



# Language Reference

# **REALbasic®**

Create your own software.™



---

# **REALbasic 2006 Language Reference**

Documentation by Dave Brandt, © 1998-2006 by REAL Software, Inc.  
WASTE text engine © 1993-2006 Marco Piovanelli  
All rights reserved.  
Printed in U.S.A.

|                           |  |
|---------------------------|--|
| Mailing Address           | REAL Software, Inc.<br>1705 South Capital of Texas Highway<br>Suite 310<br>Austin, TX 78746                      |
| Web Site                  | <a href="http://www.realsoftware.com">http://www.realsoftware.com</a>  |
| ftp Site                  | <a href="ftp://ftp.realsoftware.com">ftp://ftp.realsoftware.com</a>  |
| Support                   | REALbasic Feedback at the REAL Software web site.  |
| Bugs/<br>Feature Requests | Submit via REALbasic Feedback at the REAL Software web site.   |
| Database Plug-<br>ins     | REALbasic CD-ROM; the most current versions at<br><a href="http://www.realsoftware.com">www.realsoftware.com</a> |
| Sales                     | <a href="mailto:sales@realsoftware.com">sales@realsoftware.com</a>   |
| Phone                     | 512-328-REAL (7325)  |
| Fax                       | 512-328-7372   |

Version 2006R1, January, 2006.

# Introduction

The *Language Reference* gives you fast access to information about all the REALbasic language elements: objects, keywords, operators, constants, error messages, runtime exceptions, built-in methods and functions, and so forth. Wherever applicable, the properties and methods belonging to an object are documented, along with cross-references to related objects.

There is also a *Quickstart*, a *Tutorial*, and *User's Guide* which teach REALbasic programming concepts. Your comments (both favorable and unfavorable) are welcome. Send them to docs@realsoftware.com.

# Conventions

Property names in **bold** are read-only at runtime. This means that your code can read their values but cannot change their values.

Optional Parameters in syntax statements are surrounded by brackets- [ ].

Items in blue underline are hypertext links to other parts of the document.

In some cases, code that is irrelevant to the example is omitted for sake of clarity. Omitted code is indicated by bullets.

# The Class Hierarchy

Objects in REALbasic are arranged in a hierarchy. This means that an object inherits the properties, methods, and events of its super class (the classes above it in the hierarchy) and may contain properties, methods, and events that are unique to the object. A [PushButton](#) for example, is a subclass of a [RectControl](#), which is a subclass of [Control](#), which is a subclass of [Object](#) (the base class for all objects). You can only access their subclasses. In the Language Reference, each class's Super class is given; the properties, methods, and events of its super class (and all other classes above it in the hierarchy) are available to that class.

You can view the class hierarchy in any Window Editor in the IDE. Right+click (or Control-Click on Macintosh) to display the Window Editor contextual menu and choose Add. The full class hierarchy is shown as a multi-level hierarchical menu, with subclasses of [Object](#) at the top.

# Cross-Platform Development

## Using Target Flags

You can develop your application on one platform and compile it for up to three: Windows, Macintosh (classic and Mac OS X), and Linux. If your application will run on more than one platform, you can use the following global [Boolean](#) constants to determine which type of code is currently executing.

| Boolean Constant                   | Description  |
|------------------------------------|--|
| <a href="#">DebugBuild</a>         | The application is running within the REALbasic application, i.e., from clicking the Run button in the IDE toolbar.  |
| <a href="#">RBVersion</a>          | Returns the version of REALbasic that is being compiled. You can use this in an expression that evaluates to <a href="#">True</a> or <a href="#">False</a> to determine which version of REALbasic is compiling the application. |
| <a href="#">TargetBigEndian</a>    | The compiled application is running on a machine and OS that uses the Big Endian byte ordering.  |
| <a href="#">TargetCarbon</a>       | The compiled application is currently running Carbon/Mac OS X code.  |
| <a href="#">TargetHasGUI</a>       | The application has a GUI, i.e., it is not a <a href="#">ConsoleApplication</a> or a <a href="#">ServiceApplication</a> .  |
| <a href="#">TargetLinux</a>        | The compiled application is running Linux code.  |
| <a href="#">TargetLittleEndian</a> | The compiled application is running on a machine and OS that uses the Little Endian byte ordering.   |
| <a href="#">TargetMachO</a>        | The compiled application is running on Mac OS X, running object code in the format for the Mach kernel.  |
| <a href="#">TargetMacOS</a>        | The compiled application is currently running Macintosh code, PPC 'classic', or Mac OS X, in either the MachO or PEF formats.  |
| <a href="#">TargetMacOSClassic</a> | The compiled application is currently running PowerPC code within a 'classic' Mac OS supported by REALbasic (Mac OS 8-9).  |
| <a href="#">TargetWin32</a>        | The compiled application is currently running Win32 code.  |

## Conditional Compilation

If some of your code should be compiled only for a particular target platform (such as AppleScript-specific code or [Declare](#) statements), you can use [#If](#) statements to compile that code only for the appropriate platform. Use the following structure:

```
#If TargetBoolean
//OS-specific code
//included in built app
//only for target platform
#else (optional)
(optional code)
#elseif TargetBoolean // (optional)
//(optional) Other OS-specific code
#endif
```

---

## Language Reference

The following is an alphabetical listing of the commands available in the REALbasic language.

## - Operator

Used to find the difference between two numbers or to negate the sign of a number.

### Syntax

**result=expression1 - expression2**

| Part        | Type   | Description  |
|-------------|--------|--|
| result      | Number | The difference between <i>expression1</i> and <i>expression2</i> . |
| expression1 | Number | Any numeric expression.  |
| expression2 | Number | Any numeric expression.  |

or

**result=-expression**

| Part       | Type   | Description                               |
|------------|--------|---|
| result     | Number | <i>Expression</i> with the opposite sign. |
| expression | Number | Any numeric expression.                   |

### Notes

You can use [Operator Subtract](#) to define the subtraction operation for custom classes and [Operator Negate](#) to define the negation operation.

### Example

This example stores the difference between two numbers in a variable:

```
Dim x as Integer  
x=50-30 //x is 20
```

### See Also

[Operator Subtract](#), [Operator Negate](#) functions.

---

## " " Literal

Used to represent a [String](#) literal.

### Syntax

**"string"**

| Part   | Type                            | Description             |
|--------|---------------------------------|-------------------------|
| string | <a href="#">String</a> literal. | A string literal value. |

### Note

To write a [string](#) literal, just enclose the string in double quote marks. To write an empty [string](#), write double quote marks with no space between them.

**Example** This example concatenates two [strings](#) and stores the result in a variable:

```
Dim name as String
name="REAL" + "basic" //name = "REALbasic"
```

**See Also** [Str](#) function.

## #If...#Endif Statement

Used to control conditional compilation.

**Syntax** **#If** *TargetBoolean* [**Then**]

```
//OS specific code
[#Else]
//Other OS-specific code
[#Elseif TargetBoolean]
//Other OS-specific code for this target platform
#Endif
```

**OR**

**#If** *TargetBoolean* **Then** OS-specific code

| Part          | Type   | Description   |
|---------------|--|---|
| TargetBoolean | <a href="#">Boolean</a> constant or <a href="#">Boolean</a> constant expression. | Constant or expression that evaluates to a boolean constant used to determine the operating system that will include the code that follows. The <a href="#">DebugBuild</a> , <a href="#">RBVersion</a> , <a href="#">TargetBigEndian</a> , <a href="#">TargetCarbon</a> , <a href="#">TargetHasGUI</a> , <a href="#">TargetLinux</a> , <a href="#">TargetLittleEndian</a> , <a href="#">TargetMacOS</a> , <a href="#">TargetMachO</a> , <a href="#">TargetMacOSClassic</a> , or <a href="#">TargetWin32</a> , constants are used. |

**Notes**

**#If** statements can be written on one line if there are no **#Else** or **#Elseif** clauses. In this case, the **Then** keyword is required and the **#endif** is not part of the syntax.

Use conditional compilation to isolate platform-specific statements such as API calls, AppleEvent routines, or console application routines that are specific to Windows, Mac OS X, or Linux. The code following the **#if** statement is included only in the build for that operating system. [TargetMacOSClassic](#), and [TargetCarbon](#) are mutually exclusive subsets of [TargetMacOS](#).

## #Pragma Directives

---

The optional #ElseIf clause enables you to use a template such as this for handling all cases:

```
#If TargetWin32
//Windows specific code here
#Elseif TargetMacOS
//Macintosh code goes here.
#Elseif TargetLinux
//Linux code goes right here.
#Endif
```

The *TargetBoolean* parameter must be either a [Boolean](#) constant or a [Boolean](#) constant expression that evaluates to [True](#) or [False](#). For example, you can use [RBVersion](#) or [RBVersionString](#) in a boolean expression that determines the user's version of REALbasic.

### Example

The following example assigns values to the (user-defined) Separator property of the [App](#) class in its Open event handler. It will be used to specify full pathnames that are correct on Windows, Macintosh and Linux.

```
#If TargetWin32
Separator=" "
#Elseif TargetMacOS
Separator=":"
#Elseif TargetLinux
Separator="/"
#Endif
```

### See Also

[Declare](#) statement, [AppleEvent](#) class; [DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants.

---

## #Pragma Directives

Used to speed up code execution by turning off certain automatic tasks. When a pragma directive is used, the automatic task remains off until the end of the method in which it is called (or is enabled by calling it again and specifying [True](#)), but not in other methods that might be called within the original method.

### Syntax

**#pragma Directive**

or

## #pragma Directive True or False

| Directive              | Description  |
|------------------------|--|
| BackgroundTasks        | Enables or disables auto-yield to background threads. In addition to the pragma directive, specify <a href="#">True</a> or <a href="#">False</a> . Setting this directive to <a href="#">False</a> is the same as using disableBackgroundTasks)  |
| BoundsChecking         | Enables or disables bounds checking. In addition to the pragma directive, specify <a href="#">True</a> or <a href="#">False</a> . Specifying <a href="#">False</a> is the same as using disableBoundsChecking.   |
| DisableAutoWaitCursor  | Used to disable the automatic display of the wait cursor (or watch cursor). In versions 5.0 and above, the scope of this pragma is no longer global. The wait cursor will be disabled in the method that calls the pragma until the method ends. For example, if DisableAutoWaitCursor is called in a Pushbutton that runs a loop, the wait cursor will be disabled only until the loop runs and the Action event has completed. |
| DisableBackgroundTasks | Used to turn off automatic background task handling for code after the #pragma. It prevents REALbasic from yielding time back to the operating system, other applications, and threads. It can speed up very processor-intensive operations but prevents REALbasic from displaying the Watch cursor, may halt normal background updating of interface elements, and prevents other threads from executing.                       |
| DisableBoundsChecking  | Used to turn off array bounds checking on array index values in code after the #pragma.  |
| NilObjectChecking      | Controls whether to automatically check objects for <a href="#">Nil</a> before accessing properties and calling methods. In addition to the pragma directive, specify <a href="#">True</a> or <a href="#">False</a> .  |
| StackOverflowChecking  | Controls whether to check for stack overflows. In addition to the pragma directive, specify <a href="#">True</a> or <a href="#">False</a> .  |
| X86CallingConvention   | Accepts either StdCall or CDecl, not <a href="#">True</a> or <a href="#">False</a> . Allows you to determine which calling convention a method will be compiled with on x86. This allows you to write callback functions on Windows, which typically require the StdCall calling convention.   |

## &b Literal

---

### Notes

Four pragma directives require that you pass [True](#) or [False](#) to turn the option on or off. Since these pragmas can be enabled or disabled, you can now do things like disable background tasks for an inner loop, but leave them enabled for the outer loop:

```
For y = 0 To Height  
    #pragma BackgroundTasks False  
    For x = 0 To Width  
        //...process some pixels...  
    Next  
    #pragma BackgroundTasks True  
Next
```

---

## &b Literal

Used to represent binary literals.

### Syntax

**&b***BinaryNumber*

| Part         | Type          | Description             |
|--------------|---------------|-------------------------|
| BinaryNumber | Binary number | A binary literal value. |

### Note

To write a binary literal, precede the value with **&b**.

### Examples

This example shows a binary constant:

```
&b10010101
```

---

## &c Literal

Used to represent a color literal.

### Syntax

**&c***RRGGBB*

| Part | Type                   | Description   |
|------|------------------------|---|
| RR   | <a href="#">String</a> | The amount of Red in the color, in hexadecimal. The range is from 00 to FF.   |
| GG   | <a href="#">String</a> | The amount of Green in the color, in hexadecimal. The range is from 00 to FF. |
| BB   | <a href="#">String</a> | The amount of Blue in the color, in hexadecimal. The range is from 00 to FF.  |

**Notes**

**&c** provides an alternate way to specify a [Color](#) using the [RGB](#) model. It provides the same functionality as the [RGB](#) function, except that you specify the amount of each primary color in hexadecimal (00-FF) rather than decimal (0-255). In the Code Editor, the parameters *RR*, *GG*, and *BB* appear in the color that they represent. A color can also be specified using the [HSV](#) and [CMY](#) models.

**Example**

The following example allows the user to select a color that will be used as the fillcolor in a [Rectangle](#) control.

```
Dim c as Color
Dim b as Boolean
c=&cFFFFFF //set the default color shown in color picker
b=SelectColor(c, "Select a Color")
Rectangle1.FillColor=
```

**See Also**

[CMY](#), [HSV](#), [RGB](#) functions; [Color](#) class.

## &h Literal

Used to represent a hexadecimal number.

**Syntax**

**&h*HexNumber***

| Part      | Type               | Description                  |
|-----------|--------------------|------------------------------|
| HexNumber | Hexadecimal number | A hexadecimal literal value. |

**Note**

To write a hexadecimal literal, precede the value with **&h**.

**Example**

This example shows a hexadecimal constant:

```
&hA4
```

**See Also**

[Hex](#) function.

## &o Literal

Used to represent octal literals.

## ' Operator

---

### Syntax

#### **&oOctalNumber**

| Part        | Type         | Description            |
|-------------|--------------|------------------------|
| OctalNumber | Octal number | A octal literal value. |

### Note

To write an octal literal, precede the value with **&o**.

### Example

This example shows an octal constant:

```
&o527
```

### See Also

[Oct](#) function.

## ' Operator

---

Used to add comments to code.

### Syntax

#### **' text string**

### Notes

REALbasic's compiler will ignore any comments you enter following the ' operator. Comments are not included in stand-alone applications.

The [//](#) operator or [Rem](#) statement can be used to comment code as well. Comments can appear on separate lines or on the same line as executable code, provided they are to the right of any code that will execute. Valid comments appear in the Code Editor in red.

The Control-' keyboard shortcut (Command-' on Macintosh) toggles the selected lines of code between commented out and 'live' states. If the insertion point is somewhere in the middle of the line when you press Control-' or Command-' , the entire line is commented out. If you only want to comment out the text to the right of the insertion point, press, ' or use [//](#) or [Rem](#).

### Example

The following code uses comments to document the results of [Boolean](#) comparisons:

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean
Dim d As Boolean
a=True
b=False
d=a Or b ' Evaluates to True
d=b And c ' Evaluates to False
```

### See Also

[Rem](#) statement; [//](#) operator.

## \* Operator

Used to multiply two numbers.

### Syntax

**result=expression1 \* expression2**

| Part        | Type   | Description                                 |
|-------------|--------|---|
| result      | Number | The product of expression1 and expression2. |
| expression1 | Number | Any numeric expression.                     |
| expression2 | Number | Any numeric expression.                     |

### Notes

You can use [Operator\\_Multiply](#) to define the \* operator for classes.

### Examples

This example stores the product of two numbers in a variable:

```
Dim x as Integer
x=50*30 //x is 1500
```

### See Also

[Operator\\_Multiply](#) function.

## + Operator

Used to sum two numbers or concatenate two [string](#) values.

### Syntax

The + operator has two syntaxes:

**result=expression1 + expression2**

| Part        | Type   | Description                             |
|-------------|--------|---|
| result      | Number | The sum of expression1 and expression2. |
| expression1 | Number | Any numeric expression.                 |
| expression2 | Number | Any numeric expression.                 |

**result=expression1 + expression2**

| Part        | Type                   | Description                                       |
|-------------|------------------------|---|
| result      | <a href="#">String</a> | The concatenation of expression1 and expression2. |
| expression1 | <a href="#">String</a> | Any string expression.                            |
| expression2 | <a href="#">String</a> | Any string expression.                            |

### Notes

You can use [Operator\\_Add](#) to define the + operator for custom classes.

## / Operator

---

**Examples** This example adds several numbers together:

```
Dim x as Integer  
x=1+2+3
```

This example concatenates two strings and stores the result in a variable:

```
Dim name as String  
name="REAL"+"basic" //name = REALbasic
```

**See Also** [Operator Add](#), [Str](#), [Val](#) functions.

## / Operator

Used to perform floating point division between two numbers.

**Syntax** *result=expression1 / expression2*

| Part        | Type   | Description                                  |
|-------------|--------|--|
| result      | Number | The division of expression1 and expression2. |
| expression1 | Number | Any numeric expression.                      |
| expression2 | Number | Any numeric expression.                      |

**Notes** Use this operator when you require division that considers the decimal portion of the expressions. You can use [Operator Divide](#) to define the / operator for classes.

**Examples** This example stores the quotient of two numbers in a variable:

```
Dim x as Double  
x=50.56/30.34 //x is 1.6664469
```

**See Also** \, [Mod](#) operators; [Operator Divide](#) function.

## // Operator

Used to add comments to code.

**Syntax** *// text string*

**Notes** REALbasic's compiler will ignore any comments you enter following the // operator. Comments are not included in stand-alone applications.

The ' (apostrophe) or [Rem](#) statement can be used to comment code as well. Comments can appear on separate lines or on the same line as executable code, provided they are to the right of any code that will execute. Valid comments appear in the Code Editor in red.

The Control-' keyboard shortcut (Command-' on Macintosh) toggles the selected lines of code between commented out and 'live' states. The Code Editor toolbar has a Comment button which switches the selected line or lines into a comment line. When the selected line or lines are commented out, this button changes to Uncomment. Clicking Uncomment changes the lines into executable code.

## Example

The following code uses comments to document the results of [Boolean](#) comparisons:

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean
Dim d As Boolean
a=True
b=False
d=a Or b // Evaluates to True
d=b And c // Evaluates to False
```

## See Also

[Rem](#) statement; [`](#) operator.

## < Operator

Used to determine whether one quantitative or alphabetic expression is smaller than another. [String](#) comparisons are case-insensitive.

## Syntax

*result=expression1 < expression2*

| Part        | Type  | Description   |
|-------------|---|---|
| result      | <a href="#">Boolean</a>                                 | Returns <a href="#">True</a> if expression1 is less than expression2. |
| expression1 | <a href="#">String</a> , Number or <a href="#">date</a> | Any alphabetic or quantitative expression.                            |
| expression2 | <a href="#">String</a> , Number or <a href="#">date</a> | Any alphabetic or quantitative expression.                            |

## Note

A string is “less than” another string if it is first when the two strings are sorted alphabetically. Use [StrComp](#) to do a case-sensitive [string](#) comparison.

You can use [Operator Compare](#) to define comparisons for classes.

## <= Operator

---

**Example** This example uses the < operator to evaluate a and b.

```
Dim a,b as Integer
If Editfield1.text <> "" and EditField2.text <> "" then
a=Val(Editfield1.text)
b=Val(EditField2.text)
If a<b then
    MsgBox "A is Less Than B!"
else
    Beep
end if
else
    MsgBox "Please enter values into both boxes!"
end if
```

**See Also** [>](#), [>=](#), [<=](#), [≡](#), [<>](#), and [StrComp](#) operators; [Operator Compare](#) function.

## <= Operator

Used to determine whether one alphabetic or quantitative expression is smaller than or equal to another. [String](#) comparisons are case-insensitive.

### Syntax

*result=expression1 <= expression2*

| Part        | Type   | Description   |
|-------------|--|---|
| result      | <a href="#">Boolean</a>                                  | Returns <a href="#">True</a> if expression1 is less than or equal to expression2. |
| expression1 | <a href="#">String</a> , Number, or <a href="#">Date</a> | Any alphabetic or quantitative expression.  |
| expression2 | <a href="#">String</a> , Number, or <a href="#">Date</a> | Any alphabetic or quantitative expression.  |

### Notes

A string is “less than” another string if it is first when the two strings are sorted alphabetically. Use [StrComp](#) to do a case-sensitive [string](#) comparison.

You can use [Operator Compare](#) to define comparisons for classes.

**Examples** This example uses the <= operator to evaluate a and b.

```
Dim a,b as Integer
If Editfield1.text <> "" and EditField2.text <> "" then
a=Val(Editfield1.text)
b=Val(EditField2.text)
If a<=b then
    MsgBox "A is less than or equal to B!"
else
    Beep
end if
else
    MsgBox "Please enter values into both boxes!"
end if
```

The following example tests whether a [FolderItem](#) is [Nil](#) before assigning it to the Movie property of a [MoviePlayer](#) control.

```
Dim f As FolderItem
f=GetOpenFolderItem("video/quicktime")
If f<>Nil then
    MoviePlayer1.movie=f.OpenAsMovie
End if
```

**See Also** [>](#), [>=](#), [<](#), [<=](#), and [StrComp](#) operators; [Operator\\_Compare](#) function.

## <> Operator

Used to determine whether one expression is not equal to another. [String](#) comparisons are case-insensitive.

### Syntax

*result=expression1 <> expression2*

| Part        | Type   | Description  |
|-------------|--|--|
| result      | <a href="#">Boolean</a>  | Returns <a href="#">True</a> if expression1 is not equal to expression2. |
| expression1 | <a href="#">String</a> ,<br>Number,<br><a href="#">Object</a> , <a href="#">Color</a> ,<br>or <a href="#">Date</a> | Any expression.  |
| expression2 | <a href="#">String</a> ,<br>Number,<br><a href="#">Object</a> , <a href="#">Color</a> ,<br>or <a href="#">Date</a> | Any expression.  |

## = Operator

---

### Notes

The data types of **expression1** and **expression2** must match. You can make comparisons between objects of any data type and between objects. If you compare objects, **<>** compares their references, not their contents. For example, when you compare two **FolderItems**, **<>** determines whether they have the same reference, not whether they point to the same file.

The **<>** operator is also used with objects to test whether they are [Nil](#). For example, when you use the [NewMemoryBlock](#) function to create a new object, test whether the new object is [Nil](#) before proceeding.

Use [StrComp](#) to do a case-sensitive [string](#) comparison.

You can use [Operator Compare](#) to define comparisons for classes.

### Example

The following example uses **<>** to test for blank [EditFields](#).

```
Dim a,b as Integer
If Editfield1.text <> "" and EditField2.text <> "" then
a=Val(Editfield1.text)
b=Val(EditField2.text)
If a<b then
    MsgBox "A is Less Than B!"
else
    Beep
end if
else
    MsgBox "Please enter values into both boxes!"
end if
```

### See Also

[>](#), [>=](#), [<](#), [<=](#), [=](#), and [StrComp](#) operators; [Operator Compare](#) function.

---

## = Operator

Used to determine whether one expression is equal to another or to assign a value to a property or variable. [String](#) comparisons are case-insensitive.

### Syntax

**result=expression1 = expression2**

| Part        | Type  | Description  |
|-------------|---|--|
| result      | <a href="#">Boolean</a>   | Returns <a href="#">True</a> if expression1 is equal to expression2. |
| expression1 | <a href="#">String</a> , <a href="#">Number</a> , <a href="#">Boolean</a> , <a href="#">Color</a> , or <a href="#">Object</a> | Any expression.  |
| expression2 | <a href="#">String</a> , <a href="#">Number</a> , <a href="#">Boolean</a> , <a href="#">Color</a> or <a href="#">Object</a>   | Any expression.  |

***result=value***

| Part         | Type | Description   |
|--------------|------|---|
| result       | Any  | The property, variable, array, or array element that is assigned <i>value</i> . |
| <i>value</i> | Any  | The value assigned to <i>result</i> .   |

**Notes**

The data types of *expression1* and *expression2* must match. You can make comparisons between objects of any data type and between objects. If you compare objects, = compares their references, not their contents. For example, when you compare two FolderItems, = determines whether they have the same reference, not whether they point to the same file.

Use [StrComp](#) to do a case-sensitive [string](#) comparison.

You can use [Operator Compare](#) to define comparisons for classes.

**Example**

The following example tests whether two strings are equal.

```
Dim a,b as String
If Editfield1.text <> "" and EditField2.text <> "" then
    a>Editfield1.text
    b>EditField2.text
    If a=b then
        MsgBox "A is equal to B!"
    else
        Beep
    end if
    else
        MsgBox "Please enter values into both boxes!"
    end if
```

**See Also**

[>](#), [>=](#), [<](#), [<=](#), [<>](#), and [StrComp](#) operators; [Operator Compare](#) function.

## > Operator

Used to determine whether one alphabetic or quantitative expression is larger than another. [String](#) comparisons are case-insensitive.

**Syntax**

*result=expression1 > expression2*

| Part        | Type   | Description   |
|-------------|--|---|
| result      | <a href="#">Boolean</a>                                  | Returns <a href="#">True</a> if expression1 is larger than expression2. |
| expression1 | <a href="#">String</a> , Number, or <a href="#">Date</a> | Any alphabetic or quantitative expression.                              |

## >= Operator

---

| Part        | Type   | Description                                |
|-------------|--|--|
| expression2 | <a href="#">String</a> , Number, or <a href="#">Date</a> | Any alphabetic or quantitative expression. |

### Notes

The data types of *expression1* and *expression2* must match.

A string is “greater than” another string if it is last when the two strings are sorted alphabetically. Use [StrComp](#) to do a case-sensitive [string](#) comparison.

You can use [Operator Compare](#) to define comparisons for classes.

### Example

The following example tests whether one number is larger than another.

```
Dim a,b as Integer
If Editfield1.text <> "" and EditField2.text <> "" then
a=Val(Editfield1.text)
b=Val(EditField2.text)
If a>b then
    MsgBox "A is Greater than B!"
else
    Beep
end if
else
    MsgBox "Please enter values into both boxes!"
end if
```

### See Also

[>=](#), [<](#), [<=](#), [=](#), [<>](#), and [StrComp](#) operators; [Operator Compare](#) function.

---

## >= Operator

Used to determine whether one alphabetic or quantitative expression is greater than or equal to another. [String](#) comparisons are case-insensitive.

### Syntax

*result=expression1 >= expression2*

| Part        | Type  | Description   |
|-------------|---|---|
| result      | <a href="#">Boolean</a>                                 | Returns <a href="#">True</a> if expression1 is larger than or equal to expression2. |
| expression1 | <a href="#">String</a> , Number or <a href="#">Date</a> | Any alphabetic or quantitative expression.  |
| expression2 | <a href="#">String</a> , Number or <a href="#">Date</a> | Any alphabetic or quantitative expression.  |

### Notes

The data types of *expression1* and *expression2* must match.

A string is “greater than” another string if it is first when the two strings are sorted alphabetically. Use [StrComp](#) to do a case-sensitive [string](#) comparison.

You can use [Operator Compare](#) to define comparisons for classes.

**Example**

The following example tests whether one number is greater than or equal to another.

```
Dim a,b as Integer
If Editfield1.text <> "" and EditField2.text <> "" then
a=Val(Editfield1.text)
b=Val(EditField2.text)
If a>=b then
    MsgBox "A is greater than or equal to B!"
else
    Beep
end if
else
    MsgBox "Please enter values into both boxes!"
end if
```

**See Also**

[>](#), [<=](#), [<](#), [≡](#), [<>](#), and [StrComp](#) operators; [Operator Compare](#) function.

## \ Operator

Used to perform [Integer](#) division between two numbers.

**Syntax**

**result=expression1 \ expression2**

| Part        | Type   | Description  |
|-------------|--------|--|
| result      | Number | The <a href="#">Integer</a> division of expression1 and expression2. |
| expression1 | Number | Any numeric expression.  |
| expression2 | Number | Any numeric expression.  |

**Notes**

Use this operator when you require division that **does not** consider the decimal portion of the expressions.

You can use [Operator IntegerDivide](#) to define the \ operator for classes.

**Examples**

This example stores the result of a division of two numbers in a variable:

```
Dim x as Integer
x=50.56\30.34 //x is 1
```

**See Also**

[/](#), [Mod](#) operators; [Operator IntegerDivide](#) function.

## ^ Operator

Used to perform exponentiation.

**Syntax**

**result=expression1<sup>^</sup>expression2**

| Part        | Type   | Description   |
|-------------|--------|---|
| result      | Number | The result of raising <i>expression1</i> to the power of <i>expression2</i> . |
| expression1 | Number | Any numeric expression.   |
| expression2 | Number | Any numeric expression.   |

**Notes**

The ^ operator's precedence is higher than negation but lower than [IsA](#) or the dot operator.

Use the [Operator\\_Power](#) function to define the ^ operator for classes.

**Examples**

These examples illustrate the usage of the ^ operator.

```
Dim i as Double  
i=2^2 //returns 4  
i=(2+3)^(4-2) //returns 25  
i=3.75^3.5 //returns 102.12
```

**See Also**

[\\*\\_operator](#), [Operator\\_Power](#) function.

---

## \_ Keyword

The underscore character, “\_”, is used as the line continuation character to indicate that a line of code is being continued on the next printed line in the Code Editor.

**Syntax**

\_  
(used as the last character in a line)

**Notes**

Use the underscore character as the last character in a printed line of text in the Code Editor to indicate that you are continuing the line of code on the next printed line in the Code Editor. The compiler will then treat the second printed line as a continuation of the first line. When you use the \_ keyword, the Code Editor indents the next line automatically, as is shown in the example.

If you need to break up a text string into two lines in the Code Editor, place the \_ keyword outside the quoted string, as in the example.

To enter the `_` character and move to the next line, press Ctrl+Enter on Windows and Linux; on Macintosh, press Option-Return.

### Example

The following example uses the line continuation character to break a long string into two lines in the Code Editor. The `_` keyword is used outside the quoted string and the `+` operator concatenates the two strings.

```
Dim SaveChangesDialog as New MessageDialog
```

```
SaveChangesDialog.Explanation="Hint: If you don't save, you will " _  
    +"lose your work since your last Save."
```

## A Method with a ParamArray Cannot have any Optional Parameters Error

---

## A ParamArray cannot be Passed by Reference Error

You tried to declare a parameter using both the `ParamArray` and `ByRef` keywords. Parameters declared as `ParamArray` cannot be passed by reference.

## A ParamArray cannot have a Default Value Error

When you declared a parameter using the `ParamArray` keyword, you gave it a default value. This is not valid.

### Example

The following method declaration is not valid:

```
Sub AddNumbers(ParamArray Nums as Integer =10)
```

You must remove the assignment statement at the end of the declaration.

### See Also

[ParamArray](#) keyword.

## A ParamArray's Element Type may not be Another Array Error

You tried to use the [ParamArray](#) keyword with an array parameter in a method declaration.

**Example** This method declaration uses the [ParamArray](#) keyword with an array parameter. There's no need to do this. You can pass an indefinite number of elements simply by declaring the parameter [ParamArray](#) OR passing an array to the method. You cannot use both techniques at once.

```
Sub myMethod(ParamArray b() As String)
```

**See Also** [ParamArray](#) keyword; [Dim](#) statement.

---

## A Parameter passed by Reference Cannot have a Default Value Error

In the method declaration, you declared a parameter [ByRef](#) but also tried to assign it a default value.

**Example** The following parameter declaration is not allowed:

```
ByRef a As Integer = 5
```

You can either remove the [ByRef](#) keyword or remove the default value assignment.

**See Also** [ByRef](#) keyword.

---

## Abs Function

Returns the absolute value of the number specified.

**Syntax** **result=Abs(value)**

| Part   | Type                   | Description                                |
|--------|------------------------|--|
| value  | <a href="#">Double</a> | The number you wish the absolute value of. |
| result | <a href="#">Double</a> | The absolute value of value.               |

**Notes**

The **Abs** function returns the positive equivalent of the value specified.

**Examples**

These examples use the **Abs** function to return the absolute values of the numbers specified.

```
Dim d as Double
d=Abs(23.9) //returns 23.9
d=Abs(-23.9) //returns 23.9
```

**See Also**

[Sign](#) function.

## Acos Function

Returns the arccosine of the value specified. The arccosine is the angle whose cosine is value. The returned angle is given in radians.

**Syntax**

**result=Acos(value)**

| Part   | Type                   | Description                           |
|--------|------------------------|---------------------------------------|
| result | <a href="#">Double</a> | The arc cosine of value.              |
| value  | <a href="#">Double</a> | The value you want the arc cosine of. |

**Notes**

The **Acos** function returns the angle (in radians) of the cosine passed to it. To convert the result from radians to degrees, multiply it by 180/PI.

**Examples**

This example uses the **Acos** function to return the arc cosine of a number.

```
Dim d as Double
Const PI=3.14159265358979323846264338327950
d=Acos(.5) //returns 1.0471976
d=Acos(.5)*180/PI //returns 60
```

**See Also**

[Cos](#) function.

## AddressBook Class

Provides access to the Mac OS X Address Book. You can read from or write to the Address Book.

**Super Class**

[Object](#)

### Properties

| Name              | Type                               | Description   |
|-------------------|------------------------------------|---|
| CurrentUser       | <a href="#">AddressBookContact</a> | Get's the current user's "me" entry.  |
| HasUnsavedChanges | <a href="#">Boolean</a>            | If <a href="#">True</a> , a Save is needed. Call the Save method to save the changes. |

### Methods

| Name     | Parameters                                  | Description  |
|----------|---|--|
| Add      | Record as <a href="#">AddressBookRecord</a> | Adds the record to the Address Book.   |
| Contacts |   | Returns an array of all the contacts in the Address Book as an <a href="#">AddressBookContact</a> .  |
| Find     | Query as <a href="#">String</a>             | Returns an <a href="#">AddressBookRecord</a> . Find performs a textual search on all Contacts fields for the string passed. Returns an array of all matching Contacts records. |
| Groups   |   | Returns an array of all the groups as an <a href="#">AddressBookGroup</a> .  |
| Remove   | Record as <a href="#">AddressBookRecord</a> | Removes the record from the Address Book.  |
| Save     |   | Saves changes to the Address Book. Returns a <a href="#">Boolean</a> , <a href="#">True</a> if the save is successful.   |

### Examples

You can create an instance of this class in either of two ways. Use the [New](#) operator in the usual way or call the AddressBook property of the [System](#) object.

```
Dim book as New AddressBook  
// or  
Dim book as AddressBook  
book=System.AddressBook
```

You can obtain the contacts using the Contacts method. This code populates an the array of contacts from the user's Address Book.

```
Dim book as AddressBook  
book=System.AddressBook  
  
Dim Contacts() as AddressBookContact  
Contacts=book.Contacts
```

Once you have the array of contacts, you can get its size with the [Ubound](#) function and use the properties of the [AddressBookContact](#) class to extract any field. The [AddressBookData](#) class is able to convert to a [String](#) automatically in most cases. If the field does not contain multiple values, it will return the string.

