rounding(12.716, .8)
```

…would round to 12.

The Rounding CDF is often used to force DataEase to either round .5 **down**, or to round it **up** – whichever is desired.

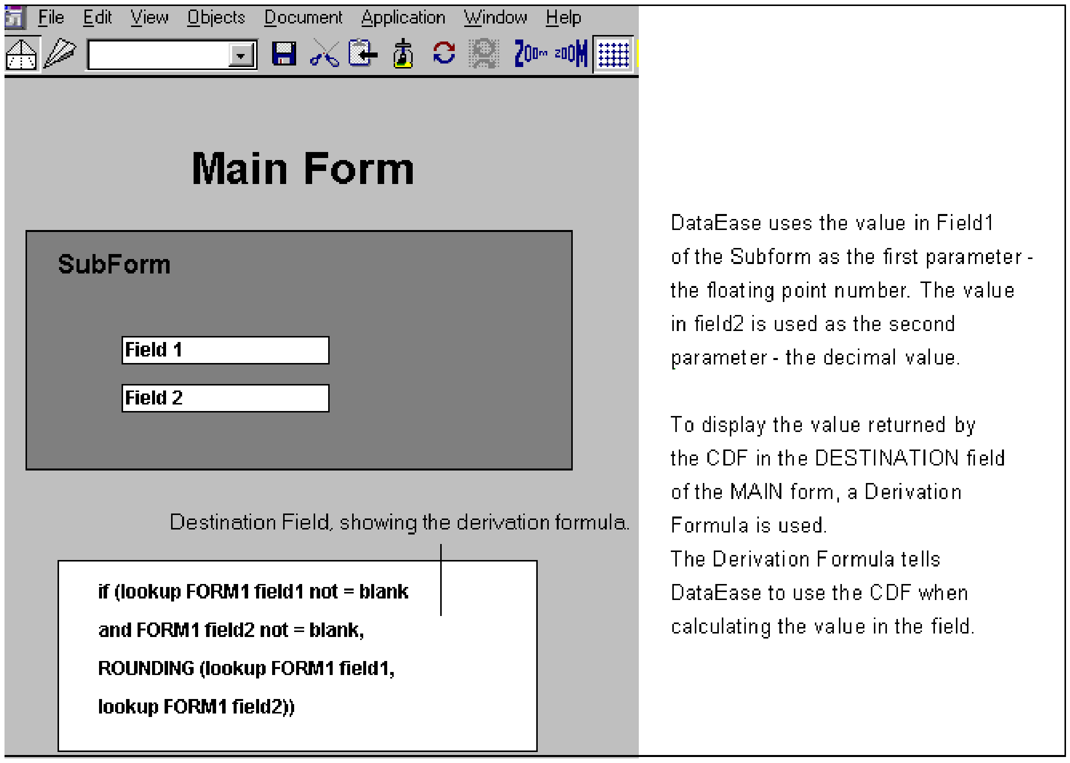
Running the Rounding CDF

To execute the Rounding CDF from a Derivation or Validation Formula, the formula might read:

```
If (float_field not = blank and Decimal_field not = blank, ROUNDING
(float_field, Decimal_field), blank)
```

...where float_field and Decimal_field may be any DataEase fields accessible to the form, or actual values. The test for blank values prevents the CDF from executing unless the two values being passed actually contain data.

The figure below illustrates how the sample CDF might be used in an application.



Destination Field, showing the derivation formula.

```
if (lookup FORM1 field1 not = blank
and FORM1 field2 not = blank,
ROUNDING (lookup FORM1 field1,
lookup FORM1 field2))
```

DataEase uses the value in Field1 of the Subform as the first parameter - the floating point number. The value in field2 is used as the second parameter - the decimal value.

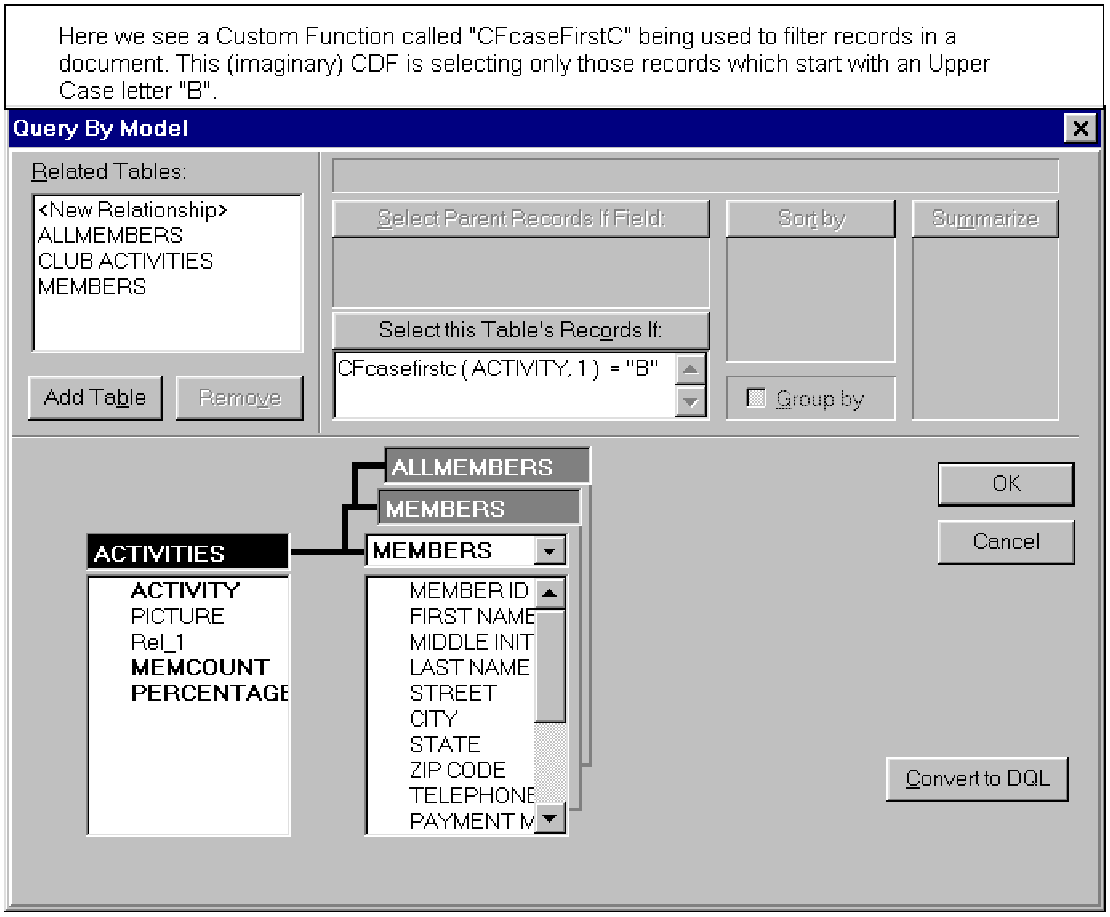
To display the value returned by the CDF in the DESTINATION field of the MAIN form, a Derivation Formula is used.
The Derivation Formula tells DataEase to use the CDF when calculating the value in the field.

For example, enter 56.75 in Field1 and .8 in Field2. Since the fractional point portion of the floating point number, .75, is less than the decimal value of 0.8, the Rounding function rounds down and returns the next lowest whole number - 56.

CDF Examples: Using CDFs in QBM and Procedures

QBM Filters

Custom Functions can be used in a QBM Record Selection Filter in exactly the same way as one of the normal DataEase built-in functions.



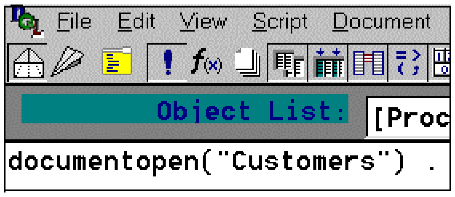
Here we see a Custom Function called "CFcaseFirstC" being used to filter records in a document. This (imaginary) CDF is selecting only those records which start with an Upper Case letter "B".

Select this Table's Records If:

```
CFcasefirstc ( ACTIVITY, 1 )  = "B"
```

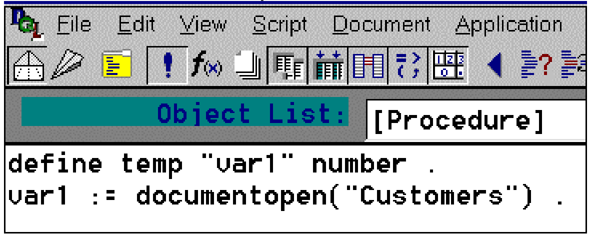
CDFs in Procedures

Custom Functions can be used inside a DQL in exactly the same way as one of the built-in DataEase functions. But remember that when a Custom Function returns a value, it must have somewhere to return that value to. Look at the example below.



```
documentopen("Customers") .
```

This looks like an acceptable Custom Function call….but it won't work.



```
define temp "var1" number .
var1 := documentopen("Customers") .
```

The example above does work, and the "Customers" form will be opened as requested. The temporary variable "Var1" exists only to hold the function's return value. It has no other purpose, and takes no further part in the procedure.

Fast Reporting

It is not a good idea to combine selection criteria with Custom Functions in a QBM filter or a DQL query, because in this scenario DataEase will not use an existing index. For example, in the query below;

```
for tblMyTable with (SomeField = GetGlobal(124)) ;
[do something]
end .
```

…any Index on **SomeField** would be ignored.

This limitation is easily overcome by the use of a temporary variable, as in;

```
Define temp "tSelectOn" Text 40.
tSelectOn := GetGlobal(124) .
for tblMyTable with (SomeField = tSelectOn) ;
[do something]
end .
```

…which will allow the index on **SomeField** to be used properly.

Custom Defined Function Library (CDFS2) version 1.01.009

Overview

The CDFS2 library provides a range of useful add-on functions which can be used to enhance your applications. The following functions are available in this version:

String Functions

StrCreate

StrDelete

StrInsert

StrNumbersOnly

StrOverwrite

NumToText

Memory functions

SetArray

GetArray

Miscellaneous

ShellAndWait

InitProgressBar

SetProgressBar

Clipboard functions

GetClipText

SetClipText

ClipClear

Dialog Functions

ShowAboutBox

GetPassword

INI file functions

IniReadString

IniWriteString

Other file functions

TextOut

TextOutLn

TextIn

FileExists

FileCopy

FileDelete

FileRename

SetUneek

GetUneek

MakeDir

RemoveDir

<u>CDFS2 Library Revisions</u>

1.01.007 January 1997

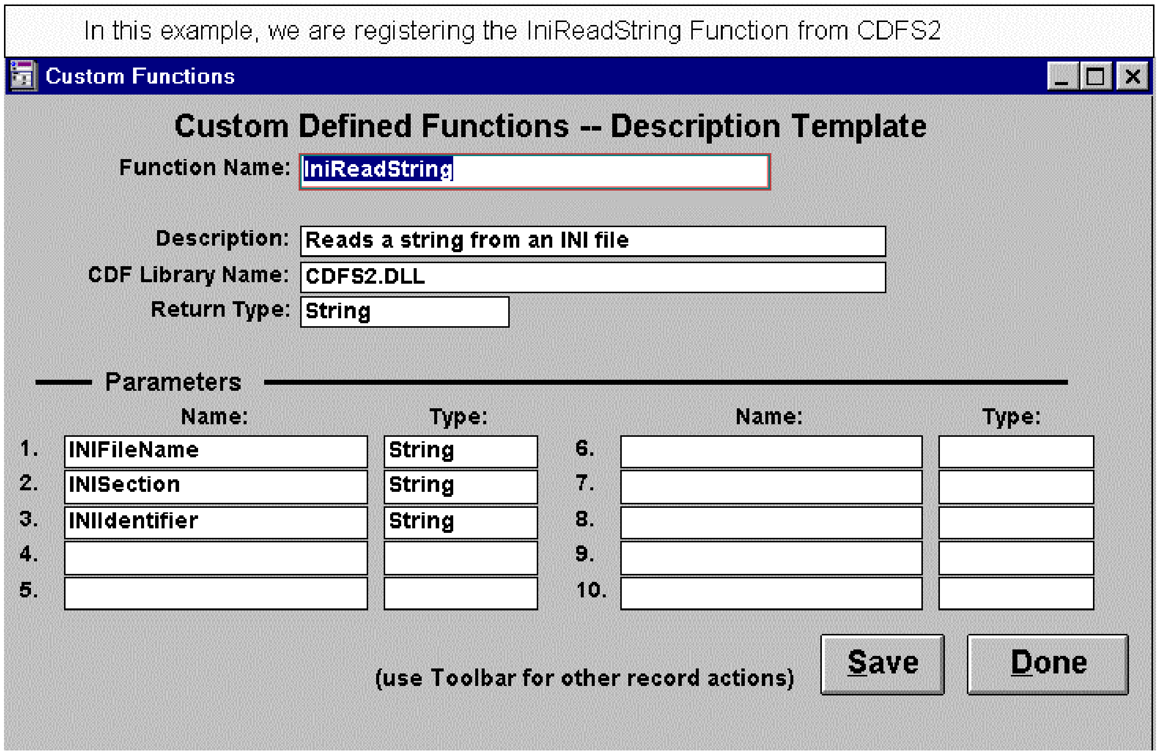
1.01.008 June 1997

1.01.009 July 2002 (32 Bit version)

CDFS2.DLL Installation

Before you can use a Custom Defined Function, you must do two things:

1. Make sure that the CDF Library file is available to the application. In the case of CDFS2.DLL, you must either copy the CDFS2.DLL file to the application directory (i.e. where the .DBM files are stored) or ensure that it is held in the DataEase program directory.
2. Each application that uses a given function must register that function's details in the DataEase Custom Defined Functions Table. Note that you only need define those functions that you intend to use. For example, if you only intend using three of the twenty nine functions held in CDFS2, then you need only register those three functions.


In this example, we are registering the IniReadString Function from CDFS2

Custom Functions

Custom Defined Functions -- Description Template

Function Name: IniReadString

Description: Reads a string from an INI file

CDF Library Name: CDFS2.DLL

Return Type: String

Parameters

	Name:	Type:		Name:	Type:
1.	INIFileName	String	6.		
2.	INISection	String	7.		
3.	INIIdentifier	String	8.		
4.			9.		
5.			10.		

(use Toolbar for other record actions)

As with all the CDF's included with DataEase, the quickest way to register a CDF in your own application is to cut and paste the appropriate CDF function from the CDF Example Application. The process is as follows:

Open the sample CDF application - called **CDF Library** - which comes with DataEase. (The default path is DE6\Samples\CDFLibs\CDF Library). Open the Main Menu, press the button on the bottom of the screen labeled **'CDFs SYSTEM FORM'**. Press F3 until you find the function you want to use in your own application. From the menu bar select **Edit>>Copy Record**. Exit the Sample CDF application and access your own application. Open the CDFs System Form, **Paste** the record, and then save it.

SetArray function

Description

This function sets the value of the specified element of a small, persistent 10 element text array, each element being able to contain a string of up to 255 characters. If N is less than 1 or more than 10, the function call is ignored. The array values will persist until reset by a further call to SetArray() or the DLL is unloaded (e.g. the calling application terminates). The function always returns 0. The SetArray function is always used in conjunction with the GetArray function, which allows you to retrieve values placed in the array.

Declaration

SetArray(Element, Value)

Parameters

Name	Type	Description
Element	integer	in the range 1..10
Value	string	a string value up to 255 characters in length

Return Type

Integer

Example