For example, you can loop through the array of [AddressBookContacts](#) and get the company names. This code puts the company names in a [ListBox](#).

```
Dim i as Integer
Dim book as AddressBook
book=System.AddressBook

Dim Contacts() as AddressBookContact
Contacts=book.Contacts
For i =0 to Ubound(Contacts)
    ListBox1.Addrow Contacts(i).CompanyName
Next
```

You can get the current user's "me" entry from this and extract a field:

```
Dim book as AddressBook
book=System.AddressBook
Msgbox book.CurrentUser.FirstName
```

Here is another way to get a property. It uses the Value method of the [AddressBookData](#) method.

```
Dim book as AddressBook
book=System.AddressBook
Msgbox book.CurrentUser.FirstName.Value
```

**See Also** [AddressBookAddress](#), [AddressBookContact](#), [AddressBookData](#), [AddressBookGroup](#), [AddressBookRecord](#) classes.

## AddressBookAddress Class

Holds the fields making up an address for a Mac OS X Address Book.

**Super Class** [Object](#)

### Properties

| Name          | Type                   | Description  |
|---------------|------------------------|--|
| City          | <a href="#">String</a> | The City entry for the Address.                            |
| Country       | <a href="#">String</a> | The Country entry for the address.                         |
| CountryCode   | <a href="#">String</a> | The ISO country code for the address. Two characters only. |
| State         | <a href="#">String</a> | The State entry for the Address.                           |
| StreetAddress | <a href="#">String</a> | The Street Address for the Address.                        |

## AddressBookContact Class

---

| Name | Type                   | Description                         |
|------|------------------------|-------------------------------------|
| Zip  | <a href="#">String</a> | The Zip Code entry for the address. |

### Methods

| Name  | Parameters  | Description  |
|-------|---|--|
| Value | PropertyName as <a href="#">String</a>  | Returns as a <a href="#">Variant</a> the requested property value. |
| Value | PropertyName as <a href="#">String</a> ,<br><a href="#">Assigns</a> newValue as <a href="#">Variant</a> | Assigns the passed new value to the specified property.            |

### Example

This example gets the current user's address and puts it in an [EditField](#).

```
Dim book as AddressBook
Dim address as AddressBookAddress
Dim data as AddressBookData
Dim s as String

book=System.AddressBook
data=book.CurrentUser.Addresses
If data.count >0 then
    Address=data.value(0)
    s=address.StreetAddress+Address.City+", "+Address.state+ " "+address.Zip
else
    s="Count is zero!"
end if
EditField1.text=s
```

### See Also

[AddressBook](#), [AddressBookContact](#), [AddressBookData](#), [AddressBookGroup](#), [AddressBookRecord](#) classes.

---

## AddressBookContact Class

Holds a Contact record in a Mac OS X Address Book.

**Super Class** [AddressBookRecord](#)

### Properties

| Name           | Type                            | Description           |
|----------------|---------------------------------|-----------------------|
| Addresses      | <a href="#">AddressBookData</a> | The Addresses entry.  |
| AIMScreenNames | <a href="#">AddressBookData</a> | AIMScreenNames entry. |
| Birthday       | <a href="#">AddressBookData</a> | Birthday entry.       |

| Name              | Type                            | Description                         |
|-------------------|---------------------------------|-------------------------------------|
| CompanyName       | <a href="#">AddressBookData</a> | Company Name entry.                 |
| EmailAddresses    | <a href="#">AddressBookData</a> | Email Addresses entries.            |
| FirstName         | <a href="#">AddressBookData</a> | FirstName entry.                    |
| HomePage          | <a href="#">AddressBookData</a> | Home page entry.                    |
| ICQNumbers        | <a href="#">AddressBookData</a> | ICQNumbers entry.                   |
| JabberScreenNames | <a href="#">AddressBookData</a> | Jabber screen names entry.          |
| JobTitle          | <a href="#">AddressBookData</a> | Job title entry.                    |
| LastName          | <a href="#">AddressBookData</a> | Last Name entry.                    |
| MiddleName        | <a href="#">AddressBookData</a> | Middle Name entry.                  |
| MSNScreenNames    | <a href="#">AddressBookData</a> | MSN screen names entry.             |
| Note              | <a href="#">AddressBookData</a> | Note entry.                         |
| PhoneNumbers      | <a href="#">AddressBookData</a> | Phone Numbers entries.              |
| VCard             | <a href="#">String</a>          | VCard format of Address Book entry. |
| YahooScreenNames  | <a href="#">AddressBookData</a> | Yahoo screen names entry.           |

## Constructor

| Name        | Parameters                         | Description  |
|-------------|------------------------------------|--|
| Constructor | [vcard as <a href="#">String</a> ] | Creates an AddressBookContact; if the optional vcard is passed, populates it with the data in the vcard. |

## Notes

The methods of the [AddressBookData](#) class give you access to the labels, names, values, and other properties of **AddressBookContact** records. Use the Value method to get and set values.

## Example

This method gets the current user's email addresses. It uses the Count and Value methods of the [AddressBookData](#) class to do so.

```

Dim Book as AddressBook
Dim i as Integer
Dim myContact as AddressBookContact
book=System.AddressBook
myContact=Book.CurrentUser
For i=0 to myContact.emailAddresses.Count-1
  ListBox1.AddRow myContact.emailAddresses.Value(i)
Next

```

The following method sets the value of the FirstName field.

```
Dim Book as AddressBook
Dim i as Integer
Dim myContact as AddressBookContact
book=System.AddressBook
myContact=Book.CurrentUser
myContact.FirstName="Boris"
//or using AddressBookData.Value method
myContact.FirstName.Value="Boris"
```

**See Also** [AddressBook](#), [AddressBookAddress](#), [AddressBookData](#), [AddressBookGroup](#), [AddressBookRecord](#) classes.

---

## AddressBookData Class

Used to manage Address Book fields that store multiple values, such as phone and fax numbers.

**Super Class** [Object](#)

### Methods

| Name        | Parameters   | Description   |
|-------------|--|---|
| ActualLabel | Index as <a href="#">Integer</a>                                       | Returns as a <a href="#">String</a> the label of this data as the system sees it.   |
| Append      | Label as <a href="#">String</a><br>NewValue as <a href="#">Variant</a> | Adds <i>NewValue</i> to the end of the array of property values, identified by <i>Label</i> . For managing multi-value properties such as phone numbers, addresses, email addresses, and so forth. See notes on labels in the description of the Insert method. |
| Count       |  | Returns as an <a href="#">Integer</a> the number of values and labels.  |

| Name   | Parameters   | Description  |
|--------|--|--|
| Insert | Index as <a href="#">Integer</a><br>Label as <a href="#">String</a><br>newValue as <a href="#">Variant</a> | <p>Inserts <i>NewValue</i> identified by <i>Label</i>. Used to insert multi-value properties such as phone numbers, email addresses, physical addresses, and so forth. The label should be any of these constants (or literals) defined on AddressBookData:</p> <p>LabelHome="Home"<br/>LabelWork="Work"<br/>LabelHomeFax="Home Fax"<br/>LabelWorkFax="Work Fax"<br/>LabelOther="Other"<br/>LabelPager="Pager"<br/>LabelMain="Main"</p> <p>For anything other than phone numbers, the only valid labels are LabelHome, LabelWork, and LabelOther. Passing anything else will raise a <a href="#">RuntimeException</a>. The Message property of the exception will explain the valid labels.</p> <p>If the property being assigned a value is a custom property or a property the REALbasic AddressBook implementation does not know about, it will pass in the label given and not manipulate it in any way.</p> |
| Label  | Index as <a href="#">Integer</a>   | Returns the label as a <a href="#">String</a> belonging to the data. The possible labels are: Home, Home Fax, Main, Mobile, Pager, Work Fax, Work. If the label isn't one of these, it returns the empty string.   |
| Name   |  | Returns as a <a href="#">String</a> the property name that this data is referring to. For example, "FirstName".  |
| Remove | Index as <a href="#">Integer</a>   | Removes the value specified by <i>Index</i> . This method is for multi-value properties such as phone number, addresses, email addresses, and so forth.  |
| Text   |  | Returns as a <a href="#">String</a> the data this object refers to. If it is a property that holds one value such as AddressBookContact.FirstName, it returns the value. If it contains multiple strings, it returns the first string.   |
| Value  | Index as <a href="#">Integer</a><br>=0   | Returns the value as a <a href="#">Variant</a> , which can be either a <a href="#">String</a> or an <a href="#">AddressBookAddress</a> . The array of values is zero-based.  |
| Value  | Index as <a href="#">Integer</a> =0,<br><a href="#">Assigns</a> newValue as <a href="#">Variant</a>        | Assigns the value passed to the specified index entry.   |

## AddressBookGroup Class

---

**Example** The **AddressBookData** class can be used to get more information about a field ([AddressBookContact](#) properties). The Name method returns the property it represents. The Label method will return what kind of property it is, like Home, Work, etc. The Label property is relevant only for **AddressBookData** objects that can contain multiple values.

This example gets the user's phone numbers and displays them in a two-column [Listbox](#). The first column displays the label ("Home", "Work", "Mobile", etc.) and the second column shows the phone number.

```
Dim book as New AddressBook
Dim data as AddressBookData
Dim c,i as Integer

data = Book.CurrentUser.PhoneNumbers
c=Data.count-1
If data.Count > 0 then
    For i=0 to c
        ListBox1.Addrow Data.Label(i)
        ListBox1.Cell(i,1)=Data.Value(i)
    next
Else
    MsgBox "No phone numbers!"
End if
```

This method uses the Value method to get the current user's first email address.

```
Dim book as AddressBook
Dim myContact as AddressBookContact
book=System.AddressBook
myContact=Book.CurrentUser
Msgbox myContact.emailAddresses.Value(0)
```

**See Also** [AddressBook](#), [AddressBookAddress](#), [AddressBookContact](#), [AddressBookGroup](#), [AddressBookRecord](#) classes.

---

## AddressBookGroup Class

Manages an Address Book group.

**Super Class** [AddressBookRecord](#)

## Properties

| Name | Type                   | Description            |
|------|------------------------|------------------------|
| Name | <a href="#">String</a> | The name of the group. |

## Methods

| Name     | Parameters                                  | Description   |
|----------|---|---|
| Add      | Record as <a href="#">AddressBookRecord</a> | Adds the record or group to the Address Book. If <i>Record</i> is a group, it adds it as a subgroup. If it is a contact, it adds it as a member of the group. |
| Contacts |   | Returns an <a href="#">AddressBookContact</a> array of people in the group.   |
| Groups   |   | Returns an <a href="#">AddressBookGroup</a> array of subgroups in the group.  |
| Remove   | Record as <a href="#">AddressBookRecord</a> | Removes the passed record, whether it is a contact or a subgroup.   |

## Example

This example gets the names of the groups in the current user's Address Book and displays them in a [ListBox](#).

```

Dim book as Addressbook
Dim groups() as AddressBookGroup
Dim i as Integer

book=System.AddressBook
groups=book.Groups
For i=0 to Ubound(groups)
    ListBox1.addrow groups(i).name
Next

```

This example gets two Contact fields from the first group and displays them in a [ListBox](#).

```

Dim book as Addressbook
Dim groups() as AddressBookGroup
Dim contacts() as AddressBookContact
Dim i as Integer

book=System.AddressBook
groups=book.Groups //get the array of groups, assume at least one group

contacts=groups(0).contacts //get array of contacts in first group
For i=0 to Ubound(Contacts)
    ListBox1.addrow contacts(i).CompanyName
    ListBox1.cell(i,1)=contacts(i).LastName
Next

```

**See Also** [AddressBook](#), [AddressBookAddress](#), [AddressBookContact](#), [AddressBookData](#), [AddressBookRecord](#) classes.

---

## AddressBookRecord Class

Stores information about an Address Book record and has methods for getting, setting, and removing values. This is the base class for [AddressBookContact](#) and [AddressBookGroup](#).

**Super Class** [Object](#)

### Properties

| Name             | Type                   | Description  |
|------------------|------------------------|--|
| CreationDate     | <a href="#">Date</a>   | Returns the creation date of the record.           |
| ModificationDate | <a href="#">Date</a>   | Returns the last modification date of the record.  |
| UniqueId         | <a href="#">String</a> | Returns the unique ID corresponding to the record. |

### Methods

| Name        | Parameters  | Description  |
|-------------|---|--|
| Value       | PropertyName as <a href="#">String</a>  | Returns the value of that property as a <a href="#">Variant</a> .        |
| Value       | PropertyName as <a href="#">String</a><br>Assigns NewValue as <a href="#">Variant</a> | Sets the value for the specified property as a <a href="#">Variant</a> . |
| RemoveValue | PropertyName as <a href="#">String</a>  | Removes the value for the specified property.                            |

**See Also** [AddressBook](#), [AddressBookAddress](#), [AddressBookContact](#), [AddressBookData](#) classes; [System](#) object.

---

## AddressOf Operator

Returns a function pointer for any method. This can be used with the [Declare](#) statement to provide a callback function to some external library.

**Syntax** *pointer=AddressOf methodName*

| Part       | Description                         |
|------------|-------------------------------------|
| pointer    | Pointer to <i>methodName</i> .      |
| methodName | Method to which you want a pointer. |

**Notes**

If your callback function raises spurious [StackOverflowExceptions](#), the system may be calling it from a separate thread, confusing REALbasic's stack overflow checker. To correct this problem, add the statement:

```
#Pragma StackOverflowChecking False
```

to the beginning of your callback function.

**See Also** [MemoryBlock](#) function; [Declare](#) statement.

---

## An Assigns Parameter cannot have a Default Value Error

You specified a default value for a parameter that used the [Assigns](#) keyword. This is not valid.

**Example** The following method declaration is not permitted.

```
Sub theVolume(a As Integer, b As Integer, Assigns c As Integer=10)
```

You must remove the assignment statement.

**See Also** [Assigns](#) keyword.

---

## An Extends Parameter cannot have a Default Value Error

You assigned a default value to a parameter that uses the [Extends](#) keyword.

**Example** The following parameter declaration is not acceptable.

```
Extends C as Color=$cFF0000, Amt as Integer
```

**See Also** [Extends](#) keyword.

## An Ordinary Parameter cannot Follow a ParamArray Error

You declared an ‘ordinary’ parameter as the second parameter after a parameter that was declared as [ParamArray](#). This is not allowed. Pass only the parameter declared as [ParamArray](#).

**Example** In the following declaration, m is the ‘ordinary’ parameter that is not allowed.

```
Sub myMethod(ParamArray n as Integer, m as Integer)
```

**See Also** [ParamArray](#) keyword.

---

## And Operator

Used to perform a logical comparison of two [boolean](#) expressions.

**Syntax** *result=expression1 And expression2*

| Part        | Description                                   |
|-------------|---|
| result      | A <a href="#">boolean</a> value.              |
| expression1 | Any valid <a href="#">boolean</a> expression. |
| expression2 | Any valid <a href="#">boolean</a> expression. |

**Notes** If both expressions evaluate to [True](#), then *result* is [True](#). If either expression evaluates to [False](#) then *result* is [False](#).

**Example** This example uses the **And** operator to perform logical comparisons.

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean
Dim d As Boolean
a=True
b=True
d=a And b //Returns True
d=a And c //Returns False
```

**See Also** [Not](#), [Or](#) operators; [Operator\\_And](#) function.

# App Function

An intrinsic instance of the [Application](#) class that represents the application.

**Syntax**      **App.property=value**

**App.method**

**Notes**      The **App** class represents the application itself (as opposed to a window or control). This allows you access to the [Application](#) class's properties, events, and methods without having to explicitly store a reference to it. If you have created a class that is a subclass of type [Application](#), the **App** class will return a reference to that class.

In the Project Editor, the App class's Super class is [Application](#) for Desktop Applications. If you create a Console Application, its Super class is [ConsoleApplication](#).

**Examples**      This example changes the application's [MouseCursor](#) to the Wait cursor:

```
App.MouseCursor=System.Cursors.Wait
```

Note: Access to the application's resource fork is supported only on Macintosh. Check the value of the [TargetMacOS](#) constant before attempting to open a resourcefork.

The following example uses the DoEvents method to yield time back to REALbasic in the middle of a loop. If you create PushButtons with and without the DoEvents method you will see how it allows REALbasic to handle other tasks while the loop is running.

```
Dim t as Integer
t=Ticks

While t+600>Ticks
    App.DoEvents
Wend
```

Often it is better to use a [Thread](#) to perform lengthy operations in the background rather than using DoEvents in this manner.

**See Also**      [Application](#), [ConsoleApplication](#), [MDIWindow](#), [ServiceApplication](#) classes; [System](#) object.

# Append Method

Appends a new element to the end of a one-dimensional array, increasing the size of the array by one.

### Syntax

**array.Append value**

| Part  | Description  |
|-------|--|
| array | Required. The array to be appended.                |
| value | The value to be assigned to the new array element. |

### Notes

The **Append** method works with one-dimensional arrays only.

### Example

This example appends a new element to the end of the aNames array.

```
aNames.append "Bill"
```

### See Also

[Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

---

## AppleEvent Class

AppleEvent objects can be used to communicate with other Macintosh applications and the Macintosh System software. If you are compiling your application for use on other operating systems, be sure to check the global [Boolean](#) constants [TargetWin32](#), [TargetMacOS](#), and [TargetLinux](#). These constants return [True](#) if the application is running on the respective operating system.

### Super Class

[Object](#)

### Properties

| Name            | Type                                    | Description   |
|-----------------|---|---|
| BooleanParam    | ParameterName as <a href="#">String</a> | A <a href="#">Boolean</a> being passed as a parameter in the AppleEvent.  |
| DescListParam   | ParameterName as <a href="#">String</a> | An <a href="#">AppleEventDescList</a> object being passed as a parameter in the AppleEvent.                     |
| DoubleParam     | ParameterName as <a href="#">String</a> | A double being passed as a parameter in the AppleEvent.   |
| EnumeratedParam | ParameterName as <a href="#">String</a> | A four character <a href="#">String</a> being passed as a parameter in the AppleEvent.                          |
| FolderItemParam | ParameterName as <a href="#">String</a> | A <a href="#">FolderItem</a> being passed as a parameter in the AppleEvent.                                     |
| IntegerParam    | ParameterName as <a href="#">String</a> | An <a href="#">Integer</a> (passed as a <a href="#">String</a> ) being passed as a parameter in the AppleEvent. |

| Name                 | Type                                      | Description  |
|----------------------|---|--|
| MacTypeParam         | ParameterName as <a href="#">String</a>   | A four character <a href="#">String</a> being passed as a parameter in the AppleEvent.                                     |
| ObjectSpecifierParam | ParameterName as <a href="#">String</a>   | An <a href="#">AppleEventObjectSpecifier</a> being passed as a parameter in the AppleEvent.                                |
| Ptr                  | <a href="#">Integer</a>                   | Pointer to underlying AppleEvent data structure, for use with <a href="#">Declare</a> statements.                          |
| RecordParam          | ParameterName as <a href="#">String</a>   | An <a href="#">AppleEventRecord</a> being passed as a parameter in the AppleEvent.   |
| ReplyBoolean         | <a href="#">Boolean</a>                   | A reply in <a href="#">Boolean</a> form. <a href="#">True</a> means that the application successfully handled the message. |
| ReplyDescList        | <a href="#">AppleEventDescriptorList</a>  | A reply in <a href="#">AppleEventDescriptorList</a> form.  |
| ReplyDouble          | <a href="#">Double</a>                    | A reply in double form.  |
| ReplyEnumerated      |   |  |
| ReplyFolderItem      | <a href="#">FolderItem</a>                | A reply in <a href="#">FolderItem</a> form.  |
| ReplyInteger         | <a href="#">Integer</a>                   | A reply in integer form.   |
| ReplyMacType         | <a href="#">String</a>                    |  |
| ReplyObjectSpecifier | <a href="#">AppleEventObjectSpecifier</a> | A reply in <a href="#">AppleEventObjectSpecifier</a> form.   |
| ReplyPtr             | <a href="#">Integer</a>                   | A reply in pointer form to the underlying AppleEvent structures for use with <a href="#">Declare</a> statements.           |
| ReplyRecord          | <a href="#">AppleEventRecord</a>          | A reply in <a href="#">AppleEventRecord</a> form.  |
| ReplySingle          | ParameterName as <a href="#">String</a>   | A reply in Single form.  |
| ReplyString          | <a href="#">String</a>                    | A reply in <a href="#">String</a> form.  |
| SingleParam          | ParameterName as <a href="#">String</a>   | A single being passed as a parameter in the AppleEvent.  |
| StringParam          | ParameterName as <a href="#">String</a>   | A <a href="#">String</a> being passed as a parameter in the AppleEvent.  |
| TimeOut              | <a href="#">Integer</a>                   | Timeout time in seconds<br>-1 = use default<br>2 = no timeout  |

## Methods

| Name             | Parameters                                     | Description                   |
|------------------|--|-------------------------------|
| LoadFromTemplate | Template as <a href="#">AppleEventTemplate</a> | Loads an AppleEvent template. |

| Name         | Parameters                              | Description   |
|--------------|---|---|
| Send         |   | Returns <a href="#">True</a> if the AppleEvent was successfully sent and <a href="#">False</a> if it was not. <a href="#">True</a> does not mean that the receiving application successfully handled the message. Use the ReplyBoolean property for that. |
| SetNullParam | ParameterName as <a href="#">String</a> | Sets the parameter specified by the Keyword to a null.  |

### Notes

AppleEvent objects are used to send and receive information between your application and other Macintosh applications or the Mac OS. To send an AppleEvent from your application to another application, create an AppleEvent with the [NewAppleEvent](#) function, fill in the AppleEvent's properties with any necessary data, then call the AppleEvent's Send function to send it.

AppleEvents can also be received by your application. When an AppleEvent is received, the [Application](#) object's HandleAppleEvent event is executed and the AppleEvent is passed to the event as a parameter. All intrinsic AppleEvents are first passed to the [Application](#) object's HandleAppleEvent event handler. If you return [True](#) from this event, the default behavior of the AppleEvent will not be executed. For more information on receiving AppleEvents, see the [Application](#) class.

### Replying To An AppleEvent

When an AppleEvent is received (via an AppleEvent handler) the ReplyBoolean, ReplyInteger, and ReplyString properties can be used to automatically send a reply back to the application that sent the AppleEvent. When sending an AppleEvent (via the Send method) the ReplyBoolean, ReplyInteger, and ReplyString properties can be used to get any reply the target application has sent back once it receives the AppleEvent.

For example, the line:

```
e.ReplyBoolean=True
```

indicates that the application successfully handled the AppleEvent message.

To determine whether the other application successfully handled a message, use code such as:

```
Dim ae as AppleEvent
Dim s as String

.

.

If ae.Send then //AE successfully sent
If ae.replyBoolean then
    s = "Yes (handled)"
else
    s = "Yes (unhandled)"
end if
else
    s = "No"
end if
```

For more information on AppleEvents, please refer to Apple's Developer section in the internet.

## Examples

This example uses an AppleEvent to tell the Finder to open the Appearance control panel:

```
Dim a as AppleEvent
a = NewAppleEvent("aevt", "odoc", "MACS")
a.FolderItemParam("----")=SpecialFolder.ControlPanels.Child("Appearance")
If Not a.Send Then
    MsgBox "The Appearance control panel could not be opened."
End If
```

In this example, the SimpleText application (which must be running for this particular example to work) is instructed to open two documents ("My Document" and "My Other Document") that are located in the folder with the REALbasic project:

```
Dim a as AppleEvent
Dim list as AppleEventDescList
a = NewAppleEvent("aevt", "odoc", "ttxt")
list = New AppleEventDescList
list.AppendFolderItem GetFolderItem("My Document")
list.AppendFolderItem GetFolderItem("My Other Document")
a.DescListParam("----") = list
If Not a.Send Then
    MsgBox "The AppleEvent could not be sent."
End If
```

## AppleEventDescList Class

---

This example displays the version of the Mac OS that is running on the user's computer:

```
Dim ae as AppleEvent
Dim obj as AppleEventObjectSpecifier
Dim version as String

If TargetMacOS then
    ae = NewAppleEvent("core", "getd", "MACS")
    obj=GetPropertyObjectDescriptor(nil, "ver2")
    ae.ObjectSpecifierParam("----") = obj

    If Not ae.send() then
        MsgBox "The event could not be sent."
    else
        version = ae.ReplyString
        If (version <> "") then
            MsgBox "The system version is apparently " " + version + " ".
        else
            MsgBox "This version of the system software does not support this event."
        end if
    end if
    else
        MsgBox "This is not a Macintosh!"
    end if
```

### See Also

[AppleEventDescList](#), [AppleEventObjectSpecifier](#) classes; [Application](#) object; [NewAppleEvent](#) function; [#If...#Else...#Endif](#) statement; [TargetLinux](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetWin32](#) constants.

---

## AppleEventDescList Class

Used to send complex information to other applications via AppleEvents.

**Super Class** [Object](#)

### Properties

| Name               | Type                             | Description   |
|--------------------|----------------------------------|---|
| <b>Count</b>       | <a href="#">Integer</a>          | The number of items in the list.  |
| <b>FolderItem</b>  | Index as <a href="#">Integer</a> | A <a href="#">FolderItem</a> array to be passed with the <a href="#">AppleEvent</a> . |
| <b>IntegerItem</b> | Index as <a href="#">Integer</a> | An <a href="#">Integer</a> array to be passed with the <a href="#">AppleEvent</a> .   |

| Name       | Type                             | Description  |
|------------|----------------------------------|--|
| RecordItem | Index as <a href="#">Integer</a> | An <a href="#">AppleEventRecord</a> array to be passed with the <a href="#">AppleEvent</a> . |
| StringItem | Index as <a href="#">Integer</a> | A StringItem array to be passed with the <a href="#">AppleEvent</a> .                        |

## Methods

| Name                  | Parameters   | Description   |
|-----------------------|--|---|
| AppendBoolean         | Value as <a href="#">Boolean</a>                   | Adds a new <a href="#">Boolean</a> value to the array                         |
| AppendDescList        | Value as <a href="#">AppleEventDescList</a>        | Adds a new AppleEventDescList value to the array.                             |
| AppendFolderItem      | Value as <a href="#">FolderItem</a>                | Adds a new <a href="#">FolderItem</a> to the FolderItem array property.       |
| AppendInteger         | Value as <a href="#">Integer</a>                   | Adds a new <a href="#">Integer</a> to the IntegerItem array property.         |
| AppendObjectSpecifier | Value as <a href="#">AppleEventObjectSpecifier</a> | Adds a new AppleEventObjectSpecifier object to the array.                     |
| AppendRecord          | Value as <a href="#">AppleEventRecord</a>          | Adds a new <a href="#">AppleEventRecord</a> to the RecordItem array property. |
| AppendString          | Value as <a href="#">String</a>                    | Adds a new <a href="#">String</a> to the StringItem array property.           |
| BooleanItem           | Index as <a href="#">Integer</a>                   | Returns <a href="#">Boolean</a> Index item.                                   |
| DescListItem          | Index as <a href="#">Integer</a>                   | Returns AppleEventDescList Index item   |
| ObjectSpecifierItem   | Index as <a href="#">Integer</a>                   | Returns AppleEventObjectSpecifier Index item                                  |

**Examples** See the [AppleEvent](#) class for an example.

**See Also** [AppleEvent](#), [AppleEventRecord](#) classes; [NewAppleEvent](#) function.

---

## AppleEventObjectSpecifier Class

An **AppleEventObjectSpecifier** specifies an object that an [AppleEvent](#) can be performed on.

**Super Class** [Object](#)

## AppleEventRecord Class

---

|                   |  |
|-------------------|--|
| <b>Properties</b> | None   |
| <b>Events</b>     | None   |
| <b>Methods</b>    | None   |
| <b>Notes</b>      | An <b>AppleEventObjectSpecifier</b> can represent any object such as a document or a pushbutton. AppleEventObjectSpecifiers are returned by several functions and are passed to the ObjectSpecifierParam property of an <a href="#">AppleEvent</a> class object.   |
| <b>Examples</b>   | See the examples for the <a href="#">AppleEvent</a> class.   |
| <b>See Also</b>   | <a href="#">AppleEvent</a> class; <a href="#">GetIndexedObjectDescriptor</a> , <a href="#">GetNamedObjectDescriptor</a> , <a href="#">GetOrdinalObjectDescriptor</a> , <a href="#">GetPropertyObjectDescriptor</a> , <a href="#">GetRangeObjectDescriptor</a> , <a href="#">GetStringComparisonObjectDescriptor</a> , <a href="#">GetTestObjectDescriptor</a> functions. |

---

## AppleEventRecord Class

A record that contains a series of different data types. AppleEventRecords can be passed using AppleEvents.

**Super Class** [Object](#)

### Properties

| Name            | Type                                    | Description   |
|-----------------|---|---|
| BooleanParam    | ParameterName as <a href="#">String</a> | A <a href="#">boolean</a> being passed as a parameter in the AppleEventRecord.  |
| DescListParam   | ParameterName as <a href="#">String</a> | An <a href="#">AppleEventDescList</a> object being passed as a parameter in the AppleEventRecord.                     |
| DoubleParam     | ParameterName as <a href="#">String</a> | A <a href="#">Double</a> being passed as a parameter in the AppleEventRecord.   |
| EnumeratedParam | ParameterName as <a href="#">String</a> | A four character <a href="#">String</a> being passed as a parameter in the AppleEventRecord.                          |
| FolderItemParam | ParameterName as <a href="#">String</a> | A <a href="#">FolderItem</a> being passed as a parameter in the AppleEventRecord.                                     |
| IntegerParam    | ParameterName as <a href="#">String</a> | An <a href="#">integer</a> (passed as a <a href="#">String</a> ) being passed as a parameter in the AppleEventRecord. |
| MacTypeParam    | ParameterName as <a href="#">String</a> | A four character <a href="#">String</a> being passed as a parameter in the AppleEventRecord.                          |

| Name                 | Type                                    | Description   |
|----------------------|---|---|
| ObjectSpecifierParam | ParameterName as <a href="#">String</a> | An <a href="#">AppleEventObjectSpecifier</a> being passed as a parameter in the AppleEventRecord. |
| RecordParam          | ParameterName as <a href="#">String</a> | An <a href="#">AppleEventRecord</a> being passed as a parameter in the AppleEventRecord.          |
| SingleParam          | ParameterName as <a href="#">String</a> | A <a href="#">Single</a> being passed as a parameter in the AppleEventRecord.                     |
| StringParam          | ParameterName as <a href="#">String</a> | A <a href="#">String</a> being passed as a parameter in the AppleEventRecord.                     |

## Methods

| Name             | Parameters                                     | Description  |
|------------------|--|--|
| LoadFromTemplate | Template as <a href="#">AppleEventTemplate</a> | Loads an Apple Event template.                         |
| SetNullParam     | ParameterName as <a href="#">String</a>        | Sets the parameter specified by the Keyword to a null. |

**See Also** [AppleEvent](#) class.

# AppleEventTarget Class

Used to send Apple Events to applications on other computers.

**Super Class** [Object](#)

## Properties

| Name     | Type                    | Description                                   |
|----------|-------------------------|---|
| Remote   | <a href="#">Boolean</a> | <a href="#">True</a> if a different computer. |
| Computer | <a href="#">String</a>  | Name of the computer.                         |
| PortType | <a href="#">String</a>  | Creator code that identifies the application. |
| Zone     | <a href="#">String</a>  | AppleTalk zone.                               |

## Methods

| Name          | Parameters  | Description   |
|---------------|---|---|
| NewAppleEvent | EventClass as <a href="#">String</a><br>EventID as <a href="#">String</a> | Creates a new AppleEvent. Returns an <a href="#">AppleEvent</a> . |

**See Also** [AppleEvent](#) class.

## AppleEventTemplate Class

A template object for creating [AppleEvent](#) objects.

**Super Class** [Object](#)

### Properties

| Name            | Parameters                     | Description  |
|-----------------|--------------------------------|--|
| IntegerParam    | Name as <a href="#">String</a> | An <a href="#">Integer</a> being passed as a parameter in the Apple Event.   |
| StringParam     | Name as <a href="#">String</a> | A <a href="#">String</a> being passed as a parameter in the Apple Event.     |
| FolderItemParam | Name as <a href="#">String</a> | A <a href="#">FolderItem</a> being passed as a parameter in the Apple Event. |

**See Also** [AppleEvent](#) class.

---

## AppleMenuItem Class

The **AppleMenuItem** class is designed to handle a menu item that you wish to place in the Apple menu under Mac OS “classic.” A menu item subclassed from

**AppleMenuItem** is automatically moved to the Apple menu for Mac OS “classic” builds and to the Application menu for Mac OS X builds. Under other operating systems, such a menu item stays where you put it in the Menu Editor. You cannot place menu items directly in the Apple or application menus in the Menu Editor.

**Super Class** [MenuItem](#)

Since this class is based on [MenuItem](#), please refer to this class for information the AppleMenuItem’s properties, methods, and event.

**Notes** To use **AppleMenuItem**, create the menu item under the menu that you want to use for the Windows and Linux builds of the application and then use the Properties pane to change its super class to **AppleMenuItem**. Otherwise, set up the menu item normally.

**See Also** [MenuItem](#), [PrefsMenuItem](#), [QuitMenuItem](#) classes.

# Application Class

Includes properties, methods, and events for managing the application as a whole.

**Super Class** [Object](#)

## Properties

| Name                      | Type                    | Description  |
|---------------------------|-------------------------|--|
| <b>AcceptFileTypes</b>    | <a href="#">String</a>  | Applies only to the initial or “blessed” instance based on the Application class and is for Macintosh only. Specifies which file types the application will accept via drag and drop. Use a File Type Sets Editor to enter the file types and in the App object’s Properties pane, click the “...” button to open a dialog box in which you can edit this list.                              |
| <b>AutoQuit</b>           | <a href="#">Boolean</a> | If <a href="#">True</a> , the application will quit when the last window is closed, whether by calling the Close method of the <a href="#">Window</a> class or the user closed the last window. AutoQuit defaults to <a href="#">True</a> on non-MDI Windows applications and <a href="#">True</a> on Linux. It defaults to <a href="#">False</a> on MDI Windows and Macintosh applications. |
| <b>BalloonHelpVisible</b> | <a href="#">Boolean</a> | Supported on Windows only. When set to <a href="#">False</a> , Balloon Help is disabled; <a href="#">True</a> enables Balloon Help.  |
| <b>BugVersion</b>         | <a href="#">Integer</a> | Bug version of the application.  |
| <b>BuildDate</b>          | <a href="#">Date</a>    | Contains the date and time when the application was built. You can also access the CreationDate property of the application’s <a href="#">FolderItem</a> by calling app.executableFile.CreationDate  |
| <b>CurrentThread</b>      | <a href="#">Thread</a>  | Obtains the currently running <a href="#">Thread</a> . If the current <a href="#">Thread</a> is the main thread (the thread that REALbasic creates and manages itself to run the application), CurrentThread returns <a href="#">Nil</a> .   |

## Application Class

---

| Name           | Type                       | Description   |
|----------------|----------------------------|---|
| DockItem       |                            | (Mac OS X only). Enables you to manipulate the dock item associated with the application. The DockItem property enables you to access the properties and methods of the <a href="#">DockItem</a> class. This class has two methods, UpdateNow and ResetIcon, and one property, Graphics. Use the methods of the <a href="#">Graphics</a> class to modify the appearance of the icon. Since a Mac OS X icon is intended to be scaled automatically, you should design it as a 128x128 pixel icon. ResetIcon resets the icon to its original state (default appearance). Call the UpdateNow method to redraw the icon. You can also use the ClearRect property of the <a href="#">Graphics</a> class to start over from a blank icon. Anything you can do with a <a href="#">Graphics</a> object you are able to do with the Dock's Graphics object (like drawing in a picture or using a <a href="#">Group2D</a> ). The DockItem property of the <a href="#">Window</a> class enables you to control the dock item for individual windows. |
| ExecutableFile | <a href="#">FolderItem</a> | Returns a <a href="#">FolderItem</a> for the actual executable application even if it is in a bundle. Use the properties and methods of the <a href="#">FolderItem</a> class to get/set attributes of the executable file and/or perform operations. For example:<br>app.ExecutableFile.AbsolutePath<br>gets the full path to the executable.<br>app.ExecutableFile.CreationDate<br>gets the date/time the executable was created.  |
| LongVersion    | <a href="#">String</a>     | Long version of the application, corresponding to the version information.  |
| MajorVersion   | <a href="#">Integer</a>    | Major version of the application, corresponding to the version information. The range is from 0 to 255.   |
| MDIWindow      | <a href="#">MDIWindow</a>  | Provides access to the properties and methods of the <a href="#">MDIWindow</a> class. Valid only for Windows applications that are built using the Multiple Document Interface option.  |
| MenuBar        | <a href="#">MenuItem</a>   | Represents the application's global menubar. Its children are the menus. You can replace the entire menubar by assigning a new <a href="#">MenuItem</a> to MenuBar. Each <a href="#">Window</a> can have a menubar assigned to it via its MenuBar property. If no menubar is assigned to the window, the application's menubar is used.   |

| Name                     | Type                         | Description  |
|--------------------------|------------------------------|--|
| <b>MinorVersion</b>      | <a href="#">Integer</a>      | Minor version of the application, corresponding to the version information. The range is from 0 to 255.  |
| <b>MouseCursor</b>       | <a href="#">MouseCursor</a>  | The cursor that is displayed while the application is running. If the Application class MouseCursor is not <a href="#">Nil</a> , the non- <a href="#">Nil</a> MouseCursor properties belonging to any <a href="#">Windows</a> or <a href="#">Controls</a> are ignored. You can use the cursors in the <a href="#">Cursors</a> object to set the mousecursor. |
| <b>NonReleaseVersion</b> | <a href="#">Integer</a>      | NonRelease version of the application, corresponding to the version information.   |
| <b>PackageInfo</b>       | <a href="#">String</a>       | Package Info of the application, corresponding to the version information.   |
| <b>RegionCode</b>        | <a href="#">Integer</a>      | Region Code of the application, corresponding to the version information. Not supported on Windows.  |
| <b>ResourceFork</b>      | <a href="#">ResourceFork</a> | Used to access the application's resources. Supported only on Macintosh. Check the value of <a href="#">TargetMacOS</a> before attempting to access the application's resources.   |
| <b>ShortVersion</b>      | <a href="#">String</a>       | Short version of the application, corresponding to the version information.  |
| <b>StageCode</b>         | <a href="#">Integer</a>      | Stage Code of the application, corresponding to the version information. Use the four Application class constants to determine the Stage:<br>0 - Development<br>1 - Alpha<br>2 - Beta<br>3 - Final   |

## Methods

| Name               | Parameters                       | Description  |
|--------------------|----------------------------------|--|
| <b>AddTrayItem</b> | Item as <a href="#">TrayItem</a> | Windows only. Adds the passed item to the System Tray via the <a href="#">TrayItem</a> class. See the <a href="#">TrayItem</a> class for an example. |

| Name               | Parameters                                 | Description  |
|--------------------|--|--|
| DoEvents           | [milliseconds as <a href="#">Integer</a> ] | <p>Yields time back to REALbasic in tight loops. Calling DoEvents inside a loop allows REALbasic to maintain the interface for the user while continuing to execute the loop. However, using DoEvents from a multithreaded application is not supported and will likely cause instability. You should consider using <a href="#">threads</a> to handle lengthy operations rather than placing them in the main thread and calling DoEvents to maintain the interface.</p> <p>If the current method is running inside a Thread, DoEvents will yield to the main thread instead of running one iteration of the event loop inside the thread, causing confusion. If you use DoEvents inside a loop, you cannot use the <a href="#">UserCancelled</a> function to detect whether the user pressed the Esc key (Windows or Linux) or Command-period (on Macintosh) to break out of the loop. Since DoEvents keeps the user interface responsive while the loop is running, you can add a button to the user interface that the user can click to stop the loop.</p> <p>The optional parameter specifies the amount of time you want the currently executing thread to sleep. DoEvents can cause a <a href="#">Thread</a> to sleep with a resolution greater than 16 milliseconds. If all threads are sleeping, then REALbasic yields back time to the system. This allows you to write applications that don't use up to 100% of the CPU during tight loops. Specifying zero milliseconds causes the next waiting thread to execute. A negative value specifies no sleep. The default is -1.</p> |
| NewDocument        |  | Calls the NewDocument event handler.   |
| OpenDocument       | File as <a href="#">FolderItem</a>         | Calls the OpenDocument event handler.  |
| RemoveTrayItem     | Item as <a href="#">TrayItem</a>           | Windows only. Removes the passed item from the System Tray via the <a href="#">TrayItem</a> class. See the <a href="#">TrayItem</a> class for an example.  |
| SleepCurrentThread | Milliseconds as <a href="#">Integer</a>    | Sleeps the current <a href="#">thread</a> for the specified number of milliseconds. It can be used for any thread.   |

| Name              | Parameters | Description  |
|-------------------|------------|--|
| YieldToNextThread |            | Triggers a context change. That is, it causes the Thread Scheduler to yield the currently executing thread's processing time to the next <a href="#">Thread</a> in the queue that is awaiting processing cycles. It is possible for the next <a href="#">Thread</a> to be the currently executing <a href="#">Thread</a> . |

## Events

| Name             | Parameters   | Description   |
|------------------|--|---|
| Activate         |  | The application is being activated.   |
| CancelClose      |  | Returns a <a href="#">Boolean</a> . This event occurs when the application is about to terminate. It occurs prior to all of the CancelClose events for each <a href="#">Window</a> . It gives you the chance to cancel the application's termination. Return <a href="#">True</a> to cause the termination to stop.   |
| Close            |  | The application is quitting.  |
| Deactivate       |  | The application is being deactivated.   |
| EnableMenuItem   |  | Executes when the user clicks in the menubar to give you the opportunity to determine which menu items to enable. Also executes when the application opens if no default window is selected in the Project Settings dialog box and when the last window is closed.  |
| HandleAppleEvent | Event as <a href="#">AppleEvent</a> , EventClass as <a href="#">String</a> , EventID as <a href="#">String</a> | An <a href="#">AppleEvent</a> has been received. Return <a href="#">True</a> to accept the <a href="#">AppleEvent</a> and <a href="#">False</a> to reject it. Intrinsic AppleEvents, such as "odoc", "quit", and "pref", are passed to this event handler first. If you return <a href="#">True</a> , the default behavior will no longer take place. For example, if you return <a href="#">True</a> for the "odoc" event, the OpenDocument event will not fire. |
| NewDocument      |  | The user launched the application by double-clicking the application icon or the NewDocument method was called.   |
| Open             |  | The application is opening.   |
| OpenDocument     | File as <a href="#">FolderItem</a>   | The user has double-clicked on a document whose creator matches the application's creator or the OpenDocument method was called.  |

| Name               | Parameters                                | Description   |
|--------------------|---|---|
| UnhandledException | error as <a href="#">RuntimeException</a> | <p>Receives otherwise missed exceptions—exceptions that could have been caught in an <a href="#">Exception Block</a> using one of the <a href="#">Runtime</a> exception handlers. If the appropriate <a href="#">Exception Block</a> is missing, the runtime exception error triggers this event and allows you to handle exceptions here. See the example.</p> <p>Returns a <a href="#">Boolean</a>. If it returns <a href="#">True</a>, the standard alert indicating an unhandled exception and quit behavior are suppressed. <a href="#">Return False</a> to allow REALbasic's default unhandled exception error processing to occur. See the <a href="#">RuntimeException</a> class.</p> |

### Notes

REALbasic adds a class based on the **Application** class to the Project Editor when you create a new Desktop Application project. This new subclass will give you access to the **Application** object's methods and events. If you created a Console Application, the App class's Super class is [ConsoleApplication](#), not **Application**.

If you wish, you can add additional subclasses based on the **Application** class to the project, but it is not necessary. If you do so, the one that is added to the project automatically is the one referred to by the [App](#) function and it is the one that will show the project's build settings.

The new subclass has its own menu handlers which can be used to handle menus items when no windows are open or for menu items that should call the same menu handler regardless of which window is frontmost. You can change the global menubar by assigning a different menubar to its [MenuBar](#) property.

See the [Control](#) class for information on changing the cursor and adding cursors to your project.

The [App](#) function returns a reference to the **Application** object. See the [App](#) function for more information.

Resource files (files of type “rsrc”) created with a resource editor can be added to your project adding each resource file to the Project. These resources are then accessible through the **Application** object's [ResourceFork](#) property. The resources from all your resource files will be copied into the built Macintosh application. In the case of a conflict, later resource files overwrite earlier ones, where the files are written in the order in which they appear in the Project Editor.

The application resources are read-only. If you need to write to resources you will need to use an external resource file. See the [ResourceFork](#) class for more information.

Note: Access to the resourcefork is supported only on Macintosh. Check the value of the [TargetMacOS](#) constant before attempting to access the application's resourcefork.

**Version Information**

The compiler autoincrements version information for a project on each compile. This includes test compiles within the debugger as well as standalone builds.

The compiler truncates version information when building Windows applications when the version information is too long to store into the executable file. The current byte limitations for these fields are as follows:

| Field         | Maximum length (bytes) |
|---------------|------------------------|
| Long Version  | 79                     |
| Short Version | 39                     |
| Package Info  | 253                    |
| Region        | 21                     |
| Release       | 11                     |

**Examples**

The following code in the Action event of a [PushButton](#) causes an [OutOfBoundsException](#) runtime error when the counter, i, reaches the value of [FontCount](#).

```
Dim i as Integer
For i=1 to FontCount
    ListBox1.Addrow Font(i)
Next
```

Since there is no exception handler within the method, the runtime exception is passed up to the [Application](#) class, triggering the [UnhandledException](#) event.

This code in the [UnhandledException](#) event of the App class “catches” the unhandled [OutOfBoundsException](#). Of course, it catches all unhandled OutOfBoundsExceptions throughout the application, so it doesn’t know where the error occurred. You could instead place an Exception Block within the [PushButton](#)’s Action event so that you can provide more specific diagnostics.

```
Function UnhandledException(error as RuntimeException) as Boolean
If error IsA OutOfBoundsException then
    MsgBox "An OutOfBounds Exception error has occurred!"
End if
Return True
```

**See Also**

[App](#), [ResourceFork](#), [System](#), objects; [AppleEvent](#), [ConsoleApplication](#), [MDIWindow](#), [ServiceApplication](#) classes.

## ApplicationSupportFolder Function

Returns a reference to the Application Support directory, if one exists.

## ApplicationSupportFolder Function

### Syntax

**result=ApplicationSupportFolder**

| Part   | Type                       | Description   |
|--------|----------------------------|---|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that refers to the Application Support folder. |

### Notes

The Application Support folder may contain code and data files needed by third-party applications. These files should usually not be written to after they are installed. In general, files deleted from this folder remove functionality from an application, unlike files in the Preferences folder, which should be non-essential.

The [SpecialFolder](#) module enables you to access many special folders that are managed by the OS.

### Windows

On Windows, **ApplicationSupportFolder** returns a reference to the \Documents and Settings\user\Local Settings\Application Data directory.

### Macintosh

On Mac OS “classic”, **ApplicationSupportFolder** returns a reference to the Application Support folder in the active System folder. On Mac OS X, **ApplicationSupportFolder** returns a reference to the /Library/ApplicationSupport folder.

### Linux

On Linux, **ApplicationSupportFolder** returns a [FolderItem](#) for the Home directory for the logged in user, /home/username/. Typically, when an application wants to create an application support file, it will create a folder in the Home directory corresponding to the application’s name. For example, for MyApp, it will use the path: “/home/username/MyApp/”.

### Example

The following example displays the full path to the Application Support folder, if there is one.

```
Dim f as FolderItem
f=ApplicationSupportFolder
If f <> Nil then
    MsgBox f.getAbsolutePath
else
    MsgBox "No Application Support folder found."
end if
```

### See Also

[DesktopFolder](#), [FontsFolder](#), [PreferencesFolder](#), [ShutdownItemsFolder](#), [StartupItemsFolder](#), [SystemFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

# ArcShape Class

Used for drawing arcs in a vector graphics environment.

**Super Class** [OvalShape](#)

## Properties

| Name       | Type                   | Description                       |
|------------|------------------------|-----------------------------------|
| ArcAngle   | <a href="#">Double</a> | The angle of the arc, in radians. |
| StartAngle | <a href="#">Double</a> | The starting angle, in radians.   |

## Notes

A StartAngle of 0 means due east, i.e., or a positive X value with Y=0. Positive angles are clockwise, just as with [Object2D](#).rotation. The value of ArcAngle can be positive or negative, extending from the starting angle. If the angle is filled, it produces a wedge; this may be useful for making pie charts, for example.

## Examples

The following method, when called from the Paint event of a [window](#), creates an arc.

```
Dim a as New ArcShape
a.arcAngle=1.57
a.startAngle=-1.57
a.FillColor=RGB(255,0,127)
g.drawObject a,100,100
```

This arc looks like this:



## See Also

[CurveShape](#), [FigureShape](#), [FolderItem](#), [Graphics](#), [Group2D](#), [Object2D](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

# Array Function

Assigns a list of values to consecutive elements of an array.

### Syntax

**result=Array(elementList)**

| Part        | Type                            | Description  |
|-------------|---------------------------------|--|
| result      | Any valid datatype for an array | Name of the array that is populated with the items in <i>ElementList</i> .   |
| ElementList | Type of <i>result</i>           | A comma-delimited list of values that are used to populate <i>result</i> . The data type of the items in <i>ElementList</i> should agree with the data type of the array <i>result</i> . If the items in ElementList are of different data types, such as Int32, UInt32, Int64 and so forth, Array will try to find the data type that can accommodate all the elements passed to it. The data type of the array that receives the elements must agree with this data type. Otherwise, you will get a <a href="#">Type Mismatch Error</a> . See the examples in Notes. |

### Notes

The **Array** function provides the same functionality as separate assignment statements for each element of an array, beginning with element zero and proceeding consecutively. If there are more elements in the array than items in the list of elements, then the “extra” elements are not disturbed.

If the *ElementList* contains numeric data of different word lengths and/or a mixture of signed and unsigned integers, **Array** will use the lowest common data type that accommodates all the elements. This may break code that worked prior to the introduction of signed/unsigned integers and integers of different word lengths. For example, in versions of REALbasic prior to 2006, the statement:

```
Array(&h01, &h02 )
```

returned an array of integers.

Because hexadecimal numbers are unsigned integers, this expression now returns an array of **UInt32s**. This means that the following code will generate a [Type Mismatch Error](#):

```
Dim i() as Integer  
i = Array(&h1, &h2)
```

You need to be careful about word length and whether or not the integer is signed. In this instance, the following will compile:

```
Dim i() as UInt32  
i = Array(&h1, &h2)
```

If you wish, you can also decide to convert one of the values to a signed integer. In that case, **Array** will return an [Int32](#) (a.k.a., [Integer](#)) array. Use either of the following ways:

```
Dim i() as Integer  
i = Array(Int32(&h1),&h2)
```

Or this:

```
Dim i() as Integer  
i = Array(1,&h2)
```

## Example

The following statement creates and populates the first three elements of the array `aNames`.

```
Dim aNames() as String  
//using the Array function  
aNames=Array("Fred","Ginger","Stanley")
```

## See Also

[Dim](#) statement; [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

---

## Array Bounds must be Integers Error

You used a non-integer in a [Dim](#) statement when trying to declare an array. You can use [Integer](#) constants to dimension an array, but not non-integers.

## Examples

```
Dim aPicture (5.2) as String  
Dim aProps (True) as Boolean
```

## See Also

[Dim](#) statement.

---

## Asc Function

Returns as an [Integer](#), the ASCII value for the first character of a [String](#).

## Syntax

`result=Asc(string)`

OR

**result=stringVariable.Asc**

| Part                  | Type                    | Description                                       |
|-----------------------|-------------------------|---|
| result                | <a href="#">Integer</a> | The ASCII value of the first character of string. |
| string                | <a href="#">String</a>  | Any valid <a href="#">string</a> expression.      |
| <i>StringVariable</i> | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .     |

### Notes

The **Asc** function returns the code point for the first character in the [String](#) passed. Characters 0 through 127 are the standard ASCII set. They will be the same on practically every platform. **Asc** returns the code point for whatever encoding the string is in. If the string is in MacRoman, you get the code point specified by MacRoman, and so forth.

If you need to get the ASCII code of the first byte of the string rather than the first character, use the [AscB](#) function.

### Examples

This example uses the **Asc** function to get the ASCII value of a character.

```
Dim a as Integer  
a = Asc("@") //returns 64
```

This example gets the code point for the “Š” symbol:

```
Dim s as String  
Dim n as Integer  
s="Š"  
n=s.Asc
```

### See Also

[Chr](#), [InStr](#), [Left](#), [Len](#), [Mid](#), [Right](#) functions; [TextEncoding](#) class.

---

## AscB Function

Returns as an [Integer](#), the value for the first byte of a [String](#).

### Syntax

**result=AscB(string)**

OR

**result=stringVariable.AscB**

| Part                  | Type                    | Description                                   |
|-----------------------|-------------------------|---|
| result                | <a href="#">Integer</a> | The value of the first character of string.   |
| string                | <a href="#">String</a>  | Any valid <a href="#">string</a> expression.  |
| <i>StringVariable</i> | <a href="#">String</a>  | Any variable of type <a href="#">String</a> . |

**Notes**

The **AscB** function returns the code for the first byte in the [String](#) passed. If you need to get the character code of the first character of the string rather than the first byte, use the [Asc](#) function.

**AscB** should be used instead of [Asc](#) when the string represents binary data or when your application will run on a one-byte character set (such as the US system) and you want case-sensitivity.

**Examples**

This example uses the **AscB** to get the value of the first byte of the [string](#) passed.

```
MsgBox Str\(AscB\("a"\)\) //returns 97
MsgBox Str\(AscB\("A"\)\) //returns 65
```

**See Also**

[Asc](#), [ChrB](#), [InStrB](#), [LeftB](#), [LenB](#), [MidB](#), [RightB](#) functions.

## Asin Function

Returns the arcsine of the value specified.

**Syntax**

**result=Asin(value)**

| Part   | Type                   | Description  |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The arc sine of value.   |
| value  | <a href="#">Double</a> | The value you want the arc sine of. Value is the <a href="#">Sin</a> of the angle you want and must be between -1 and 1. |

**Notes**

The arcsine is the angle whose sine is value. The **Asin** function returns the angle (in radians) of the sine passed to it. To express the arcsine in degrees, multiply the result by 180/PI.

**Examples**

This example uses the **Asin** function to return the arcsine of a number.

```
Dim d as Double
Const PI=3.14159265358979323846264338327950
d=Asin(.5) //returns 0.5235988
d=Asin(.5)*180/PI //returns 30
```

**See Also**

[Sin](#) function.

## Assigns Keyword

Used to indicate that a value will be passed via a parameter to a method via the [Assignment](#) operator.

### Syntax

**Assigns parameter As DataType**

| Part      | Type                | Description                               |
|-----------|---------------------|---|
| parameter |                     | The name of the parameter being declared. |
| DataType  | Any valid data type | The data type of the parameter.           |

### Notes

Use the **Assigns** keyword when you want to assign a value to a parameter of a method with the equals sign rather than as an argument.

The **Assigns** keyword can appear in a [Method](#) declaration dialog box for the only parameter passed to the method or, if more than one parameter is being passed, for the last parameter. When **Assigns** is used, you use a different syntax when calling the method. The value for the parameter must be passed using the assignment operator rather than as an argument.

### Example

The following declaration uses **Assigns** for the last parameter:

```
Sub theVolume(a as Integer, b as Integer, Assigns c as Integer)
```

When this method is called, only the values for a and b are passed as arguments; the value for c is passed using the assignment operator. For example:

```
theVolume(5,4)=10
```

If you try to use the syntax:

```
theVolume(5,4,10)
```

you will receive a [Syntax](#) error.

### See Also

[Assigns can only be used on the last parameter](#) Error.

## Assigns can only be used on the last parameter Error

You used the [Assigns](#) keyword in a Method declaration for a parameter other than the last parameter. If there is only one parameter, the use of [Assigns](#) is permitted.

**Example** The following declaration uses the [Assigns](#) keyword for the first parameter.

```
Sub SquareIt(Assigns x as Double, y as Double)
```

**See Also** [Assigns](#) keyword.

## Atan Function

Returns the arctangent of the value specified. The arctangent is the angle whose tangent is value.

**Syntax** *result=Atan(value)*

| Part   | Type                   | Description                            |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The arc tangent of value.              |
| value  | <a href="#">Double</a> | The value you want the arc tangent of. |

**Notes** The **Atan** function returns the angle (in radians) of the number passed to it. To express the arctangent in degrees, multiply the result by 180/PI.

**Examples** This example uses the **Atan** function to return the arc tangent of a number.

```
Dim d as Double
Const PI=3.14159265358979323846264338327950
d=Atan(1) //returns 0.785398 (PI/4 radians)
d=Atan(1)*180/PI // returns 45
```

**See Also** [Atan2](#), [Tan](#) functions.

## Atan2 Function

Returns the arctangent of the point whose coordinates are *x* and *y*. The arctangent is the angle from the x-axis to a line drawn through the origin (0,0) and a point with coordinates *x*, *y*.

**Syntax** *result=Atan2 (y, x)*

| Part     | Type                   | Description   |
|----------|------------------------|---|
| result   | <a href="#">Double</a> | Arctangent of the point ( <i>y</i> , <i>x</i> ) in radians. |
| <i>y</i> | <a href="#">Double</a> | <i>y</i> coordinate of the point.                           |
| <i>x</i> | <a href="#">Double</a> | <i>x</i> coordinate of the point.                           |

## AutoDiscovery Class

---

- Notes** The result is expressed in radians. To convert it to degrees, multiply it by 180/PI.  
The converse operations are done with [Cos](#) and [Sin](#). That is, if you have an angle and want to find an x, y pair along the line described by 0,0 and x,y, you can do so with:

```
x = Cos(angle)*radius  
y= Sin(angle)*radius
```

- Examples** This example uses the **Atan2** function to return the arctangent of a point directly above the origin.

```
Dim d as Double  
Const PI=3.14159265358979323846264338327950  
d=Atan2(1,0) //returns 1.57  
d=Atan2(1,0)*180/PI //returns 90
```

- See Also** [Atan](#), [Tan](#) functions.

## AutoDiscovery Class

**AutoDiscovery** uses the [EasyUDPSocket](#) class to automatically discover other REALbasic applications on the network.

**Super Class** [EasyUDPSocket](#)

### Methods

| Name               | Parameters  | Description  |
|--------------------|---|--|
| GetMemberList      |   | Returns as a <a href="#">String</a> array of IP addresses belonging to the list of machines on the network who are in the group. |
| SendMessageToGroup | Command as <a href="#">Integer</a> , Data as <a href="#">String</a> | Sends the message consisting of <i>Command</i> and <i>Data</i> to the group.   |
| UpdateMemberList   |   | Clears the internal list of connected members and queries the network for all members of the current group.                      |

### Events

| Name         | Parameters                          | Description  |
|--------------|-------------------------------------|--|
| MemberJoined | IPAddress as <a href="#">String</a> | A machine identified as <i>IPAddress</i> has joined. |
| MemberLeft   | IPAddress as <a href="#">String</a> | A machine identified as <i>IPAddress</i> has left.   |

**Notes**

The **AutoDiscovery** class lets you automatically discover other machines on the local network that are communicating using the [EasyUDPSocket](#) class. It does so by checking to see what other applications are using the same group name that you pass to the Register function of the [EasyUDPSocket](#) class. When a member joins (this includes your application when you first call the Register method), you will get a MemberJoined event with the IP address of the member that joined. When a member leaves, then you get a MemberLeft event with their IP as well. If you would like a list of the currently connected members, you can get an array of their IPs by calling the GetMemberList method. You can refresh the list by calling the UpdateMemberList method. It clears the internal list of connected members and re-queries the network for members.

The **AutoDiscovery** class is intended to make communication among network users as easy as possible. To do so, it makes use of a proprietary protocol. Because of this, it is unable to discover other communications protocols that may also be running on the network, such as iChat. Use the generic classes such as [TCPSocket](#), [UDPSocket](#), [HTTPSocket](#), and so forth.

**Examples**

Registering as a group member. Everyone on the network who wants to join the group needs to use the same group name.

```
AutoDiscovery1.Register("MyGroup")
AutoDiscovery1.UpdateMemberList
```

Getting the member list and displaying it in a [ListBox](#):

```
Dim s(-1) as String
Dim i as Integer
s= AutoDiscovery1.GetMemberList
For i=0 to Ubound(s)
    ListBox1.addrow s(i)
Next
```

Leaving the group.

```
AutoDiscovery1.Unregister("MyGroup")
```

Sending a message to the group.

```
AutoDiscovery1.SendMessageToGroup(100,"Hello world")
```

Receiving a message using the ReceivedMessage event:

```
Sub ReceivedMessage (FromIP as String, Command as Integer, Data as String)
    MsgBox fromIP + " sent us " + Str(command) + ":" + Data
```

**See Also**

[EasyUDPSocket](#), [UDPSocket](#) classes.

## Beep Method

Plays your computer's default beep sound.

**Syntax**      **Beep**

**Notes**      The **Beep** method plays the default Beep sound selected on your computer.

**Example**      This example plays the alert sound three times.

```
Beep  
Beep  
Beep
```

---

## BevelButton Control

Used for creating a bevel button. A **BevelButton** can use text, a graphic, a pop-up menu, or several of these interface elements in combination.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| AcceptFocus | <a href="#">Boolean</a> | If <a href="#">True</a> , the BevelButton will be included in the Tab order and can accept the focus. The GotFocus and LostFocus events will fire at the appropriate times. When the BevelButton has the focus, it has a selection rectangle in its interior. If the user clicks the BevelButton or presses either the Space bar or the Enter key, the Action event will fire. The ability of the BevelButton to accept the focus is Windows-only. The default value is <a href="#">False</a> . |

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| Bevel            | <a href="#">Integer</a> | Bevel style. The Bevel property affects the appearance and in some cases the shape of the BevelButton. Examples of all bevel styles are in the Users Guide.<br>0 - Small bevel<br>1 - Normal bevel<br>2 - Large bevel<br>3 - Rounded bevel (Max OS X only)<br>4 - No Bevel (Windows XP only; see Notes)<br>5 - Round (Max OS X only)<br>6 - Large Round (Mac OS X only)<br>7 - Disclosure (Mac OS X only)<br>The Mac OS X-only bevel styles appear as the Small Bevel style on other platforms. |
| Bold             | <a href="#">Boolean</a> | Applies the bold style to the button caption.   |
| ButtonType       | <a href="#">Integer</a> | 0 - Button. Remains in 'down' position until mouse is released.<br>1 - Toggles. Remains in 'down' position until clicked again.<br>2 - Sticky. Remains in 'down' position when clicked.   |
| Caption          | <a href="#">String</a>  | The button's text.  |
| CaptionAlign     | <a href="#">Integer</a> | Alignment of caption.<br>0 - Flush left<br>1 - Flush right<br>2 - Sys direction<br>3 - Center   |
| CaptionDelta     | <a href="#">Integer</a> | Distance in pixels of the caption from the left of the button.  |
| CaptionPlacement | <a href="#">Integer</a> | Placement of caption relative to graphic (Icon).<br>0 - Sys Direction<br>1 - Normally<br>2 - Right of graphic<br>3 - Left of graphic<br>4 - Below graphic<br>5 - Above graphic  |
| Icon             |                         | Name of graphic to use as icon. Drag the graphic to the Project Editor or import it using File ▶ Import. Supports pictures from any source, including those created at runtime with the <a href="#">NewPicture</a> function. Supports transparency, either by setting the Transparent property of the <a href="#">Picture</a> to 1 or via the <a href="#">Picture</a> 's Mask. See the first example.   |

## BevelButton Control

---

| Name          | Type                    | Description  |
|---------------|-------------------------|--|
| IconAlign     | <a href="#">Integer</a> | Alignment of graphic within BevelButton<br>0 - Sys Direction<br>1 - Center<br>2 - Left<br>3 - Right<br>4 - Top<br>5 - Bottom<br>6 - Top left<br>7 - Bottom left<br>8 - Top right<br>9 - Bottom right |
| IconDx        | <a href="#">Integer</a> | Distance in pixels from 'flush' left or right, depending on alignment. If center is chosen, IconDx does nothing.   |
| IconDy        | <a href="#">Integer</a> | Distance in pixels from 'flush' top or bottom, depending on alignment. If center is chosen, IconDy does nothing.   |
| Italic        | <a href="#">Boolean</a> | Applies the italic style to the button caption.  |
| LockBottom    | <a href="#">Boolean</a> | Determines whether the bottom edge of the control should stay at a set distance from the bottom edge of the owning window.   |
| LockLeft      | <a href="#">Boolean</a> | Determines whether the left edge of the control should stay at a set distance from the left edge of the owning window. LockLeft has no effect unless LockRight is <a href="#">True</a> .             |
| LockRight     | <a href="#">Boolean</a> | Determines whether the right edge of the control should stay at a set distance from the right edge of the owning window.   |
| LockTop       | <a href="#">Boolean</a> | Determines whether the top edge of the control should stay at a set distance from the top edge of the owning window. LockTop has no effect unless LockBottom is <a href="#">True</a> .               |
| HasMenu       | <a href="#">Integer</a> | Controls whether BevelButton control has a popup menu.<br>0 - No menu<br>1 - Normal menu<br>2 - Menu on right  |
| MenuItemValue | <a href="#">Integer</a> | The number of the menu item the user selects. The first menu item is number zero. A separator cannot be selected, but "counts" as a menu value.  |
| TextFont      | <a href="#">String</a>  | Name of the font used to display the button caption.   |
| TextSize      | <a href="#">Integer</a> | Size of the font used to display the button caption.   |
| Underline     | <a href="#">Boolean</a> | Applies the underline style to the button caption.   |
| Value         | <a href="#">Boolean</a> | If <a href="#">True</a> , the button initially appears as if it is pressed.  |

## Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| Action    |  | The BevelButton has been clicked. A right-click does not trigger the Action event.  |
| GotFocus  |  | (Windows only) The BevelButton has received the focus and has a selection rectangle around its caption. The AcceptFocus property must be set to <a href="#">True</a> for the BevelButton to be capable of getting the focus.                        |
| LostFocus |  | (Windows only) The control has lost the focus.  |
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the BevelButton's region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown. The Action event will not execute and the state of the object will not change. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the BevelButton's region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. The return value is ignored.                               |

## Methods

| Name          | Parameters   | Description  |
|---------------|--|--|
| AddRow        | Text as <a href="#">String</a>                                   | Adds a row to the bottom of the menu and uses Text as the menu item.   |
| AddSeparator  |  | Adds a row to the bottom of the menu and uses a separator as the menu "item."  |
| DeleteAllRows |  | Deletes all rows of the current Bevel button menu.   |
| InsertRow     | Row as <a href="#">Integer</a><br>Text as <a href="#">String</a> | Inserts a row in the position indicated by Row using Text as the menu item.  |
| List          | Row as <a href="#">Integer</a>                                   | Returns the text of the menu item as a <a href="#">string</a> . The first menu item is zero. If the Row parameter is out of bounds, it triggers an <a href="#">OutOfBoundsException</a> exception. |
| RemoveRow     | Row as <a href="#">Integer</a>                                   | Removes Row from the menu. The first menu item is zero.  |

## Notes

A **BevelButton** can display a graphic, a label (caption), or both.

The first row in a menu has an index of zero.

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. For example, if you set the Caption property to "Bob & Ray", only "Bob Ray" will display. You need to set it to "Bob && Ray". The ampersand character is used on Windows for keyboard accelerators. This is done to make applications on all platforms behave consistently.

The "No Bevel" option for the Bevel property is for the Windows XP operating system. The button appears to have no border until the mouse enters the region of BevelButton

## Bin Function

---

control. Then the outline of the button appears. On non-XP operating systems, the “No Bevel” option has the same appearance of “Small Bevel.”

### Examples

The following example places an icon in the bevel button, aligns it, and specifies the placement of the caption. The graphic, “databasequery,” has been added to the Project.