```
  i := SetArray(3,"Colin Davies")
```

GetArray Function

Description

This function returns the string value of the specified element of a small, persistent 10 element text array, each element being able to contain a string of up to 255 characters. If N is less than 1 or more than 10, a null string is returned. The array values will persist until reset by SetArray or the DLL is unloaded (e.g. the calling application terminates).

Declaration

GetArray(Element)

Parameters

Name	Type	Description
Element	integer	in the range 1..10

Return Type

String

Example

MyVariable `:= GetArray(6)`

NumToText()

Description

This function will return a formatted number as a string.

Declaration

NumToText(Num, Format)

Parameters

Name	Type	Description
Num	double	The number to be formatted (any format)
Format	string	The format mask (see below)

Return Type

String

Format Mask

The following format specifiers are supported in the format parameter:

Specifier	**Represents**
0	Digit placeholder. If the value being formatted has a digit in the position where the '0' appears in the format, then that digit is copied to the returned string. Otherwise, a '0' is stored in that position in the returned string.
#	Digit placeholder. If the value being formatted has a digit in the position where the '#' appears in the format, then that digit is copied to the returned string. Otherwise, nothing is stored in that position in the returned string.
.	Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value; any additional '.' characters are ignored. The actual character used as the decimal separator in the output string is determined by the DecimalSeparator global variable. The default value of DecimalSeparator is specified in the Number Format of the International section in the Windows Control Panel.
,	Thousand separator. If the format string contains one or more ',' characters, the output will have thousand separators inserted between each group of three digits to the left of the decimal point. The placement and number of ',' characters in the format string does not affect the output, except to indicate that thousand separators are wanted. The actual character used as the thousand separator in the output is determined by the ThousandSeparator global variable. The default value of ThousandSeparator is specified in the Number Format of the International section in the Windows Control Panel.
E+	Scientific notation. If any of the strings 'E+', 'E-', 'e+', or 'e-' are

	contained in the format string, the number is formatted using scientific notation. A group of up to four '0' characters can immediately follow the 'E+', 'E-', 'e+', or 'e-' to determine the minimum number of digits in the exponent. The 'E+' and 'e+' formats cause a plus sign to be output for positive exponents and a minus sign to be output for negative exponents. The 'E-' and 'e-' formats output a sign character only for negative exponents.
'xx'	Characters enclosed in single quotes are output as-is, and do not affect formatting.
;	Separates sections for positive, negative, and zero numbers in the format string.

The locations of the leftmost '0' before the decimal point in the format string and the rightmost '0' after the decimal point in the format string determine the range of digits that are always present in the output string.

The number being formatted is always rounded to as many decimal places as there are digit placeholders ('0' or '#') to the right of the decimal point. If the format string contains no decimal point, the value being formatted is rounded to the nearest whole number.

If the number being formatted has more digits to the left of the decimal separator than there are digit placeholders to the left of the '.' character in the format string, the extra digits are output before the first digit placeholder.

To allow different formats for positive, negative, and zero values, the format string can contain between one and three sections separated by semicolons.

One section:	The format string applies to all values.
Two sections:	The first section applies to positive values and zeros, and the second section applies to negative values.
Three sections:	The first section applies to positive values, the second applies to negative values, and the third applies to zeros.

If the section for negative values or the section for zero values is empty, that is if there is nothing between the semicolons that delimit the section, the section for positive values is used instead.

If the section for positive values is empty, or if the entire format string is empty, the value is formatted using general floating-point formatting.

Examples

Example numbers and their resulting formats...

Format string	**1234**	**-1234**	**0.5**	**0**
--				
	1234	-1234	0.5	0
0	1234	-1234	1	0
0.00	1234.00	-1234.00	0.50	0.00
#.##	1234	-1234	.5	
#,##0.00	1,234.00	-1,234.00	0.50	0.00
#,##0.00;(#,##0.00)	1,234.00	(1,234.00)	0.50	0.00
#,##0.00;;'Zero'	1,234.00	-1,234.00	0.50	Zero
0.000E+00	1.234E+03	-1.234E+03	5.000E-01	0.000E+00
#.###E-0	1.234E3	-1.234E3	5E-1	0E0

IniReadString Function

Description

This function returns the value from the specified INI file section and identifier as a string. For example: Assume an INI file named 'C:\TEMP\LAST.INI' containing the following entries:

```
[SIGNON]
LastUser=Fred
SignOnDate=12/09/99
SignOnTime=10:00:00

[USAGE]
Fred=14
Frank=176
Harold=2

[OVERNIGHT_RUN]
LastRun=10/09/99
NoInvoicesGenerated=27
NoFaxesSent=152
```

In this case, the function call (in pseudo-syntax):

```
S := IniReadString("C:\TEMP\LAST.INI", "Usage", "Frank")
```

would return the value "176" into the string variable **s**.

Declaration

IniReadString(Filename,Section,Identifier)

Parameters

Name	Type	Description
Filename	string	Fully qualified file path and name of required INI file
Section	string	The name of the INI file section required, without leading and trailing [] characters. If no filepath is specified then the \WINDOWS directory is assumed
Identifier	string	The name of the INI file identifier section required, without trailing = character

Return Type

string

IniWriteString Function

Description

This function sets the value in the specified INI file section and identifier. If the specified file does not yet exist, it will be created automatically. If the Section and Identifier exist already then the identifier's current value will be modified. If the Section and/or Identifier do not exist, then they will be added automatically. If the function was unsuccessful for any reason (e.g. invalid filename), it will return a non-zero value, otherwise it will return zero.

For example: assume an INI file named 'C:\APP\LAST.INI' containing the following entries:

```
[SIGNON]
LastUser=Fred
SignOnDate=12/09/99
SignOnTime=10:00:00
[USAGE]
Fred=14
Frank=176
Harold=2

[OVERNIGHT_RUN]
LastRun=10/09/99
NoInvoicesGenerated=27
NoFaxesSent=152
```

In this case, after the following function calls (in pseudo-syntax):

```
i := IniWriteString("C:\TEMP\LAST.INI", "Usage", "Frank","183")
i := IniWriteString("C:\TEMP\LAST.INI", "Usage", "Bill","3")
i := IniWriteString("C:\TEMP\LAST.INI", "NEW_SECTION",
"Budget","40,000")
```

the INI file would now look like:

```
[SIGNON]
LastUser=Fred
SignOnDate=12/09/99
SignOnTime=10:00:00

[USAGE]
Bill=3          <-- new value added
Fred=14
Frank=183       <-- existing value updated
Harold=2
```

```
 [OVERNIGHT_RUN]
LastRun=10/09/99
NoInvoicesGenerated=27
NoFaxesSent=152

 [NEW_SECTION]    <-- whole new section added
Budget=40,000
```

Declaration

IniWriteString(Filename, Section, Identifier, Value)

Parameters

Name	Type	Description
Filename	string	Fully qualified file path and name of required INI file
Section	string	The name of the INI file section required, without leading and trailing [] characters. If no file path is specified then the current WINDOWS directory is assumed
Identifier	string	The name of the INI file identifier section required, without trailing = character
Value	string	The value that you wish to set the Identifier to.

Return Type

integer

TextOut function

Description

This function will append the specified text onto the end of the specified file. If the file does not exist, it will be created automatically. If there is any file I/O problem during the function call (e.g. 'invalid file name', 'disk full') it will return -1, otherwise it will return 0. Possibly useful for creating non-standard ASCII export files. Please note that that this is probably not as fast a function as it could be since (for safety) each individual call has the overhead of opening, appending, then closing the file.

Declaration

TextOut(Filename, Str)

Parameters

Name	Type	Description
Filename	string	The full filepath, filename and extension of the text file that is to receive the appends.
Str	string	The string to be appended onto the file *Filename*

Return Type

Integer

Example

The following calls:

```
TextOut("C:\TEMP\LOG.DAT","AAAAAA|");
TextOut("C:\TEMP\LOG.DAT","BBBBBB|");
TextOutLn("C:\TEMP\LOG.DAT","CCCC");
TextOut("C:\TEMP\LOG.DAT","DDDDDD|");
TextOut("C:\TEMP\LOG.DAT","EEEEEE|");
TextOutLn("C:\TEMP\LOG.DAT","FFFF");
```

will create, or append to, the file C:\TEMP\LOG.DAT :

```
AAAAAA|BBBBBB|CCCC
DDDDDD|EEEEE|FFFF
```

TextIn function

Description

This function will read and return the NTH line of the specified text file as a string. If the line is longer than 255 characters it will be truncated. If the file does not exist, or there is any other file I/O problem during the function call, it will return a null string..

Declaration

TextOut(Filename, N)

Parameters

Name	Type	Description
Filename	string	The full filepath, filename and extension of the text file that is to be read.
N	integer	The physical line number of the text file to be read.

Return Type

String

Example

Given the text file C:\TEMP\ERRMSGS.DAT containing the following lines:

```
File not found
Invalid drive
Invalid file format
Disk full
```

the following call:

```
s := TextIn("C:\TEMP\ERRMSGS.DAT",3)
```

will return in `s` a value of :

```
"Invalid file format"
```

StrCreate function

Description

This function will return a string of identical characters of the specified length.

Declaration

StrCreate(Char, N)

Parameters

Name	Type	Description
Char	string	The character that will be repeated N times. If this string contains more than one character, only the first character will be used
N	integer	Must be between 1 and 255 (will be automatically adjusted to fall within this range if necessary)

Return Type

String

Example

The following call will return to the string `s` the value: `"XXXXXXXXXXXXXXXXXXXX"`

```
s := StrCreate("X",20)
```

StrInsert function

Description

This function will insert the specified substring at the specified character position, shifting existing characters to the right and return the resulting string.

Declaration

StrInsert(Str, SubStr, Position)

Parameters

Name	Type	Description
Str	string	The source string
SubStr	string	The substring to be inserted into *Str*
Position	integer	The character position at which to insert *SubStr*. Must be between 1 and 254 (will be automatically adjusted to fall within this range if necessary). if the resulting string is longer than 255 characters it will be truncated.

Return Type

String

Example

The following call will return, in the string `s`, the value: `"John Winston Lennon"`

```
s := StrInsert("John Lennon","Winston",6)
```

StrOverwrite function

Description

This function will insert the specified substring in the source string at the specified character position *overwriting* the existing characters.

Declaration

StrOverwrite(Str, SubStr, Position)

Parameters

Name	Type	Description
Str	string	The source string. Must be <= 255 characters in length.
SubStr	string	The substring to be overwritten into *Str.*
Position	integer	The character position at which to overwrite *SubStr*. Must be between 1 and the length of *Str* - 1) - will be automatically adjusted to fall within this range if necessary.

Return Type

String

Example

The following call will return, in the string `s`, the value: `"John Stephen Lennon"`

```
  s := StrOverwrite("John Winston Lennon","Stephen",6)
```

StrDelete function

Description

This function will delete the specified number of characters from a string, starting at a specified character position and return the resulting string.

Declaration

StrDelete(Str, Position, Count)

Parameters

Name	Type	Description
Str	string	The source string
Pos	integer	The character position from which the deletion will take place. Must be between 1 and 255 (will be automatically adjusted to fall within this range if necessary). If *Pos* is greater than the number of characters in *Str*, then no deletion takes place
Count	integer	The number of characters to be deleted. Must be between 1 and 255 (will be automatically adjusted to fall within this range if necessary). If *Pos* + *Count* is greater than the number of characters in *Str*, then all characters from character position *Pos* are deleted.

Return Type

String

Example

The following call will return, in the string `s`, the value: `"John Lennon"`

```
s := StrDelete("John Winston Lennon",6,8)
```

FileExists function

Description

This function returns 0 if the file does not exist, otherwise it returns 1.

Declaration

FileExists(FileName)

Parameters

Name	Type	Description
FileName	string	The full path, name and extension of the specified file.

Return Type

integer

StrNumbersOnly function

Description

This function will accept a string of mixed characters and return a string with all non-numeric characters stripped out.

Declaration

StrNumbersOnly(Str)

Parameters

Name	Type	Description
Str	string	The source string

Return Type

String

Example

The following call will return, in the string `s`, the value: `"01713335555"`

```
s := StrNumbersOnly("0171-333-5555")
```

GetClipText function

Description

This function will return any text currently stored in the Windows clipboard, truncated to 255 characters. Clipboard data stored in any other format apart from text will be ignored.

Declaration

GetClipText

Parameters

Name	Type	Description
None		

Return Type

String

Example

```
  s := GetClipText .
```

SetClipText function

Description

This function will replace the current contents of the Windows clipboard with the specified text. The function always returns an integer value of 0.

Declaration

SetClipText (Value)

Parameters

Name	Type	Description
Value	String	The text value to be stored on the clipboard

Return Type

integer

Example

```
  i := SetClipText('This is just atext - OK?') .
```

ClipClear function

Description

This function will clear the current contents of the Windows clipboard. It always returns 0.

Declaration

ClipClear

Parameters

Name	Type	Description
None		

Return Type

Integer

Example

```
  i := ClipClear .
```

ShowAboutBox function

Description

This function will display a modal 'About' box.

Declaration

ShowAboutBox (Product, Version, Copyright, Comments)

Parameters

Name	Type	Description
Product	String	Any text up to 50 characters
Version	String	Any text up to 50 characters
Copyright	String	Any text up to 50 characters
Comments	String	Any text up to 255 characters

Return Type

Integer

Example

```
i := ShowAboutBox(       "CDFS2 Dynamic Link Library",
                         "Version 1.01",
                         "Colin Davies 1996",
                         "General purpose library of string, file, dialog and
clipboard functions") .
```

GetPassword function

Description

This function will display a modal window requesting a password to be entered. If the user enters a password then clicks OK, the entered password is compared to the Password parameter (case-insensitive) and the function returns 1 if it matches or 0 if it doesn't. If the user clicks Cancel then the function always returns 0 .

Declaration

GetPassword (Password)

Parameters

Name	Type	Description
Password	String	Any text up to 20 characters

Return Type

Integer

Example

```
if GetPassword("Colin") = 0 then
  .... Show message "Password incorrect!"
else
  .... other processing
```

TextOutLn

Description

This function will append a line of text plus a CR/LF onto the end of the specified file. If the file does not exist, it will be created automatically. If there is any file I/O problem during the function call (e.g. 'invalid file name', 'disk full') it will return -1, otherwise it will return 0. Useful for creating a text log 'on the fly' for later debugging.

Declaration

TextOutLn(Filename, Str)

Parameters

Name	Type	Description
Filename	string	The full filepath, filename and extension of the text file that is to receive the appends.
Str	string	The string to be appended onto the file *Filename*

Return Type

Integer

Example

The following call:

```
TextOut("C:\TEMP\LOG.DAT","Customer name = JONES")
```

will update the file C:\TEMP\LOG.DAT from:

```
Customer name = ICI
Customer name = HARRODS
```

to

```
Customer name = ICI
Customer name = HARRODS
Customer name = JONES
```

FileCopy

Description

This function returns 0 if the file copy operation was successful, otherwise it returns 1.

Declaration

FileCopy(FromFileName, ToFileName)

Parameters

Name	Datatype	Description
FromFilename	string	The full path, name and extension of the source file
ToFileName	string	The full path, name and extension of the destination file.

Return Type

integer

SetUneek function

Description

Seeds the long integer value held in an external binary file that will later be called by **GetUneek**. If the specified file does not exist, it will be created automatically. The resultant file will be exactly 4 bytes in length. If, for any reason, the file cannot be updated or created or there is any other file I/O error, the function will return -1, otherwise it will return 0.

For example, to seed the value of 1000 in the file C:\TEMP\INVOICE.BIN:

```
i := SetUneek("C:\TEMP\INVOICE.BIN",1000)
```

Declaration

SetUneek(FileName, Value)

Parameters

Name	Type	Description
FileName	string	The full path, name and extension of the external file that will hold the next 'uneek' number.
Value	long integer	The value to set the next 'uneek' number to.

Return Type

Integer

GetUneek function

Description

Retrieves the long integer value held in the external binary file specified in *Filename* and then increments it by *Increment.* If, for any reason, the file cannot be read or updated or there is any other file I/O error, the function will return -1, otherwise it will return the 'uneek' value.

For example, given an external file named C:\TEMP\INVOICE.BIN, and containing a 'uneek' value of 1000, the following calls would return successive values of 1000,1005 and 1010 into the variable `i`:

```
i := GetUneek("C:\TEMP\INVOICE.BIN",5)   <-- returns 1000
i := GetUneek("C:\TEMP\INVOICE.BIN",5)   <-- returns 1005
i := GetUneek("C:\TEMP\INVOICE.BIN",5)   <-- returns 1010
```

Declaration

GetUneek(FileName, Increment)

Parameters

Name	Type	Description
FileName	string	The full path, name and extension of the external binary file that holds hold the next 'uneek' number.
Increment	integer	The value by which the 'uneek' value held in the file will be incremented after a successful 'get'.

Return Type

Long integer

FileDelete

Description

This function returns 0 if the file does not exist, otherwise it returns 1.

Declaration

FileDelete(FileName)

Parameters

Name	Datatype	Description
Filename	string	The full path, name and extension of the file to be deleted

Return Type

integer

FileRename

Description

This function returns 0 if the FileRename operation was successful, otherwise it returns 1.

Declaration

FileRename(OldFileName, NewFileName)

Parameters

Name	Datatype	Description
OldFilename	string	The full path, name and extension of the file to be renamed
NewFileName	string	The full path, name and extension of the new file

Return Type

integer

MakeDir

Description

This function returns 0 if the directory was successfully created, otherwise it returns -1.

Declaration

MakeDir(DirectoryName)

Parameters

Name	Type	Description
DirectoryName	string	The full directory path with no trailing '\'

Return Type

Integer

Example

```
i := MakeDir("C:\TEMP\INSTALL") .
```

RemoveDir

Description

This function returns 0 if the directory was successfully removed, otherwise it returns -1.

Declaration

RemoveDir(DirectoryName)

Parameters

Name	Type	Description
DirectoryName	string	The full directory path with no trailing '\'

Return Type

Integer

Example

```
i := RemoveDir("C:\TEMP\INSTALL") .
```

ShellAndWait function

Description

This function will **suspend** your application then attempt to execute the specified executable, along with any specified parameters. Only when the shelled program has finished and exited will your application be reactivated. This applies to shelling Windows or DOS programs. The function returns 32 if the shelled program could be executed, otherwise it returns one of the following values:

Error Code	Description
0	System was out of memory, executable file was corrupt, or relocations were invalid.
2	File was not found.
3	Path was not found.
5	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
6	Library required separate data segments for each task.
8	There was insufficient memory to start the application.
10	Windows version was incorrect.
11	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
12	Application was designed for a different operating system.
13	Application was designed for MS-DOS 4.0.
14	Type of executable file was unknown.
15	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
16	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
19	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
20	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
21	Application requires 32-bit extensions.

Declaration

ShellAndWait(Program, ShowMode)

Parameters

Name	Type	Description
Program	string	The full path, name and extension of the program file plus any parameters
ShowMode	integer	determines the window style of the shelled program: 0 - Hidden (**be careful** to ensure that the shell program will definitely complete otherwise you will have no way of closing the shelled program and your application will remain suspended - doesn't work for DOS windows)

1 - Restored

2 - Minimised

3 - Maximised

Return Type

integer

Example

```
i := ShellAndWait("PKUNZIP C:\IMPORTS\ASCII.ZIP
C:\App\temp\imports\*.*",2)
```

This example will suspend your application, launch PKUNZIP in a DOS window, then exit and awaken your application when it has completed. Note that using just the WINDOWS API function WinExec() would launch the above DOS program too, but your application code would *continue* to run even though the launched process had not yet finished. If, continuing the above example, your application needed to wait for the PKUNZIP to finish in order to process the decompressed ASCII import files then WinExec() would not work but ShellAndWait() would.

InitProgressBar function

Description

Initialises the progress bar with its window caption, gauge label and maximum (100%) value. The minimum value is always 0. You may only have one progress bar active at any one time. You can adjust the height and width, within limits, of the progress bar dialog by setting the Height and Width parameters. The limits are:

Height	150 - 600
Width	432 - 800

Height & Width values outside these limits will be trimmed accordingly.

NB: You **must always ensure** that, at some logical point, your code destroys the progress bar by calling the **KillProgressBar** function

Declaration

InitProgressBar

Parameters

Name	Type	Description
MaxValue	integer	The value representing 100% progress
Caption	string	The progress bar window title
Description	string	The text of the label that appears just above the progress bar gauge
Height	integer	The height of the progress bar dialog in pixels. Use 0 for default minimum height
Width	Integer	The width of the progress bar dialog in pixels. Use 0 for default minimum width

Return Type

Integer

Example (DFW)

```
Define "X" number .
X := InitProgressBar(CountOf(Customers), "Customer Mailshot",
"Compiling mailshot, please wait...",0,0) .
  for CUSTOMERS ;
  .. do some processing ..
    X := SetProgressBar(current item number, jointext("Processing " +
current item number))
  end
X := KillProgressMeter()
```

SetProgressBar function

Description

Sets the progress and gauge label of an existing (initialised) progress bar. The length of the progress bar and the percentage figure displayed inside the bar is determined by the calculation *Progress / MaxValue * 100* where *MaxValue* was set using the **InitProgressBar** function. This function returns 1 if successful, 0 if not (e.g. if progress bar not initialised).

Declaration

SetProgressBar

Parameters

Name	Type	Description
Progress	integer	The value representing current % progress
Description	string	The text of the label that appears just above the progress bar gauge

Return Type

Integer

Example

```
Define "X" number .
X := InitProgressBar(CountOf(Customers), "Customer Mailshot",
"Compiling mailshot, please wait...") .
  for CUSTOMERS ;
  .. do some processing ..
    X := SetProgressBar(current item number, jointext("Processing " +
current item number))
  end
X := KillProgressMeter()
```

KillProgressBar function

Description

Destroys the current progress bar and recoups the memory it used.

NB: You **must always ensure** that, at some logical point, your code destroys the progress bar by calling this function.

Declaration

KillProgressBar

Parameters
none

Return Type

Integer

Example

```
Define "X" number .
X := InitProgressBar(CountOf(Customers), "Customer Mailshot",
"Compiling mailshot, please wait...") .
  for CUSTOMERS ;
  .. do some processing ..
    X := SetProgressBar(current item number, jointext("Processing " +
current item number))
  end
X := KillProgressMeter()
```

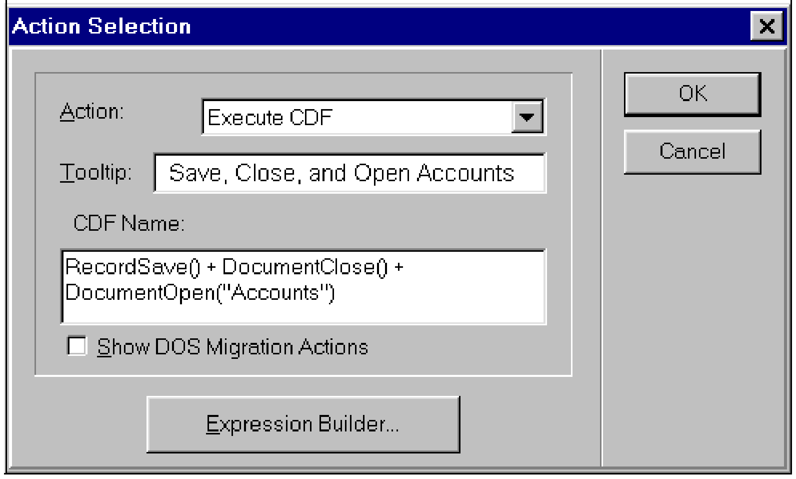
DFW Acts Library: Introduction

DFWActs is an unusual and extremely powerful Library. While other CDF libraries add new functions to DataEase, DFWActs allows you to access the existing **commands** inside DataEase – but in ways which DataEase normally wouldn't allow.

Example: A DataEase button or picture object allows you to carry out one 'Action'. It can "Save Record", or "Clear Form", and so on. Obviously very powerful, but you're stuck with one action per button. But DFWActs enhances this considerably. Using them you can:

1) Run multiple DataEase Actions on a button.
2) Use conditional statements.
3) Pass derived parameters to the Action.
4) Run any combination of the above on a single button.

Here is an example of a DataEase Button running multiple actions, using three of the CDFs from DFWActs -" RecordSave", to save the current record, "DocumentClose", to close the current document, and "DocumentOpen" to open a specified new document.



Action: Execute CDF

Tooltip: Save, Close, and Open Accounts

CDF Name:

```
RecordSave() + DocumentClose() +
DocumentOpen("Accounts")
```

☐ Show DOS Migration Actions

Important Notes

Inside the button/picture "CDF Name" box, you can join Multiple Functions together with a "+" between them, as shown above.

You are allowed a maximum of 254 characters inside the button/image "CDF name" box. If you type more than 254 characters, these characters will be truncated when you save.

The button/picture expression builder (the pick-lists of available commands) does NOT list CDFs – so if you have to have a good memory, or better yet a written list of the commands you may want to use.

DFW Acts Function List

The full list of available commands is shown below. Click on a name to see its description.

AccessAllRecords	AccessNoRecord	AccessReadOnly
AddFilterAND	AddFilterOR	AppDelete
ApplicationBackup	ApplicationClose	ApplicationInstall
ApplicationLock	ApplicationRestore	ApplicationUnlock
AppMigrate	AppNew	AppOpen
AppPreferences	AppRename	ClearQueryFilter
ClearSelectionFilter	ClearSort	CopyRecord
CopySelected	CopySpecial	CustomFunctions
CutSelected	DatabaseLinks	DataExport
DataImport	DesignerView	DocumentClose
DocumentDelete	DocumentOpen	DocumentsInfo
DOSAdminMenu	DOSBackup	DOSCheckDisk
DOSDefineForm	DOSDefineImport	DOSDefineTable
DOSDeleteForm	DOSDeleteImport	DOSDeleteProcedure
DOSDeleteReport	DOSFileList	DOSFormatDisk
DOSFormsMenu	DOSImportMenu	DOSInstallForm
DOSInstallProcedure	DOSLoadProcedure	DOSLoadReport
DOSMainMenu	DOSMaintMenu	DOSOneTimeImport
DOSOSMenu	DOSProcedure	DOSProgramCall
DOSPrompt	DOSQueryMenu	DOSRecordEntry
DOSRecordsMenu	DOSReportFields	DOSReportFormat
DOSReportRecords	DOSReportsMenu	DOSRestore
DOSRunImport	DOSRunProcedure	DOSRunReport
DOSStatus	DOSUserMenu	DOSUtilMenu
DOSViewForm	DOSViewImport	EndPrintToWindow
ExecuteFile	ExitDataEase	FieldClear
FilterClear	FilterSet	FormClear
FormOpenRelated	FormReorganize	HelpAbout
HelpDesktop	HelpDQL	HelpGlossary
HelpHowTo	HelpIndex	HelpMenus
HelpSearch	HelpToolbar	HelpUser
HideWin	IfMaxHide	IfMaxMin
IfMaxNormalize	IfNormalHide	IfNormalMax
IfNormalMin	ImportsInfo	IsWinMax
LookupTo	MaxWin	MinWin
NewForm	NewMenu	NewProcedure
NewReport	NormalizeWin	OLELinks

OpenForm	OpenMenu	OpenProcedure
OpenReport	Paste	PrintDocument
PrinterSetup	PrintPreview	QBMNewReport
RecordDelete	RecordFirst	RecordFirstPos
RecordLast	RecordLastPos	RecordNext
RecordNextPage	RecordNextPagePos	RecordNextPos
RecordPrevious	RecordPreviousPage	RecordPreviousPos
RecordPrevPagePos	RecordRestore	RecordSave
RecordSaveNew	RecordsInfo	RecordsSort
RefreshContinuously	RefreshScreen	RefreshSetTime
Relationships	ReturnDataToDoc	ReturnToParentDoc
SelectionFilter	SelectRecords	ServersInfon
SortAscending	SortDescending	ToggleCatalog
ToggleNormalMax	ToggleSQL	ToggleStatusbar
ToggleToolbar	UndoEdit	Users
ViewAsForm	ViewAsTable	WindowsArrangeIcons
WindowsCascade	WindowsCloseAll	WindowsTileAcross
WindowsTileDown	ZoomBy	ZoomCustom
ZoomFillAcross	ZoomFillDown	ZoomFillWindow
ZoomIn	ZoomNormal	ZoomOut
ZoomPrevious	ZoomTo	

AccessAllRecords

Description

Set locking to **all record** access.

Parameters

None

Return Value

Status. Integer.

Usage

Allow All Records to be Accessed by other users.

DataEase Action(s)

Access All Records

AccessReadOnly

Description

Set locking to **read record** access.

Parameters

None

Return Value

Status. Integer.

Usage

Allow other users to Read all Records.

DataEase Action(s)

Access Read Only

AccessNoRecord

Description

Set locking to **no record** access.

Parameters

None

Return Value

Status. Integer.

Usage

Allow No Records to be Accessed by other users.

DataEase Action(s)

Access No Record

AddFilterAND

Description

ANDS some new criteria to the current selection filter. If there is no current filter, then this becomes the filter.

Parameters

Name	Type	Description
FilterLSide	string	A string representing the additional selection criteria. A space is appended to FilterLSide, if it does not already end with a space.
FilterRSide	string	A string representing the “right hand side” of the additional selection criteria

Return Value

Status. Integer.

Usage

ANDs an additional selection filter to the current filter. If there is no existing filter, it sets this as the filter. A space is appended to FilterLSide if it does not end with a space. An operator **must** be included in **either** FilterLSide or FilterRSide but not in **both**.

Example

AddFilterAND("MyOtherField =", "This one too!")

DataEase Action(s)

Add Filter (AND)

AddFilterOR

Description

ORs current selection filter

Parameters

Name	Type	Description
FilterLSide	string	A string representing the **or**'d selection criteria. A space is appended to FilterLSide, if it does not already end with a space.
FilterRSide	string	A string representing the "right hand side" of the **or**'d selection criteria.

Return Value

Status. Integer.

Usage

Creates an additional **OR**'d selection filter to the current filter. If there is no filter it sets this as the filter. A space is appended to FilterLSide if it does not end with a space. An operator **must** be included in **either** FilterLSide or FilterRSide but not in **both**.

Example

AddFilterOr("MyField =", "or this one")

DataEase Action(s)

Add Filter (OR)

ApplicationClose

Description

Closes application.

Parameters

None

Return Value

Status. Integer.

Usage

Closes the current application.

DataEase Action(s)

Application Close

ApplicationBackup

Description

Opens the application Backup dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Backup the current Application.

DataEase Action(s)

Application Backup

AppRename

Description

Goes to Rename Application dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Change Application Name.

DataEase Action(s)

Application Rename

AppNew

Description

Goes to New Application dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Create new application

DataEase Action(s)

Application New.

AppOpen

Description

Goes to Open Application dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Open existing application

DataEase Action(s)

Application Open

AppPreferences

Description

Display the Application Preferences dialog.

Parameters

None

Return Value

Status. Integer.

Usage

View or Modify Application-Level options.

DataEase Action(s)

Application Preferences

ApplicationRestore

Description

Opens the application Restore dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Restore an Application from a DataEase backup.

DataEase Action(s)

Application Restore

ApplicationInstall

Description

Opens the Install dialog.

Parameters

Name	Type	Description
InstallName	string	A string holding the name of .DIW file which contains a description of the application components to be installed.

Return Value

Status. Integer.

Usage

Install parts of a DataEase Application.

DataEase Action(s)

Application Install >> Enter: Install File (.diw) Name

ApplicationLock

Description

Lock the application.

Parameters

None

Return Value

Status. Integer.

Usage

Ensure that you are the exclusive user of this application.

DataEase Action(s)

Application Lock

AppDelete

Description

Goes to Delete Application dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Delete exiting application

DataEase Action(s)

Application Delete

AppMigrate

Description

Goes to Migrate Application dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Migrate DFD 4.53 application

DataEase Action(s)

Application Migrate

ApplicationUnlock

Description

Unlock the application.

Parameters

None

Return Value

Status. Integer.

Usage

Allow others to use this application.

DataEase Action(s)

Application Unlock

ClearSort

Description

Clear sorting filters.

Parameters

None

Return Value

Status. Integer.

Usage

Clear record sorting filter.

DataEase Action(s)

Clear Sort

ClearSelectionFilter

Description

Clears the selection filter.

Parameters

None

Return Value

Status. Integer.

Usage

Clear the Main record selection filter.

DataEase Action(s)

Clear Selection Filter

ClearQueryFilters

Description

Clears the selection filters

Parameters

None

Return Value

Status. Integer.

Usage

Clears all selection runtime selection filters.

DataEase Action(s)

Clear Query Filters

CutSelected

Description

Deletes selected data from document and places it in the clipboard.

Parameters

None

Return Value

Status. Integer.

Usage

Cut the selected object to the clipboard.

DataEase Action(s)

Cut

CopySelected

Description

Copies selected data to the clipboard.

Parameters

None

Return Value

Status. Integer.

Usage

Copy the selected object to the clipboard.

DataEase Action(s)

Copy

CopyRecord

Description

copies selected record(s)

Parameters

None

Return Value

Status. Integer.

Usage

Copies the selected record to the clipboard.

DataEase Action(s)

Copy Record(s)

CopySpecial

Description

Copies selected data values to the clipboard

Parameters

None

Return Value

Status. Integer.

Usage

Copy the selected record(s) data to the clipboard.

DataEase Action(s)

Copy Special

CustomFunctions

Description

Open the Customer Functions System form.

Parameters

None

Return Value

Status. Integer.

Usage

Register Custom-Defined Functions.

DataEase Action(s)

DatabaseLinks

Description

Open the Database Links dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Define and maintain links to external databases.

DataEase Action(s)

Database Links

DataExport

Description

Exports data.

Parameters

None

Return Value

Status. Integer.

Usage

Displays the export dialog.

DataEase Action(s)

Data Export

DataImport

Description

Imports data.

Parameters

Name	Type	Description
ImportName	string	A character string, enclosed in quotes, representing the name of the import .dbi file you wish to run.

Return Value

Status. Integer.

Usage

Runs an import (.DBI), or displays the import dialog.

DataEase Action(s)

Data Import >> Enter: Import File (.dbi) Name

DocumentsInfo

Description

Run Documents Status.

Parameters

None

Return Value

Status. Integer.

Usage

Display Status of Application Documents.

DataEase Action(s)

Documents Information

DocumentClose

Description

Closes the current document.

Parameters

None

Return Value

Status. Integer.

Usage

Closes the current document.

DataEase Action(s)

Document Close

DocumentDelete

Description

Deletes the specified document or displays the Delete Document dialog

Parameters

Name	Type	Description
DocName	string	A character string, enclosed in quotes, representing the name of the document you wish to delete.

Return Value

Status. Integer.

Usage

Deletes the specified document or, if no document is specified, displays the Delete Document dialog. If the specified document is open it will not be deleted and the system will beep.

DataEase Action(s)

Document Delete >> Enter: Document Name

DocumentOpen

Description

Opens a specified document.

Parameters

Name	Type	Description
DocName	string	A character string, enclosed in quotes, representing the name of the document you wish to open. For example, **DocumentOpen("MyCustomers")**

Return Value

Status. Integer.

Usage

If DocName is not empty the specified document will be opened. If DocName is empty, then a dialog will display with all documents.

DataEase Action(s)

Document Open >>Enter: Document Name

DesignerView

Description

Puts the current document into designer mode.

Parameters

None

Return Value

Status. Integer.

Usage

Puts document into design mode.

DataEase Action(s)

Designer View

DOSMainMenu

Description

Display the DOS Main Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Main Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Main Menu

DOSUserMenu

Description

Display the DOS User Menu.

Parameters

Name	Type	Description
MenuName	string	A string holding the name of the Menu you wish to display.

Return Value

Status. Integer.

Usage

Display/execute an existing Menu document.

DataEase Action(s)

DOS User Menu

DOSRecordEntry

Description

Open a form in user view.

Parameters

Name	Type	Description
FormName	string	A string holding the name of the Form you wish to display.

Return Value

Status. Integer.

Usage

Open an existing Form document in User View.

DataEase Action(s)

DOS Record Entry

DOSQueryMenu

Description

Display the DOS Query Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the DQL Query Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Query Menu

DOSProcedure

Description

Open a procedure in user view.

Parameters

Name	Type	Description
ProcedureName	string	A string holding the name of the procedure you wish to display.

Return Value

Status. Integer.

Usage

Execute an existing Procedure or display DQL Menu.

DataEase Action(s)

DOS Procedure

DOSStatus

Description

Run DOS Document Status.

Parameters

None

Return Value

Status. Integer.

Usage

Run the Document Status Report.

DataEase Action(s)

DOS Status

DOSUtilMenu

Description

DOS Utilities Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Utilities Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Utilities Menu

DOSImportMenu

Description

Open the DOS Import Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Utilities Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Import Menu

DOSProgramCall

Description

Execute DOS command.

Parameters

Name	Type	Description
ProgramName	string	A string holding the name of the program you wish to run.

Return Value

Status. Integer.

Usage

Execute a DOS command with coded parameter substitution.

DataEase Action(s)

DOS Program Call

DOSFormsMenu

Description

Open DOS Forms Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Form Definition Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Forms Menu

DOSRecordsMenu

Description

Open DOS Forms Menu for user view.

Parameters

None

Return Value

Status. Integer.

Usage

Display a menu of Forms available for record entry.

DataEase Action(s)

DOS Records Menu

DOSMaintMenu

Description

Open DOS Maintenance Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Maintenance Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Maintenance Menu

DOSReportsMenu

Description

Open DOS Quick Reports Menu

Parameters

None

Return Value

Status. Integer.

Usage

Display the Quick Reports Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Reports Menu

DOSDefineTable

Description

Display New Form dialog for new table definition.

Parameters

None

Return Value

Status. Integer.

Usage

Define a new Form and new Table.

DataEase Action(s)

DOS Define Table

DOSDefineForm

Description

Open the DOS Forms Menu

Parameters

None

Return Value

Status. Integer.

Usage

Define a new Form on an existing Table.

DataEase Action(s)

DOS Define Form

DOSViewForm

Description

Open Form in designer mode.

Parameters

None

Return Value

Status. Integer.

Usage

View or Modify a Form Document.

DataEase Action(s)

DOS View Fom

DOSDeleteForm

Description

Display Delete dialog for forms

Parameters

None

Return Value

Status. Integer.

Usage

Delete a Form Document.

DataEase Action(s)

DOS Delete Form

DOSOSMenu

Description

Open the DOS Forms Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Operating System Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Operating System Menu

DOSFileList

Description

Run the DOS directory listing.

Parameters

Name	Type	Description
FileSpec	string	

Return Value

Status. Integer.

Usage

Application Directory DOS File List.

DataEase Action(s)

DOS File List

DOSCheckDisk

Description

Run DOS Check Disk

Parameters

None

Return Value

Status. Integer.

Usage

Check Application Disk for errors and space available.

DataEase Action(s)

DOS Check Disk

DOSFormatDisk

Description

Format a diskette.

Parameters

Name	Type	Description
DiskDrive	string	A letter representing the drive to be formatted. For example, DOSFormatDisk("A")

Return Value

Status. Integer.

Usage

Format a New Disk.

DataEase Action(s)

DOS Format Disk

DOSBackup

Description

Perform a DOS Backup

Parameters

Name	Type	Description
DiskDrive	string	A letter representing the drive where the backup will be made to.

Return Value

Status. Integer.

Usage

Operating System Backup of application.

DataEase Action(s)

DOS Backup

DOSRestore

Description

Perform a DOS Restore.

Parameters

Name	Type	Description
DiskDrive	string	A letter representing the drive where the backup will be restored from.

Return Value

Status. Integer.

Usage

Operating System Restore of application.

DataEase Action(s)

DOS Restore

DOSPrompt

Description

Run a DOS window.

Parameters

None

Return Value

Status. Integer.

Usage

Other DOS Operating System commands.

DataEase Action(s)

DOSPrompt

DOSAdminMenu

Description

Open DOS Admin Menu.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Administration Menu from the DOS or OS/2 version.

DataEase Action(s)

DOS Administration Menu

DOSRunProcedure

Description

Open an existing procedure

Parameters

None

Return Value

Status. Integer.

Usage

Execute an existing Procedure document.

DataEase Action(s)

DOS Run Procedure

DOSLoadProcedure

Description

Load a procedure.

Parameters

None

Return Value

Status. Integer.

Usage

Load a Procedure document.

DataEase Action(s)

DOS Load Procedure

DOSDeleteProcedure

Description

Open the Delete dialog for procedures

Parameters

None

Return Value

Status. Integer.

Usage

Delete a Procedure document.

DataEase Action(s)

DOS Delete Procedure

DOSRunReport

Description

Run a report.

Parameters

None

Return Value

Status. Integer.

Usage

Execute an existing Report document.

DataEase Action(s)

DOS Run Report

DOSReportRecords

Description

Run report records

Parameters

None

Return Value

Status. Integer.

Usage

Define selection of Report Document Records.

DataEase Action(s)

DOSReportFields

Description

Define report.

Parameters

None

Return Value

Status. Integer.

Usage

Define Report Document Fields.

DataEase Action(s)

DOS Report Fields

DOSReportFormat

Description

Define a report format.

Parameters

None

Return Value

Status. Integer.

Usage

Define New Report Format.

DataEase Action(s)

DOSReportFormat

DOSLoadReport

Description

Load a report document.

Parameters

None

Return Value

Status. Integer.

Usage

Load a Report document.

DataEase Action(s)

DOS Load Report

DOSDeleteReport

Description