```
Me.icon=databasequery
Me.iconalign=6
Me.captionalign=0
Me.captionplacement=2
Me.icondx=2
Me.icondy=1
```

The following example creates a bevel button menu.

```
Dim s as String
Dim i as Integer
Dim last as Integer
s="January,February,March,April,May,June,July,August,September, " _
    +"October,November,December"
last=CountFields(s, ", ")
Me.caption="Month"
Me.captionalign=0 //flush left
Me.hasMenu=2 //menu on right
For i=1 to last
    me.addRow NthField(s, ", ",i)
Next
```

The following line of code in the **BevelButton's** Action event handler sets the **BevelButton** caption to the value that the user selects:

```
Me.caption=Me.list(Me.MenuValue)
```

### See Also

[ComboBox](#), [ContextMenu](#), [PopupMenu](#), [PushButton](#) controls.

---

## Bin Function

Returns the binary version of the number passed as a [string](#).

### Syntax

**result=Bin(value)**

| Part   | Type                    | Description                           |
|--------|-------------------------|---------------------------------------|
| result | <a href="#">String</a>  | Value converted to binary.            |
| value  | <a href="#">Integer</a> | The number to be converted to binary. |

**Notes**

If the value is not a whole number, the decimal part will be truncated.

You can specify binary, hex, or octal numbers by preceding the number with the & symbol and the letter that indicates the number base. The letter b indicates binary, h indicates hex, and o indicates octal.

**Examples**

Below are examples of various numbers converted to binary:

```
Dim binVersion As String
binVersion=Bin(15) //returns "1111"
binVersion=Bin(15.5) //returns "1111"
binVersion=Bin(75) //returns "1001011"
binVersion=Bin(&hF) //returns "1111"
```

**See Also**

[&b](#), [&h](#), [&o](#) literals; [Hex](#), [Oct](#) functions.

## BinaryStream Class

**BinaryStream** objects are used to read and write data to and from a binary file. The benefit of using BinaryStreams rather than text streams is that you can read from and write to any position in the file. Text files must be read sequentially from the start to the end.

**Super Class** [Object](#)

**Constructors** Use these constructors to back the BinaryStream from a [MemoryBlock](#) or a [String](#) instead of a file on disk.

| Name                         | Parameters                        | Description   |
|------------------------------|-----------------------------------|---|
| <a href="#">BinaryStream</a> | mb as <a href="#">MemoryBlock</a> | Creates a BinaryStream from a <a href="#">MemoryBlock</a> . All BinaryStream functions are supported. If the <a href="#">MemoryBlock</a> was created from a <a href="#">Declare</a> , then the Length property of the BinaryStream will always report -1. |

## BinaryStream Class

---

| Name         | Parameters   | Description  |
|--------------|--|--|
| BinaryStream | Handle as <a href="#">Integer</a> ,<br>Type as <a href="#">Integer</a> | Type is one of the HandleType class constants and Handle is the appropriate handle type specified by the Type parameter. The HandleType class constants are as follows:<br>1- HandleTypeWin32Handle. A Windows32 OS handle.<br>2- HandleTypeFilePointer. A file pointer.<br>3- HandleTypeFileNumber. A file descriptor.<br>4- HandleTypeMacFileRefNum. A file reference number.<br>5- HandleTypeMacFileSpecPointer. An FSSpec. For instance, you can use a <a href="#">Declare</a> to open a file with whatever permissions that you wish, and then pass the Handle to a stream object's constructor |
| BinaryStream | s as <a href="#">String</a>  | Creates a BinaryStream from a <a href="#">String</a> . All BinaryStream functions except writing and setting the length are supported. Since the string is immutable, the BinaryStream behavior is read only and the write methods of the stream produce an error.   |

## Properties

| Name          | Type                    | Description   |
|---------------|-------------------------|---|
| EOF           | <a href="#">Boolean</a> | The stream has reached the end of the file.   |
| Handle        | <a href="#">Integer</a> | Parameter is Type as <a href="#">Integer</a> . The BinaryStream class constants given below can be passed as the parameter.<br>1- HandleTypeWin32Handle. A Windows32 OS handle.<br>2- HandleTypeFilePointer. A file pointer.<br>3- HandleTypeFileNumber. A file descriptor.<br>4- HandleTypeMacFileRefNum. A file reference number.<br>5- HandleTypeMacFileSpecPointer. An FSSpec.<br>Handle returns a handle of the Type passed or -1 if the requested Type cannot be retrieved. |
| LastErrorCode | <a href="#">Integer</a> | Reports the last file I/O error. Error numbers are given as original file system error codes.   |
| Length        | <a href="#">Integer</a> | The length of the file's data fork. If you set the Length property to a value smaller than its current value, it will truncate the file.  |
| LittleEndian  | <a href="#">Boolean</a> | If <a href="#">True</a> , the byte order is assumed to be low byte, high byte. By default, BinaryStream reads data in the BigEndian byte order.   |
| Position      | <a href="#">Integer</a> | The current file position in the binarystream. This property is automatically incremented by all of the Read and Write methods.   |

## Methods

| Name        | Parameters   | Description  |
|-------------|--|--|
| Close       |  | Closes the stream (and hence the file opened by the stream).   |
| Read        | Count as <a href="#">Integer</a><br><a href="#">[,Encoding as</a><br><a href="#">TextEncoding]</a> | Reads Count characters from the stream starting at the Position property and returns them as a <a href="#">String</a> . The optional parameter <i>Encoding</i> enables you to specify the text encoding to be defined on the returned string. Use the <a href="#">Encodings</a> object to specify the text encoding. If you omit this parameter (or pass <a href="#">Nil</a> ), then the encoding of the returned string will be <a href="#">Nil</a> . |
| ReadBoolean |  | Reads a boolean value from the stream and returns it as a <a href="#">Boolean</a> .  |
| ReadDouble  |  | Reads a double word from the stream and returns it as a <a href="#">Double</a> .   |
| ReadInt16   |  | Reads a two-byte value from the stream and returns it as an <a href="#">Int16</a> .  |
| ReadInt32   |  | Reads a four-byte value from the stream and returns it as an <a href="#">Int32</a> .   |
| ReadInt64   |  | Reads an eight-byte value from the stream and returns it as an <a href="#">Int64</a> .   |
| ReadInt8    |  | Reads a one-byte value from the stream and returns it as an <a href="#">Int8</a> .   |
| ReadPString | [Encoding as<br><a href="#">TextEncoding]</a>  | Reads a Pascal string from the stream and returns it as a <a href="#">String</a> . The optional parameter <i>Encoding</i> enables you to specify the encoding to be defined on the returned string. Use the <a href="#">Encodings</a> object to specify the text encoding. If you omit this parameter (or pass <a href="#">Nil</a> ), then the encoding of the returned string will be <a href="#">Nil</a> .   |
| ReadSingle  |  | Reads a word from the stream and returns it as a <a href="#">Double</a> .  |
| ReadUInt16  |  | Reads a two-byte value from the stream and returns it as a <a href="#">UInt16</a> .  |
| ReadUInt32  |  | Reads a four-byte value from the stream and returns it as a <a href="#">UInt32</a> .   |
| ReadUInt64  |  | Reads an eight-byte value from the stream and returns it as a <a href="#">UInt64</a> .   |
| ReadUInt8   |  | Reads a one-byte value from the stream and returns it as a <a href="#">UInt8</a> .   |
| Write       | Data as <a href="#">String</a>   | Writes Data to the file starting at the Position property.   |

| Name         | Parameters                       | Description  |
|--------------|----------------------------------|--|
| WriteBoolean | Value as <a href="#">Boolean</a> | Writes the value passed as a boolean to the stream.      |
| WriteDouble  | Value as <a href="#">Double</a>  | Writes the value passed as a double word to the stream.  |
| WriteInt16   | Value as <a href="#">Int16</a>   | Writes the passed <a href="#">Int16</a> to the stream.   |
| WriteInt32   | Value as <a href="#">Int32</a>   | Writes the passed <a href="#">Int32</a> to the steam.    |
| WriteInt64   | Value as <a href="#">Int64</a>   | Writes the passed <a href="#">Int64</a> to the stream.   |
| WriteInt8    | Value as <a href="#">Int8</a>    | Writes the passed <a href="#">Int8</a> to the stream.    |
| WritePString | Text as <a href="#">String</a>   | Writes the text passed as a Pascal string to the stream. |
| WriteSingle  | Value as <a href="#">Double</a>  | Writes the value passed as a single word to the stream.  |
| WriteUInt16  | Value as <a href="#">UInt16</a>  | Writes the passed <a href="#">UInt16</a> to the stream.  |
| WriteInt32   | Value as <a href="#">Uint32</a>  | Writes the passed <a href="#">Uint32</a> to the steam.   |
| WriteInt64   | Value as <a href="#">UInt64</a>  | Writes the passed <a href="#">UInt64</a> to the stream.  |
| WriteInt8    | Value as <a href="#">UInt8</a>   | Writes the passed <a href="#">UInt8</a> to the stream.   |

### Notes

What is the LittleEndian property? The Windows and Linux operating systems store binary values in the reverse order from the Mac OS. If you were using the ReadShort or ReadLong methods to read data from a file that was in Little Endian format, you would get incorrect data. REALbasic reads data in Big Endian format. Most Macintosh files are in Big Endian format. If you are reading a file that is in Little Endian format, you will need to set the Little Endian property to [True](#) before you begin reading the file. This applies to writing data with WriteShort and WriteLong.

You can use the constants [TargetLittleEndian](#) and [TargetBigEndian](#) to determine which byte order is being used for a particular compile.

For example, in big endian (like the Mac OS), the value 258 would be stored as:

01 02

while in Little Endian, it would be stored as:

02 01

If the LittleEndian property is set incorrectly, then you would read the value as 513.

Because REALbasic has the LittleEndian property, you can write your code to be OS-independent. Set the LittleEndian property to [True](#) if the file format is intrinsically little endian (i.e. GIF files), otherwise leave it as [False](#).

The **BinaryStream** class implements the [Readable](#) and [Writable](#) class interfaces. If you implement either or both of these interfaces, you must provide the methods and parameters as specified by those class interfaces.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” on page 458 of the *User’s Guide*.

## Example

This example reads each pair of bytes from a file and writes them in reverse order to a new file. The user chooses the source file using the Open-file dialog box and saves the new file using the Save as dialog box.

```
Dim WriteToFile as BinaryStream
Dim ReadFromFile as BinaryStream
Dim f as FolderItem
f=GetOpenFolderItem("text")
If f <> Nil Then
    ReadFromFile=f.OpenAsBinaryFile(False)
    ReadFromFile.littleEndian=True
    f=GetSaveFolderItem(" "," ")
    If f <> Nil Then
        WriteToFile=f.CreateBinaryFile("text")
        While Not ReadFromFile.EOF
            WriteToFile.WriteShort ReadFromFile.ReadShort
        wend
        WriteToFile.close
    End If
    ReadFromFile.close
End If
```

## See Also

[FolderItem](#), [MemoryBlock](#), [TextInputStream](#), [TextOutputStream](#) classes;  
[TargetBigEndian](#), [TargetLittleEndian](#) constants.

---

# Bitwise Class

Performs bitwise operations on integers.

Super Class [Object](#)

## Methods

| Name   | Parameters   | Description   |
|--------|--|---|
| BitAnd | value1 <b>as</b> <a href="#">Integer</a><br>value2 <b>as</b> <a href="#">Integer</a><br>[...valueN <b>as</b> <a href="#">Integer</a> ] | Returns an <a href="#">Integer</a> . Performs a bitwise And on value1 and value2. If any optional parameters value3 to valueN are passed, BitAnd returns the progressive results of each operation. |
| BitOr  | value1 <b>as</b> <a href="#">Integer</a><br>value2 <b>as</b> <a href="#">Integer</a><br>[...valueN <b>as</b> <a href="#">Integer</a> ] | Returns an <a href="#">Integer</a> . Performs a bitwise Or on value1 and value2. If any optional parameters value3 to valueN are passed, BitOr returns the progressive results of each operation.   |

| Name           | Parameters  | Description  |
|----------------|---|--|
| BitXor         | value1 as <a href="#">Integer</a><br>value2 as <a href="#">Integer</a><br>[...valueN as <a href="#">Integer</a> ] | Returns an <a href="#">Integer</a> . Performs a bitwise Or on value1 and value2. If any optional parameters value3 to valueN are passed, BitXor returns the progressive results of each operation. |
| OnesComplement | value as <a href="#">Integer</a>  | Performs a one's compliment operation on value. Each bit of the number is inverted—zeros replaced with ones and vice versa.  |
| ShiftLeft      | value as <a href="#">Integer</a><br>shift as <a href="#">Integer</a><br>[,numBits as <a href="#">Integer</a> ]    | Shifts value to the left by shift. Only shifts bits within numBits field size. numBits defaults to 32 bits.  |
| ShiftRight     | value as <a href="#">Integer</a><br>shift as <a href="#">Integer</a><br>[,numBits as <a href="#">Integer</a> ]    | Shifts value to the right by shift. Only shifts bits within numBits field size. numBits defaults to 32 bits.   |

**Notes**

The [BitwiseAnd](#), [BitwiseOr](#), and [BitwiseXor](#) functions are obsolete. Use the corresponding methods of the **Bitwise** class.

You do not need to create an instance of the **Bitwise** class in order to access its methods. It's a special object, like [System](#) or [Application](#), that always exists.

The BitAnd method returns an [integer](#) that is the result of comparing each bit of the two integers passed (or contiguous integers passed if passing three or more integers) and assigning 1 to the bit position in the integer returned if both bits in the same position in the integers passed are 1. Otherwise, 0 is assigned to the bit position.

The BitOr method returns an [Integer](#) that is the result of comparing each bit of the two integers passed (or contiguous integers passed if passing three or more integers) and assigning 1 to the bit position in the integer returned if either of the bits in the same position in the integers passed are 1. Otherwise, 0 is assigned to the bit position.

The BitXor method returns an [Integer](#) that is the result of comparing each bit of the two integers passed (or contiguous integers passed if passing three or more integers) and assigning 1 to the bit position in the integer returned if both bits in the same position in the integers passed are not equal. Otherwise, 0 is assigned to the bit position.

The following table shows the results:

| Integer1 | Integer2 | BitAnd | BitOr | BitXor |
|----------|----------|--------|-------|--------|
| 0        | 0        | 0      | 0     | 0      |
| 0        | 1        | 0      | 1     | 1      |
| 1        | 0        | 0      | 1     | 1      |
| 1        | 1        | 1      | 1     | 0      |
|          |          |        |       |        |

Ones complement is sometimes used to represent positive and negative numbers. Positive numbers start with zeros and negative numbers start with ones. The only problem is that zero is represented two ways

| Decimal | Ones complement | Signed Decimal |
|---------|-----------------|----------------|
| 0       | 000             | 0              |
| 1       | 001             | 1              |
| 2       | 010             | 2              |
| 3       | 011             | 3              |
| 4       | 100             | -3             |
| 5       | 101             | -2             |
| 6       | 110             | -1             |
| 7       | 111             | -0             |

## Example

```
Dim i as Integer
i=Bitwise.bitand(5,3) //returns 1
i=Bitwise.bitor(5,3) //returns 7
i=Bitwise.bitxor(5,3) //returns 6
```

The following example illustrates how to re-express a bit expression that was written in C in REALbasic. The following expression in C:  
`(0xE0 | ((c >> 12) & 0x0F))`

would become:

```
Bitwise.BitOr(&hE0, Bitwise.BitAnd(Bitwise.ShiftRight(c, 12), &h0F))
```

## Boolean Data Type

**Boolean** is an intrinsic data type in REALbasic. A **Boolean** can only take on the values of [True](#) or [False](#). The default value is [False](#).

**Boolean** values are [False](#) by default but can be set to [True](#) using REALbasic's [True](#) keyword and back to [False](#) using the [False](#) keyword. Some of the properties of objects in REALbasic are **Boolean** values. In the Properties pane, they appear as checkboxes.

The [VarType](#) function returns 11 when passed a **Boolean** variable.

## Example

The following statement disables an [EditField](#), making it non-enterable and non-selectable (i.e., the user can't tab into it).

```
EditField1.Enabled=False
```

**See Also** [False](#), [True](#), [VarType](#) functions; [Dim](#) statement; [And](#), [Not](#), [Or](#) operators.

## Bounds3D Class

Represents a bounding volume, useful for simple collision and visibility testing in [RB3DSpace](#) controls. A **Bounds3D** can represent either an axis-aligned bounding box or and a bounding sphere.

**Super Class** [Object](#)

**Constructor Syntax** This table shows how to create either a bounding box or sphere.

| Syntax                       | Parameters   | Description                                   |
|------------------------------|--|---|
| <a href="#">New Bounds3D</a> | Center as <a href="#">Vector3D</a><br>Radius as <a href="#">Double</a>     | The constructor for a <b>Bounds3D</b> sphere. |
| <a href="#">New Bounds3D</a> | Minimum as <a href="#">Vector3D</a><br>Maximum as <a href="#">Vector3D</a> | The constructor for a <b>Bounds3D</b> box.    |

## Properties

| Name    | Type                     | Description  |
|---------|--------------------------|--|
| Center  | <a href="#">Vector3D</a> | The center of the sphere or box.                                     |
| Maximum | <a href="#">Vector3D</a> | The maximum x, y, and z or the bounding box.                         |
| Minimum | <a href="#">Vector3D</a> | The minimum x, y, and z of the bounding box.                         |
| Radius  | <a href="#">Double</a>   | The radius of the bounding sphere.                                   |
| Type    | <a href="#">Integer</a>  | The type of the bounding object.<br>1=Sphere<br>2=Box<br>0 =Invalid. |

## Methods

| Name       | Parameters                        | Description  |
|------------|-----------------------------------|--|
| Intersects | Other as <b>Bounds3D</b>          | Returns a <a href="#">Boolean</a> . Tests whether this bounding volume overlaps another. When in doubt, it returns <a href="#">True</a> .  |
| InView     | Rb3D as <a href="#">RB3DSpace</a> | Returns a <a href="#">Boolean</a> . Tests whether the Bounds3D volume is at least partly in the viewable part of the <a href="#">RB3DSpace</a> , <a href="#">RB3D</a> . When in doubt, it returns <a href="#">True</a> . |

| Name                    | Parameters   | Description   |
|-------------------------|--|---|
| ContainsPoint           | Pt as <a href="#">Vector3D</a>                                     | Returns a <a href="#">Boolean</a> . Tests whether this bounding volume contains the 3D point, Pt.   |
| LineIntersection        | Pt1 as <a href="#">Vector3D</a><br>Pt2 as <a href="#">Vector3D</a> | Returns a <a href="#">Vector3D</a> . Finds the point at which a line through Pt1 and Pt2 intersects the bounding volume. If it does not intersect, it returns <a href="#">Nil</a> .                             |
| LineSegmentIntersection | Pt1 as <a href="#">Vector3D</a><br>Pt2 as <a href="#">Vector3D</a> | Returns a <a href="#">Vector3D</a> . Finds the point at which a line through Pt1 and Pt2 intersects the bounding volume. If the intersection point is not between Pt1 and Pt2, it returns <a href="#">Nil</a> . |

**Notes**

As indicated in the Methods table, some of the tests are not exact; in hard-to-decide edge cases, **Bounds3D** returns [True](#), which indicates that two objects overlap or an object is visible (InView). This means that a return value of [True](#) may indicate a need for more detailed tests, but a return value of [False](#) indicates that no further testing is necessary.

Currently, the Line intersection tests treat all **Bounds3D** volumes as bounding spheres.

**See Also**

[ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [RB3DSpace](#) control.

## Break Keyword

When running in the IDE, **Break** causes the compiler to break into the debugger.

**Syntax**

**Break**

**Notes**

The **Break** keyword has the same effect as setting a breakpoint in the Code Editor. **Break** statements are ignored in standalone applications.

**See Also**

[Function](#), [Sub](#) statements.

## ByRef Keyword

Used to pass a parameter by reference.

### Syntax

### ByRef parameter

| Part      | Type          | Description   |
|-----------|---------------|---|
| parameter | Any data type | Parameter to be passed by reference. Parameter can be an array. |

### Notes

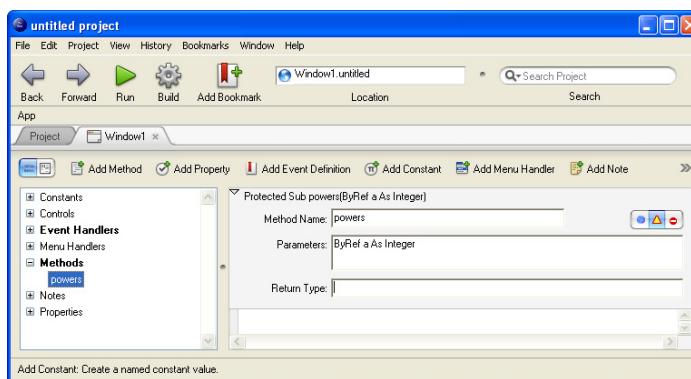
By default, you pass parameters to methods by value. When you do so, the method receives a copy of the data in the object that you pass. Your method receives the data and can perform operations on it.

To pass a parameter by reference, you use the **ByRef** keyword in the parameter declaration area within the Add Method pane. If you precede a parameter name by the keyword **ByRef**, you pass information by reference. When you pass information by reference, you actually pass a pointer to the object containing the information. The practical advantage of this technique is that the method can **change the values** of each parameter. When you pass parameters by value, you can't do this because the parameter only represents a copy of the data itself.

If the parameter is an array, **ByRef** works in the same way as if the parameter were a variable. Your method can replace the array or only some elements in the array.

### Example

The following method declaration uses **ByRef**.



When you click OK to save the new method, the Sub statement in the Code Editor shows that the parameter has been declared **ByRef**:

```
Protected Sub powers(ByRef a As Integer)
```

The Powers method takes one parameter, an [integer](#), a, that is called **ByRef**.

The method is simply:

```
a=a*a
```

Powers is called in the following code:

```
Dim a as Integer
a=3
powers a
EditField1.text=Str(a)
```

The [EditField](#) displays the value of 9.

**See Also** [ByVal](#) keyword.

## Byte Data Type

An 8-bit (1 byte) signed integer. Its value can range from -128 to 127. This data type can be used either with a [Declare](#) statement or within REALbasic. Within REALbasic, it is an alias for [Int8](#).

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type                        | Number of Bytes | Range                           |
|----------------------------------|-----------------|---------------------------------|
| <a href="#">Int8 or Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>            | 2               | -32,768 to 32,767               |
| <a href="#">Int32 or Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>           | 8               | 0 to 2^64-1                     |

**See Also** [Declare](#) statement; [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

## ByVal Keyword

Used to pass a parameter by value.

**Syntax** **ByVal** *parameter*

| Part             | Type          | Description                      |
|------------------|---------------|----------------------------------|
| <i>parameter</i> | Any Data Type | Parameter to be passed by value. |

## Call Statement

---

### Notes

**ByVal** can be used in the Method Declaration area to denote that a parameter is to be passed by value. By default, parameters are passed by value. Therefore, the **ByVal** keyword is optional.

Parameters passed by value are treated as local variables inside the method—just like variables that are created using the [Dim](#) statement. This means that you can modify the values of the parameters themselves rather than first assigning the parameter to a local variable. For example, if you pass a value in the parameter “x”, you can increment or decrement the value of x rather than assigning the value of x to a local variable that is created using [Dim](#).

### See Also

[ByRef](#) keyword.

## Call Statement

---

Enables you to call a function without handling the value returned by the function.

### Syntax

**Call** *functionName*

| Part         | Description   |
|--------------|---------------|
| functionName | Any function. |

### Notes

Use the **Call** statement only when you want to call a function without using the value that it returns. This enables you to have the function perform its job without declaring an extra local variable to store the function’s result.

### Example

The following calls the [SelectColor](#) function without using its ([Boolean](#)) result.

```
Dim c as Color
c=CMY(.35,.9,.6) //choose the default color shown in color picker
Call SelectColor(c,"Select a Color")
Rectangle1.FillColor=c
```

### See Also

[Function](#) statement.

## Cannot Assign a Value to this Property Error

---

You tried to assign a value to a constant or other object type that does not accept a value.

**Example** Assigning a value to a constant:

```
Const SerialPort=1080
SerialPort=1040
```

**See Also** [Const](#), [Dim](#) statements.

## Cannot Get this Property's Value Error

You tried to read a value for a property when it had no value.

## Canvas Control

Canvas controls are very useful for implementing your own graphical controls because you can use the [Graphics](#) class drawing commands or the [Object2D](#) classes to draw inside the Canvas region. It can also be used to display existing graphics, like the [ImageWell](#).

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) class for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| AcceptFocus | <a href="#">Boolean</a> | If <a href="#">True</a> , the control can have the focus. It can accept the GotFocus and LostFocus events and can display a focus ring if UseFocusRing is set to <a href="#">True</a> . (Macintosh only). The default is <a href="#">False</a> .  |
| AcceptTabs  | <a href="#">Boolean</a> | If <a href="#">True</a> and AcceptFocus is <a href="#">True</a> , then pressing Tab triggers the KeyDown event for processing. If <a href="#">False</a> , pressing the Tab key does not trigger the KeyDown event; pressing Tab triggers the LostFocus Event and selects the next object in the window that can accept the focus. |
| Backdrop    | <a href="#">Picture</a> | The <a href="#">Picture</a> that will automatically be drawn into the area. Must be a PICT on Macintosh or BMP or PNG on Windows and Linux unless the user has QuickTime installed (Windows and Macintosh only).  |

| Name            | Type                     | Description  |
|-----------------|--------------------------|--|
| EraseBackground | <a href="#">Boolean</a>  | Indicates whether REALbasic should erase the entire Canvas when doing operations like resizing. It defaults to True for backwards compatibility. It is supported only on Windows since this is the only platform on which REALbasic erases the background.   |
| <b>Graphics</b> | <a href="#">Graphics</a> | Allows access to all the <a href="#">Graphics</a> methods for drawing into the area. Whatever objects are drawn using the <a href="#">Graphics</a> methods are drawn after the Backdrop image is drawn, in a separate layer. This allows you to overlay objects on top of the Backdrop image. If the Canvas has no code in its Paint event, then this setting is ignored because it pertains only to Paint operations. |
| UseFocusRing    | <a href="#">Boolean</a>  | If <a href="#">True</a> , the Canvas indicates that it has the focus with a ring around its border; if <a href="#">False</a> , the appearance of the object does not change when it has the focus. Setting this property to <a href="#">True</a> has no effect on Windows or Linux. The default is <a href="#">True</a> .  |

## Events

| Name            | Parameters   | Description  |
|-----------------|--|--|
| EnableMenuItems |  | The Canvas control has received the focus. Menu handlers are invoked if the Canvas control has the focus.  |
| GotFocus        |  | The Canvas control has received the focus.   |
| KeyDown         | Key as <a href="#">String</a>                                  | The key that has been passed to the event was pressed while the Canvas control has the focus.  |
| LostFocus       |  | The Canvas control has lost the focus.   |
| MouseDown       | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the canvas region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown.  |
| MouseDrag       | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Canvas control and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |
| MouseUp         | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the Canvas region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.  |

| Name  | Parameters                    | Description   |
|-------|-------------------------------|---|
| Paint | g as <a href="#">Graphics</a> | The area needs to be redrawn, such as when it has been covered by another window and then uncovered. The parameter passed gives you access to the Graphics class methods for drawing. Use the Paint event, rather than the Open event, if you want objects drawn with the Graphics class methods to be persistent. On Windows only, the EraseBackground property determines whether REALbasic should erase the entire Canvas when repainting. |

## Methods

| Name   | Parameters   | Description   |
|--------|--|---|
| Scroll | DeltaX as <a href="#">Integer</a> ,<br>DeltaY as <a href="#">Integer</a> ,<br>[Left as <a href="#">Integer</a> ],<br>[Top as <a href="#">Integer</a> ],<br>[Width as <a href="#">Integer</a> ],<br>[Height as <a href="#">Integer</a> ],<br>[ScrollControls as <a href="#">Boolean</a> ] | DeltaX and DeltaY are the number of pixels to scroll horizontally and vertically relative to the current position of the picture in the Canvas control. Positive values scroll right and down while negative values scroll left and up. The Left, Top, Width, and Height parameters specify the portion of the picture to be scrolled. If these are not passed, the entire picture will be scrolled. ScrollControls indicates whether controls positioned on top of the Canvas control should be scrolled as well. ScrollControls is <a href="#">True</a> by default. |

## Notes

Coordinates passed to Canvas events are local to the Canvas object.

To use the Scroll method to scroll the picture in a Canvas control, you need to store the last scroll value for the axis you are scrolling so you can use this to calculate the amount to scroll. This can be done by adding a property to the window that contains the Canvas control or by creating a new class based on the Canvas control that contains properties to hold the last X scroll amount and last Y scroll amount.

If the ScrollControls parameter is [True](#), any controls on top of the Canvas control will also be scrolled. This allows the implementation of a scrolling pane of controls.

**Examples** This example draws a 3D rectangle with a raised look.

```
Sub Paint(g as Graphics)
    g.forecolor=RGB(255, 255, 255) //white
    g.drawline 1,1,canvas1.width,1
    g.drawline 1,canvas1.height-1,1,1
    g.forecolor=RGB(140,140,140) //dark gray
    g.drawline canvas1.width-1,2,canvas1.width-1,canvas1.height
    g.drawline 1,canvas1.height-1,canvas1.width,canvas1.height-1
    //fill in the light gray rectangle
    g.forecolor=RGB(239,239,239)
    g.fillRect 2,2,canvas1.width-3,canvas1.height-3
```

This example assigns a picture that has been added to the Project Editor to the Backdrop property:

```
Me.backdrop=OSLogo
```

You can then use the methods of the [Graphics](#) class to modify the picture in any way you like. For example, the following code in the object's Paint event handler adds a caption to the picture:

```
Me.Graphics.TextFont="Helvetica"
Me.Graphics.TextSize=14
Me.Graphics.ForeColor=RGB(255,255,255)
Me.Graphics.DrawString "Joe Bob",10,100
```

If you instead assigned the graphic to the BackDrop property using [NewPicture](#), as shown below, you could manipulate the graphic at the pixel level using the RGBSurface property of the [Picture](#) object.

```
Me.backdrop=NewPicture(210,30,32)
Me.backdrop.Graphics.DrawPicture madeWithREALbasicLogo,0,0
```

See also the examples for the [Control](#) class, which gives an example of dragging from a Canvas control and the [ImageWell](#) class example, which show drag and drop to and from the [ImageWell.Image](#) property.

### Simulating a Focus Ring on Windows and Linux

Unfortunately, a Focus Ring does not appear on Windows or Linux when a Canvas control gets the focus, but the GotFocus and LostFocus events fire normally. You can easily use them to simulate a focus ring.

In the GotFocus event, use:

```
#If TargetWin32 or TargetLinux
Me.Graphics.forecolor=HighlightColor //or FrameColor, whichever you wish
Me.Graphics.DrawRect 0,0,me.width-1,me.height-1
#endif
```

In the LostFocus event, use:

```
#If TargetWin32
Me.Graphics.forecolor=RGB(178,178,178) //gray
Me.Graphics.DrawRect 0,0,me.width-1,me.height-1
#endif
```

The simulated focus ring looks like this:



**See Also** [Graphics](#), [Picture](#) classes; [NewPicture](#) function.

## Catch Statement

Used to handle [RuntimeException](#) errors in a [Try](#) block.

### Syntax

**Catch [[*ErrorParameter*] [As *ErrorType*]]**

| Part           | Description   |
|----------------|---|
| ErrorParameter | Optional, used to determine the type of runtime exception.  |
| ErrorType      | Optional, if used, it must be used with ErrorParameter. Used to 'catch' a particular type of runtime error. If used, the Catch statement will handle only that type of runtime error. |

### Notes

The [Try](#) block is similar to the [Exception](#) block. Exceptions that occur within the [Try](#) block are caught by the **Catch** statement. It has the same syntax as the [Exception](#) statement. See the [RuntimeException](#) statement or the Runtime Errors theme for descriptions of each type of runtime error.

Unlike the [Exception](#) block, [Try](#) blocks can be nested. If a [Try](#) block does not handle an exception or re-raises it, it can be handled by the next outermost [Try](#) block, or by the [Exception](#) block itself if there are no containing [Try](#) blocks.

Local variables that are declared inside a **Catch** statement exist only within the **Catch** statement's scope rather than inside the whole method's scope. This means that multiple **Catch** blocks at the same level can use the same exception variable name.

## CDbl Function

---

A [Try](#) block can contain its own [Finally](#) statement. The code in the [Finally](#) statement will execute regardless of whether or not an exception was raised within the [Try](#) block.

### Example

This example uses the **Catch** statement to handle an out of range key that is passed to a [Dictionary](#).

```
Try
    SomeValue=myDictionary.Value("badKey")
Catch err as KeyNotFoundException
    MsgBox "The key could not be located."
End Try
```

### See Also

[Exception](#), [Try](#) blocks; [Finally](#), [Function](#), [Sub](#) statements; [RuntimeException](#) error.

---

## CDbl Function

This function is the same as the [Val](#) function but is used when you need to pass a [string](#) that uses a character other than the period (.) as the decimal separator. It uses the character specified by the operating system as the decimal separator.

See the [Val](#) function for more information.

### See Also

[CStr](#), [Str](#), [Val](#) functions.

---

## Ceil Function

Returns the value specified rounded up to the nearest [integer](#).

### Syntax

**result=Ceil (value)**

| Part   | Type                   | Description                        |
|--------|------------------------|------------------------------------|
| result | <a href="#">Double</a> | The ceiling of value.              |
| value  | <a href="#">Double</a> | The value you want the ceiling of. |

### Notes

The **Ceil** function returns the value passed to it rounded up to the nearest [integer](#).

### Examples

This example uses the **Ceil** function to return ceiling of a number.

```
Dim d as Double
d=Ceil(1.234) //returns 2
```

### See Also

[Floor](#), [Round](#) functions.

# CFStringRef Data Type

Passes the string as a CFString reference. This is typically used on Mac OS X. You can pass a regular REALbasic string to it and will be treated as a pointer to the memory contained within the [MemoryBlock](#).

This data type can be used only with the [Declare](#) statement and it cannot be used to declare REALbasic variables, properties, and methods.

## See Also

[Declare](#) statement; [Byte](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

# Checkbox Control

The standard checkbox button control.

## Super Class

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name       | Type                        | Description   |
|------------|-----------------------------|---|
| Bold       | <a href="#">Boolean</a>     | Applies the bold style to the Checkbox caption.   |
| Caption    | <a href="#">String</a>      | The Checkbox's label.   |
| DataField  | <a href="#">String</a>      | Relevant only if the CheckBox is used in conjunction with a <a href="#">DataControl</a> to display the contents of a field in a database table. Name of a Field in the table referenced by <a href="#">DataControl</a> .  |
| DataSource | <a href="#">DataControl</a> | The <a href="#">DataControl</a> that references a table in a database whose fields are displayed. Use the DataField property to link a field to be displayed in a CheckBox. In the IDE, a popup menu of field names will appear in the Properties pane. When setting in code, it takes a <a href="#">String</a> . Assign it to the Name property of a <a href="#">DataControl</a> . Note: Database fields can also be displayed using <a href="#">EditFields</a> , <a href="#">ListBoxes</a> , and <a href="#">StaticText</a> controls. |
| Italic     | <a href="#">Boolean</a>     | Applies the italic style to the caption.  |
| TextFont   | <a href="#">String</a>      | Name of the font used to display the caption.   |
| TextSize   | <a href="#">Integer</a>     | Size of the font used to display the caption.   |
| Underline  | <a href="#">Boolean</a>     | Applies the underline style to the caption.   |

## Chr Function

---

| Name  | Type                    | Description   |
|-------|-------------------------|---|
| Value | <a href="#">Boolean</a> | The value of the checkbox. <a href="#">True</a> indicates it is checked. The Action event fires when the Value property is changed. |

## Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| Action    |  | The CheckBox has been clicked or the value has been changed programmatically. A right-click does not trigger the Action event.  |
| GotFocus  |  | (Windows and Linux) The control has received the focus and has a selection marquee around its caption.  |
| LostFocus |  | (Windows and Linux) The control has lost the focus.   |
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the control's region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown. The Action event will not execute and the state of the object will not change. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the control's region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. The return value is ignored.                               |

## Note

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. For example, if the caption should read “Bread & Butter”, enter “Bread && Butter”.

This is done to make applications on all operating systems behave consistently. The ampersand is used to denote a keyboard accelerator on Windows.

## Example

The following code in the CheckBox's Action event handler checks the value of the CheckBox.

```
If CheckBox1.value then  
    EditField1.text="True"  
else  
    EditField1.text="False"  
end if
```

## See Also

[RadioButton](#) control; [RectControl](#) class.

---

## Chr Function

Returns the character whose ASCII value is passed.

**Syntax*****result=Chr(value)***

| Part   | Type   | Description   |
|--------|--|---|
| result | <a href="#">String</a>   | The character whose ASCII value was passed.                                     |
| value  | <a href="#">Integer</a> ,<br><a href="#">Single</a> , or<br><a href="#">Double</a> | The numeric value ("code point") of the character you want, in the range 0-127. |

**Notes**

The **Chr** function returns the character whose value is specified. It is valid for the standard ASCII character set only. For this reason, **Chr** may return unexpected results unless you use it only for ASCII values (0-127).

In general, you should use the **Chr** method of the [TextEncoding](#) class. With it, you specify both the encoding and the character code in that encoding. Please see the example in the [TextEncoding](#) class and the last line in this example. In effect, the **Chr** function supports the special case in which the encoding is ASCII and *value* is less than 128.

The **Chr** function will return a single byte [string](#) when running on single byte systems and return a double byte string when running on double byte systems. If you need to get a single byte string regardless of whether the system software is single or double byte, use the [ChrB](#) function.

**Examples**

These examples use the **Chr** function to return the characters whose ASCII values are specified.

```
Dim Tab,CR as String
Tab=Chr(9) //returns a tab
CR=Chr(13) //returns carriage return
CR=Encodings.ASCII.Chr(13) //also returns carriage return
```

**See Also**

[Asc](#), [Encoding](#), [InStr](#), [Left](#), [Len](#), [Mid](#), [Right](#) functions; [EndOfLine](#), [TextEncoding](#) classes; [Encodings](#) object.

## ChrB Function

Returns a single byte [string](#) whose value is passed.

**Syntax*****result=ChrB(value)***

| Part   | Type   | Description                                    |
|--------|--|--|
| result | <a href="#">String</a>   | The single byte string whose value was passed. |
| value  | <a href="#">Integer</a> ,<br><a href="#">Single</a> , or<br><a href="#">Double</a> | The value of the character you want.           |

## ClearFocus Method

---

### Notes

The **ChrB** function returns a single byte string whose value is specified. **ChrB** should be used rather than [Chr](#) when *value* represents binary data.

If you need to get a single byte string when running on single byte system software and a double byte string when running on double byte system software, use the [Chr](#) function.

If you need to specify a code point above 127, use the [Chr](#) property of the [TextEncoding](#) class. With it, you specify both the encoding and the character code in that encoding.

### Examples

These examples use the **ChrB** function to return the characters whose values are specified.

```
Dim s as String  
s=ChrB(32) //returns a space  
s=ChrB(13) //returns carriage return
```

### See Also

[AscB](#), [Chr](#), [Encoding](#), [InStrB](#), [LeftB](#), [LenB](#), [MidB](#), [RightB](#) functions; [EndOfLine](#), [TextEncoding](#) classes; [Encodings](#) object.

## ClearFocus Method

---

Removes the focus from the control that has the focus, leaving no control with the focus.

### Syntax

**ClearFocus**

### Notes

Calling ClearFocus has the same effect as calling the [Window](#) class method, SetFocus. You can set the focus using the SetFocus property of the [RectControl](#) class. Please note that the SetFocus method of the [RectControl](#) class does nothing when the control it is being applied to is not capable of having the focus on the platform on which the code is running.

### Example

The following code in the Action event of a [PushButton](#) removes the focus from the object in the window that has the focus.

```
ClearFocus
```

### See Also

[RectControl](#), [Window](#) classes.

# Clipboard Class

Used to read and write information to and from the Clipboard.

**Super Class** [Object](#)

## Properties

| Name                    | Type                    | Description   |
|-------------------------|-------------------------|---|
| Picture                 | <a href="#">Picture</a> | The picture on the Clipboard.                                     |
| <b>PictureAvailable</b> | <a href="#">Boolean</a> | Returns <a href="#">True</a> if the Clipboard contains a picture. |
| Text                    | <a href="#">String</a>  | The text on the Clipboard.  |

## Methods

| Name             | Parameters                        | Description  |
|------------------|-----------------------------------|--|
| AddRawData       |                                   | Copies the passed data to the Clipboard  |
| Close            |                                   | Closes the Clipboard. The Clipboard must be closed within the event handler it was opened in or an error will occur. |
| RawData          |                                   | Returns the binary data from the Clipboard as a <a href="#">string</a> for the specified type.                       |
| RawDataAvailable | macType as <a href="#">String</a> | Returns <a href="#">True</a> if the Clipboard contains data.   |
| SetText          | Text as <a href="#">String</a>    | Sets the Clipboard to the text passed.   |
| TextAvailable    |                                   | Returns <a href="#">True</a> if the data on the Clipboard is text.   |

## Notes

In order to read text from or write text to the Clipboard, you must create an object of type **Clipboard** and then access the Text or Picture properties of the Clipboard object you create. You can place a picture on the Clipboard by setting the Picture property. You can use the SetText and AddRawData methods to write text or binary data to the Clipboard.

## Examples

This example gets the text on the Clipboard and copies it to a variable.

```
Dim c as New Clipboard
Dim s as String
s=c.text
c.close
```

## CLong Function

---

This example puts text on the Clipboard.

```
Dim c as New Clipboard  
c.text="The Quick Brown Fox outran the Lazy Dog"  
c.close
```

This example checks to see if the Clipboard is text data and if so, copies the contents of the clipboard to an [EditField](#).

```
Dim c as New Clipboard  
If c.TextAvailable then  
    EditField1.text=c.Text  
End if  
c.close
```

This example copies a PICT image to the Clipboard:

```
Dim c as New Clipboard  
If ImageWell1.image <> Nil then  
    c.Picture=ImageWell1.image  
    c.Close  
else  
    MsgBox "No picture is available!"  
End if
```

When you want to put more than one item on the clipboard, you can't use the properties of this class to append new text or graphics to existing material. That is, the following code won't put both text strings on the Clipboard:

```
Dim c as New Clipboard  
c.text="This is the first item being put on the clipboard!"  
c.text="This is the second item being put on the clipboard"  
c.close
```

You need to do the appending in your code and just use one call to the Text property.

---

## CLong Function

Returns the numeric form of a [string](#) as an [Int64](#).

**Syntax**      **result=CLong(string)**

**OR**

**result=stringVariable.CLong**

| Part           | Type                   | Description                                   |
|----------------|------------------------|---|
| result         | <a href="#">Int64</a>  | The numeric equivalent of the string passed.  |
| string         | <a href="#">String</a> | Any valid <a href="#">string</a> expression.  |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> . |

**Notes**

The **CLong** function stops reading the [string](#) at the first character it doesn't recognize as part of a number. All other characters are automatically stripped.

It does recognize prefixes [&o](#) (octal), [&b](#) (binary), and [&h](#) (hexadecimal). However, spaces are not allowed in front of the ampersand. That is, “[&hFF](#)” returns 0, but “[&h FF](#)” returns 255.

**CLong** returns zero if string contains no numbers.

**Examples**

These examples use the **CLong** function to return the numbers contained in a [string](#) literal or a [string](#) variable

```
Dim n As Int64
Dim s As String
n = CLong("12345") //returns 12345
n = CLong(" 12345") //returns 0
n = CLong("123 45") //returns 123
n = CLong("&hFFF") //returns 4095
n = CLong("&b1111") //returns 15

s="12345"
n=s.CLong //returns 12345
```

**See Also**

[CDbl](#), [CStr](#), [Str](#), [Val](#) functions; [&b](#), [&h](#), [&o](#) literals; [Int64](#) data type.

## CMY Function

Returns a [Color](#) based on the CMY (cyan, magenta, yellow) color model.

**Syntax****result=CMY(cyan, magenta, yellow)**

| Part    | Type                   | Description  |
|---------|------------------------|--|
| result  | <a href="#">Color</a>  | An object that represents the color based on the cyan, magenta, and yellow values. |
| cyan    | <a href="#">Double</a> | The value of Cyan in the color (0-1).  |
| magenta | <a href="#">Double</a> | The value of Magenta in the color (0-1).   |
| yellow  | <a href="#">Double</a> | The value of Yellow in the color (0-1).  |

### Notes

The **CMY** function returns a [Color](#) based on the amounts of Cyan, Magenta, and Yellow passed. These amounts are represented by [doubles](#) between 0 and 1 and are stored internally as three bytes. You can also use either the [RGB](#) or [HSV](#) models to assign a color. You can specify a color literal with the [&c](#) literal.

### Examples

This example uses the **CMY** function to assign a [Color](#) to the Fillcolor property of a [Rectangle](#) control.

```
rectangle1.fillcolor=CMY(.35,.9,.6)
```

### See Also

[Color](#) data type; [HSV](#), [RGB](#), [SelectColor](#) functions; [&c](#) literal.

---

## Collection Class

Used to store a set of related items in a larger structure. The concept of a collection is similar to that of an array, except that each element in a collection can be a different data type. A collection can be thought of as an array of [variants](#). Also, each element in a collection can be named. Elements in a collection can be referred to either by number or name.

The [Dictionary](#) class provides all the functionality of the **Collection** class and offers several advantages: With the **Collection** class, the time taken to locate an item is a function of the number of items in the **Collection** because the search is sequential. A [Dictionary](#) uses a hash table, making the time (relatively) independent of the number of items. It is designed for high-speed lookups. Also, the key parameter in a [Dictionary](#) is a [Variant](#), but is a [String](#) in the **Collection** class. Therefore, we recommend that you use the [Dictionary](#) class rather than the **Collection** class whenever possible.

The **Collection** class is included mainly for backward compatibility and for conversion of Visual Basic projects.

### Super Class

[Object](#)

### Methods

| Name  | Parameters  | Description   |
|-------|---|---|
| Add   | Value as <a href="#">Variant</a><br>Key as <a href="#">String</a>       | Value is added as the last element of the collection; if Key is provided, it may be used to refer to the element using the Item property. |
| Count |   | Number of elements in the collection. Returns an <a href="#">integer</a> .  |
| Item  | Index as <a href="#">Integer</a><br>OR<br>key as <a href="#">String</a> | Refers to an element of a collection. Returns a <a href="#">Variant</a> . Index is 1-based.   |

| Name   | Parameters  | Description  |
|--------|---|--|
| Remove | Index as <a href="#">Integer</a><br>OR<br>key as <a href="#">String</a> | Removes the specified element of a collection. Index is 1-based. |

## Example

The following example creates a collection, populates it with both [string](#) and numeric values, and displays each element in [EditFields](#) or a [Canvas](#) control (The picture “lois” has been added to the project). Note that a collection is much like a database record.

```
Dim c as New Collection
c.add 1, "ID"
c.add "Lois Lane", "Name"
c.add "Reporter", "JobTitle"
c.add 85000, "Salary"
c.add lois, "Picture"
EditField1.text=c.item("ID")
EditField2.text=c.item(2) //returns "Lois Lane"
EditField3.text=c.item("JobTitle")
EditField4.text=c.item("Salary")
Canvas1.backdrop=c.item("Picture")
```

If you want to use the Item or Remove methods to refer to an item, use parentheses around the parameter passed to the method. This is because the compiler doesn't know which data type you are passing. For example, use

```
c.remove("Name")
```

rather than

```
c.Remove "Name"
```

to remove the “Lois Lane” record from the collection.

## See Also

[Dictionary](#), [Variant](#) classes; [VarType](#) function; [Nil](#) keyword.

# Color Data Type

A **Color** is an intrinsic data type that consists of three bytes that define the color. A color can be set using the [RGB](#), [HSV](#), or [CMY](#) models or the [&c](#) literal. Each set of properties is used with the appropriate function to specify a color. The values of any of the nine properties can then be read. The default value of a **Color** is [&c000000](#) (white).

### Properties

| Name       | Type                    | Description                                       |
|------------|-------------------------|---|
| Red        | <a href="#">Integer</a> | The amount (0-255) of red in the color (RGB)      |
| Green      | <a href="#">Integer</a> | The amount (0-255) of green in the color (RGB).   |
| Blue       | <a href="#">Integer</a> | The amount (0-255) of blue in the color (RGB).    |
| Hue        | <a href="#">Double</a>  | The value of Hue (0-1) in the color (HSV).        |
| Saturation | <a href="#">Double</a>  | The value of Saturation (0-1) in the color (HSV). |
| Value      | <a href="#">Double</a>  | The value of Value (0-1) in the color (HSV).      |
| Cyan       | <a href="#">Double</a>  | The value of Cyan (0-1) in the color (CMY).       |
| Magenta    | <a href="#">Double</a>  | The value of Magenta (0-1) in the color (CMY).    |
| Yellow     | <a href="#">Double</a>  | The value of Yellow (0-1) in the color (CMY).     |

### Notes

Colors are often used to assign colors to properties of type **Color**. Use the [RGB](#), [HSV](#), or [CMY](#) functions to assign a color to an object or use the format:

```
&cRRGGBB
```

where *RR* is the RGB value of Red in hexadecimal, *GG* is the value Green in hexadecimal, and *BB* is the value of Blue in hexadecimal. You can choose the color that you want to assign to a property by clicking on the existing color in the Properties pane to open the Color picker. Choose the color via the Color picker and REALbasic will figure out the hexadecimal values that represent that color.

You can use the [VarType](#) function to determine whether a property or a variable is a **Color**. If it is, [VarType](#) returns 16.

### Examples

The following example uses the [RGB](#) model to set the forecolor property of a [Canvas](#) control and draw a square using the current forecolor. The code is placed in the [Canvas](#) control's Paint event.

```
Canvas1.graphics.foreColor=RGB(255,0,0)  
Canvas1.graphics.drawrect 0,0,50,50
```

The following example assigns a color directly using the RGB color model. The RGB values must be specified in hexadecimal:

```
Canvas1.graphics.foreColor=&cFF0000  
Canvas1.graphics.drawrect 0,0,50,50
```

The following example uses the [CMY](#) model to set the fillcolor of a [Rectangle](#) control, *r*.

```
r.fillColor=CMY(.1,.3,.5)
```

The following example inverts the fillcolor:

```
Dim c as Color
c=RGB(255-r.fillcolor.red,255-r.fillcolor.green, 255-r.fillColor.blue)
r.fillColor=c
```

## See Also

[CMY](#), [DarkTingeColor](#), [FillColor](#), [FrameColor](#), [HighlightColor](#), [HSV](#), [LightBevelColor](#), [LightTingeColor](#), [RGB](#), [SelectColor](#), [TextColor](#), [Variant](#), [VarType](#) functions; [&c](#) literal; [Dim](#) statement.

---

## ColorList Class

Used to specify the vertex colors of a [Trimesh](#).

Super Class [Object](#)

### Properties

| Name    | Type                    | Description  |
|---------|-------------------------|--|
| Enabled | <a href="#">Boolean</a> | If <a href="#">True</a> , the <a href="#">Trimesh</a> has vertex colors; if <a href="#">False</a> , it has no colors and an attempt to access it will raise an <a href="#">OutOfBoundsException</a> exception. |

### Methods

| Name        | Parameters  | Description   |
|-------------|---|---|
| AddRGBToAll | Red as <a href="#">Integer</a> , Green as <a href="#">Integer</a> , Blue as <a href="#">Integer</a> | Adds the passed RGB color to all colors in the list. The parameters take on values of 0 to 255. |
| Copy        | other as <a href="#">ColorList</a>  | Copies the passed <a href="#">ColorList</a> into the current list.                              |
| Item        | Index as <a href="#">Integer</a>  | Gets or sets the color of the passed item. Returns a <a href="#">Color</a> .                    |

### Example

The following sets the colors of each vertex in a triangle to red, green, and blue, respectively.

```
Dim tm as Trimesh  
tm=New Trimesh  
  
// define the vertices  
tm.VertexCount = 3  
tm.VertexPositions.SetXYZ(0, 5, 5, 0 )  
tm.VertexPositions.SetXYZ(1, 5, 10, 0 )  
tm.VertexPositions.SetXYZ(2, 10, 10, 0 )  
  
tm.HasVertexColors=True  
tm.VertexColors.Item(0) = &cFF0000  
tm.VertexColors.Item(1) = &c00FF00  
tm.VertexColors.Item(2) = &c0000FF
```

### See Also

[Bounds3D](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

## ComboBox Control

A data entry control that is similar to the [PopupMenu](#), but allows for either selection from a list or keyboard entry. It behaves like a single-line [EditField](#) that has a popup menu attached to it. The **ComboBox** can get the focus on all platforms.

### Super Class

[PopupMenu](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name         | Type                    | Description   |
|--------------|-------------------------|---|
| AutoComplete | <a href="#">Boolean</a> | If <a href="#">True</a> , the user can use the autocomplete feature in REALbasic to complete text entries.  |
| SelStart     | <a href="#">Integer</a> | The position of the insertion point. A value of zero means that the insertion point is before the first character. Use SelStart to set the position of the insertion point in the text. Use SelStart in conjunction with SelLength to select a portion of the text in the ComboBox, beginning with <i>SelStart</i> and extending for <i>SelLength</i> characters. |

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| SelLength    | <a href="#">Integer</a> | The number of highlighted characters. A SelLength of 0 means an insertion point rather than a selection. A value greater than the number of characters in the ComboBox means that the selection is from SelStart to the end of the ComboBox. |
| Text         | <a href="#">String</a>  | Gets or sets the text of the current item.   |
| UseFocusRing | <a href="#">Boolean</a> | If <a href="#">True</a> , the control indicates that it has the focus with a ring around its border; if <a href="#">False</a> , the appearance of the control does not change when it has the focus.   |

## Events

| Name        |                               | Description   |
|-------------|-------------------------------|---|
| GotFocus    |                               | The ComboBox has received the focus. If the user tabbed to select it, the current item is highlighted.  |
| KeyDown     | key as <a href="#">String</a> | Returns a <a href="#">Boolean</a> . The user has pressed the Key passed while the ComboBox has the focus. Returning <a href="#">True</a> means that no further processing is to be done with the Key. |
| LostFocus   |                               | The ComboBox has lost the focus.  |
| TextChanged |                               | The text in the ComboBox has changed.   |

## Examples

This code in the Open event handler populates a **ComboBox** and sets the initial value to the current month:

```
Dim s as String
Dim i,last as Integer
Dim d as New Date
s="January,February,March,April,May,June,July, "
    +"August,September,October,November,December"
last=CountFields(s,",")
For i=1 to last
    me.addRow NthField(s,",",i)
Next
me.ListIndex=d.Month-1
```

The value of the ListIndex property contains the index of the selected item, but it does not indicate whether the user has entered text into the **ComboBox**. Examine the Text property to get the current menu selection or the text entered by the user. For example, the following line in the TextChanged event handler displays either the currently selected menu item or the text typed into the **ComboBox**.

```
StaticText1.Text=Me.Text
```

## ConsoleApplication Class

---

This example adds an item to a **ComboBox** in its Open event handler.

```
Me.addrow "October"
```

This example populates the RowTag identifier with a sequence number:

```
Dim i,nItems as Integer  
nItems=Me.ListCount-1  
For i=0 to nItems  
  Me.rowTag(i)=i  
Next
```

Since RowTag is a [Variant](#), you must first convert it to an [Integer](#) if you want to compare it to another integer. Do this with a simple assignment statement such as:

```
Dim RecID as Integer  
RecID=ComboBox1.Rowtag(1)
```

Then compare RecID to another [Integer](#).

This example opens a new window when an item is chosen.

```
Sub Change()  
Dim w As ListEditorWindow  
If ComboBox1.text="Edit List..." Then  
  w=New ListEditorWindow  
End If
```

The following line changes the selected item in a **ComboBox**:

```
ComboBox1.listIndex=3
```

Displaying the RowTag of the selected menu item:

```
EditField1.text=ComboBox1.rowtag(ComboBox1.listindex)
```

**See Also** [ContextMenu](#), [EditField](#), [PopupMenu](#) controls; [RectControl](#) class.

---

## ConsoleApplication Class

Used to run console applications on Windows, Mac OS X, and Linux.

**SuperClass** [Object](#)

## Properties

| Name           | Type                       | Description   |
|----------------|----------------------------|---|
| ExecutableFile | <a href="#">FolderItem</a> | Points to the actual executable file even if it is inside a bundle. |

## Methods

| Name      | Parameters                                 | Description  |
|-----------|--|--|
| Daemonize |  | Returns a <a href="#">Boolean</a> . Converts the application from a regular console application to a daemon on Mac OS X or Linux. See the Notes for an example.  |
| DoEvents  | [milliseconds as <a href="#">Integer</a> ] | Yields time back to REALbasic so that it can handle other events. The optional parameter specifies the amount of time you want the currently executing thread to sleep. If all threads are sleeping, then REALbasic yields back time to the system. This allows you to write applications that don't use up to 100% of the CPU during tight loops.<br>Specifying zero milliseconds causes the next waiting thread to execute. A negative value specifies no sleep. The default is 10 milliseconds. |

## Events

| Name               | Parameters                     | Description   |
|--------------------|--------------------------------|---|
| Run                | args as <a href="#">String</a> | Returns an <a href="#">Integer</a> . Args is an array of command line parameters that get passed to the application. The first parameter will always be the application itself. The return value will be passed to the operating system as the return value for the entire application. |
| UnhandledException |                                | Occurs when a <code>RuntimeException</code> occurs that is not otherwise handled. This event allows you to do any last-minute clean-up, but the application does not resume after this event. The application terminates after this event.  |

## Notes

A console application differs from a desktop application in that it contains no graphical user interface and works only from the command line. In Mac OS X, a console application runs within the Terminal application; on Windows, it runs from the command line prompt, and on Linux it runs from the command line or a Terminal window.

To create a console application, choose File ▶ New Project and choose Console Application from the New Project dialog box. This will create a new project without

items for the default window and menubar. The Project Editor will only have the App class, which is subclassed from **ConsoleApplication** instead of **Application**.

Because a console application has no windows or menus of its own, it communicates with the user is through the [Print](#) and [Input](#) commands or the [StandardInputStream](#) and [StandardOutputStream](#) classes, which provide equivalent functionality.

When you create a GUI application, the program execution begins in the Open event of the [Application](#) class and halts when you call the [Quit](#) method or the user quits your application by choosing File ▶ Quit. Your program will stay loaded in memory and running until a [Quit](#) command is received.

A console application behaves differently from a desktop application. The program execution begins in the Run event of the **ConsoleApplication** class and terminates when you exit the Run event or call the [Quit](#) method. In other words, the entire application executes inside of the Run event.

If you would like your application to behave in a more GUI-like way, where it continues to run until the user interacts with it, then you can do that by placing a [While](#) loop in the Run event.

A [ServiceApplication](#) is a special type of console application that is designed to run without user intervention of any kind. The typical type of service application is a an internet server, such as an HTTP, FTP, or WebDAV server, which is capable of running without any user logged into the machine.

On Linux, a console application does not require GTK, GDK, or CUPS.

A console application cannot use the [Graphics](#) class. This rules out the use of the [Picture](#) class, the [Movie](#) class, and all other graphics-oriented classes.

### The Daemonize Method

A typical use of the Daemonize method is as follows:

```
If (args(1) = "start" or args(1) = "-d") then
    If Not App.Daemonize then
        System.Log( System.LogLevelCritical, "Could not daemonize the "
            +"application")
    Return -1
end
end
```

### See Also

[Input](#), [Print](#), [StdErr](#), [StdIn](#), [StdOut](#) methods; [ServiceApplication](#), [StandardInputStream](#), [StandardOutputStream](#) classes; [TargetHasGUI](#) constant.

---

## Const Statement

Declares a value as a local constant.

**Syntax**    **Const** *constantname* = *value*

**Notes**    The **Const** statement can be used in place of the [Dim](#) statement followed by an assignment statement when you are sure that the value of the variable should not change within the method. Using **Const** instead of [Dim](#) provides a convenient way to manage such values.

A **Const** statement can be placed anywhere in a method, including inside a conditional structure, such as an [If](#) statement, or a looping structure.

Constants declared in this manner are local to the method. You can also create constants in a window, class, or module, as described in the [User's Guide](#). Constants that belong to windows or classes can be public (accessible throughout the application, protected (accessible only within the object that owns the constant and its subclasses), or private (accessible only within the object that owns it). Modules can have global constants, accessible throughout the application.

**Examples**    The following constant is used to set a button's caption.

```
Const Accept="OK"
BevelButton1.caption=Accept
```

The following sets the value of "Pi" for the method.

```
Const Pi=3.14159265358979323846264338327950
```

**See Also**    [Dim](#) statement.

## Constructor Method

Fired automatically when an object is instantiated.

**Syntax**    **Constructor** [*parameter list*]

| Part           | Type | Description                  |
|----------------|------|------------------------------|
| Parameter List | Any  | Optional list of parameters. |

**Notes**    When you create a new object, you will sometimes want to perform some sort of initialization on the object. The constructor is a mechanism for doing this. A class's constructor is the method that will be executed automatically when an instance of the class is created.

You write a constructor for a custom class by creating a new method for the class and naming it "Constructor". See the chapter on custom classes in the *Users Guide* for more information on writing constructors.

**See Also** Destructor Method

## ContainerControl Class

Used to embed a group of controls in a [window](#) or in another [control](#). This is a Professional-only feature.

**Super Class** [Window](#)

### Properties

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| AcceptFocus | <a href="#">Boolean</a> | If <b>True</b> , the <b>ContainerControl</b> can have the focus. It can accept the GotFocus and LostFocus events.           |
| Parent      | <a href="#">Control</a> | Gets the containing control, if any.  |
| Window      | <a href="#">Window</a>  | Gets the containing window. Multiple levels of embedding are permitted, so this window may not be the highest level window. |

### Methods

| Name        | Parameters  | Description   |
|-------------|---|---|
| EmbedWithin | ContainingWindow as <a href="#">Window</a><br>[,left as <a href="#">Integer</a> ]<br>[,top as <a href="#">Integer</a> ]<br>[,width as <a href="#">Integer</a> ]<br>[,height as <a href="#">Integer</a> ]<br>or<br>ContainerControl as <a href="#">RectControl</a><br>[,left as <a href="#">Integer</a> ]<br>[,top as <a href="#">Integer</a> ]<br>[,width as <a href="#">Integer</a> ]<br>[,height as <a href="#">Integer</a> ] | Embeds the <b>ContainerControl</b> in <i>ContainingWindow</i> or <i>ContainerControl</i> without drawing the window frame, title bar, title bar widgets, grow handle, and so forth. If a control is passed instead of a window, the <b>ContainerControl</b> will embed on the passed control as the parent of the <b>ContainerControl</b> . If the <b>ContainerControl</b> is embedded on a <a href="#">PagePanel</a> or <a href="#">TabPanel</a> , it is embedded on the current page. However, we recommend using <b>EmbedWithinPanel</b> for this purpose.<br>The optional <i>Left</i> and <i>Top</i> parameters determine the location of the top-left corner, relative to the containing window or containing control. The optional parameters <i>Width</i> and <i>Height</i> determine the size of the embedded window. |

| Name             | Parameters  | Description   |
|------------------|---|---|
| EmbedWithinPanel | ContainingPanel as <a href="#">PagePanel</a><br>Page as <a href="#">Integer</a><br>[,left as <a href="#">Integer</a> ]<br>[,top as <a href="#">Integer</a> ]<br>[,width as <a href="#">Integer</a> ]<br>[,height as <a href="#">Integer</a> ] | Embeds the <b>ContainerControl</b> on a page in the passed <a href="#">PagePanel</a> or <a href="#">TabPanel</a> . The <b>ContainerControl</b> is embedded on the passed page and the containing control is the parent of the <b>ContainerControl</b> . The optional <i>Left</i> and <i>Top</i> parameters determine the location of the top-left corner, relative to the containing control, not the parent window. The optional parameters <i>Width</i> and <i>Height</i> determine the size of the <b>ContainerControl</b> . |

## Notes

To create a **ContainerControl**, add an instance of this class to your project. Do this either by clicking the Add Container Control button in the Project Editor toolbar or with the Project ▶ Add ▶ Add Container Control menu item. When you do so, REALbasic adds an instance of a **ContainerControl** to the project.

Double-click the **ContainerControl** item in the Project panel to open a Container Control Editor. The Container Control Editor looks like a Window Editor, except that it works only for ContainerControls. A **ContainerControl** doesn't have a Frame property (or a frame, for that matter), and has no Title bar, grow handle, or the other widgets that a window can have. The area of the **ContainerControl** is represented by a gray box.

The Container Control editor has the same Controls list as a Window Editor. Use it to add the controls or other ContainerControls to this **ContainerControl**.

When your design for the **ContainerControl** is complete, you can embed it in a window or control in either the IDE or via code. Multiple levels of embedding are supported.

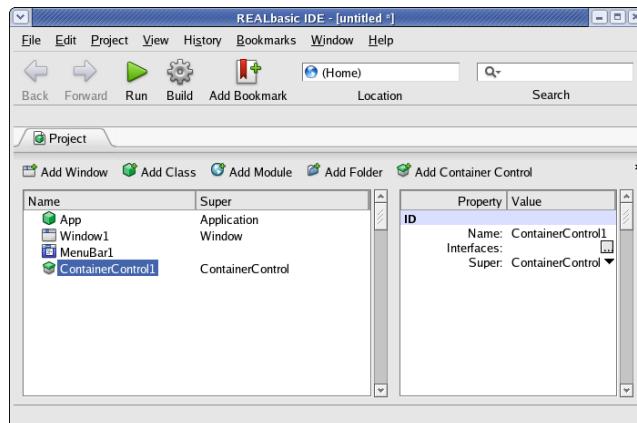
In the IDE, you can embed the **ContainerControl** in the following way:

- 1 **Switch to the Window Editor for the window that will contain the ContainerControl.**
- 2 **Use the popup menu above the list of controls to display the Project controls.**
- 3 **Add the ContainerControl to the window by dragging, double-clicking, or by clicking and drawing a region in the window.**
- 4 **Resize and reposition it as needed.**

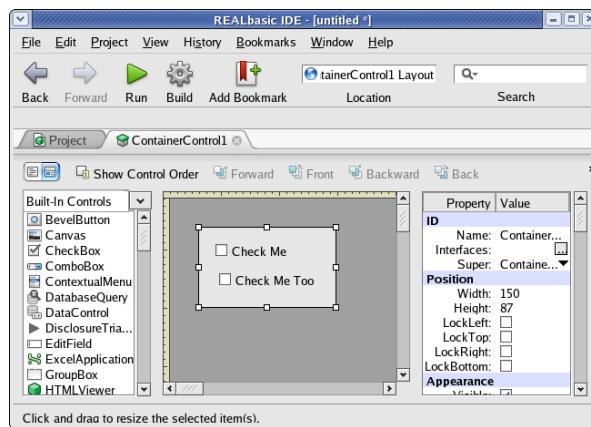
To add the ContainerControl to a window via code, use either the EmbedWithin or EmbedWithinPanel methods. Use EmbedWithin to embed the **ContainerControl** in either a window or a control, depending on whether the first parameter is a [window](#) or a [control](#). For the special case of embedding within a [PagePanel](#) or a [TabPanel](#), use EmbedWithinPanel instead. It allows you to pass the page number on which the **ContainerControl** will be embedded.

## ContainerControl Class

The following example uses a **ContainerControl** that has been added to the project.



You double-click a **ContainerControl** in the Project Window to display its ContainerControl Editor. Two checkboxes have been added.



The following statement embeds a **ContainerControl** at so that its top-left corner is 50 pixels from the left side of the window and 100 pixels from the top.