Open the Delete dialog for reports.

Parameters

None

Return Value

Status. Integer.

Usage

Delete a Report document.

DataEase Action(s)

DOS Delete Report

DOSOneTimeImport

Description

Open the Import dialog.

Parameters

None

Return Value

Status. Integer.

Usage

One-Time Import Dialog.

DataEase Action(s)

DOS One-Time Import

DOSRunImport

Description

Run an import.

Parameters

Name	Type	Description
ImportName	string	A string representing the name of the import DBI file.

Return Value

Status. Integer.

Usage

Run a Predefined Import.

DataEase Action(s)

DOS Run Import

DOSDefineImport

Description

Open the Import dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Define an Import.

DataEase Action(s)

DOS Define Import

DOSViewImport

Description

Display the Import dialog.

Parameters

Name	Type	Description
ImportName	string	A string representing the name of the import DBI file you wish to view.

Return Value

Status. Integer.

Usage

View or Modify a Predefined Import.

DataEase Action(s)

DOS View Import

DOSDeleteImport

Description

Open the Import dialog

Parameters

Name	Type	Description
ImportName	string	A string representing the name of the import DBI file you wish to delete.

Return Value

Status. Integer.

Usage

Delete a Predefined Import.

DataEase Action(s)

DOS Delete Import

DOSInstallForm

Description

Open the Install dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Install a Form Document.

DataEase Action(s)

DOS Install Form

DOSInstallProcedure

Description

Open the Install dialog

Parameters

None

Return Value

Status. Integer.

Usage

Install a Procedure Document.

DataEase Action(s)

DOS Install Procedure

EndPrintToWindow

Description

End print-to-window.

Parameters

None

Return Value

Status. Integer.

Usage

End the Printer-To-Window display.

DataEase Action(s)

End Printer-To-Window

ExitDataEase

Description

Closes DataEase

Parameters

None

Return Value

Status. Integer.

Usage

Closes DataEase.EXE, thus shutting DataEase.

DataEase Action(s)

Exit DataEase

ExecuteFile

Description

Executes specified File.

Parameters

Name	Type	Description
ExecuteString	string	A string holding the name of the file you wish to execute.

Return Value

Status. Integer.

Usage

Executes the specified file.   .

DataEase Action(s)

Execute File -- File Name:

FilterClear

Description

Clear the selection.

Parameters

None

Return Value

Status. Integer.

Usage

Display all records, regardless of selection filter/sort order.

DataEase Action(s)

Filter Clear

FilterSet

Description

Set filters/sorts.

Parameters

None

Return Value

Status. Integer.

Usage

Display records according to the specified selection filter/sort order.

DataEase Action(s)

Filter Set

FieldClear

Description

Clears the currently selected field.

Parameters

None

Return Value

Status. Integer

Usage

Currently does nothing for fields or buttons.

DataEase Action(s)

Field Clear

FormReorganize

Description

Reorganizes TableName, or opens the Reorganize Form dialog.

Parameters

Name	Type	Description
TableName	string	A string holding the name of the table which you wish to reorganize.

Return Value

Status. Integer.

Usage

If a table name is specified, then that table is reorganized. If not, the Reorganize Form dialog is displayed.

DataEase Action(s)

Form Reorganize >> Enter: Table Name

FormClear

Description

Goes to a new record

Parameters

None

Return Value

Status.   Integer.

Usage

Clears the form.

DataEase Action(s)

Form Clear

FormOpenRelated

Description

Opens related document

Parameters

Name	Type	Description
DocName	string	A string representing the name of the Related Document which you wish to open.

Return Value

Status. Integer.

Usage

If DocName is specified it is opened (assuming it is related to the parent document). If DocName is empty then the Open Related dialog is displayed.

DataEase Action(s)

Form Open Related >> Enter RELATION[,Form]:

HelpDesktop

Description

Display Help About the desktop.

Parameters

None

Return Value

Status. Integer.

Usage

Explains the parts of the screen and use of the keyboard and mouse.

DataEase Action(s)

Help Desktop

HelpIndex

Description

Display help index.

Parameters

None

Return Value

Status. Integer.

Usage

Displays a list of Help topics.

DataEase Action(s)

Help Index

HelpMenus

Description

Display help about the menus

Parameters

None

Return Value

Status. Integer.

Usage

Displays the contents of all pull-down menus.

DataEase Action(s)

Help Menus

HelpToolbar

Description

Display help about the toolbars.

Parameters

None

Return Value

Status. Integer.

Usage

Displays basic instructions on the use of the toolbar icons.

Notes:

DataEase Action(s)

Help Toolbar

HelpHowTo

Description

Display help on “how to”.

Parameters

None

Return Value

Status. Integer.

Usage

Displays basic instructions for common operations and procedures.

DataEase Action(s)

Help How To

HelpGlossary

Description

Display the help glossary.

Parameters

None

Return Value

Status. Integer.

Usage

Alphabetical list of dialogs and defined terms.

DataEase Action(s)

Help Glossary

HelpUser

Description

Display help for user of document.

Parameters

None

Return Value

Status. Integer.

Usage

Help for the user of this Document.

DataEase Action(s)

Help User

HelpAbout

Description

Displays the About DataEase dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Display the Program Version Number and Copyright Information.

DataEase Action(s)

Help About

HelpSearch

Description

Perform a help search.

Parameters

Name	Type	Description
Keyword	string	A string holding the name of the DataEase keyword you wish to search for help on.

Return Value

Status. Integer.

Usage

Displays a list of Help topics.

DataEase Action(s)

Search for Help

HelpDQL

Description

Displays help on DQL

Parameters

None

Return Value

Status. Integer.

Usage

Displays a list of Help topics.

DataEase Action(s)

Help DQL

HideWin

Description

Hides current window

Parameters

None

Return Value

Status. Integer.

Usage

Hides the current window.

DataEase Action(s)

None

IfMaxNormalize

Description

Normalizes current window if maximized.

Parameters

UseSize	integer	An integer value used to specify how the application window will display. 0 - Normalized 1 - Maximized 2 - Minimized 3 – Normalized, then positioned with xPos and yPos and also sized with xSize and ySize **Note**: UseSize set to 1 or 3 will not behave as expected with DOS applications. PIF files control the ShowWindow mode for DOS applications and can only be overridden with UseSize 2.
xPos	integer	Horizontal position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
yPos	integer	Vertical position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
xSize	integer	Horizontal size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
ySize	integer	Vertical size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.

Return Value

Status. Integer.

Usage

If the show state of the current window is maximized it is normalized. Size and position can be optionally specified.

DataEase Action(s)

None

IfNormalMax

Description

Maximizes current window if normalized.

Parameters

None

Return Value

Status. Integer.

Usage

If the show state of the current window is normal it is maximized. Size and position can be optionally specified.

DataEase Action(s)

None

IfMaxHide

Description

Hides current window if maximized.

Parameters

None

Return Value

Status. Integer.

Usage

If the current window is maximized it is hidden.

DataEase Action(s)

None

IfNormalHide

Description

Hides current window if normal.

Parameters

None

Return Value

Status. Integer.

Usage

If the current window is normalized it is hidden.

DataEase Action(s)

None

ImportsInfo

Description

Run Imports Status.

Parameters

None

Return Value

Status. Integer.

Usage

Display Status of Application Imports.

DataEase Action(s)

Imports Information

IfMaxMin

Description

Minimizes current window if maximized.

Parameters

None

Return Value

Status. Integer.

Usage

If the show state of the current window is maximized it is minimized.

DataEase Action(s)

None

IfNormalMin

Description

Minimizes current window if normal

Parameters

None

Return Value

Status. Integer.

Usage

If normalized the current window is minimized.

DataEase Action(s)

None

IsWinMax

Description

Reports on the window size.

Parameters

None

Return Value

Status. Integer.

Usage

Returns 2 if the current window is maximized and 1 if the current window is normalized.

Notes: Could be enhanced to specify document by name.

DataEase Action(s)

None

LookupTo

Description

Does relational lookup to specified table.

Parameters

Name	Type	Description
TableName	string	A string representing the name of the TableName from which you wish to perform a lookup.

Return Value

Status. Integer.

Usage

Displays the Lookup dialog if a table is specified. If notable is specified the Table dialog displays first.

DataEase Action(s)

Lookup -- Enter RELATION:

MaxWin

Description

Maximizes current window.

Parameters

None

Return Value

Status. Integer.

Usage

Maximizes the current window.

DataEase Action(s)

None

MinWin

Description

Minimizes current window

Parameters

None

Return Value

Status. Integer.

Usage

Minimizes the current window.

DataEase Action(s)

None

NewForm

Description

Displays the **New Document** dialog box with **Form** selected.

Parameters

None

Return Value

Status. Integer.

Usage

The New Document dialog is displayed, with Form ready selected.

DataEase Action(s)

New Form

NewMenu

Description

Displays the **New Document** dialog with **Menu** selected .

Parameters

None

Return Value

Status. Integer

Usage

The New Document dialog is displayed with Menu selected.

DataEase Action(s)

New Menu

NewProcedure

Description

Displays the **New Document** dialog with **Procedure** selected.

Parameters

None

Return Value

Status. Integer.

Usage

The New Document dialog is displayed with Procedure selected.

DataEase Action(s)

New Procedure

NewReport

Description

Displays the **New Document** dialog with **Report** selected.

Parameters

None

Return Value

Satus. Integer.

Usage:

The New Document dialog is displayed with Report selected.

DataEase Action(s)

New Report.

NextRecord

Description

Goes to the next record

Parameters

None

Return Value

Status. Integer.

Usage

Goes to the next record

DataEase Action(s)

Record Next

NormalizeWin

Description

Normalizes the currently active window.

Parameters

UseSize	integer	An integer value used to specify how the application window will display. 0 - Normalized 1 - Maximized 2 - Minimized 3 – Normalized, then positioned with xPos and yPos and also sized with xSize and ySize **Note**: UseSize set to 1 or 3 will not behave as expected with DOS applications. PIF files control the ShowWindow mode for DOS applications and can only be overridden with UseSize 2.
xPos	integer	Horizontal position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
yPos	integer	Vertical position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
xSize	integer	Horizontal size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
ySize	integer	Vertical size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.

Return Value

Status. Integer.

Usage

Normalizes the current window. The window size and position can be optionally specified.

DataEase Action(s)

None

OpenForm

Description

Opens a specified form or displays dialog.

Parameters

Name	Type	Description
FormName	string	A character string, enclosed in quotes, representing

the name of the form you wish to open.

Return Value

Status. Integer.

Usage

If FormName is not empty the specified document will be opened. If FormName is empty then a dialog will display with all documents.

DataEase Action(s)

Open Form >> Enter: Form Name

OpenMenu

Description

Opens a specified menu, or displays the documents dialog.

Parameters

Name	Type	Description
MenuName	string	A character string, enclosed in quotes, representing the name of the menu you wish to open.

Return Value

Status. Integer.

Usage

If MenuName is not empty the specified document will be opened. If MenuName is empty then a dialog will display with all documents.

DataEase Action(s)

Open Menu >> Enter: Menu Name

OpenReport

Description

Opens a specified report, or displays the dialog box.

Parameters

Name	Type	Description
ReportName	string	A character string, enclosed in quotes, representing the name of the Report you wish to open. For example, OpenReport(“FinanceRep1”).

Return Value

Status. Integer.

Usage

If ReportName is not empty, the specified document will be opened. If ReportName is empty then a dialog will display with all documents.

DataEase Action(s)

Open Report >> Enter: Report Name

OpenProcedure

Description

Opens a specified procedure, or displays the Open Document dialog.

Parameters

Name	Type	Description
ProcedureName	string	A character string, enclosed in quotes, representing the name of the Procedure you wish to open.

Return Value

Status. Integer.

Usage

If ProcedureName is not empty the specified document will be opened. If ProcedureName is empty then a dialog will display with all documents.

DataEase Action(s)

Open Procedure >> Enter: Procedure Name

OLELinks

Description

View, Update, Open, or Remove OLE links.

Parameters

None

Return Value

Status. Integer.

Usage

View, Update, Open, or Remove OLE links.

DataEase Action(s)

Links

Paste

Description

Pastes data from the clipboard.

Parameters

None

Return Value

Status. Integer.

Usage

Pastes data values from the clipboard.

DataEase Action(s)

Paste

PrintDocument

Description

Prints the current document.

Parameters

None

Return Value

Status. Integer.

Usage

Displays the Print dialog, or prints document depending on settings in Print Options.

DataEase Action(s)

Print Document

PrintPreview

Description

Prints the current document in **zoomed out** mode.

Parameters

None

Return Value

Status. Integer.

Usage

Zooms out, then displays Print dialog or prints document, depending on settings in Print Options.

DataEase Action(s)

Print Preview

PrinterSetup

Description

Displays the **Printer Setup** dialog

Parameters

None

Return Value

Status. Integer.

Usage

Displays Printer Setup dialog.

DataEase Action(s)

Printer Setup

QBM_NewReport

Description

Displays the **QBM** dialog with the **MultiView** for the current document.

Parameters

None

Return Value

Status. Integer.

Usage

The QBM dialog is displayed with a copy of the MultiView for the current document.
DataEase Action(s)

QBM - New Report

RecordSave

Description

Saves changed record(s)

Parameters

None

Return Value

Status.   Integer.

Usage

Saves modifications to the current record.

DataEase Action(s)

Record Save

RecordSaveNew

Description

Saves new record(s)

Parameters

None

Return Value

Status.   Integer.

Usage

Saves current record as a new record.

DataEase Action(s)

Record Save As New

RecordDelete

Description

Deletes current record

Parameters

None

Return Value

Status.   Integer.

Usage

Deletes the current record.

DataEase Action(s)

Record Delete

RecordRestore

Description

Restores deleted record

Parameters

None

Return Value

Status.   Integer.

Usage

Restores the current record.

DataEase Action(s)

Record Restore

RecordLast

Description

Goes to the last record

Parameters

None

Return Value

Status. Integer.

Usage

Goes to the last record.

DataEase Action(s)

Record Last

RecordNext

Description

Goto the next record.

Parameters

None

Return Value

Status. Integer.

Usage

Display the next record in table.

DataEase Action(s)

Next Record

ReturnToParentDoc

Description

Return to parent document

Parameters

None

Return Value

Status. Integer.

Usage

Closes the current document and returns focus to the parent document.

DataEase Action(s)

Return to

RefreshScreen

Description

Refreshes data.

Parameters

None

Return Value

Status. Integer.

Usage

Refresh the screen to show data changed by other users.

DataEase Action(s)

Refresh Screen

RefreshContinuously

Description

Continuously refreshes data.

Parameters

None

Return Value

Status. Integer.

Usage

Continuously Refresh the screen to show data changed by other users.

DataEase Action(s)

Refresh Continuously

RefreshSetTime

Description

Sets refresh time.

Parameters

None

Return Value

Status. Integer.

Usage

Set the Time interval for continuous Refresh in seconds.

DataEase Action(s)

Refresh Set Time

Notes: Current action does not accept a Time parameter.

RecordFirst

Description

Goes to the first record.

Parameters

None

Return Value

Status. Integer.

Usage

Goes to the first record.

DataEase Action(s)

Record First

RecordPrevious

Description

Goes to the previous record

Parameters

None

Return Value

Status. Integer.

Usage

Goes to the previous record

DataEase Action(s)

Record Previous

Relationships

Description

Open the Relationships form.

Parameters

None

Return Value

Status. Integer.

Usage

Define or modify Relationships between tables.

DataEase Action(s)

Relationships

ReturnDataToDoc

Description

Returns data to parent document

Parameters

None

Return Value

Status. Integer.

Usage

Closes current document and causes PRISM to return the data to the parent document.

DataEase Action(s)

Return Data to

RecordsSort

Description

Goto set sort order mode.

Parameters

None

Return Value

Status. Integer.

Usage

Set record sorting filter.

DataEase Action(s)

Records Sort

RecordPreviousPage

Description

Goto previous page of records.

Parameters

None

Return Value

Status. Integer.

Usage

Display the previous window of records in the database.

DataEase Action(s)

Record Previous Page

RecordNextPage

Description

Goto next page of records.

Parameters

None

Return Value

Status. Integer.

Usage

Display the next window of records in the database.

DataEase Action(s)

Record Next Page

RecordsInfo

Description

Run Records Status.

Parameters

None

Return Value

Status. Integer.

Usage

Display Status of Application Records.

DataEase Action(s)

Records Information

RecordFirstPos

Description

Record first, positionally dependent.

Parameters

None

Return Value

Status. Integer.