```
ContainerControl1.EmbedWithin(Self,50,100)
```

When the project is run, the controls in the ContainerControl appear in the default window, Window1. Use the same approach to embed the ContainerControl in a control other than a PagePanel or TabPanel; for the latter types of controls, use EmbedWithinPanel and pass the name of the control and the desired panel number.

## Uses

ContainerControls have multiple uses. You can:

- Organize groups of controls into reusable interface components,
- Create custom controls made up of several constituent controls,

- Increase encapsulation of complex window layouts,
- Create dynamic layouts.

**Behaviors** For the most part ContainerControls act exactly how you would expect. For example, if you put code in the MouseMove event of an embedded window, the event will fire when your mouse moves over the embedded window's boundaries. There are a few things you need to be aware of:

The Handle property of a **ContainerControl** and the Handle property of controls of an **ContainerControl** are [Nil](#) until the Open event. All of the other properties can be manipulated before the Open event.

**ContainerControl** either have their own keyboard focus and menu handling or they share it with their containing window. Which behavior is chosen depends on the state of the AcceptFocus flag when the window is embedded. When AcceptFocus is [True](#), the **ContainerControl** does not share focus with the window. If a containing window has embedded that which share focus with it, those windows will get a first crack at handling it. If none handle it, the containing window will get a try. This applies to KeyDown and MenuCommands. It also affects how menu commands are enabled.

Some properties are new to **ContainerControls** and relate to its behavior when embedded; these behave like the corresponding properties of the [Canvas](#) control. These include the following: LockLeft, LockTop, LockRight, LockBottom, Enabled, AutoDeactivate, HelpTag, DisabledBalloonHelp, UseFocusRing, AcceptFocus, AcceptTabs, Parent and Window.

**Miscellaneous Issues** The Moved, Resized, and Resizing events fire on embedded windows when the embedded window is moved or resized.

The Show/Hide and the Visible property can be used to set the visibility of embedded windows.

Nesting ContainerControls is allowed. However, you can't embed a **ContainerControl** such that the containing **ContainerControl** or a **ContainerControl** higher in the containing chain is another instance of the same **ContainerControl**; in other words, you can't recursively nest ContainerControls in other instances of themselves.

Nested ContainerControl coordinates for controls and events are automatically transformed for you. You don't need to worry about them unless you are dealing with global coordinates as you would for the [MenuItem's](#) Popup method.

The CancelClose event will fire for any **ContainerControl** or nested ContainerControls. Returning [True](#) from any of these ContainerControls will prevent the window or any of its controls from closing.

**See Also** [RectControl](#), [Window](#) classes.

# ContextMenu Control

Used to display and control contextual menus.

## Super Class [Control](#)

Because this is a [Control](#), see the [Control](#) class for other properties and events that are common to all [Control](#) objects.

## Events

| Name   | Parameters                     | Description   |
|--------|--------------------------------|---|
| Action | Item as <a href="#">String</a> | The <i>Item</i> from the contextual menu that was chosen. |

## Properties

| Name      | Type                    | Description  |
|-----------|-------------------------|--|
| UseMacCMM | <a href="#">Boolean</a> | If <a href="#">True</a> , the Help item is displayed. If <a href="#">False</a> , the Help item is omitted. This has no effect on Windows or Linux. |

## Methods

| Name          | Parameters                     | Description   |
|---------------|--------------------------------|---|
| AddRow        | Item as <a href="#">String</a> | Appends <i>Item</i> in a new row to the end of the list. Pass a dash to add a separator as a menu item. |
| AddSeparator  |                                | Appends a separator line to the end of the list.  |
| DeleteAllRows |                                | Deletes all rows in the list.   |
| Open          |                                | Displays the contextual menu.   |

## Notes

The ConstructContextMenu events of the [Window](#) and [RectControl](#) class are the recommended places to create contextual menus. It takes as a parameter a MenuItem that serves the role as the ‘menubar’ for a contextual menu. Any menu items that you add to this menu bar are displayed when you return [True](#) from this event.

The ConstructContextMenu event will fire when the time is right to display the contextual menu. This is not restricted to mouse-related events and, for mouse events, it can vary by platform. On Macintosh, it is displayed on a MouseDown event; for Windows and Linux, it is displayed on MouseUp. Also, the user may have pressed the contextual menu key on a Windows keyboard or pressed Shift+F10 instead of clicking the mouse. The ConstructContextMenu event simplifies all this for you by just firing whenever the user has requested the contextual menu in the window or the control.

If you build your contextual menu in the ConstructContextMenu and have no Menu Handlers for the items, you handle the selected menu item with the ContextualMenuAction event. It is passed the selected MenuItem from the contextual

menu and you can inspect its Text or Tag property to determine which item was selected.

All of this eliminates the need for the **ContextMenu** control and the [IsContextualClick](#) function.

See the [RectControl](#) entry for an example of these two event handlers.

## Using the Contextual-Menu Control

Contextual menus are created in the same way [PopupMenu](#) controls are. However, they are no longer needed if you use the ConstructContextMenu and ContextualMenuAction event handlers.

If you have a static set of items for a contextual menu, you can build the menu in the ContextualMenu's Open event handler. If the items in the menu change, you may want to build the contextual menu on the fly—after testing [IsContextualClick](#)—in a MouseDown event handler. To display the contextual menu, call its Open method.

You display a contextual menu by calling the Popup method of the [MenuItem](#) class. This method displays the [MenuItem](#) as a contextual menu. When the user chooses a menu item, it will first try to fire the [MenuItem's](#) Action event handler. If that returns [False](#) or is not implemented, then it will try its menu handlers. If those return [False](#) or are not implemented, it will return the [MenuItem](#) that was selected. If any of these handlers returned [True](#) (indicating that the action was handled), then the Popup method returns [Nil](#).

If these methods fail to handle the menu selection, the ContextualMenuAction event is fired. It gives you a chance to handle the menu selection by inspecting the Text and/or Tag properties of the selected menu item.

There are two choices for implementing contextual menus when you have more than one control in a window that requires a contextual menu. The simplest method is to add one ContextualMenu control for each control that requires a contextual menu. The second method would require you to add a property of type [Control](#) or [RectControl](#) to the window and store a reference to the object that requires the display of a contextual menu. Then a single ContextualMenu control can be used to display and control the contextual menus because it can use the property you have added to determine which object the user is clicking. This second method is a bit more complex but will be more efficient when you have several controls in a window that all require the same contextual menu.

## Example

The following code in a ContextualMenu's Open event handler populates the control:

```
Dim s as String
Dim i,last as Integer
s="Undo,-,Cut,Copy,Paste" //dash adds a separator below Undo
last=CountFields(s, ",")
For i=1 to last
  Me.addRow NthField(s, ",",i)
Next
```

## Continue Statement

---

The following code in a control's MouseDown event handler displays the contextual menu when the user right+clicks in the control (or Control-clicks on Macintosh):

```
If IsContextualClick then  
    ContextualMenu1.open  
    Return True  
End if
```

When the user chooses a menu item, the ContextualMenu's Action event handler runs. The selected menu item is passed to the event handler, so you know which item the user has chosen.

**See Also** [Control](#), [MenuItem](#) classes; [IsContextualClick](#) function.

---

## Continue Statement

Continues execution with the next iteration of a loop.

**Syntax** **Continue**

or

**Continue [For | While | Do]**

or

**Continue For LoopVariable**

| Part         | Type  | Description   |
|--------------|---|---|
| LoopVariable | Datatype of a <a href="#">For</a> loop variable: Integer, Single, or Double | The loop variable that controls iteration of the For statement that you want to continue. |

**Notes**

The Continue statement enables you to jump to the end of a loop and continue execution without executing the lines of code between the Continue statement and the end of the loop. If there is ambiguity concerning which loop you mean, you can use the second or third syntaxes to clarify the situation. For example, if you have nested [For](#) loops and want to jump from a line in the innermost loop to the outermost loop, you can use the last syntax to identify the loop variable that controls the outermost loop.

**See Also** [Do](#), [Exit](#), [For](#), [While](#) statements.

# Control Class

Control classes include the [NotePlayer](#) and [ContextMenu](#) controls. The [RecrControl](#) class is subclassed from the Control class.

**Super Class** [Object](#)

## Properties

| Name               | Type                    | Description  |
|--------------------|-------------------------|--|
| <b>Index</b>       | <a href="#">Integer</a> | If the control is used in a control array, the control's index in the array.   |
| <b>Handle</b>      | <a href="#">Integer</a> | Returns a handle to the control.   |
| <b>MouseX</b>      | <a href="#">Integer</a> | The X coordinate of the mouse (pixels). Measured from the top-left corner of the window.   |
| <b>MouseY</b>      | <a href="#">Integer</a> | The Y coordinate of the mouse (pixels). Measured from the top-left corner of the window.   |
| <b>Name</b>        | <a href="#">String</a>  | The name of the control.   |
| <b>PanellIndex</b> | <a href="#">Integer</a> | Used to get or set the panel of a <a href="#">TabPanel</a> or <a href="#">PagePanel</a> on which the control has been placed. The first panel is numbered zero. If the control has been placed on a panel of a <a href="#">TabPanel</a> or <a href="#">PagePanel</a> control, it returns the panel number. If the control is not on a <a href="#">PagePanel</a> or <a href="#">TabPanel</a> , it returns -1. If you change the PanellIndex to a nonexistent panel, the control will disappear until you give it a PanellIndex value that corresponds to a panel that exists. |
| <b>Window</b>      | <a href="#">Window</a>  | The control's parent window.   |

## Events

| Name  | Parameters | Description                   |
|-------|------------|-------------------------------|
| Close |            | The window is about to close. |
| Open  |            | The window is about to open.  |

## Methods

| Name  | Parameters | Description   |
|-------|------------|---|
| Close |            | Closes a control. Closing a control permanently removes the control from memory, making it impossible to access. You can close both non-indexed controls and indexed controls. When you close an indexed control, the indexes for the remaining controls will shift downward so that the indexes start with zero and are consecutive. |

## ConvertEncoding Function

---

### Notes

**Changing the Cursor** The MouseCursor property controls which cursor will be displayed while the mouse is within the control, assuming that the MouseCursor properties of the [Application](#) and parent [Window](#) classes are [Nil](#). You can, for example, assign different MouseCursors to different controls within a window and the shape of the pointer will change whenever it is over a particular control. The Cursors module contains a library of custom cursors that you access with the syntax [System.Cursors.MouseCursorName](#). Please see the [MouseCursor](#) class for information about the library of custom cursors.

If you also need custom cursors at the Window or Application levels, you need to manage them along the lines described in the section on the [MouseCursor](#) class.

### Custom Cursors in Windows

If you need to use custom cursors in the Windows build of your application, you can do so using a slightly modified version of this technique.

1. With a resource editor, create a CURS resource that contains only one cursor. Give the resource file a name that indicates the type of cursor contained in the resource file.
2. Repeat this process for each custom cursor.
3. Add all the resource files to the Project Editor.
4. You can then access each custom cursor by name, e.g.,

```
Me.MouseCursor=Pen
```

**See Also** [NotePlayer](#) control; [Object](#), [RectControl](#), classes.

---

## ConvertEncoding Function

Provides a quick way to convert a string of known encoding to some other encoding, without having to create a [TextConverter](#) object.

### Syntax

**result=ConvertEncoding(Str, NewEncoding)**

OR

**result=Str.ConvertEncoding(NewEncoding)**

| Part        | Type                         | Description   |
|-------------|------------------------------|---|
| result      | <a href="#">String</a>       | The converted string using the <i>NewEncoding</i> TextEncoding. |
| str         | <a href="#">String</a>       | The string to be converted.                                     |
| newEncoding | <a href="#">TextEncoding</a> | The encoding to be used in the conversion.                      |

**Notes**

When you need to write text to a file that will be opened by another application that expects a particular encoding, use **ConvertEncoding** to convert the text to that encoding before you call the Write method. Here is an example that converts the text in an [EditField](#) to the MacRoman encoding.

```
Dim file As FolderItem
Dim fileStream As TextOutputStream
file=GetSaveFolderItem("plain/text", "My Info")
fileStream=file.CreateTextFile
fileStream.Write ConvertEncoding(namefield.Text,Encodings.MacRoman)
fileStream.Close
```

**Example**

The following example use the [Encodings](#) object to convert the text in an [EditField](#) to the ANSI encoding.

```
Dim s as String
s=ConvertEncoding(Editfield1.text,Encodings.WindowsANSI)
```

**See Also**

[TextConverter](#), [TextEncoding](#), [TextOutputStream](#) classes; [DefineEncoding](#), [Encoding](#) functions; [Encodings](#) object.

## Cos Function

Returns the cosine of the given angle.

**Syntax**

**result=Cos (value)**

| Part   | Type                   | Description                                    |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The cosine of value.                           |
| value  | <a href="#">Double</a> | The value (in radians) you want the cosine of. |

**Notes**

The **Cos** function returns the cosine of the angle (in radians) passed to it. If the angle is in degrees, multiply it by PI/180 to convert it to radians.

**Examples**

This example uses the **Cos** function to return the cosine of an angle.

```
Dim d as Double
Const PI=3.14159
d=Cos(45*PI/180) //returns 0.707
```

**See Also**

[Acos](#) function.

## CountFields Function

Returns the number of values (fields) in the string passed that are separated by the separator string passed. If the source string is binary data or you require case-sensitivity, use [CountFieldsB](#) instead.

### Syntax

**result=CountFields(source, separator)**

| Part      | Type                    | Description   |
|-----------|-------------------------|---|
| result    | <a href="#">Integer</a> | The number of values in source that are separated by separator. |
| source    | <a href="#">String</a>  | The original string.  |
| separator | <a href="#">String</a>  | The character or characters that separate the values in source. |

### Notes

The **CountFields** function is useful for reading columns of data from a text file where the columns (fields) are delimited with a specific character or characters.

If the separator is not found within source, **CountFields** returns 1. If *source* is null, **CountFields** returns zero.

### Examples

The example below returns 5.

```
Dim count as Integer  
Dim s as String  
s="Dan*Smith*11/22/69*5125554323*Male"  
count=CountFields(s, "*")
```

The following example returns three because it counts the null “field” after the (unnecessary) final field delimiter.

```
Dim count as Integer  
Dim s as String  
s="Dan*Smith*"  
count=CountFields(s, "*")
```

See also the example that illustrates how to populate a [PopupMenu](#) control.

### See Also

[CountFieldsB](#), [NthField](#), [Split](#) functions; [TextInputStream](#) object example.

---

## CountFieldsB Function

Returns the number of values (fields) in the string passed that are separated by the separator string passed. **CountFieldsB** is identical to [CountFields](#), except that it treats

the source string as binary data. Use this instead of CountFields if you need the function to be case-sensitive.

**Syntax**

**result=CountFieldsB(source, separator)**

| Part      | Type                    | Description  |
|-----------|-------------------------|--|
| result    | <a href="#">Integer</a> | The number of values in source that are separated by separator.                                  |
| source    | <a href="#">String</a>  | The original string.   |
| separator | <a href="#">String</a>  | The character or characters that separates the values in source. Separator can be of any length. |

**Notes**

The **CountFieldsB** function is useful for reading columns of data from a text file where the columns (fields) are delimited with a specific character or characters.

If the separator is not found within source, **CountFieldsB** returns 1. If source is null, **CountFieldsB** returns zero.

**Examples**

The example below returns 5.

```
Dim count as Integer
Dim s as String
s="Dan*Smith*11/22/69*5125554323*Male"
count=CountFieldsB(s, "*")
```

The following example returns three because it counts the null “field” after the (unnecessary) final field delimiter.

```
Dim count as Integer
Dim s as String
s="Dan*Smith*"
count=CountFieldsB(s, "*")
```

See also the example that illustrates how to populate a [PopupMenu](#) control.

**See Also**

[CountFields](#), [NthFieldB](#), [SplitB](#) functions; [TextInputStream](#) object example.

## CriticalSection Class

Used to protect a resource in a multithreaded environment.

**Super Class** [Object](#)

## Methods

| Name     | Parameters | Description  |
|----------|------------|--|
| Enter    |            | Attempts to get a lock on the resource managed by the CriticalSection. When the call to Enter succeeds, the function returns and your code has exclusive access to the protected resource. If the lock cannot be obtained immediately, Enter will block the current thread from continuing to run. It will wait for the resource to become available.<br>Enter differs from the Signal method of the <a href="#">Semaphore</a> class in that it can be called multiple times from the currently executing thread. If the calling thread already owns the lock, the method returns immediately. This makes CriticalSections very useful for calling a method recursively. |
| Leave    |            | Call Leave when you are finished using the protected resource and give it back to the CriticalSection. Every time you call Enter or TryEnter and succeed, you must call Leave. This includes calling it recursively Otherwise the resource will be protected indefinitely.   |
| TryEnter |            | Similar to Enter but returns a <a href="#">Boolean</a> . Attempts to get a lock on the resource managed by the CriticalSection. If it succeeds, it returns <a href="#">True</a> and the thread has exclusive use of the resource. If it fails, it returns <a href="#">False</a> but does not block execution of the thread.  |

## Notes

A **CriticalSection** is similar to a [Semaphore](#), except a **CriticalSection** protects only one resource. The [Semaphore](#) class, on the other hand, can protect more than one resource. The [Mutex](#) class is similar to a **CriticalSection**, but its scope is all the applications that are running on the user's computer, not just the current REALbasic application. You can use a [Mutex](#), for example, check whether another copy of the application is running and is using a needed resource.

You use critical sections in conjunction with [Threads](#). In a situation in which two or more of the threads might try to access the same item, then you should enclose the code with calls to the Enter (or TryEnter) and Leave methods.

One strategy is this:

- Create a subclass of [Thread](#) which does the operations that might compete for access to a resource,
- Create a **CriticalSection** property of Public scope in, for example, the window that contains the control that calls the threads.
- In the Run event handler of the [Thread](#), call Enter and Leave before and after the code that tries to access the shared resource.

## See Also

[Mutex](#), [Semaphore](#), [Thread](#) classes.

## CStr Function

Use to convert the passed data type to a [String](#). You can also pass a class instance as long as the class implements the [Operator\\_Convert](#) method that returns a [String](#).

See the [Str](#) function for more information.

### Syntax

**result=CStr(Data)**

| Part   | Type   | Description   |
|--------|--|---|
| result | <a href="#">String</a>                           | The string representation of the value passed.              |
| data   | Variable or a class with a string representation | The variable or class that will be represented as a string. |

### Examples

These examples return the string representation of two numbers and a [Boolean](#).

```
Dim s as String
Dim d as New Date
s=CStr(1) //returns "1", as a string
s=CStr(d.Day) //returns the day number as a string
s=CStr(True) //returns "True"
```

### See Also

[CDbl](#), [Str](#), [Val](#) functions.

## CString Data Type

A null-terminated [String](#). You can pass a regular REALbasic [String](#) to it and it will be terminated with a null byte automatically. CString can only be used with the [Declare](#) statement and cannot be used to declare REALbasic variables, properties, or methods.

### See Also

[Declare](#) statement; [Byte](#), [CFStringRef](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

## Cursors Object

Contains a library of standard mouse cursors.

## Cursors Object

---

### Notes

The **Cursors** object contains a library of standard mouse cursors that you can access by calling [System.Cursors.MouseCursorName](#), where MouseCursorName is one of the following:

| MouseCursorName         | Description   |
|-------------------------|---|
| ArrowAllDirections      | This is a set of four arrows pointing in both the North/South and East/West dimensions. It is typically used when moving objects on Windows.      |
| ArrowEastWest           | A pair of arrows pointing in the East and West directions. It is typically used when resizing something horizontally.                             |
| ArrowNortheastSouthwest | A pair of arrows pointing Northeast and Southwest directions. It is typically used when resizing something in this diagonal direction.            |
| ArrowNorthSouth         | A pair of arrows pointing North and South. It is typically used when resizing something vertically.   |
| ArrowNorthwestSoutheast | A pair of arrows pointing in the Northwest and Southeast directions. It is typically used when resizing something in this diagonal direction.     |
| FingerPointer           | This cursor shows one finger pointing up indicating the presence of a hyperlink.  |
| HandClosed              | This cursor is typically used in conjunction with the HandOpen cursor to indicate that something has been "grabbed" and the drag is taking place. |
| HandOpen                | This is the open hand cursor that is typically used to indicate that something can be "grabbed."  |
| IBeam                   | The text insertion cursor that indicates that text can be entered into the object.  |
| SplitterEastWest        | A pair of arrows pointing East and West with a vertical bar between them. This is typically used when dragging a vertical splitter control.       |
| SplitterNorthSouth      | A pair of arrows pointing North and South with a horizontal bar between them. This is typically used when dragging a horizontal splitter control. |
| StandardPointer         | The standard system "arrow" cursor that is used for normal operations. You can use this in place of the global function ArrowCursor.              |
| Wait                    | The system "wait" cursor that is used when a long operation is in progress and you cannot provide other feedback.                                 |

### Example

The following line in the MouseDown event of a [Canvas](#) changes the cursor to the HandClosed cursor.

```
Me.MouseCursor=System.Cursors.HandClosed
```

### See Also

[MouseCursor](#) class; [System](#) object.

# CurveShape Class

Used for drawing lines and curves in a vector graphics environment.

**Super Class** [Object2D](#)

## Properties

| Name     | Type                    | Description   |
|----------|-------------------------|---|
| ControlX | <a href="#">Double</a>  | Parameter is Index as <a href="#">Integer</a> . A zero-based array of control points (horizontal position)  |
| ControlY | <a href="#">Double</a>  | Parameter is Index as <a href="#">Integer</a> . A zero-based array of control points (vertical position).   |
| Order    | <a href="#">Integer</a> | The number of off-curve control points that are used. It is one of the following values:<br>0—A straight line from x,y to x2,y2 is used.<br>1—A quadratic Bezier curve is drawn using one control point.<br>2—A cubic Bezier curve is drawn using two control points. |
| Segments | <a href="#">Integer</a> | Number of straight line segments to use to approximate a curve. The default value of zero tells REALbasic to choose a 'sensible' number of segments.  |
| X2       | <a href="#">Double</a>  | The horizontal position of the end of the line or curve.  |
| Y2       | <a href="#">Double</a>  | The vertical position of the end of the line or curve.  |

## Notes

For CurveShapes, the default value for the Border property is 100 and the default value for Fill is 0.

When you set Order to zero, you get a straight line from x,y to x2, y2. You use control points when you want to 'bend' the line in a particular way. When you set Order to 1, the line bends towards ControlX(0),ControlY(0) on its way from X,Y to X2,Y2. This is what happens in the example. The further away the control point is, the more the line will deviate in that direction.

If Order is set to 2, there are two control points. The line will bend first towards ControlX(0),ControlY(0), and then towards ControlX(1),ControlY(1). At the end points, the curve points directly at the associated control point. The curve is guaranteed to always stay within the polygon formed by the endpoints and the control points (if any).

## Example

The following method, when placed in the MouseDown event of a [window](#), draws a simple little curve when the user presses the mouse button. The negative value of

## DarkBevelColor Function

---

ControlY(0) places the control point above the imaginary straight line from x,y to x2,y2.

```
Dim c As New CurveShape  
c.controlx(0)=120  
c.controly(0)=-40  
c.order=1  
c.x=10  
c.y=10  
c.x2=250  
c.y2=10  
  
Graphics.DrawObject c,x,y
```

The curve looks like this:



### See Also

[FigureShape](#), [FolderItem](#), [Group2D](#), [Graphics](#), [Object2D](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

---

## DarkBevelColor Function

The currently selected color for drawing dark lines in dividing lines and group boxes.

### Syntax

**result=DarkBevelColor**

| Part   | Type                  | Description  |
|--------|-----------------------|--|
| result | <a href="#">Color</a> | The color used for drawing the dark lines in dividing lines and group boxes. |

### Notes

This value is useful when you are using [Canvas](#) controls to create custom controls. When drawing controls like dividing [Lines](#) and [GroupBoxes](#), use this color for the dark portions of the object (usually the right and bottom sides of the object).

This value can be changed by the user, so you should access this value in the Paint event handler rather than storing the value.

### See Also

[CMY](#), [DarkTingeColor](#), [FillColor](#), [FrameColor](#), [HighlightColor](#), [HSV](#), [LightBevelColor](#), [LightTingeColor](#), [RGB](#), [TextColor](#) functions; [Color](#) data type.

## DarkTingeColor Function

The currently selected color for drawing dark lines inside frames on the bottom and right sides.

### Syntax

**result=DarkTingeColor**

| Part   | Type                  | Description  |
|--------|-----------------------|--|
| result | <a href="#">Color</a> | The color used for drawing the dark lines inside frames on the bottom and right. |

### Notes

This value is useful when you are using [Canvas](#) controls to create custom controls. When drawing beveled objects, use this color for the dark portions of the object (usually the right and bottom sides of the object). In a [CheckBox](#) control, this color is used to draw the dark lines that give it the beveled appearance just inside the checkbox on the right and bottom sides.

This value can be changed by the user, so you should access this value in the Paint event handler rather than storing the value.

### See Also

[DarkBevelColor](#), [FillColor](#), [FrameColor](#), [HighlightColor](#), [LightBevelColor](#), [LightTingeColor](#), [TextColor](#) functions.

---

## DataAvailableProvider Interface Class

Used to program custom object bindings.

### Properties

| Name          | Type                    | Description                               |
|---------------|-------------------------|---|
| dataAvailable | <a href="#">Boolean</a> | <a href="#">True</a> if data is available |

### See Also

[DataNotifier Interface](#) class.

---

## Database Class

A **Database** object represents an open database that can be accessed by REALbasic's "front-end" database commands. You can create or open a REAL SQL Database using the [REALSQLdatabase](#) class. Use the [Database4DServer](#), [ODBCDatabase](#), [OpenBaseDatabase](#), [MySQLDatabase](#), [OracleDatabase](#), or [PostgreSQLDatabase](#) classes to return **Database** objects.

## Database Class

---

Except for the REAL SQL Database, you need to install the appropriate database plug-in in the REALbasic Plugins folder to use the other data sources. Database plug-ins may be updated more frequently than the REALbasic application itself. You can obtain the most current versions of all database plug-ins at [www.realsoftware.com](http://www.realsoftware.com).

FrontBase can also be accessed using a third-party plug-in. The FrontBase plug-in is free and is available on the REALbasic CD.

**Super Class** [Object](#)

### Properties

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| DatabaseName | <a href="#">String</a>  | The database name to open.   |
| Error        | <a href="#">Boolean</a> | <a href="#">True</a> if an error is returned from the database engine.   |
| ErrorCode    | <a href="#">Integer</a> | Error code returned from database engine. Error codes and error messages are different for each engine.                |
| ErrorMessage | <a href="#">String</a>  | Text of error message returned from the database engine. Error codes and error messages are different for each engine. |
| Host         | <a href="#">String</a>  | The database host name or IP address of server to connect to.  |
| Password     | <a href="#">String</a>  | The password required for access to the database.  |
| UserName     | <a href="#">String</a>  | The username required for access to the database   |

### Methods

| Name        | Parameters                          | Description  |
|-------------|-------------------------------------|--|
| Close       |                                     | Closes the database.   |
| Commit      |                                     | Commits (saves) changes to records. If you quit the application after making changes to a <a href="#">RecordSet</a> , REALbasic issues an implicit Commit. Use Commit and Rollback to manage transactions.     |
| Connect     |                                     | Connects to the database server and opens the database for access. Returns a <a href="#">Boolean</a> . Before proceeding with database operations, test to be sure that Connect returns <a href="#">True</a> . |
| FieldSchema | TableName as <a href="#">String</a> | Returns a <a href="#">RecordSet</a> with information about all fields in the table. See notes, below.  |
| GetProperty | Name as <a href="#">String</a>      | Returns the database property specified by Name from the data source. Currently supported only for 4D Server and for getting the connection string. See example.   |

| Name         | Parameters   | Description  |
|--------------|--|--|
| IndexSchema  | TableName as <a href="#">String</a>  | Returns a <a href="#">RecordSet</a> containing the list of indexes for the passed table or <a href="#">Nil</a> if the table has no indexes or the database source does not support indexes. The <a href="#">RecordSet</a> has one field, IndexName, for the names of the indexes, and one row per index. |
| InsertRecord | TableName as <a href="#">String</a> , Data as <a href="#">DatabaseRecord</a> | Inserts Data as the last row of TableName.   |
| Rollback     |  | Cancels a set of changes to records. Use this method or Commit to cancel or save transactions.   |
| SQLExecute   | ExecuteString as <a href="#">String</a>                                      | The SQL Execute statement. <i>ExecuteString</i> is a valid SQL statement (other than Select) that is supported by the data source you are using. Use the SQLSelect method with the SQL Select statement.   |
| SQLSelect    | SelectString as <a href="#">String</a>                                       | The SQL Select statement. <i>SelectString</i> is a valid SQL Select statement. Returns a <a href="#">RecordSet</a> .   |
| TableSchema  |  | Returns a <a href="#">RecordSet</a> with a list of all tables in the database. See notes, below.   |

## Notes

### SQL for the REALdatabase data source

The [REALSQLdatabase](#) data source supports a subset of SQL/92 and SQL/99. This subset supports selecting, inserting, and deleting records, creating, modifying, and deleting tables, and building indexes. For more information, see Appendix A, *REALSQLdatabase SQL Language Reference* in the pdf version of the *Language Reference* or the SQLite web site at <http://www.sqlite.org>.

### Connecting to a data source

Use the following subclasses of the **Database** class to connect to data sources.

| Class                              | Description  |
|------------------------------------|--|
| <a href="#">Database4DServer</a>   | Supports 4D versions 6.8 and above, including 4D 2003 and above.   |
| <a href="#">MySQLDatabase</a>      | Provides support for MySQL databases; this functionality was previously available only via third-party plug-ins. |
| <a href="#">ODBCDatabase</a>       | Supports ODBC-based databases.   |
| <a href="#">OpenBaseDatabase</a>   | Supports OpenBaseDatabase.   |
| <a href="#">OracleDatabase</a>     | Connects to Oracle 8i and above.   |
| <a href="#">PostgreSQLDatabase</a> | Supports PostgreSQL.   |
| <a href="#">REALSQLdatabase</a>    | Supports the built-in REAL SQL Database data source and is built-into REALbasic (no plug-in required).           |

Third-parties may supply plug-ins that work with other data sources.

When establishing a connection to any data source, do a test like this before proceeding:

```
Dim db as Database  
. . .  
If db.Connect() then  
//connection is successful  
else  
//connection failed  
end if
```

In general, you will use the Username, Host, and Password properties of the **Database** class to establish the connection. Specific requirements are different for different data sources.

### FieldSchema and TableSchema

FieldSchema returns a [RecordSet](#) with five fields: ColumnName ([String](#)), FieldType ([Integer](#)), IsPrimary ([Boolean](#)), NotNull ([Boolean](#)), and Length in bytes ([Integer](#)) for Text fields. *FieldType* uses the following values. Also, some field types have implementation-specific features, such as numeric precision and maximum length.

The following table contains information about SQL data storage types.

| FieldType       | Value | Description  |
|-----------------|-------|--|
| Null            | 0     | Denotes the absence of any value, i.e., a missing value.   |
| Byte            | 1     | Stores the byte representation of a character string.  |
| SmallInt        | 2     | A numeric data type with no fractional part. The maximum number of digits is implementation-specific, but is usually less than or equal to INTEGER. The REAL database supports 2-byte smallints, which allow you to store values in the range of $\pm 32,767$ . If you are using another data source, check the documentation of your data source. |
| Integer         | 3     | A numeric data type with no fractional part. The maximum number of digits is implementation-specific. The REAL database supports 4-byte integers, which provide a range of $\pm 2,000,000,000$ . If you are using another data source, check the documentation of your data source.  |
| Char            | 4     | Stores alphabetic data, in which you specify the maximum number of characters for the field, i.e., CHAR (20) for a 20 character field. If a record contains fewer than the maximum number of characters for the field, the remaining characters will be padded with blanks.  |
| Text or VarChar | 5     | Stores alphabetic data, in which the number of characters vary from record to record, but you don't want to pad the unused characters with blanks.<br>For example, "VARCHAR (20)" specifies a VARCHAR field with a maximum length of 20 characters.  |

| FieldType             | Value | Description  |
|-----------------------|-------|--|
| Float                 | 6     | Stores floating-point numeric values with a precision that you specify, i.e., FLOAT (5).   |
| Double                | 7     | Stores double-precision floating-point numbers.  |
| Date                  | 8     | Stores year, month, and day values of a date in the format YYYY-MM-DD. The year value is four digits; the month and day values are two digits.   |
| Time                  | 9     | Stores hour, minute, and second values of a time in the format HH:MM:SS. The hours and minutes are two digits. The seconds values is also two digits, may include a optional fractional part, e.g., 09:55:25.248. The default length of the fractional part is zero.   |
| TimeStamp             | 10    | Stores both date and time information in the format YYYY-MM-DD HH:MM:SS. The lengths of the components of a TimeStamp are the same as for Time and Date, except that the default length of the fractional part of the time component is six digits rather than zero. If a TimeStamp values has no fractional component, then its length is 19 digits If it has a fractional component, its length is 20 digits, plus the length of the fractional component. |
| Currency              | 11    | Stores a decimal amount with a Dollar sign.  |
| Boolean               | 12    | Stores the values of TRUE or FALSE.  |
| Decimal               | 13    | Stores a numeric value that can have both an integral and fractional part. You specify the total number of digits and the number of digits to the right of the decimal place, i.e., DECIMAL (5,2) specifies a decimal field that can contain values up to 999.99. DECIMAL (5) specifies a field that can contain values up to 99,999.  |
| Binary                | 14    | Stores code, images, and hexadecimal data. Consult the documentation of your data source for information on the maximum size of a Binary field.  |
| Long Text (Blob)      | 15    | Stores a text object. Consult the documentation of your data source for information on the maximum size of a Blob.   |
| Long VarBinary (Blob) | 16    | Stores a binary object. The REAL SQL Database supports blobs of up to any size. Furthermore, a blob can be stored in a column of any declared data affinity. If you are using another data source, check the documentation of your data source.  |
| MacPICT               | 17    | Stores a Macintosh PICT image.   |
| String                | 18    | Text up to $2^{31}$ bytes. The same as VarChar.  |
| Unknown               | 255   | Unrecognized data type.  |

The values of IsPrimary and NotNull are set using the SQL Select statement that created the table.

TableSchema returns a [RecordSet](#) with one field, TableName ([String](#)).

## Database Class

---

| <b>The Property Method</b>              | The purpose of the Property method is to retrieve miscellaneous information from the data source. For example, if you are using 4th Dimension as the data source, you can pass "connectionString" to the Property method and it will return the connection string for the 4D Server database. It contains the name of the database. ConnectionString is supported by 4th Dimension but may not be supported by other data sources.  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
|---|---|----------|------------|--------------|--|--------------|--|-------------------|---|------------|--|-----|---|-----|--|
| <b>SQLSelect and SQLExecute Methods</b> | You use the SQLSelect and SQLExecute methods to communicate with your data source via SQL commands. You use SQLSelect to call the SELECT statement and the SQLExecute statement for all other statements. See Appendix A for detailed information on SQL supported by the REAL SQL Database or the SQLite web site at <a href="http://www.sqlite.org">http://www.sqlite.org</a> .   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| <b>SQL Aggregate Functions</b>          | The following aggregate functions are supported. Each calculates a value from a group of rows that is chosen by the WHERE clause in the SELECT statement in which the Aggregate function is used:   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
|   | <table border="1"><thead><tr><th>Function</th><th>Definition</th></tr></thead><tbody><tr><td>AVG</td><td>Returns the arithmetic average of a numeric expression.</td></tr><tr><td>COUNT</td><td>Returns the total number of rows accessed by the expression.</td></tr><tr><td>COUNT (*)</td><td>Returns the total number of rows accessed by the expression. No list of fields is passed and Null and duplicate values are included in the count.</td></tr><tr><td>MAX</td><td>Returns the maximum value of the specified field or fields.</td></tr><tr><td>MIN</td><td>Returns the minimum value of the specified field or fields.</td></tr><tr><td>SUM</td><td>Returns the total value in the specified numeric fields.</td></tr></tbody></table> | Function | Definition | AVG          | Returns the arithmetic average of a numeric expression.  | COUNT        | Returns the total number of rows accessed by the expression. | COUNT (*)         | Returns the total number of rows accessed by the expression. No list of fields is passed and Null and duplicate values are included in the count. | MAX        | Returns the maximum value of the specified field or fields.  | MIN | Returns the minimum value of the specified field or fields. | SUM | Returns the total value in the specified numeric fields. |
| Function                                | Definition  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| AVG                                     | Returns the arithmetic average of a numeric expression.   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| COUNT                                   | Returns the total number of rows accessed by the expression.  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| COUNT (*)                               | Returns the total number of rows accessed by the expression. No list of fields is passed and Null and duplicate values are included in the count.   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| MAX                                     | Returns the maximum value of the specified field or fields.   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| MIN                                     | Returns the minimum value of the specified field or fields.   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| SUM                                     | Returns the total value in the specified numeric fields.  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| <b>Date/Time and Misc. Functions</b>    | The following date and time functions are supported by the REALdatabase data source.  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
|   | <table border="1"><thead><tr><th>Function</th><th>Definition</th></tr></thead><tbody><tr><td>CURRENT_DATE</td><td>Returns the current date in SQL Date format, YYYY-MM-DD.</td></tr><tr><td>CURRENT_TIME</td><td>Returns the current time in SQL Time, HH:MM:SS.</td></tr><tr><td>CURRENT_TIMESTAMP</td><td>Returns the current date-time in SQLTimeStamp format, YYYY-MM-DD HH:MM:SS, as a SQLTimeStamp data type.</td></tr><tr><td>LAST_ROWID</td><td>Returns the value of the last _rowID for the table passed. This value is needed for certain types of relational joins.</td></tr></tbody></table>  | Function | Definition | CURRENT_DATE | Returns the current date in SQL Date format, YYYY-MM-DD. | CURRENT_TIME | Returns the current time in SQL Time, HH:MM:SS.              | CURRENT_TIMESTAMP | Returns the current date-time in SQLTimeStamp format, YYYY-MM-DD HH:MM:SS, as a SQLTimeStamp data type.   | LAST_ROWID | Returns the value of the last _rowID for the table passed. This value is needed for certain types of relational joins. |     |   |     |  |
| Function                                | Definition  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| CURRENT_DATE                            | Returns the current date in SQL Date format, YYYY-MM-DD.  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| CURRENT_TIME                            | Returns the current time in SQL Time, HH:MM:SS.   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| CURRENT_TIMESTAMP                       | Returns the current date-time in SQLTimeStamp format, YYYY-MM-DD HH:MM:SS, as a SQLTimeStamp data type.   |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |
| LAST_ROWID                              | Returns the value of the last _rowID for the table passed. This value is needed for certain types of relational joins.  |          |            |              |  |              |  |                   |   |            |  |     |   |     |  |

## Examples

### Creating a REAL database

The following code creates an internal database and uses SQLExecute to create a table.

```
Dim db as REALSQLdatabase
Dim f as FolderItem
Dim result as Boolean
f=New FolderItem("mydb")
db=New REALSQLdatabase
db.databaseFile=f
result=db.CreateDatabaseFile
If db.Connect() then
    db.SQLExecute("create table invoices(id integer ,Cust_ID integer,Amount
double, Date date")
    db.Commit
else
    MsgBox "Database not created"
end if
```

The following example inserts a row in this table:

```
dim r as DatabaseRecord
dim mybool as Boolean
dim mydate as Date

.
.

r=New DatabaseRecord
r.IntegerColumn("id")=4
r.IntegerColumn("Cust_ID")=2
r.DoubleColumn("Amount") =9.98
mybool=ParseDate("10/22/98",mydate)
r.DateColumn("Date") = mydate
db.InsertRecord("invoices",r)
```

### RecordSets

The following example creates a [RecordSet](#) that contains all rows and columns for a particular customer sorted by amount:

```
dim rs as RecordSet
.
.
.
rs = db.SQLSelect("select * from invoices where cust_id=02 ORDER BY
amount DESC")
```

The following Select statement retrieves all rows and columns from the Customers table in which the customer's last name begins with the letter 'L.'

```
rs=db.SQLSelect("select * from Customers WHERE customer.name LIKE 'L%'"")
```

## Database4DServer Class

---

This Select statement retrieves all rows and columns in which the customer's last name does not begin with the letter 'L.'

```
rs=db.SQLSelect("select * from customer WHERE customer.name NOT LIKE  
'L%'")
```

The following example returns a [RecordSet](#) that contains information on the fields in the customers table:

```
Dim dbFile as FolderItem  
Dim db as REALSQLdatabase  
Dim rs as RecordSet  
rs=New RecordSet  
db=New RecordSet  
dbFile = GetFolderItem("Pubs")  
db.DatabaseFile=dbFile  
If db.Connect() then  
    rs=db.FieldSchema("Customers")  
else  
    Beep  
    MsgBox db.ErrorMessage  
end if
```

### See Also

[Database4DServer](#), [DatabaseField](#), [DatabaseRecord](#), [MySQLDatabase](#), [ODBCDatabase](#), [OpenBaseDatabase](#), [OracleDatabase](#), [PostgreSQLDatabase](#), [REALSQLdatabase](#), [RecordSet](#) classes; [OpenCSVCursor](#), [OpenDBFCursor](#), [OpenDTFDatabase](#) functions.

---

## Database4DServer Class

Use to open a 4th Dimension/4D Server data source.

**Database4DServer** supports 4D Server version 6.8 and above and 4D 2003 and above.

**Super Class** [Database](#)

### Properties

| Name | Type                   | Description  |
|------|------------------------|--|
| Task | <a href="#">String</a> | Description of the connection. The <i>task</i> parameter is the name of the process that is created on 4D Server when you establish your connection. The value of task is displayed in the 4D Server window. |

### Notes

In order to use this class, you must install the 4D database plug-in in your plugins folder, which is available at the REALbasic web site. There are separate versions of the

plug-in for Windows and Macintosh. You can cross-compile from the Macintosh but not from Windows.

The 4D plug-in is not supported on Windows 98/ME; it may cause the REALbasic IDE to crash on launch.

4DOpen.dll is required for built Windows applications. Copy the 4DOpen.DLL file (provided with the Windows version of the plug-in) into the same directory as your application. This will ensure that the application can find this library and that the proper version of the library is used.

The set of database plug-ins is included on the REALbasic CD, but you may find more recent versions at the REAL Software web site, <http://www.realsoftware.com>.

When the 4D plug-in is installed, the Add Data Source menu item in the File menu of the IDE contains an submenu item for 4D Server. When you choose this menu item, a dialog box is presented, enabling you to choose a 4D Server on the network.

To access 4D Server, it must be configured to allow 4D Open connections and the user **username** must be in the group that has 4D Open access.

**Example** The following code makes a connection to a 4D Server via TCP/IP.

```
Dim db as Database4DServer
db = New Database4DServer
db.host = "127.0.0.1"
db.username = "Jimmy"
db.password = "motion"
db.task="DataEntry"

If db.connect then
//proceed with database operations.
else
//error connecting
end if
```

**See Also** [Database](#), [RecordSet](#) classes.

## DatabaseField Class

Used to access the values of fields in a record in a database table.

**Super Class** [Object](#)

## Properties

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| BooleanValue | <a href="#">Boolean</a> | Used to get and set the values of <a href="#">Boolean</a> field types.   |
| DateValue    | <a href="#">Date</a>    | Used to get and set the values of <a href="#">Date</a> field types.  |
| DoubleValue  | <a href="#">Double</a>  | Used to get and set the values of <a href="#">Double</a> field types.  |
| IntegerValue | <a href="#">Integer</a> | Used to get and set the values of <a href="#">Integer</a> field types.   |
| JPEGValue    | <a href="#">Picture</a> | Used to get and set the values of <a href="#">Picture</a> field types.   |
| MacPICTValue | <a href="#">Picture</a> | Used to get and set the values of <a href="#">Picture</a> field types.   |
| Name         | <a href="#">String</a>  | Used to get the name of the column.  |
| StringValue  | <a href="#">String</a>  | Used to get and set the values of <a href="#">String</a> /Character field types.   |
| Value        | <a href="#">Variant</a> | Used to get and set the value of a field of any data type. The properties that get and set the values of specific data types are recommended over Value. Set Value to <a href="#">Nil</a> to set the field to NULL. The NULL value is currently supported in the MySQL, OpenBase, and PostgreSQL plug-ins. |

## Notes

Assignments to the DateValue property store the time as well as the date, as with [DatabaseRecords](#). When getting a value, the resulting [Date](#) object may contain a time as well as a date. For fields of type [Date](#), the time will be 00:00:00 (midnight); and for fields of type Time, the date will be January 1, 0001. Fields of typeTimeStamp contain valid data for both the date and time.

The conversion operator, [Operator\\_Convert](#), has been added to StringValue, BooleanValue, DateValue, IntegerValue, and DoubleValue.

## Examples

The following example uses the Value property to set the values of fields of different data types.

```
Dim MyDB as Database
Dim rs as RecordSet
rs = MyDB.SQLSelect("select * from mytable")
rs.Edit
rs.field("MyName").value = Nil // NULL this field,
//does not work for all database plugins
rs.field("MyID").value = 23
rs.field("MyText").value = "Test"
rs.update
MyDB.Commit
```

The following method populates a [ListBox](#) with a [RecordSet](#). It uses the Name and StringValue properties to obtain the fieldnames and values:

```
Sub populateCursor(d as RecordSet)
    dim i as Integer
    dim v as String

    ListBox1.deleteAllRows
    ListBox1.columnCount = d.fieldCount
    ListBox1.addRow d.IdxField(1).Name
    ListBox1.cellBold(ListBox1.lastIndex, 0) = true
    for i = 2 to d.fieldCount
        ListBox1.cell(ListBox1.lastIndex,i-1) =d.IdxField(i).name
        ListBox1.cellBold(ListBox1.lastIndex, i-1) = true
    next
    While not d.eof
        v = d.IdxField(1).StringValue
        ListBox1.addRow v
        For i = 2 to d.fieldCount
            ListBox1.cell(ListBox1.lastIndex,i-1)=d.IdxField(i).StringValue
        next
        d.MoveNext
    wend
```

**See Also** [Database](#), [DatabaseRecord](#), [RecordSet](#) classes.

## DatabaseQuery Control

The control by which you query databases. You ordinarily use a **DatabaseQuery** tool by binding it to the [ListBox](#) that is used to display the results of SQL queries.

Alternatively, you can build databases without the **DatabaseQuery** control using only the commands in the Database theme.

Because its super class is [Object](#), you can instantiate it with code, eliminating the need to actually add the control to a window.

**Super Class** [Object](#)

### Properties

| Name     | Type                     | Description   |
|----------|--------------------------|---|
| Database | <a href="#">Database</a> | The database that will be queried.                    |
| SQLQuery | <a href="#">String</a>   | The text of the SQL query to be run against Database. |

## DatabaseRecord Class

---

### Events

| Name          | Parameters | Description                      |
|---------------|------------|----------------------------------|
| QueryComplete |            | Runs when the query is finished. |

### Methods

| Name     | Parameters | Description   |
|----------|------------|---|
| RunQuery |            | Executes the query defined by the <i>SQLQuery</i> property. |

**Examples** See the examples in the chapter “Creating Databases with REALbasic” in the User’s Guide.

**See Also** [Database](#), [Database4DServer](#), [DatabaseField](#), [DatabaseRecord](#), [MySQLDatabase](#), [ODBCDatabase](#), [OpenBaseDatabase](#), [OracleDatabase](#), [PostgreSQLDatabase](#), [REALSQLdatabase](#), [RecordSet](#) classes; [OpenCSVCursor](#), [OpenDBFCursor](#), [OpenDTFDatabase](#) functions; [DataControl](#) control.

---

## DatabaseRecord Class

Used to create new [Database](#) records. The methods are used to populate the fields in a record. Call it once per column in the record.

**Super Class** [Object](#)

### Methods

| Name          | Parameters                     | Description   |
|---------------|--------------------------------|---|
| BlobColumn    | Name as <a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified blob.   |
| BooleanColumn | Name as <a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified boolean value. The <a href="#">REALSQLdatabase</a> inserts 0 for <a href="#">False</a> and 1 for <a href="#">True</a> . |
| Column        | Name as <a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified string value.   |
| DateColumn    | Name as <a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified date value.   |
| DoubleColumn  | Name as <a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified double value.   |
| IntegerColumn | Name as <a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified integer value.  |

| Name          | Parameters                        | Description   |
|---------------|-----------------------------------|---|
| JPEGColumn    | Name as<br><a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified JPEG image.           |
| MacPictColumn | Name as<br><a href="#">String</a> | Parameter is the column name. Sets the specified field to the specified Macintosh PICT image. |

**Notes**

Assignments to the DateColumn property now store the time as well as the date, to support the SQLTimeStamp and Time field types (as well as Date). Note that if the date part is January 1, 0001, then this is converted to SQL as only a time (e.g., "18:54:00"), whereas if the date is anything else, it converts to SQL in full form (e.g., "2000-5-30 18:54:00").

**Example**

The following example creates a new [REALSQLdatabase](#), an employees table, and adds a record to the table.

```

Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
Dim mydate as New Date
Dim rec as DatabaseRecord
dbFile = GetFolderItem\("Pubs"\)
db.DatabaseFile = dbFile
If db.CreateDatabaseFile Then
    db.SQLExecute("create table employees(id integer, name varchar, "
        +"jobtitle varchar, DOB date, Salary double")
rec = New DatabaseRecord

rec.IntegerColumn("id") =1
rec.Column("name") = "Lois Lane"
rec.Column("jobtitle")="Pundit"
mybool=ParseDate\("1Jan1950",mydate)
rec.DateColumn("DOB")=mydate
rec.DoubleColumn("Salary")=105000
db.InsertRecord("employees",rec)
db.Commit
else
    MsgBox "Error: "+db.ErrorMessage
End if

```

**See Also**

[Database](#), [Database4DServer](#), [DatabaseField](#), [ODBCDatabase](#), [OpenBaseDatabase](#), [OracleDatabase](#), [PostgreSQLDatabase](#), [REALSQLdatabase](#), [RecordSet](#) classes; [OpenCSVCursor](#), [OpenDBFCursor](#), [OpenDTFDatabase](#) functions.

# DataControl Control

Allows the user to browse among records in a database table by clicking buttons. It consists of First Record, Previous Record, Next Record, and Last Record navigation buttons and a label that displays the Caption property. Built-in events fire when the user clicks any button. Methods allow you to navigate among rows programmatically.

A **DataControl** is designed to display data using [CheckBoxes](#), [EditFields](#), [ListBoxes](#), [StaticTexts](#), [PopupMenus](#), and [ComboBoxes](#). You can also create custom classes that are based on any of these classes and use them in conjunction with a **DataControl**.

Each of these controls (or their subclasses) has two properties that are specific to the **DataControl**: *DataSource* and *DataField*. *DataSource* accepts a **DataControl** object and *DataField* accepts the name of a field in the table to which the **DataControl** is linked. You set up a database interface by linking a **DataControl** to a table in a database and then linking each field whose values you wish to display to a [CheckBox](#), [EditField](#), [ListBox](#), [StaticText](#), [PopupMenu](#) or [ComboBox](#) using the *DataField* and *DataSource* properties.

**SuperClass** [RectControl](#)

## Properties

| Property     | Type                         | Description  |
|--------------|------------------------------|--|
| Caption      | <a href="#">String</a>       | The text shown between the two sets of navigation buttons.   |
| Commit       | <a href="#">Boolean</a>      | If <a href="#">True</a> , automatically commits inserts, updates, and deletions to the database.   |
| Database     | <a href="#">Database</a>     | The database to which the control is linked.   |
| Encoding     | <a href="#">TextEncoding</a> | When the DataControl retrieves a string value, it will set the Encoding of the string and when you update or insert a string value using the DataControl. REALbasic will convert that encoding to this encoding. The default Encoding is UTF8. |
| ReadOnly     | <a href="#">Boolean</a>      | Set to <a href="#">True</a> to prevent the user from adding, modifying, or deleting records. When <a href="#">True</a> , the record navigation controls are disabled.  |
| RecordLocked | <a href="#">Boolean</a>      | A value of <a href="#">True</a> means that the record is locked, preventing changes to the record; <a href="#">False</a> indicates that it is unlocked.  |
| RecordSet    | <a href="#">RecordSet</a>    | The <a href="#">RecordSet</a> managed by the DataControl.  |
| SQLQuery     | <a href="#">String</a>       | SQL query used to obtain the default <a href="#">RecordSet</a> .   |
| TableName    | <a href="#">String</a>       | Name of the table in Database whose records will be displayed by the controls linked to the DataControl.   |

## Events

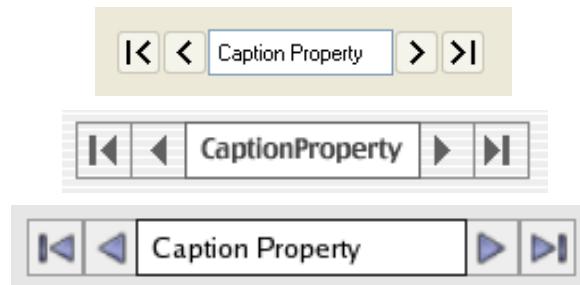
| Name         | Parameters                        | Description  |
|--------------|-----------------------------------|--|
| Delete       |                                   | Occurs after the record is validated for deletion. The record is already deleted at this point.  |
| Insert       |                                   | Occurs after the new record is validated but before data aware fields are cleared.   |
| MoveFirst    |                                   | Occurs when the First Record button has been pressed but before moving to the first record. Returning <a href="#">True</a> prevents the record pointer from moving to the first record.  |
| MoveLast     |                                   | Occurs when the Last Record button has been pressed but before moving to the last record. Returning <a href="#">True</a> prevents the record pointer from moving to the last record.   |
| MoveNext     |                                   | Occurs when the Next Button has been pressed but before moving to the next record. Returning <a href="#">True</a> prevents the record pointer from moving to the next record.  |
| MovePrevious |                                   | Occurs when the Previous Button has been pressed but before moving to the previous record. Returning <a href="#">True</a> prevents the record pointer from moving to the previous record.  |
| Reposition   | Index as <a href="#">Integer</a>  | Occurs after initialization and when the record pointer has moved to another record. The Index parameter indicates the row number after the reposition.  |
| Update       |                                   | Occurs after the record is validated for updating. The record is already updated at this point.  |
| Validate     | Action as <a href="#">Integer</a> | Occurs when you have modified the current record via data aware <a href="#">EditFields</a> . Returning <a href="#">False</a> (the default) updates the record with your changes. Returning <a href="#">True</a> prevents the update from happening. Action specifies what kind of action is being performed:<br>0=AddNew<br>1=Update<br>2=Delete |

## Methods

| Method       | Parameters                       | Description  |
|--------------|----------------------------------|--|
| Delete       |                                  | Deletes the current record from the selection, leaving the record pointer at the same row, if possible. When deleting the last record, the record pointer will move to the previous record.<br>Delete triggers the following events:<br>Validate<br>Reposition<br>Delete                                     |
| FieldCount   |                                  | Returns the number of fields in TableName.   |
| Insert       |                                  | By default, adds a new record to the table, closes the <a href="#">RecordSet</a> , and clears any data aware fields. This behavior can be overridden in the Insert event. Insert triggers the following events:<br>Validate (and closes <a href="#">RecordSet</a> )<br>Insert (and clears data-aware fields) |
| MoveFirst    |                                  | Moves to the first record in the <a href="#">RecordSet</a> .   |
| MoveLast     |                                  | Moves to the last record in the <a href="#">RecordSet</a> .  |
| MoveNext     |                                  | Moves to the next record in the <a href="#">RecordSet</a> .  |
| MovePrevious |                                  | Moves to the previous record in the <a href="#">RecordSet</a> .  |
| MoveTo       | Index as <a href="#">Integer</a> | Moves to the Index row in the <a href="#">RecordSet</a> .  |
| NewRecord    |                                  | Inserts a new record in the table specified by <i>TableName</i> and navigates to it.   |
| RecordCount  |                                  | Returns the number of records in <i>TableName</i> .  |
| RecordSet    |                                  | Returns a <a href="#">RecordSet</a> .  |
| Row          |                                  | Returns the current row in the <a href="#">RecordSet</a> .   |
| RowCount     |                                  | Returns the number of rows in TableName.<br>Obsolete; use RecordCount instead.   |
| RunQuery     |                                  | Repopulates the <a href="#">RecordSet</a> by running SQLQuery.   |
| Update       |                                  | Updates the current record in the selection, leaving the <a href="#">RecordSet</a> at the same row. Update triggers the following events:<br>Validate<br>Update  |

**Notes**

When you add a DataControl to a window, it looks like this on Windows, Macintosh, and Linux:



You use a DataControl by creating a window that uses [Checkboxes](#), [EditFields](#), [ListBoxes](#), [PopupMenu](#)s, [ComboBoxes](#), or [StaticText](#) controls to display and edit data. Each of these controls has two properties that are meaningful only when used in conjunction with a **DataControl**, DataField and DataSource.

You can use Object Binding to bind objects such as a [PushButton](#) to a **DataControl**. Supported actions are: Add, Insert, Update, and Delete Record when the PushButton is pushed. To create an object bind, select the control and the **DataControl** and then choose Project ▶ Add ▶ Binding (If you have added the Add Binding button to the Window Editor toolbar, you can click that button to display the dialog). A dialog box that lists all possible bindings between the selected control and the **DataControl** will be listed. Choose the desired type of binding.

To remove a binding, choose View ▶ List Bindings. A dialog that lists all the current bindings in the window appears. Highlight the binding you wish to remove and then click the Delete button. (If you have added the List Bindings button to the Window Editor toolbar, you can click that button to display the dialog.)

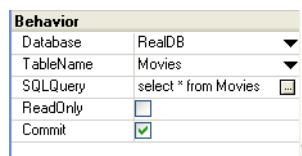
When you build an interface using a **DataControl**, it makes it easy to modify and save records. When the user saves changes, it automatically updates the values in the database back end with the values in the controls bound to the DataControl. If you need to update other values, you need to do it conventionally using the RecordSet method of the DataControl.

## DataControl Control

**Example** The following example database displays fields in [EditFields](#) and uses a **DataControl** for record navigation.

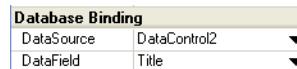


The DataControl's properties in the Behavior group in its Properties Window specify that it will manage the records in the Movies table in the RealDB database.



The SQLQuery property is the SQL query that will run when the form is opened. This query finds all the records in the Movies table. This means that the DataControl's navigation buttons can be used to browse through all the records right after the form opens.

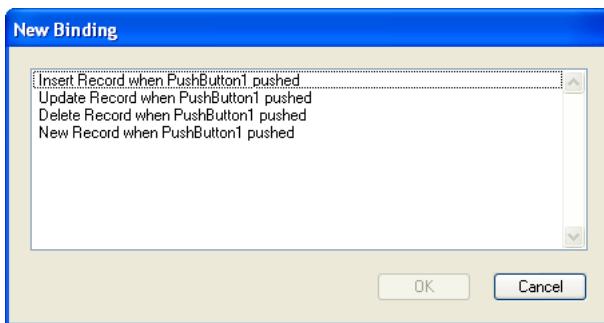
Each [EditField](#) is linked to the **DataControl** in its Properties Window by setting the DataSource and DataField properties. When you set the DataSource, a pop-up menu of fields from the table specified by Database becomes available for the DataField property. For example, the [EditField](#) for the movie title is bound to the column "Title" in the table to which the **DataControl** is linked.



Each field on the form is bound to a field in the Movies table in this manner.

The Add button is linked to the **DataControl** using object binding. To establish the bind, select both the [PushButton](#) and the **DataControl** and choose Project ▶ Add ▶ Binding to display the New Binding dialog box. Select "New Record

when addButton pushed". (If you have added the Add Binding button to the Window Editor toolbar, you can click that instead of choosing the menu command.)



Use the same process to bind the Delete button to the **DataControl** and use "Delete Record when DeleteButton pushed." Bind the Insert button to the "Insert record when InsertButton pushed" and bind the Update button to the "Update record when UpdateButton pushed.

To add a new record, click the Add button, enter the information into the fields, and click Insert. To modify a record, navigate to it using the buttons in the DataControl, edit the values, and click Update. To delete a record, navigate to it using the buttons in the DataControl and click Delete.

**See Also** [Database](#), [DatabaseField](#), [DatabaseRecord](#), [RecordSet](#) classes.

## Datagram Class

Used to send and receive data using the [UDPSocket](#) class.

**Super Class** [Object](#)

### Properties

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Address | <a href="#">String</a>  | The address of the remote machine that you are sending data to, or receiving data from.   |
| Data    | <a href="#">String</a>  | The data you are sending or receiving.  |
| Port    | <a href="#">Integer</a> | The port that the packet will be sent to when calling the Write method. If the Port property is set or left to zero, then the port that it is locally bound to is used.<br>When reading packets, the Port property specifies the port that the Datagram was sent from on the remote computer. |

## DataNotificationReceiver Interface Class

---

### Notes

A **Datagram** consists of two parts, the IP address of the remote machine which sent you the data, and the data itself. When you attempt to send data, you must specify information in the form of a **Datagram**. This information is usually the remote address of the machine you want to receive your packet, the port it should be sent to, and the data you wish to send the remote machine. Please see the [UDPSocket](#) for information on how to use the **Datagram**.

### See Also

[SocketCore](#), [UDPSocket](#) classes.

---

## DataNotificationReceiver Interface Class

Used to program custom object bindings.

### Methods

| Name   | Parameters | Description                         |
|--------|------------|-------------------------------------|
| Reload |            | Reloads data notifier after changes |
| Commit |            | Commits changes to data.            |

---

## DataNotifier Interface Class

Used to program custom object bindings.

### Methods

| Name                           | Parameters  | Description  |
|--------------------------------|---|--|
| addDataNotificationReceiver    | receiver as<br><a href="#">DataNotificationReceiver</a> | Adds a data notification receiver to the interface class       |
| removeDataNotificationReceiver | receiver as<br><a href="#">DataNotificationReceiver</a> | Removes a data notification receiver from the interface class. |

---

## Date Class

A **Date** object stores the number of seconds since 12:00AM, January 1, 1904.

Properties of a **Date** enable you to get and set a day value only, a date/time, or only a time.

### Super Class

[Object](#)

## Properties

| Name                   | Type                    | Description   |
|------------------------|-------------------------|---|
| <b>AbbreviatedDate</b> | <a href="#">String</a>  | Reports the date in the user's abbreviated date format as a <a href="#">string</a> based on the user's locale and formatting even if the user's locale is a Unicode-only locale Example (US format: Wed, Dec. 31, 1997).                |
| Day                    | <a href="#">Integer</a> | The day number of the date.   |
| <b>DayOfWeek</b>       | <a href="#">Integer</a> | The day of the week as an integer: 1=Sunday, 7=Saturday.  |
| <b>DayOfYear</b>       | <a href="#">Integer</a> | The number days into the year that the date falls on.   |
| Hour                   | <a href="#">Integer</a> | The hour number of the time portion of the date.  |
| <b>LongDate</b>        | <a href="#">String</a>  | Reports the date in the user's long date format as a <a href="#">string</a> based on the user's locale and formatting even if the user's locale is a Unicode-only locale. Example (US format): Wednesday, December 31, 1997.            |
| <b>LongTime</b>        | <a href="#">String</a>  | Reports the date in the user's long time format as a <a href="#">string</a> based on the user's locale and formatting even if the user's locale is a Unicode-only locale. Example (US format): 2:32:40 PM                               |
| Minute                 | <a href="#">Integer</a> | The minute number of the time portion of the date.  |
| Month                  | <a href="#">Integer</a> | The month number of the date.   |
| Second                 | <a href="#">Integer</a> | The second number of the time portion of the date.  |
| <b>ShortDate</b>       | <a href="#">String</a>  | Reports the date in the user's short date format as a <a href="#">string</a> based on the user's locale and formatting even if the user's locale is a Unicode-only locale. Example (US format): 12/31/97                                |
| <b>ShortTime</b>       | <a href="#">String</a>  | Reports the date in the user's short time format as a <a href="#">string</a> based on the user's locale and formatting even if the user's locale is a Unicode-only locale. Example (US format): 2:32 PM.                                |
| <b>SQLDate</b>         | <a href="#">String</a>  | Gets and sets the date in SQL date format, YYYY-MM-DD. Example: 2003-09-03  |
| <b>SQLDateTime</b>     | <a href="#">String</a>  | Gets and sets the date in SQL date/time format, YYYY-MM-DD HH:MM:SS. Example: 2003-09-03 13:39:16   |
| <b>TotalSeconds</b>    | <a href="#">Double</a>  | The number of seconds since 12:00AM, January 1, 1904. TotalSeconds is the 'master' property from which other ways of expressing date/time are derived. A negative value of TotalSeconds indicates a Date/Time prior to January 1, 1904. |
| <b>WeekOfYear</b>      | <a href="#">Integer</a> | The number of the week of the year the date falls in. The first week is numbered 1. The first week may be incomplete. If January 1 falls on a Saturday, then the next day is in week 2.   |
| <b>Year</b>            | <a href="#">Integer</a> | The year portion of the date.   |

## Methods

| Name                             | Parameters | Description  |
|----------------------------------|------------|--|
| <a href="#">Operator_Compare</a> | d as Date  | Returns an <a href="#">integer</a> whose meaning is as follows: <0 means that the date associated with the current <b>Date</b> object is before the passed date, 0 means that it is equal to the passed date, and >0 means that it is after the passed date. |

## Notes

When you create and instantiate a **Date** object, it is initialized to the current date and time. If you try to set the date or date/time and the format is incorrect, REALbasic will raise an [UnsupportedFormatException](#).

The date properties of [FolderItems](#) can be accessed via the CreationDate and ModificationDate properties of [FolderItem](#) objects. You can get the current date and time by creating a new date and reading the values of the Year, Month, Day, Hour, Minute, and Second properties.

In the following code:

```
Dim v as Variant  
Dim d as Date  
d=New Date  
v=d
```

What is actually happening is that the [Variant](#) stores the value of the TotalSeconds property as a [Double](#), along with the type information that it is a **Date** (the Variant's Type property = 7).

Although **Date** is a class that is subclassed from [Object](#), [VarType](#) identifies a **Date** as a Date data type (Type=7) rather than an Object (Type=9).

Use the [ParseDate](#) function to convert a date string to a **Date** value.

The TotalSeconds property is the 'master' property that stores the date/time associated with a **Date**. The other property values are derived from TotalSeconds. If you change the value of the TotalSeconds property, the values of the Year, Month, Day, Hour, Minute, and Second properties change to reflect the second on which TotalSeconds occurs. Conversely, if you change any of these properties, the value of TotalSeconds changes commensurately.

The **Date** properties that return formatted date or time information are affected by the user's operating system settings. For example, the Regional and Language Options panel on Windows XP determines how dates are formatted.



If you need to control the exact appearance of date/time information, the best way is to extract the information yourself and manage the formatting using [string](#) manipulation functions.

## Examples

This example creates a **Date** and displays the current date in a message box.

```
Dim d as Date
d=New Date
MsgBox d.shortdate
```

You can write this more compactly by instantiating the **Date** variable within the [Dim](#) statement, as in this example:

```
Dim d as New Date
MsgBox d.shortdate
```

This example sets a **Date** object to 10 February 1954.

```
Dim d as New Date
d.Year=1954
d.Month=2
d.Day=10
MsgBox d.ShortDate
```

## DebugBuild Constant

---

The following example compares that date to the current date:

```
Dim d as New Date  
Dim today as New Date  
Dim i as Integer  
d.Year=1954  
d.Month=2  
d.Day=10  
i=today.Operator_Compare(d)
```

**See Also** [Microseconds](#), [ParseDate](#), [Ticks](#), functions; [FolderItem](#) class.

## DebugBuild Constant

Used to determine whether the current operating environment is the IDE.

**DebugBuild** is [True](#) when you compile the project by clicking the Run button in the Toolbar.

**Syntax** **result=DebugBuild**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | <a href="#">True</a> if the application is running within REALbasic, i.e., you chose Debug ► Run. |

**Example** The following example displays a message box only in the IDE.