Usage

Display the first matching record.

DataEase Action(s)

Record First

RecordPreviousPos

Description

Record previous, positionally dependent.

Parameters

None

Return Value

Status. Integer.

Usage

Display the previous matching record.

DataEase Action(s)

Record Previous

RecordNextPos

Description

Record next, positionally dependent.

Parameters

None

Return Value

Status. Integer.

Usage

Display the next matching record.

DataEase Action(s)

Record Next

RecordLastPos

Description

Record last, positionally dependent.

Parameters

None

Return Value

Status. Integer.

Usage

Display the last matching record.

DataEase Action(s)

Record Last

RecordPrevPagePos

Description

Record previous page, positionally dependent.

Parameters

None

Return Value

Status. Integer.

Usage

Display the previous window of records in the database.

DataEase Action(s)

Record Previous Page

RecordNextPagePos

Description

Record next page, positionally dependent.

Parameters

None

Return Value

Status. Integer.

Usage

Display the next window of records in the database.

DataEase Action(s)

Record Next Page

ServersInfo

Description

Run Servers Status.

Parameters

None

Return Value

Status. Integer.

Usage

Display Status of Application Servers.

DataEase Action(s)

Servers Information

SelectRecords

Description

Goto **Set Filters** mode.

Parameters

None

Return Value

Status. Integer.

Usage

Enter selection criteria into fields.

DataEase Action(s)

Select Records

SelectionFilter

Description

Sets selection filter

Parameters

Name	Type	Description
FilterLSide	string	A string representing the selection criteria. A space is appended to FilterLSide, if it does not already end with a space.
FilterRSide	string	A string representing the right-hand side of the selection criteria.

Return Value

Status. Integer.

Usage

Sets a selection filter. A space is appended to FilterLSide if it does not end with a space. An operator must be included in **either** FilterLSide or FilterRSide, but not in **both**.

Example

SelectionFilter(“MyField =”, “This one!”)

DataEase Action(s)

Selection Filter

SortAsending

Description

Set field for ascending sort order.

Parameters

None

Return Value

Status. Integer.

Usage

Sort this field in Ascending order.

DataEase Action(s)

Records Sort Ascending

SortDescending

Description

Set field for descending sort order.

Parameters

None

Return Value

Status. Integer.

Usage

Sort this field in Descending order.

DataEase Action(s)

Records Sort Descending

ToggleToolbar

Description

Switches the Toolbar on or off.

Parameters

Name	Type	Description
Switch	string	Enter (1) to show the Toolbar, or (0) to hide the toolbar. If no Switch value is entered, then the Toolbar setting is toggled

Return Value

Status. Integer.

Usage

Show (1) or hide (0) the toolbar. If no Switch then it toggles the setting.

DataEase Action(s)

Toggle Toolbar -- Enter: 1 (set ON) or 0 (set OFF)

ToggleStatusbar

Description

Switches the Status Bar on or off.

Parameters

Name	Type	Description
Switch	string	Enter (1) to show the Status Bar, or (0) to hide the toolbar. If no Switch value is entered, then the Toolbar setting is toggled

Return Value

Status. Integer.

Usage

Show (1) or hide (0) the status bar. If no Switch then it toggles the setting.

DataEase Action(s)

Toggle Status Bar -- Enter: 1 (set ON) or 0 (set OFF)

ToggleCatalog

Description

Switches the Catalog on or off.

Parameters

Name	Type	Description
Switch	string	Enter (1) to show the Status Bar, or (0) to hide the toolbar. If no Switch value is entered, then the Toolbar setting is toggled

Return Value

Status. Integer.

Usage

Show (1) or hide (0) the DataEase Catalog. If no Switch is specified, then it toggles the setting.

DataEase Action(s)

Toggle Catalog -- Enter: 1 (set ON) or 0 (set OFF)

ToggleSQL

Description

Turns the Generated SQL window on or off.

Parameters

Name	Type	Description
Switch	string	Enter (1) to show the Generated SQL Window, or (0) to hide the toolbar. If no Switch value is entered, then the SQL Window setting is toggled

Return Value

Status. Integer.

Usage

Show (1) or hide (0) the SQL window. If no Switch then it toggles the setting.

DataEase Action(s)

Toggle Generated SQL -- Enter: 1 (set ON) or 0 (set OFF)

ToggleNormalMax

Description

Toggles between normalized and maximized.

Parameters

UseSize	integer	An integer value used to specify how the application window will display. 0 - Normalized 1 - Maximized 2 - Minimized 3 – Normalized, then positioned with xPos and yPos and also sized with xSize and ySize

		Note: UseSize set to 1 or 3 will not behave as expected with DOS applications. PIF files control the ShowWindow mode for DOS applications and can only be overridden with UseSize 2.
xPos	integer	Horizontal position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
yPos	integer	Vertical position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
xSize	integer	Horizontal size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.
ySize	integer	Vertical size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if UseSize is not equal to 3.

Return Value

Status. Integer.

Usage

If the show state of the current window is maximized it is normalized. If the show state is normalized it is maximized. If normalizing size and position can be optionally specified.

DataEase Action(s)

None

UndoEdit

Description

Undo last action

Parameters

None

Return Value

Status. Integer.

Usage

Undoes the last edit action.

DataEase Action(s)

Undo

Users

Description

Open Users form.

Parameters

None

Return Value

Status. Integer.

Usage

Used for grant users permission to access the application.

DataEase Action(s)

Users

ViewAsForm

Description

View as a form (rather than as a Table)

Parameters

None

Return Value

Status. Integer.

Usage

Set document to be displayed in single record format.

DataEase Action(s)

View As a Form

ViewAsTable

Description

View as a table (Table View rather than Form View).

Parameters

None

Return Value

Status. Integer.

Usage

Display multiple records in a tabular format.

DataEase Action(s)

View As Table

WindowsArrangeIcons

Description

Arrange window icons.

Parameters

None

Return Value

Status. Integer.

Usage

Arrange all minimized document windows.

DataEase Action(s)

Windows Arrange Icons

WindowsTileAcross

Description

Tiles windows horizontally.

Parameters

None

Return Value

Status. Integer.

Usage

Horizontally tile all open document windows.

DataEase Action(s)

Windows Tile Across

WindowsTileDown

Description

Tiles windows vertically.

Parameters

None

Return Value

Status. Integer.

Usage

Vertically tile all open document windows.

DataEase Action(s)

Windows Tile Down

WindowsCascade

Description

Cascades all windows

Parameters

None

Return Value

Status. Integer.

Usage

Cascade all document windows.

DataEase Action(s)

Windows Cascade

WindowsCloseAll

Description

Closes all documents

Parameters

None

Return Value

Status. Integer.

Usage

Closes all documents open in the MDI.

DataEase Action(s)

Windows Close All

ZoomNormal

Description

Turns zoom off.

Parameters

None

Return Value

Status. Integer.

Usage

Restore display of objects to normal size.

DataEase Action(s)

Zoom Normal

ZoomIn

Description

Zoom in.

Parameters

None

Return Value

Status. Integer.

Usage

Enlarge display of objects by %d percent of current size.

DataEase Action(s)

Zoom In

ZoomOut

Description

Zooms out.

Parameters

None

Return Value

Status. Integer.

Usage

Reduce display of objects by a percent of current size.

DataEase Action(s)

Zoom Out

ZoomFillWindow

Description

Zooms to fill window.

Parameters

None

Return Value

Status. Integer.

Usage

Zoom to fill the Document window.

DataEase Action(s)

Zoom Fill Window

ZoomFillAcross

Description

Zooms to fill window width.

Parameters

None

Return Value

Status. Integer.

Usage

Zoom to fill the Width of the Document window.

DataEase Action(s)

Zoom Fill Across

ZoomFillDown

Description

Zooms to window height.

Parameters

None

Return Value

Status. Integer.

Usage

Zoom to fill the Height of the Document window.

DataEase Action(s)

Zoom Fill Down

ZoomPrevious

Description

Zoom to previous setting.

Parameters

None

Return Value

Status. Integer.

Usage

Restore display of objects to Previous Zoomed size.

DataEase Action(s)

Zoom Previous

ZoomCustom

Description

Opens Custom Zoom dialog.

Parameters

None

Return Value

Status. Integer.

Usage

Use Custom Zoom dialog to specify a percentage and/or set options.

DataEase Action(s)

Zoom Custom

ZoomTo

Description

Zooms to specified percentage.

Parameters

Name	Type	Description
ZoomTo	string	The percentage of the Normal size you wish to zoom to.

Return Value

Status. Integer.

Usage

Specify a percentage of the NORMAL size and Zoom TO it.

DataEase Action(s)

Zoom To -- Enter: Percentage to Zoom to

ZoomBy

Description

Zoom by specified percentage.

Parameters

Name	Type	Description
ZoomBy	string	The percentage of the Normal size you wish to zoom by.

Return Value

Status. Integer.

Usage

Specify a percentage of the NORMAL size and Zoom by it.

DataEase Action(s)

Zoom By -- Enter: Percentage to Zoom By

StrFunc Library

Introduction

The **StrFunc** CDF DLL contains eight functions for manipulating character strings. When using these functions, all strings should be enclosed within double quotes, otherwise they may be treated as field names.

The eight functions are:

StripChar	PadChar	PasteFromClip	CopyToClip
JoinStrings	Convert	FileRead	FileWrite

Installing the StrFunc Functions

The quickest way to register one or more functions from StrFunc into your own application is to cut and paste the registration information.

Open the **Sample CDF application** and on the Main menu, press the button on the bottom of the screen labeled **'CDFs SYSTEM FORM'**. Press F3 until you find the function you want to use in your own application. From the menu bar select **edit>>copy>>record**. Exit the Sample CDF application and access your own application. Open the CDFs System Form and Paste the record.

Note: You may need to change the CDF LIBRARY NAME field to make sure DataEase can find the DLL file.

StripChar

Description

This Function will search a string for specific character(s), and if found, they will be striped out.

Declaration

StripChar (targetstring/field , <character(s) to strip>)

Parameters

Name	Type	Description
Value1	string	The target string or field name. Can be any string value up to 255 characters in length.
Value2	string	The character(s) to be stripped. Can be any string value up to 255 characters in length.

Return Type

String. The target string, minus the stripped out characters.

Example

StripChar(MyFieldValue, “e”)

…when placed into the derivation formula of a field called MyFieldValue, this example would strip the letter "e” from whatever text was typed into MyFieldValue.

PadChar

Description

This Function will duplicate a character string a specified number of times. Note that if the target string contains more than one character, then only the first character of the string will be repeated.

Declaration

PadChar (character string, number)

Parameters

Name	Type	Description
Char	string	The character that will be repeated N times. If this string contains more than one character, only the **first** character will be used
N	integer	A number, between 1 and 255, representing the number of times which the string should be repeated.

Return Type

String. The duplicated string.

Example

In a field called MyFieldTwo, place the following derivation:

PadChar (MyField, 12)

…when you type “A” into MyField, then MyFieldTwo will derive as “AAAAAAAAAAAA”.

PasteFromClip

Description

This Function will paste data from the clipboard.

Declaration

PasteFromClip ()

Parameters

None

Return Type

Text. The text from the clipboard.

CopyToClip

Description

This Function will copy text to the clipboard.

Declaration

CopyToClip (character string or field)

Parameters

Name	Type	Description
Char	string	The character string or field name to be copied to the clipboard.

Return Type

None

JoinStrings

Description

This Function will concatenate a series of strings together.

Declaration

JoinStrings (string1 , string2 , ... , string10)

Parameters

Name	Type	Description
Char1	string	The first character string or field name to be joined.
Char2	string	The second character string or field name to be joined.
Char3	string	The third character string or field name to be joined.
Char4	string	The fourth character string or field name to be joined.
Char5	string	The fifth character string or field name to be joined.
Char6	string	The sixth character string or field name to be joined.
Char7	string	The seventh character string or field name to be joined.
Char8	string	The eighth character string or field name to be joined.
Char9	string	The ninth character string or field name to be joined.
Char10	string	The tenth character string or field name to be joined.

Note: If you have fewer than ten strings to join together, then use double quotations to represent missing strings. For example:

JoinStrings (string1 , string2 , “”,””,””,””,””,””,””, string10)

….would join three strings together.

Return Type

String. The Concatenated string.

Convert

Description

This function will convert between **characters** and their **ASCII decimal** equivalents. The <input type> is the letter '**c**' for character or '**n**' for decimal. If the input type is **c**, then the return value is decimal. Likewise if the input type is **n** then the return value is character.

Declaration

Convert (input type, target)

Parameters

Name	Type	Description
Type	string	A single character, which tells the function whether the either a “c” for character, or “n” for decimal.
Swapped	string	The target string or field name.

Return Type

A character, if the input was of type **decimal,** or a decimal number if the input type was of type **character**.

FileRead

Description

This function will read a string of characters from a file, starting at a specified position within the file, and continuing for a specified number of characters. Optionally, the string can be displayed within a message window by specifying "Y" or "N" as the messageflag.

Declaration

FileRead (filename, no. of characters to read, start position, messageflag)

Parameters

Name	Type	Description
FileName	string	The target file name.
HowMany	integer	The number of characters to read.
Start	integer	The start position in the target string.
MessageFlag	string	The Message Flag value. Specify “Y” to display in a message window, or “N” for no message window.

Return Type

String. A string of characters, holding the characters you have selected.

FileWrite

Description

This function will write a string of characters to a file, the user will specify the name of the target file, the starting position within the file, and the string or field value to write. Optionally, the string can be displayed within a message window by specifying "Y" or "N" as the messageflag.

Declaration

FileWrite (filename, start position, string to write or field, messageflag)

Parameters

Name	Type	Description
FileName	string	The target file name.
Start	integer	The start position in the target string.
String	string	The string to be written into the file.
MessageFlag	string	The Message Flag value. Specify “Y” to display in a message window, or “N” for no message window.

Return Type

"Success" or "Failure".

MsgBox Library

Description

The primary CDF in **MsgBox** is called **Message**. It displays a custom Windows message box allowing the application developer to specify a custom message, message box title, a message icon, various buttons which allow the user to respond to the message box, and finally an audible beep to alert the user to the fact that something requires attention. This function can be used alone or in conjunction with other DataEase for Windows functions. It returns an integer value corresponding to the Users response in dispatching the dialog.

Note that the 'beep' sound is actually performed by a separate function (called **MsgBeep**), which is included in the MsgBox.DLL -so don't forget to register both **Message** and **MsgBeep**.

Declaration

MESSAGE (MessageText , CaptionText , IconDisplay , Buttonselection , BeepSound)

Parameters

Name	Type	Description
TextMessage	string	The text message you wish to display.
CaptionTitle	string	A fifty character string used as the title for the Message Dialog box.
IconDisplay	integer	An integer value between 1 and 5 inclusive. Values are as shown below; 1 Information Icon (i) 2 Exclamation Icon (!) 3 Stop (Stop sign) 4 Question (?) 5 No icon
ButtonSelection	integer	An integer value between 0 and 5 inclusive that specifies what combination of buttons to be displayed in the message dialog for the user to use in dispatching the dialog and sending return values back to the function. Possible values are: 0 **OK** (default) 1 **OK CANCEL** 2 **RETRY CANCEL** 3 **ABORT RETRY CANCEL** 4 **YES NO** 5 **YES NO CANCEL**
BeepSound	integer	An integer value between 0 and 7 inclusive, which specifies which type of audible beep to generate according to a given system alert level. The sound for each alert level is determined by an entry in the [sounds] section of WIN.INI. Possible values are: 0 or 1 No sound (default)

2	System sound
3	Information sound
4	Exclamation sound
5	Stop sound
6	Question sound
7	OK sound

Return Values

Each message dialog you display will include at least one push button, which will enable the user to dispatch the dialog. When a button is pressed, the Message function returns an integer value indicating that it has received a user response, whereupon DataEase will continue processing. The following return values correspond to the available buttons for display:

1	ABORT
2	CANCEL
3	IGNORE
4	NO
5	OK
6	RETRY
7	YES
0	Out of Memory

Example

We have a form with a date field called DEPARTURE DATE, and we want to display a message when the user enters a date that is prior to today's date (e.g. it's difficult to leave when you already have). However, if the user enters a date after Today's date, we want to display a message that indicates on what day of the week that date occurs.

```
Field: DEPARTURE DATE
 Derivation Formula:

 if ( DEPARTURE DATE < current date ,
     if ( Message
         ( jointext ( "The date you entered: " ,
         jointext ( DEPARTURE DATE , " is invalid. \n\t
                     Do you want today's date?" )
            ) ,
         "Invalid Date Message" , 4, 4, 6 ) = 7 ,
     if ( Message
         ( jointext ( "Your entry date will be replaced with
         today's date: " , current date ) ,
         "Enter Current Date", 1, 0, 3 ) = 5, current date ,
         current date ) ,
     if ( Message
         ("Your entry will be discarded and the field will be
         left blank." ,
         "Discard Entry Date" , 2, 6, 4 ) = 5 , blank, blank )
         ) ,
     if ( Message
         ( jointext ( "The date you entered: " ,
         jointext ( DEPARTURE DATE,
         jointext ( " is a " ,
         spellweekday ( weekday ( DEPARTURE DATE ) ) ) ) )
         "Day of the Week", 4, 4, 6 ) = 5 ,
     DEPARTURE DATE , DEPARTURE DATE ))
```

NOTE: the symbols /n/t on line 4 tell Windows to display a <newline> and <tab>.

MsgBeep

Description

Generates an audible sound from the PC's speaker.

Declaration

MsgBeep(beepSound)

Parameters

Name	Type	Description
BeepSound	integer	An integer value between 0 and 7 inclusive, which specifies which type of audible beep to generate according to a given system alert level. The sound for each alert level is determined by an entry in the [sounds] section of WIN.INI. Possible values are:

0 or 1	No sound (default)
2	System sound
3	Information sound
4	Exclamation sound
5	Stop sound
6	Question sound
7	OK sound

Return Value

The value returned by MsgBeep means nothing and can be discarded or ignored.

PlayVid Library

Description

The **PlayVid** CDF DLL Library contains just one function – PLAYVID.

PlayVid passes a string (Video file name, e.g. Windsurf.AVI) to the Windows VidPlaySound function. The Windows function then immediately plays the video file via the Media Player program. Your Windows configuration must be set up with an appropriate Video driver.

Declaration

PlayVideo (WaveFileString)

Parameters

Name	Type	Description
VideoFileString	string	A character string consisting of a Video file's drive, path, and filename, eg. C:\videos\MyVideo1.AVI

Return Type

None

Example

User has a table where each record stores the name of a different video file. The user wants to be able to play the video for any record by pushing a button on the screen.

In this example we build a simple form with two fields and one button.

Fields

Name of Video.	Text:25
Video File.	Text:25

Button

Button Text is “Play Video”, and Action is PlayVideo(VideoFile)

PlaySound Library

Description

The **PlaySnd** CDF DLL Library contains one function – PlaySound. This function passes a string (wave file name, e.g. chimes.wav) to the Windows SndPlaySound function. The Windows function then immediately plays the wave file. Your Windows configuration must be set up with an appropriate Speaker driver.

Declaration

PlaySound (WaveFileString)

Parameters

Name	Type	Description
WaveFileString	string	A character string consisting of a Sound file's drive, path, and filename, eg. C:\sounds\MySound1.WAV

Return Type

None

Example

User has a table where each record stores the name of a different sound file. The user wants to be able to play the sound for any record by pushing a button on the screen.

In this example we build a simple form with two fields and one button.

Fields

Name of Sound.	Text:25
Sound File.	Text:25

Button

Button Text is “Play Video”, and Action is PlaySound(SoundFile)

DeMacro Library

Introduction

The **DeMacro** CDF DLL Library contains two functions – **KeyMacro** and **KeyStrokes**. These functions can be used to automate the input of commonly-used strings or commands. The KeyMacro function reads this string from a file, while the KeyStrokes function reads a string which would typically be hard-coded into a field derivation by the application designer.

For example, imagine we have a Form into which we will Import data regularly using a standard file name. We wish to automate this process by pressing a button on the Form. The macro should bring up the IMPORT dialog, select VARIABLE Length Text as the file format, Tab to the Source Filename edit control, type in the text: "PRICES.DAT", and press the <enter> key to execute the import.

KeyMacro

Description

This CDF will automatically send a series of keystrokes stored in a command file to DataEase for immediate execution. The keystrokes are entered into an ASCII text file of your choice using a normal ASCII text editor. The keystrokes sent can be Function Keys, Alt or Ctrl key combinations, Cursor movement keys and most typing characters from the keyboard. Conventions for building this file are explained below. Once built, the command file can be called as part of a CDF passing argument assigned to an Action Button or Image placed on a Form or a Live report.

Declaration

KeyMacro(FileName)

Parameters

Name	Type	Description
FileName	string	An ASCII text file containing the set of keystrokes to automatically execute.

KeyStrokes

Description

KeyStrokes is essentially identical to KeyMacro. But instead of passing the name of a file to the function, KeyStrokes passes a text string which itself consists of the keystrokes you wish executed. As you will appreciate, KeyStrokes is easier to use, while KeyMacro is more versatile, because you can change the contents of the command file.

Declaration

KeyStrokes(KeyString

Parameters

Name	Type	Description
KeyString	string	An ASCII text string containing the set of keystrokes to automatically execute.

Note: The Windows System Queue defaults to 120 events. Every keystroke is two events (KEYDOWN and KEYUP). This means that any single DeMacro script can contain only about 60 keystrokes. Characters can be two or more keystrokes. For example "a" is two keystrokes, "A" is four keystrokes, and "Ç" is ten keystrokes! (it gets sent as alt-0199). There is a way round this limitation. You can increase the size of the Windows System Queue with the undocumented command:

TypeAhead = xxxx

…where xxxx is an integer value. This command should be put in your WIN.INI file in the [windows] group.

Command File Keystroke Conventions:

To send text keystrokes, simply type them into the file. For instance, to send PRICES.DAT, just type it into the file. No quotes or special characters are needed. Spaces are allowed anywhere in the text you type but are ignored (collapsed) when passed to DataEase. So, the string: "Jeremiah Johnson" would be sent to DataEase as "JeremiahJohnson". To embed hard spaces you must use the token "_sp". Therefore: "Jeremiah_sp Johnson" would be sent as "Jeremiah Johnson". Blank lines in the command file are also ignored. Comment lines can be added if they begin with the string "/*" and end with the string "*/. Failure to close a comment line with "*/" will cause an error when the CDF is fired.

To send a combination of keystrokes such as ALT-PRTSC (to copy a picture of the current window to the Clipboard), you embed the second command within parentheses such as: _alt(_prtsc). When sending an Accelerator Key combination, make sure to put the text character in lower case. So to invoke FILE-OPEN you would enter the command _ctrl(o)

If you want to use a Button to automatically enter text into a field you must be sure to TAB back into or forward into the target field. So if you have a button that will write the text "Federal Express" into the DELIVERY field which is 2 tab order positions behind the button, you must precede the text string with two _shift(_tab) commands.

Once the command file is defined, you can call it via the DEMACRO CDF from a button. The Button Action should be set to EXPRESSION(CDF) and the Parameter box should have the following:

DEMARCO("path\command-filename").

Token Table:

_SHIFT	SHIFT KEY key,
_CTRL	CONTROL key,
_ALT	ALT key,
_bksp	BACK SPACE key,
_break	BREAK key,
_caps	CAPS LOCK TOGGLE key,
_del	DELETE key,
_down	DOWN ARROW key,
_end	END key,
_enter	RETURN key,
_esc	ESCAPE key,
_home	HOME key,
_ins	INSERT TOGGLE key,

_left	LEFT ARROW key,
_pgdn	PGDN KEY key,
_pgup	PGUP KEY key,
_prtsc	PRINT SCREEN key,
_right	RIGHT ARROW key,
_tab	TAB key,
_up	UP ARROW key,
_sp	SPACE key,
_under	UNDERSCORE key,
_lpar	LEFT PAREN key,
_rpar	RIGHT PAREN key,
_scroll	SCROLL LOCK key,
_numlock	NUMLOCK key,
_F1	F1 key,
_F2	F2 key,
_F3	F3 key,
_F4	F4 key,
_F5	F5 key,
_F6	F6 key,
_F7	F7 key,
_F8	F8 key,
_F9	F9 key,
_F10	F10 key,
_F11	F11 key,
_F12	F12 key,

OsFunc Library

Introduction

The **OsFunc** CDF DLL Library contains thirteen functions which allow you to carry out basic operating system tasks – such as creating new directories - from within DataEase. The functions are:

MakeDir	FreeDiskSpace	GetDateTimeOf
RemoveDir	IsFile	OSVersion
SearchFileInPath	GetCurrentDiskLetter	CopyAFile
GetCurrentDirName	GetSizeOf	RenameFile
ChangeDir		

MakeDir

Description

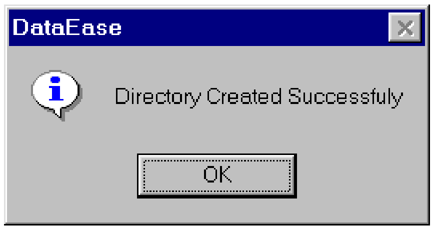
This Function will create a new Directory. Optionally, the OS response can be displayed within a message window by specifying "Y" or "N" as the messageflag.

Declaration

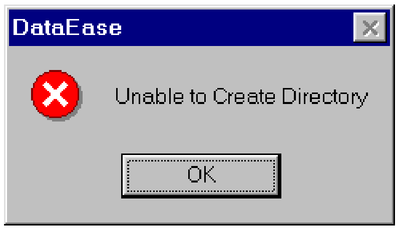
MakeDir (path/file name , messageflag)

Parameters

Name	Type	Description
PathandName	string	A character string representing the complete path and directory name, enclosed in quotes. For example, “C:\Data\NewDirectory”
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window. Entering a **Y** is normally the best choice, since it enables the user to see if his action has been successful or not.


DataEase

Directory Created Successfuly


DataEase

Unable to Create Directory

Example

MakeDir (“C:\data\MyNewDirectory”, “Y”)

RemoveDir

Description

This Function will delete a Directory. Optionally, the OS response can be displayed within a message window by specifying "Y" or "N" as the messageflag.

Declaration

RemoveDir (path/file name , messageflag)

Parameters

Name	Type	Description
PathandName	string	A character string representing the complete path and name of the directory to be deleted, enclosed in quotes. For example, “C:\Data\NewDirectory”
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window. Entering a **Y** is normally the best choice, since it enables the user to see if his action has been successful or not.

Example

RemoveDir(“E:\data\oldstuff”, “Y”)

SearchFileInPath

Description

This Function will search for a file in the specified Directory. If the file is not found here, then it will search the directories in your PC’s PATH environment statement. Optionally, the OS response can be displayed within a message window by specifying "Y" as the messageflag.

Declaration

SearchFileInPath (path/file name , messageflag)

Parameters

Name	Type	Description
PathandName	string	A character string representing the complete path and name of the directory to be searched, and the file name to be searched for. The complete string should be enclosed in quotes. For example, “D:\Data\address.txt”
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window. Entering a **Y** is normally the best choice, since it enables the user to see if his action has been successful or not.

Return Value

If found, this function returns the file name and its path.

FreeDiskSpace

Description

This Function will report the amount of free disk space remaining on the specified drive. Optionally, the OS response can be displayed within a message window by specifying "Y" or "N" as the messageflag.

Declaration

FreeDiskSpace (drive letter, messageflag)

Parameters

Name	Type	Description
PathandName	string	A character string, enclosed in quotes, representing the Drive to be checked. For example, “C”.
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

A string, representing the Amount of Disk space left.

IsFile

Description

This Function will search for a file in the specified Directory. If the file is not found there, it will search the directories in your PC’s path statement. This function will also search for files with a specified Attribute, e.g. a **Hidden** file, or a **System** file.

Declaration

IsFile (path/file name, file attribute)

Parameters

Name	Type	Description
PathandName	string	A character string, enclosed in quotes, representing the path and file name to be checked. For example, “C:\data\log.dat”.
FileAttribute	integer	Can be any of the following values:

00	Normal file, no attributes
01	Read only attribute
02	Hidden file
04	System file
08	Volume label
10	Directory
20	Archive

Return Value

If the specified file is found then a string reading "YES" will be returned. If not found, the string will read "NO".

Example

IsFile(“D:\mydata\log.txt”, 02)

GetCurrentDiskLetter

Description

This Function will get the current disk drive letter. Optionally, the OS response can be displayed within a message window by specifying "Y" as the messageflag.

Declaration

GetCurrentDiskLetter (messageflag)

Parameters

Name	Type	Description
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

A text string, containing the current Drive letter.

GetSizeOf

Description

This Function will calculate the size of all files within a directory, or a specified group of files (i.e.: c:\*.sys). Optionally, the OS response can be displayed within a message window by specifying "Y" as the messageflag value.

Declaration

GetSizeOf (path/file specification, messageflag)

Parameters

Name	Type	Description
PathandName	string	A character string, enclosed in quotes, representing the Drive/Directory and type of file to be checked. For example “C:\sysfiles\*.*”, or “D:\data\*.txt”.
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

A string, holding the size in bytes of the specified directory /files.

GetCurrentDirName

Description

This Function returns the current Directory Name. Optionally, the OS response can be displayed within a message window by specifying "Y" as the messageflag value.

Declaration

GetCurrentDirName (messageflag)

Parameters

Name	Type	Description
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

A text string holding the name of the current directory.

GetDateTimeOf

Description

This Function will get the Date Time of a file. Optionally, the OS response can be displayed within a message window by specifying "Y" " as the messageflag value.

Declaration

GetDateTimeOf (path/file name, messageflag)

Parameters

Name	Type	Description
PathandName	string	A character string, enclosed in quotes, representing the full Drive/Directory and full name of the file to be checked. For example “F:\datafiles\Fridaylog.doc”.
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

A string holding the Date and Time data for the specified file.

OSVersion

Description

This Function will return a string holding the Operating System Version.

Declaration

OSVersion (messageflag)

Parameters

Name	Type	Description
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

A string holding the Version Number of your PC’s Operating System.

CopyAFile

Description

This Function will copy a file. While copying, the file name can be changed, if desired. Optionally, the OS response can be displayed within a message window by specifying "Y" " as the messageflag value

Declaration

CopyAFile (source filename, destination file name, messageflag)

Parameters

Name	Type	Description
SourceFile	string	A character string, enclosed in quotes, holding the full drive/directory/file name of the target file. For example “C:\datafiles\Todaylog.dat”.
DestinationFile	string	A character string, enclosed in quotes, representing the full Drive/Directory/file name of the destination. For example “C:\datafiles\backup\Backuplog.dat”.
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

None

Example

CopyAFile (“C:\logs\log1.dat”, “d:\backups\log1backup.dat”, “N”)

RenameFile

Description

This Function will rename a file.

Declaration

RenameFile (old file Path/File, new file name, messageflag)

Parameters

Name	Type	Description
SourceFileName	string	A character string, enclosed in quotes, holding the full drive/directory/file name of the target file. For example “C:\datafiles\Todaylog.dat”.
NewFileName	string	A character string, enclosed in quotes, representing the full Drive/Directory/file name of the destination. For example “C:\datafiles\Backuplog.dat”.
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

None

Note: The source drive\path and the destination drive\path must be identical, otherwise you will receive an error.

ChangeDir

Description

This Function will change the current directory.

Declaration

ChangeDir (new directory name)

Parameters

Name	Type	Description
NewDirectory	string	A character string, enclosed in quotes, representing the full Drive/Directory that you wish to make current. For example, “D:\data”.
MessageFlag	string	Can be either “Y” or “N”. If you enter **Y**, then the response from the operating system is displayed in a message window.

Return Value

None.

File_CDF Library

The **File_CDF** CDF DLL Library contains just one function – **FileExecCDF**.

Description

This CDF passes a string to the Windows WinExec function. The WinExec function parses the string and executes the command and also passes open mode to ShowWindow. If The drive, path or execution filename are invalid, an error message will be displayed.

This function can be extremely useful, allowing you to call a named program, the file it is to open, and any parameters that might be needed, plus the type of Window it will open in.

Declaration

FileExecCDF(ExecuteString, iWndMode, ixWndPos, iyWndPos, ixWndSize, iyWndSize)

Parameters

Name	Type	Description
ExecuteString:	string	A character string consisting of an execution file's drive, path, and filename. The string may also include data file drive, path, filename and any other parameters the user desires. For example, **Notepad example.doc**
iWndMode:	integer	An integer value used to specify how the application window will display. 0 - Normalized 1 - Maximized 2 - Minimized 3 – Normalized, then positioned with ixWndPos and iyWndPos and also sized with ixWndSize and iyWndSize **Note**: iWndMode set to 1 or 3 will not behave as expected with DOS applications. PIF files control the ShowWindow mode for DOS applications and can only be overridden with iWndMode 2.
ixWndPos	integer	Horizontal position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if iWndMode is not equal to 3.
iyWndPos	integer	Vertical position of the upper left corner of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if iWndMode is not equal to 3.
ixWndSize	integer	Horizontal size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if iWndMode is not equal to 3.
iyndSize	integer	Vertical size of the application window. Units are in pixels and are relative to the screen. This parameter is ignored if iWndMode is not equal to 3.

Return Value

None

Example

A user has a table where each record keeps track of a document. The user wants to be able to view and edit the particular document referenced by a record in the table. The Form has the following objects:

Fields

ExecFile	Text:50
DataFile	Text:50
ExecParam	Text:10
CombinedString	Text:160, Virtual, Prevent Entry

The derivation for CombinedString field is as follows:

jointext(ExecFile, jointext(" ", jointext(DataFile, jointext(" ", ExecParam))))

Button

Button Text	View File
Button Action	FileExecCDF(CombinedString, 3, 300, 300, 350, 250)

Win API

Not every Custom Function has to be specifically written for DataEase. The Windows Application Programmers Interface (Win API) contains many pre-written functions which will work very happily as CDFs with DataEase.

To use these functions you will have to be familiar with the Win API. Microsoft's MSDN documentation covers this subject very well.

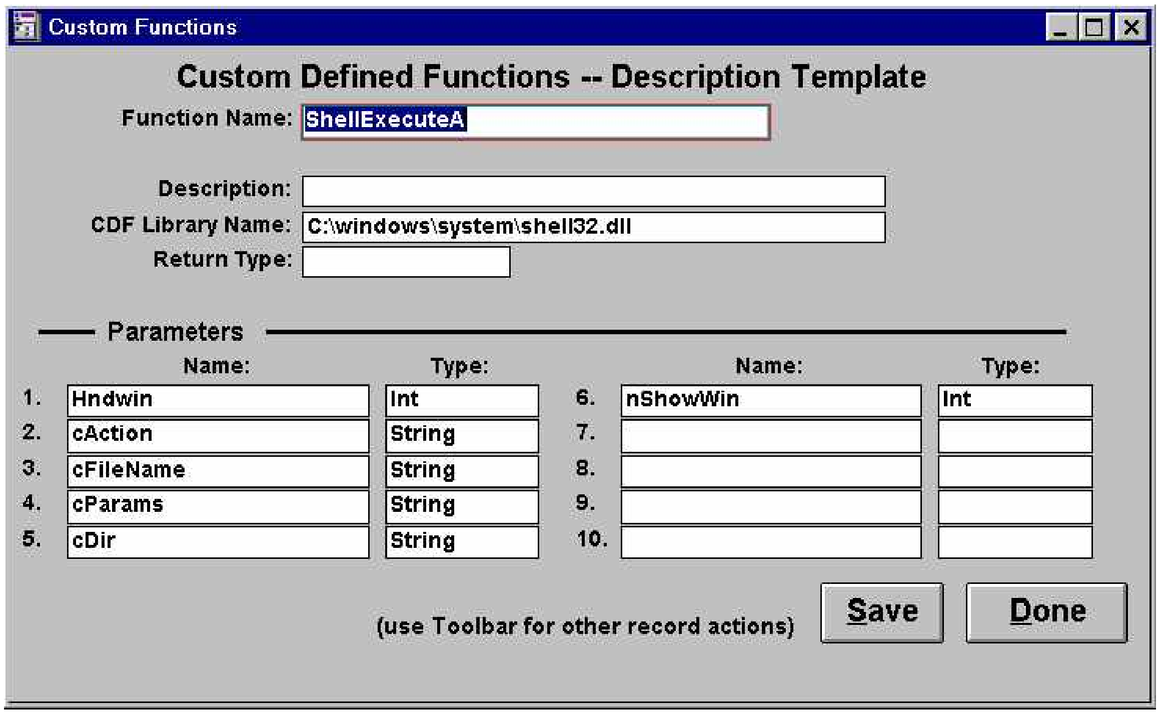
Rather than attempt to document Win API, in this section we will demonstrate how two of the Win API functions – ShellExecute and SwapMouseButton – can be used with DataEase.

ShellExecute

The Windows API ShellExecute() function is part of the Shell32.dll. It can be used to automatically start the application associated with a given document extension. For example, you could use it to open NotePad by passing the filename MyNotes.txt to the ShellExecute() function, or open Word by passing MyNotes.doc, and so on.

In this example we'll try to do something mildly useful, and create a database of favorite Web Sites. The database contains one table, which in turn contains just three fields and one button.

Step 1: Register the CDF.



Custom Defined Functions -- Description Template

Function Name: ShellExecuteA

Description:

CDF Library Name: C:\windows\system\shell32.dll

Return Type:

Parameters

	Name:	Type:		Name:	Type:
1.	Hndwin	Int	6.	nShowWin	Int
2.	cAction	String	7.		
3.	cFileName	String	8.		
4.	cParams	String	9.		
5.	cDir	String	10.		

Make sure that the path leading to the CDF Library Name is correct. It will vary according to your own version of Windows. It might be c:\WinNT\System32\shell32.dll, or whatever. You can run a quick search on "shell32.dll" to check your machine's path.

Note that ShellExecuteA is the 32 bit version of ShellExecute. Microsoft commonly add an "A" to the filename to distinguish between them.

Step 2: Create a new DataEase 6 database (we're using an Event Script, so it has to be v6 or later – earlier versions of DataEase did not support Event Wscripting.), and create a Table called "Favorites". In this Table create three fields called:

Web_Address.	Text 100. Unique. Indexed. Derivation: Web_Address := concat ("www.",Web_Address) .
Site_Summary	Text 80.
Site_Description	Memo 1000

…the actual lengths are irrelevant, so long as WebAddress is long enough to hold the longest URL you're likely to use.

Now create a button, with a text label "**Go**". In the button's clicked event, put the following script:

```
define "retcode" Number .
retcode := ShellExecuteA(0,"open",Web_Address.Value,"","",1) .
```

And that's that. There is no step 3, because the example database is now finished. Go into the form in user view, type **dataease.com** into the web_Address field, and save the record. Whenever you want to visit **dataease.com**, bring that record up on screen and hit the Go button. The ShellExecuteA function will open the application associated with the key file extension (WWW) – this will be Explorer, or NetScape, etc – pass it the URL held in Web_Address, and off you go. The web browser will start you dial-up connection, if needed.

The two other fields hold general information about the site. Site_Summary holds a one-liner written by you, and the Site_Description Memo field can be used to hold a lengthier description of the site. Most web sites have a "Site Description" or something like it, so you can simply cut and paste this into your memo field.

Now that we've described 'what it does', we can proceed to describe 'how it does it.

ShellExecuteA performs an operation on a specified file. The nature of that operation and the name of the file are two of the parameters that we pass it in our button script. The six parameters are:

Hndwin
Handle to a parent window. This window receives any message boxes that an application produces, such as error reporting. Defaults to 0 .

In our example database we pass the value **0** as our parameter.

cAction
A string - referred to as a verb - which specifies the action to be performed. Possible verbs are:

Edit

Explore

Find

Open

Print

Which verbs are available depends on the particular file or folder you're passing as a parameter. For instance, trying to use the verb Print with an executable file extension will result in failure.

A quick (though not infallible) way to find out which actions are available to a particular object is to simply right-click on the file/folder in Windows, which brings up the object's shortcut menu. This menu normally lists the available verbs – edit, open, print, etc.

In more detail, the six verbs operate as follows;

Edit:	Launches an editor and opens the document for editing. If cFileName is not a document file, the function will fail.
Explore:	Explores the folder specified by cFileName.
Find:	Initiates a search starting from the specified directory.
Open:	Opens the file specified by the lpFile parameter. The file can be an executable file, a document file, or a folder.
Print:	Prints the document file specified by cFileName. If cFileName is not a document file, the function will fail.

In our example database we pass the verb "open" as this parameter.

cFileName
The string that specifies the file or object on which to execute the specified verb.
Remember that not all verbs are supported on all objects. For example, not all document types support the "print" verb.

In our example database we're passing "Web_Address.Value" as this parameter.
cParams
If the cFileName parameter specifies an executable file, cParams is a string that specifies the parameters to be passed to the application. The format of this string is determined by the verb that is to be invoked. If cFileName specifies a document file, cParams should be "".
cDir
String that specifies the default directory. In our example we pass "", which leaves us in the DataEase data directory. If you were using ShellExecureA to open document files, for example, then you'd place your path to the document directory in this parameter.
nShowWin
Flags that specify how an application is to be displayed when it is opened.
If cFileName specifies a document file, the flag is simply passed to the associated application. It is up to the application to decide how to handle it.

0 - Hides the window and activates another window.

3 - Maximizes the specified window.

6 - Minimizes the specified window and activates the next top-level window in the z-order.

9 - Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.

5 - Activates the window and displays it in its current size and position.

3 - Activates the window and displays it as a maximized window.

2 - Activates the window and displays it as a minimized window.

7 - Displays the window as a minimized window. The active window remains active.

8 - Displays the window in its current state. The active window remains active.

4 - Displays a window in its most recent size and position. The active window remains active.

1 - Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

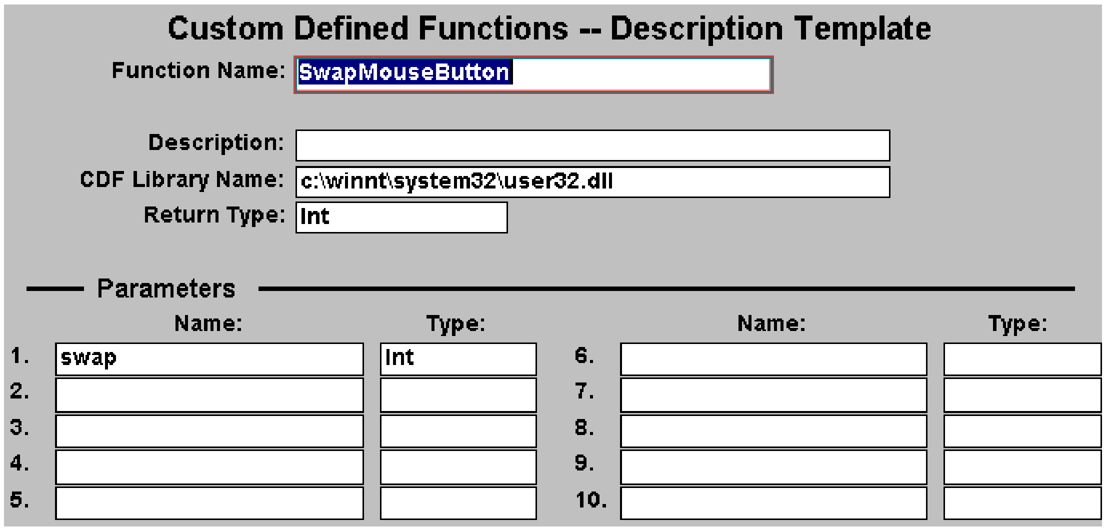
Our example database passes a "1" here.

SwapMouseButton

The MouseSwapButton function h swaps the left and right buttons on a mouse, to keep both lefties and righties happy. Not something which you'd need on your own personal PC, but it could be useful on a general office PC.

Normally you can only swap mouse buttons via the **Settings>>Control Panel>> Mouse>>Buttons** routine. But you can also swap buttons by calling the SwapMouseButton function, which is part of the Windows User32.dll library.

First off you have to register the function with DataEase, as shown below.



Custom Defined Functions -- Description Template

Function Name: SwapMouseButton

Description:

CDF Library Name: c:\winnt\system32\user32.dll

Return Type: Int

Parameters

	Name:	Type:		Name:	Type:
1.	swap	Int	6.		
2.			7.		
3.			8.		
4.			9.		
5.			10.		

Note that the path specified in the CDF Library Name field may need to be changed for your PC. Do a quick search for user32.dll if you're not sure of the right path.

This function needs only one parameter - swap - which is actually a boolean expression, though in DataEase we define it as an integer.

Once the CDF has been registered, there are a number of ways to use it. One simple method would be to add two buttons to the user's startup document. The buttons could be called "Left Hander" and "Right Hander". (If you were feeling really flash, you could call this CDF as part of the user's DataEase Login script, and use a Case statement to set the Left/Right button automatically. Needlessly complex, in my opinion, so back to the two button approach).

The **Left Hand** button would have the following script placed in its Clicked Event:

define "retcode" number .

retcode := SwapMouseButton (0) .

The **right hand** button would have the following script in its Clicked Event:

define "retcode" number .

retcode := SwapMouseButton (1) .

Remember, though, that you can't call CDF's from a DataEase Menu document - yet another reason to dump Menu documents in the bin and use form documents instead.

The mouse-button change is global, by the way, so it will affect the whole PC, not just DataEase.

MemArr32 Library

The MemArr32 DLL Library contains a matched pair of functions – **SetGlobal** and **GetGlobal** - which allows the developer to create a global array of up to 255 elements.

SetGlobal

Description

SetGlobal is the function you use to populate an array element with a value of your choice. Once populated, the value can be accessed with the **GetGlobal** function.

Declaration

SetGlobal(ArrayNumber, TextString)

Parameters

Name	Type	Description
ArrayNumber	int	A value between 0 and 254, representing the array element number.
TextString	string	The value to be stored in this element, enclosed in quotes. You can store numbers as well as text characters. The string has a maximum length of 39 characters. Example: SetGlobal(15, “Store This”)

Return Value

Integer.

GetGlobal

Description

GetGlobal is the function you use to retrieve a value from an array element.

Declaration

GetGlobal(ArrayNumber)

Parameters

Name	Type	Description
ArrayNumber	int	A value between 0 and 254, representing the target array element number from which you wish to retrieve a value.

Return Value

Text. A text string, containing the retrieved value.

Example

MyVariable := GetArray(120) .

Index