```
#if DebugBuild then  
    MsgBox "You're in the IDE"  
#endif
```

**See Also** [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetWin32](#) constants; [#If](#) statement.

## DebugDumpObjects Method

Used to determine if your application is leaking memory.

**Syntax** **DebugDumpObjects (filename)**

| Part     | Type                   | Description   |
|----------|------------------------|---|
| filename | <a href="#">String</a> | The name of the text file that will be created when the method is called. |

**Notes**

REALbasic manages memory for you. This means that it allocates memory for new objects your application creates, keeps track of which objects your application is currently using and removes objects from memory when they are no longer in use. If your code creates circular references this might prevent objects from being released when you expect them to. See, for example, the discussion of circular references in [StackOverflowException](#).

The **DebugDumpObjects** method can help determine if you have a memory leak. When you call **DebugDumpObjects**, the method creates a text file in the same location as your application or the same location as REALbasic if you are running your application in the IDE. This text file contains the amount of memory currently being used by objects like windows, controls, instances of classes, etc. The text file also lists all of the objects in memory. You can determine if a memory leak exists by calling this method at various times and then comparing the text files that are created to determine if the objects in memory are what you are expecting to be in memory.

**Example**

The following line of code creates the text file described above:

```
DebugDumpObjects ("dumpfile")
```

---

## Declaration: Contains a Reserved Word Error

You used a [Dim](#) statement while declaring a property in the Add Property area. [Dim](#) is used only for declaring local variables.

The [Dim](#) keyword should be omitted this declaration:

```
Dim TextHasChanged as Boolean
```

**See Also**

[Dim](#) statement.

---

## Declare Statement

Used to make API calls. Both PPC and x86 machines are supported. DLLs can be called from a built Windows application. Shared libraries can be called from Macintosh and Linux applications.

**Syntax**

**[Soft] Declare Sub/Function Name Lib *LibraryName* [Alias *AliasName*]  
([*Parameters*]) [*As ReturnType*]**

| Part | Type | Description   |
|------|------|---|
| Name | Name | Name of the API call. Use Name in your code to refer to the API call. |

## Declare Statement

| Part        | Type                   | Description  |
|-------------|------------------------|--|
| LibraryName | <a href="#">String</a> | Library in which API call is found.  |
| AliasName   | <a href="#">String</a> | Optional: If the API call has the same name as a REALbasic method, declare the call with a different name and use the alias to refer to the actual API call. |
| Parameters  |                        | Parameters of the API call. You can pass <a href="#">Nil</a> to a parameter of type <a href="#">Ptr</a> .  |
| ReturnType  |                        | Optional: The data type of the value returned by the call if it is a function.   |

### Notes

Function names for the OS are case sensitive. For example, SetLocalTime works on Win32 but SetlocalTime does not.

**Declare** statements must not call strictly “classic” libraries like AppearanceLib to run on Mac OS X. If they do not, the application will run in the “classic” environment within Mac OS X.

The **Declare** statement allows use of constants for Lib names.

When building a Carbon application in the Mach-O format, the Lib parameter to a **Declare** statement now accepts the name of a system framework in place of a full path to the library. If you wrote a declare against “CoreMIDI”, for example, the compiler would see that the Lib string did not contain any path separators, and would expand it to “/System/Library/Frameworks/CoreMIDI.framework/CoreMIDI”.

You can pass [Nil](#) to a **Declare** parameter of type [Ptr](#). Passing [Nil](#) is not the same as passing a [Nil MemoryBlock](#). The former is a constant and the latter require a conversion operation at run time. If the [MemoryBlock](#) is [Nil](#), then the conversion operator will generate a [NilObjectException](#). However, this isn’t a problem when passing the constant [Nil](#), as no conversion is needed.

**Soft Declares** In versions of REALbasic prior to 2005, the only supported type of declares was a “hard” declare. A hard declare of the form:

```
Declare Sub Foo Lib "Bar" ()
```

tells REALbasic that Foo will always be in Bar, and Bar will always be installed on the user’s system. When building your application, REALbasic puts this dependency in your application’s Import table. When the system loader tries to load your application, it will automatically find all of these Declares and try to resolve them. If it can’t resolve a library or a function from a library, your application would refuse to load and terminate.

The ‘Soft’ Declare keyword provides some important flexibility. Soft declares are not put into your application’s import table so the system loader doesn’t try to resolve them. Instead, when you first try to use the softly declared method, REALbasic is responsible for resolving the library and functions. This provides you with a few handy features.

First, it means that you now have a graceful way to deal with the situation of a function or library that isn't found. Instead of your application just not launching, you now have a [FunctionNotFound](#) exception that you can catch. Second, REALbasic can be more flexible about loading the library.

For example, LibC on many newer Linux distributions is really a ld loader script that GCC uses to determine which version of LibC to actually link against. This means that you can't write hard declares that link against LibC in REALbasic because libc.so isn't really a symbolic link to the proper shared library (like most other .so files are). You would have to write a **Declare** against the actual LibC version you wanted, which provides for a very poor experience for users who don't have the same version of LibC installed as you. However, with a soft declare, REALbasic can resolve the loader script for LibC and find the proper shared library to load against. In other words, what used to be declared like this:

```
Declare Function getpid Lib "libc-2.3.2.so" () as Integer
```

Can now be declared like this:

```
Soft Declare Function getpid Lib "LibC" () as Integer
```

With the Soft keyword, you don't have to try to link against a specific version of that library.

On Windows, any function that uses a string will have two different versions, the A version, and the W version. This is the Win32 way of dealing with Unicode. The disadvantage is that the W version of the method is usually available only on Windows NT machines, not 9x ones. If you could not use a Soft declare, you'd have to make two different versions of your application. Using Soft declares, you can do this:

```
Soft Declare Function MessageBoxA Lib "User32" ( hwnd as Integer, _
msg as CString, title as CString, flags as Integer ) as Integer
Soft Declare Function MessageBoxW Lib "User32" ( hwnd as Integer, _
msg as WString, title as WString, flags as Integer ) as Integer
Try
Call MessageBoxW( hWnd, "This is a string that handles unicode", _
    "Title here", 0 )
Catch
Call MessageBoxA( hWnd, "This is a string that doesn't handle unicode", _
    "Title here", 0 )
End
```

This code runs on any version of Windows. The reason is simple: if the wide char version (W version) is available, then your call will not throw an exception, and because of the WString data type, all your strings will be automatically converted into UTF-16 strings when passed into the function. However, if the call doesn't resolve because

## Declare Statement

---

you're running on Windows 98 (or another 9x machine), then the exception will be thrown and will use the ANSI (A version) of the function call.

Here is a feature of Soft declares for Macintosh builds. You can call into Mach-O bundles from within a CFM application. For example, if you wanted to use the BSD function called "socket", you can use a soft declare like this:

**Soft Declare** [Function](#) socket Lib "System.framework" ( domain as [Integer](#), type as [Integer](#), protocol as [Integer](#) ) as [Integer](#)

### Checking that a Function is Available

To check whether a function can be resolved, you can use the [IsFunctionAvailable](#) function of the [System](#) object. It determines whether a function can be resolved without having to actually declare it and try to call into it.

### Data Types

The following data types are for use with Declares only:

| Type                        | Description                |
|-----------------------------|----------------------------|
| <a href="#">Byte</a>        | Signed 8-bit integer       |
| <a href="#">CFStringRef</a> | A CFString reference       |
| <a href="#">CString</a>     | A null-terminated string.  |
| <a href="#">OSType</a>      | A four char code           |
| <a href="#">PString</a>     | A "Pascal" string.         |
| <a href="#">Ptr</a>         | 4-byte pointer to memory   |
| <a href="#">Short</a>       | 2-byte signed integer      |
| <a href="#">UByte</a>       | Unsigned 8-bit integer     |
| <a href="#">UShort</a>      | Unsigned 2-byte integer    |
| <a href="#">WindowPtr</a>   | 4-byte integer to a handle |
| <a href="#">WString</a>     | A UTF-16 string.           |

See the descriptions of each data type for more information.

### Automatic Types

There are two new automatic types: [WString](#) and [CFStringRef](#). You can pass [Nil](#) in place of a [CString](#), [WString](#), [CFStringRef](#), or [PString](#).

You can use these two new types with declares that need a wchar\_t \* or [CFStringRef](#), respectively. Their usage is the same as with [CString](#), and all data is cleaned up when appropriate. For example:

**Declare** [Sub](#) Foo Lib "Bar"(utf16Str as [WString](#))

## Examples

### Name Conflicts

Suppose you want to write a function that got the various system metrics. You use a method declaration like this:

[Function](#) GetSystemMetrics (whichMetric as [Integer](#)) as [Integer](#)

However, this conflicts with the Win32 API which the function wraps. In this case, you can rename the declare using an alias such as this:

```
Declare Function MyGetSystemMetrics Lib "User32" Alias "GetSystemMetrics" _  
    type as Integer ) as Integer
```

### Defining Double-clicks

The following example displays the maximum number of ticks between mouse clicks for two clicks to be considered a “double-click.” Above this value, two clicks are considered separate mouse events.

```
Dim DoubleClickTime as Integer  
#if TargetMacOS  
    Declare Function GetDblTime Lib "InterfaceLib" () as Integer  
    doubleClickTime = GetDblTime()  
    #endif  
  
#if TargetWin32  
    Declare Function GetDoubleClickTime Lib "User32.DLL" () as Integer  
    doubleClickTime = GetDoubleClickTime()  
    #endif  
    MsgBox Str(doubleclicktime)
```

To modify the code to run under Carbon, substitute “CarbonLib” for “InterfaceLib”:

```
Dim DoubleClickTime as Integer  
#if TargetCarbon  
    Declare Function GetDblTime Lib "CarbonLib" () as Integer  
    doubleClickTime = GetDblTime()  
    #endif  
    MsgBox Str(doubleclicktime)
```

## Declare Statement

---

The following example gets the current process ID and illustrates Declares for Mach-O, Mac 'classic', Windows, and Linux.

```
Function GetPid() As Integer
Dim pid as Integer
Dim psn as MemoryBlock

#if TargetWin32
' On Windows, we're going to use the GetCurrentProcessId
' function to figure out our pid
Declare Function GetCurrentProcessId Lib "Kernel32" () as Integer
' Get the pid according to windows
id = GetCurrentProcessId

#elseif TargetMachO
' For a Mach-O app, we can just use the system's
' getpid function in the kernel framework, which
' is automatically included in the system framework.
Declare Function getpid Lib "/System/Library/Frameworks/System.framework/System" _
() as Integer
' Get the pid according to mach
pid = getpid
#elseif TargetLinux
' On Linux, we get the pid from the libc library. It
' is worth noting that I am hard coding the version
' of libc in this call. It requires libc version 2.3.2
Declare Function getpid Lib "libc-2.3.2.so" () as Integer
' Get the pid according to the kernel
pid = getpid

#elseif TargetCarbon or TargetMacOSClassic
' On Carbon PEF or Classic we will use the lower
' 32 bits of the ProcessSerialNumber.
Declare Function GetCurrentProcess Lib "CarbonLib" (psn as Ptr) as Integer

' Make a 64 bit structure
psn = New MemoryBlock( 8 )
' Get the PSN from it
Call GetCurrentProcess( psn )
' And store off the lower 32 bits (the upper
' 32 seemed to always be 0 anyways)
pid = psn.Long( 4 )
#endif

' Return the pid we found (note, if we didn't
' find a pid, it'll return 0)
Return pid
```

The following example adds the ability to detect a double-click to a Canvas control. The code is placed in a Canvas's MouseUp event handler.

```
Sub MouseUp (X as Integer, Y as Integer)
Dim doubleClickTime, currentClickTicks as Integer

#If TargetMacOS then
#if TargetCarbon then
    Declare Function GetDblTime Lib "CarbonLib" () as Integer
#endif
    DoubleClickTime = GetDblTime()
#endif

#if TargetWin32 then
    Declare Function GetDoubleClickTime Lib "User32.DLL" () as Integer
    doubleClickTime = GetDoubleClickTime()
#endif

#if TargetLinux then
    Declare Function gtk_settings_get_default lib "libgtk-x11-2.0.so" as Ptr
    Declare Sub g_object_get lib "libgtk-x11-2.0.so" (Obj as Ptr, _
        first_property_name as CString, byref doubleClicktime as Integer, _
        Null as Integer)

    Dim gtkSettings as MemoryBlock

    gtkSettings = gtk_settings_get_default()

    g_object_get(gtkSettings, "gtk-double-click-time",doubleClickTime, 0)
        // DoubleClickTime now holds the number of milliseconds
    DoubleClickTime = DoubleClickTime / 1000.0 * 60
#endif

    currentClickTicks = ticks
    //if the two clicks happened close enough together in time
    if (currentClickTicks - lastClickTicks) <= doubleClickTime then
        //if the two clicks occurred close enough together in space
        If Abs(X - lastClickX) <= 5 and Abs(Y - LastClickY) <= 5 then
            DoubleClick //a double click has occurred so call the event
            end if
        end if
        lastClickTicks = currentClickTicks
        lastClickX = X
        lastClickY = Y
```

**See Also**

[TargetCarbon](#), [TargetLinux](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; [System](#) object, [FunctionNotFound](#) exception; [Byte](#),

## DecodeBase64 Function

---

[CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

## DecodeBase64 Function

---

Decodes a Base64 string back to its original form.

### Syntax

**result=DecodeBase64(str [, encoding])**

| Part            | Type                         | Description  |
|-----------------|------------------------------|--|
| <i>result</i>   | <a href="#">String</a>       | The decoded value of <i>str</i> .  |
| <i>str</i>      | <a href="#">String</a>       | The Base64 <a href="#">String</a> to be decoded.   |
| <i>encoding</i> | <a href="#">TextEncoding</a> | Optional: The text encoding of the passed string. If you pass an encoding, it has the same effect as calling <a href="#">DefineEncoding</a> on the returned <a href="#">String</a> . |

### Notes

The **DecodeBase64** function performs the reverse operation of [EncodeBase64](#). An encoded string passed to **DecodeBase64** returns the original string.

### See Also

[EncodeBase64](#) function.

## DecodeQuotedPrintable Function

---

Decodes into ‘normal’ text a string that has been encoded by [EncodeQuotedPrintable](#).

### Syntax

**result=DecodeQuotedPrintable(str[, encoding])**

| Part            | Type                         | Description  |
|-----------------|------------------------------|--|
| <i>result</i>   | <a href="#">String</a>       | The result of processing <i>str</i> .  |
| <i>str</i>      | <a href="#">String</a>       | The string expression to be decoded back to its original form.   |
| <i>encoding</i> | <a href="#">TextEncoding</a> | Optional: The text encoding of the passed string. If you pass an encoding, it has the same effect as calling <a href="#">DefineEncoding</a> on the returned <a href="#">String</a> . |

### Notes

**DecodeQuotedPrintable** converts text representations of unprintable characters (i.e., control characters, returns and tabs) back into their original form. It is designed for handling email or usenet messages. It undoes the encoding done by [EncodeQuotedPrintable](#).

The **DecodeQuotedPrintable** function performs the function in reverse.

**DecodeQuotedPrintable** can decode strings in which hexadecimal values are represented in lowercase as well as the standard uppercase.

**See Also** [EncodeQuotedPrintable](#) function.

## DecodeURLComponent Function

Decodes the components of a URL that were encoded by [EncodeURLComponent](#) or equivalent.

**Syntax** ***result=DecodeURLComponent(str)***

| Part   | Type                   | Description            |
|--------|------------------------|------------------------|
| result | <a href="#">String</a> | The 'original' string. |
| str    | <a href="#">String</a> | The encoded URL.       |

**Notes** A valid URL is a series of components that are separated by component separators. They are the “.”, “/”, “;”, “&”, and “?”. [EncodeURLComponent](#) works with each component part of the URL. It assumes that any component separators in a component represent text and must be encoded. An encoded URL is decoded by **DecodeURLComponent**.

**Example** Here is an example of how an encoded URL is decoded.

```
Dim s as String
s=DecodeURLComponent("www.bob%26ray.com") //returns "www.bob&ray.com"
```

**See Also** [EncodeURLComponent](#) function.

## DefineEncoding Function

Returns a [String](#) with the same data as the given string, but with the encoding of the given encoding. This function is useful when you have a string whose encoding is known to you but not to REALbasic. REALbasic ‘knows’ the encoding of the text it creates, so you don’t have to use **DefineEncoding** for any such text.

**Syntax** ***result=DefineEncoding(str, enc)***

OR

## DesktopFolder Function

---

**result=**[str.DefineEncoding\(enc\)](#)

| Part          | Type                         | Description  |
|---------------|------------------------------|--|
| <i>result</i> | <a href="#">String</a>       | The result of encoding <i>str</i> using the encoding specified by <i>enc</i> . |
| <i>str</i>    | <a href="#">String</a>       | The <a href="#">String</a> to be encoded.                                      |
| <i>enc</i>    | <a href="#">TextEncoding</a> | The <a href="#">TextEncoding</a> to be used to encode <i>str</i> .             |

### Notes

Consult the values of *Base* entry for [TextEncoding](#) when creating the [TextEncoding](#) object using the [GetTextEncoding](#) function.

### Example

The following example takes 8 bytes from a [MemoryBlock](#) and sets the encoding to UTF16.

```
Editfield1.text=DefineEncoding(MyMemoryBlock.StringValue(0,8),  
Encodings.UTF16 )
```

This example uses **DefineEncoding** when reading text via a [TCPSocket](#).

```
EditField1.Text = DefineEncoding( TCPSocket1.ReadAll, Encodings.ASCII )
```

### See Also

[TextEncoding](#) class; [ConvertEncoding](#), [Encoding](#), [GetTextEncoding](#) functions;  
[Encodings](#) object.

---

## DesktopFolder Function

Used to access to the desktop directory. The [SpecialFolder](#) object supports access to this and many other system special folders.

### Syntax

**result=**[DesktopFolder](#)

| Part          | Type                       | Description  |
|---------------|----------------------------|--|
| <i>result</i> | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the desktop folder. |

### Notes

Use the **DesktopFolder** function to access items on the desktop. Though they don't appear to be in a folder, in fact, all items on the desktop are in a directory or folder.

### Windows

On Windows, **DesktopFolder** returns a [FolderItem](#) that references the Desktop directory for the current user: \Documents and Settings\user\Desktop, where *user* is the name of the currently-logged in user.

### Macintosh

On Mac OS X, **DesktopFolder** returns a [FolderItem](#) that references the Desktop directory for the current user: /Users/*user/Desktop*:

**Linux**

On Linux, **DesktopFolder** tries to return the desktop folder for the current user's Window Manager. If there is a folder named "Desktop", it will return a [FolderItem](#) to that. Otherwise, if there is a folder in the current user's home directory named ".gnome-desktop", it will return a [FolderItem](#) to that.

Volumes and the Trash Can or Recycle Bin are not included in the Desktop Folder; however, desktop printers are. The Recycle Bin/Trash Can can be accessed via the [TrashFolder](#) function and the [SpecialFolder](#)'s [Trash](#) property.

**Example**

This example displays the absolute path to the user's Desktop folder, if it exists on the user's machine.

```
Dim f As FolderItem
f=DesktopFolder
if f <> Nil Then
  MsgBox f.AbsolutePath
Else
  MsgBox "The folderItem does not exist."
End if
```

**See Also**

[ApplicationSupportFolder](#), [FontsFolder](#), [PreferencesFolder](#), [StartupItemsFolder](#), [SystemFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

## Destructor Method

Executed automatically when a class goes out of scope.

**Syntax****Destructor****Notes**

The **Destructor** method is called automatically, so you would never call it yourself. It takes no parameters and does not return a value.

Destructors are typically used to do any "clean up" that REALbasic doesn't do automatically. If it exists, a class's destructor is called automatically when an instance of the class is deleted or goes out of scope — for example, when the user closes the window.

To create the Destructor for a custom class, add a method to the class and name it "Destructor".

Destructors are called when the last reference to an object is removed, even if execution is in a destructor for another object. Note that this means you can cause a stack overflow if your destructor triggers other destructors in a deep recursion. However, such overflow will not happen as long as properties of the object are being cleaned up

automatically. So, it is generally preferable to *not* set properties to [Nil](#) in your destructor, but instead let REALbasic clean them up for you.

**See Also** [Constructor](#) method; [Destructors Can't Have Parameters](#) error.

---

## Destructors Can't Have Parameters Error

You cannot define parameters for a class's destructor.

A destructor is a special method of a class that is named Destructor. It executes automatically when the class goes out of scope.

---

## Dictionary Class

An object that contains a list of *key-value* pairs. Both keys and values are [Variants](#).

**Super Class** [Object](#)

### Properties

| Name     | Type                    | Description  |
|----------|-------------------------|--|
| BinCount | <a href="#">Integer</a> | <p>The number of bins the hash table uses. This is a measure of the hash table size, independent of the number of items the <b>Dictionary</b> contains.</p> <p>Normally, you do not need to be concerned with bins and the hash table. BinCount would be of use to you only if you have very large dictionaries consisting of thousands of entries and you want to try to optimize performance.</p> <p>If you set BinCount to a positive value, the <b>Dictionary</b> will use that number of bins regardless of the number of items in the <b>Dictionary</b>.</p> <p>If you set BinCount to a value equal to or less than zero, the <b>Dictionary</b> will pick whatever value it thinks is appropriate for the number of items in the dictionary. It will dynamically change the number of bins as items are added or removed from the <b>Dictionary</b> (or until you assign a positive value to BinCount).</p> <p>The latter is the default behaviour.</p> |
| Count    | <a href="#">Integer</a> | Number of key-value pairs in the <b>Dictionary</b> .   |
| Value    | <a href="#">Variant</a> | Takes one parameter, key as <a href="#">Variant</a> . Assigns a value to the key item in the <b>Dictionary</b> or retrieves the value associated with the key item. If you attempt to read a value for a key that is not in the dictionary, a <a href="#">KeyNotFoundException</a> error is raised.  |

## Methods

| Name   | Parameters   | Description   |
|--------|--|---|
| Clear  |  | Clears all the key-value pairs in the <b>Dictionary</b> .   |
| HasKey | Key as <a href="#">Variant</a>   | Returns a <a href="#">Boolean</a> . Returns <a href="#">True</a> if Key is in the <b>Dictionary</b> and <a href="#">False</a> if it is not.   |
| Key    | Index as <a href="#">Integer</a>   | Returns a <a href="#">Variant</a> . Returns the value of key for the <i>Index</i> <sup>th</sup> sequential item in the <b>Dictionary</b> . The first item is numbered zero. If there is no <i>Index</i> <sup>th</sup> item in the <b>Dictionary</b> , a call generates an <a href="#">OutOfBoundsException</a> error. |
| Keys   |  | Returns all the keys in the <b>Dictionary</b> as an array of <a href="#">Variants</a> . The order is stable and matches the order returned by the Values method at least until the <b>Dictionary</b> is modified. Use this method with <a href="#">ForEach</a> to loop through all the keys.                          |
| Lookup | key as <a href="#">Variant</a> , defaultValue as <a href="#">Variant</a> | Returns a <a href="#">Variant</a> . Looks up the passed value of Key. If Key is found, it returns the corresponding value. If Key is not found, it returns the passed <i>defaultValue</i> .   |
| Remove | Key as <a href="#">Variant</a>   | Removes the key value-key pair from the <b>Dictionary</b> . If Key is not in the <b>Dictionary</b> , it raises a <a href="#">KeyNotFoundException</a> error.  |
| Values |  | Returns all the values in the <b>Dictionary</b> as an array of <a href="#">Variants</a> . The order is stable and matches the order returned by Keys at least until the <b>Dictionary</b> is modified. Use this method with <a href="#">ForEach</a> to loop through all the values.                                   |

## Notes

The **Dictionary** class provides the functionality of the [Collection](#) class and offers several advantages: With the [Collection](#) class, the time taken to locate an item is a function of the number of items in the [Collection](#) because the search is sequential. A **Dictionary** uses a hash table, making the time (relatively) independent of the number of items. It is designed for high-speed lookups. Also, the key parameter in a **Dictionary** is a [Variant](#), but is a [String](#) in the [Collection](#) class, giving you greater flexibility. On the other hand, the Collection can store multiple values per key. When you assign a value to a key that is already in the **Dictionary**, the new value replaces the old one.

## Dim Statement

---

### Example

The following example takes advantage of the fact that the key is a [Variant](#), and not necessarily a number. In this example, the keys are colors and the values are descriptions.

```
Dim d as New Dictionary
d.Value(RGB\(255,0,0\))="This is pure red."
d.Value(RGB\(0,255,255\))="This is pure cyan."
MsgBox d.value(d.Key(1)) //retrieves cyan
Exception err as RuntimeException
If err IsA KeyNotFoundException then
    MsgBox "Key not in the dictionary"
End if
If err IsA OutOfBoundsException then
    MsgBox "The index of the key is out of bounds!"
End if
```

If the index passed to the Key method in the line:

```
MsgBox d.value(d.Key(1))
```

is not an element in the dictionary, an [OutOfBoundsException](#) occurs and the [Exception block](#) at the end of the method will trap it. You could also retrieve this value in the dictionary by passing the Value method the key rather than the key's index:

```
MsgBox dict.value(RGB\(0,255,255\))
```

In this case, if you pass a key that is not in the dictionary, a [KeyNotFoundException](#) will occur and the [Exception block](#) will also trap it and display the appropriate error message.

Instead of the [Exception](#) block, you could first use the HasKey method before trying to access the value:

```
If d.HasKey(RGB\(0,255,255\)) then
    MsgBox d.value(RGB\(0,255,255\))
Else
    MsgBox "Key not in the dictionary!"
End If
```

### See Also

[KeyNotFoundException](#) error; [Collection](#), [Variant](#) classes.

---

## Dim Statement

Creates a local variable or array with the name and size (in the case of an array) and data type specified. The [Static](#) statement provides similar functionality except that the variables retain their values from one invocation of the method to the next.

## Syntax

The **Dim** statement has two syntaxes:

**Dim variableName [,variableNameN] As [New] dataType [= initialValue]**

| Part          | Type                  | Description   |
|---------------|-----------------------|---|
| variableName  | Variable name         | The name of the new variable.   |
| variableNameN | Variable name         | Optional. The names of other variables you wish to create with the same data type. Each name should be separated with a comma.  |
| dataType      | Data type name        | The data type of the variable.  |
| InitialValue  | Same type as dataType | Initial value for variableName. If the datatype is an object that requires instantiation via the New operator, you can do that instead of assigning an initial value. |

**Dim arrayName(size [,sizeN]) as dataType**

| Part      | Type                    | Description  |
|-----------|-------------------------|--|
| arrayName | Variable name           | The name of the new array.   |
| size      | <a href="#">Integer</a> | The size (number of elements) for the array.   |
| sizeN     | <a href="#">Integer</a> | Optional. The size of the next dimension of the array if you are creating a multi-dimensional array. |
| dataType  | Data type name          | The data type of the array.  |

## Notes

In the first syntax, the **Dim** statement is used to create new variables. A variable is an object stored in RAM that can hold a value. In the second syntax, the **Dim** statement is used to create a new array of the size specified. An array is simply a variable that can contain multiple values that are all the same data type. Passing more than one size parameter creates a multi-dimensional array, with the number of dimensions equal to the number of size parameters passed.

You can optionally provide an initial value to a variable declared with a **Dim** statement. If the variable is one of REALbasic's built-in data types ([String](#), [Integer](#), [Single](#), [Double](#), [Boolean](#), or [Color](#)) you can use the assignment statement to assign the initial value. Here are some examples:

```
Dim a as Integer =5
Dim b as Double =87.87
Dim s as String="Howdy"
Dim c as Color = &cFF4B51
```

The following statement initializes three variables to an initial value:

```
Dim x, y, z As Integer = 10
```

## Dim Statement

---

After the statement, all three variables have the same initial value.

If the data type is an object, you can optionally instantiate the object with the [New](#) operator in the **Dim** statement. This code declares and instantiates a [Date](#) object.

```
Dim d as New Date
MsgBox d.shortdate
```

Otherwise, you need a second line of code to instantiate the variable, *d*, as in the following:

```
Dim d as Date
d=New Date
MsgBox d.shortdate
```

If the call to the object's constructor takes parameters, you can pass the parameters as well. Here is an example:

```
Dim mb as New MemoryBlock(512)
```

If you write:

```
Dim a, b as New Memoryblock(10)
```

Both *a* and *b* point to the same [Memoryblock](#).

The **Dim** statement can be placed anywhere in a method or function, including inside a conditional structure (an [If](#) or [Case](#) statement).

Variables created with the **Dim** statement are local in scope. This means that they are created each time the method is called and are destroyed when the method is finished. The value of a local variable can be accessed only from within the method.

If you don't know the size of the array you need at the time you declare it, you can declare it as a null array, i.e., an array with no elements, and use the [Redim](#) command to resize it later. You do this by giving it an index of -1 or by leaving the parentheses empty. This means "an array of no elements." For example, the statement:

```
Dim aNames (-1) as String
```

creates the array *aNames* with no elements. If your program needs to load a list of names that the user enters, you can wait to size the array until you know how many names the user has entered. You can also accomplish this by leaving out the -1. The following statement is equivalent:

```
Dim aNames () as String
```

When you assign values to an array variable, such as with the [Split](#) function, you don't need to know the number of elements ahead of time.

You can also assign an array to another array. For example, the following is valid:

```
Dim myArray() as String  
Dim newArray() as String  
myArray(0)="Anthony"  
myArray(1)="Aardvark"  
myArray(2)="Accountant"  
newArray()=myArray //newArray set equal to myArray
```

The [Static](#) statement provides an alternative to **Dim**. [Static](#) creates a local variable or local array with the name and size (in the case of an array) and data type specified. A variable declared with the [Static](#) statement and assigned a value retains its value from one invocation of the method to the next. In contrast, variables declared with the **Dim** statement are completely local to the method and are destroyed when each invocation of the method goes out of scope.

## Examples

This example uses the **Dim** statement to create an [integer](#) variable called "age" and assigns it the value 33.

```
Dim age As Integer  
age=33
```

This example uses the **Dim** statement to create two [string](#) variables.

```
Dim FirstName, LastName As String
```

This example uses the **Dim** statement to create an array called aNames with 11 elements (remember, arrays have a zero element).

```
Dim aNames(10) As String
```

This example uses the **Dim** statement to create an array called aNames with one element.

```
Dim aNames(0) As String
```

This example uses the **Dim** statement to create a two-dimensional array called aNames with 11 rows and 4 columns.

```
Dim aNames(10,3) As String
```

## See Also

[Append](#), [Insert](#), [Redim](#), [Remove](#), [Sort](#) methods; [UBound](#) function; [Collection](#), [Dictionary](#) classes; [Static](#) statement.

# DisclosureTriangle Control

Used to add a disclosure triangle to a window. A **DisclosureTriangle** can face either left or right in its “closed” position. When the user clicks a **DisclosureTriangle** control, the value of *Value* is toggled automatically.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| AcceptFocus | <a href="#">Boolean</a> | If <a href="#">True</a> , the DisclosureTriangle will be included in the Tab order and can accept the focus. The GotFocus and LostFocus events will fire at the appropriate times. When the DisclosureTriangle has the focus, it has a selection rectangle. If the user clicks the DisclosureTriangle or presses either the Space bar or the Enter key, the Action event will fire. The ability of the DisclosureTriangle to accept the focus is Windows-only. The default value is <a href="#">False</a> . |
| Facing      | <a href="#">Integer</a> | 0 - Right facing<br>1 - Left facing   |
| Value       | <a href="#">Boolean</a> | <a href="#">True</a> corresponds to downward; <a href="#">False</a> corresponds to either left or right depending on the value of Facing.   |

## Events

| Name      | Parameters | Description   |
|-----------|------------|---|
| GotFocus  |            | (Windows only) The DisclosureTriangle has received the focus and has a selection rectangle. The AcceptFocus property must be set to <a href="#">True</a> for the DisclosureTriangle to be capable of getting the focus. |
| LostFocus |            | (Windows only) The control has lost the focus.  |

**See Also** [RectControl](#) class.

---

# DockItem Class

Used to manage the icons for the REALbasic application and application windows in the Mac OS X dock.

**Super Class** [Object](#)

## Properties

| Name            | Type                     | Description   |
|-----------------|--------------------------|---|
| <b>Graphics</b> | <a href="#">Graphics</a> | Allows access to all the <a href="#">Graphics</a> methods for drawing into the dock icon. |

## Methods

| Name             | Parameters | Description  |
|------------------|------------|--|
| <b>ResetIcon</b> |            | Resets the icon to its original state, its default appearance. |
| <b>UpdateNow</b> |            | Redraws the icon.  |

## Notes

The **DockItem** class is for Mac OS X only. It enables you to manipulate the dock item associated with your application or with the application's windows. To manipulate the application's dock icon, use the DockItem property of the [Application](#) class. To manipulate a window's dock icon, use the DockItem property of the [Window](#) class.

Use the methods of the [Graphics](#) class to modify the appearance of the icon. Since a Mac OS X icon is intended to be scaled automatically, you should design it as a 128x128 pixel icon.

The **DockItem** class has two methods, UpdateNow and ResetIcon. ResetIcon resets the icon to its original state. Call the UpdateNow method to redraw the icon. You can also use the ClearRect property of the [Graphics](#) class to start over from a blank icon.

Anything you can do with a [Graphics](#) object you are able to do with the DockItem's Graphics property, such as drawing in a picture or using a [Group2D](#).

The [Graphics](#) property may be [Nil](#); it is non-[Nil](#) for a window only when a Mac OS X window has been minimized to the dock.

**See Also** [Application](#), [Graphics](#), [Window](#) classes.

---

## DocumentsFolder Function

Used to access the user's Documents folder.

## Syntax

**result=DocumentsFolder**

| Part          | Type                       | Description  |
|---------------|----------------------------|--|
| <b>result</b> | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the Documents folder. |

## Do...Loop Statement

---

|                  |   |
|------------------|---|
| <b>Notes</b>     | The <b>DocumentsFolder</b> function provides a way to access the Documents folder that will work under different language systems and all platforms. The <a href="#">SpecialFolder</a> module contains many functions that return special folders managed by the OS.  |
| <b>Windows</b>   | On Windows XP, a call to DocumentsFolder returns the My Documents folder in the current user's directory, e.g., C:\Documents and Settings\username\My Documents.  |
| <b>Macintosh</b> | On Macintosh, it returns the Documents folder for the current user.   |
| <b>Linux</b>     | On Linux, DocumentsFolder tries to get the directory "Documents" in the logged-in user's Home directory. If this directory does not exist, it returns a <a href="#">FolderItem</a> to the user's Home directory.<br><br>The <a href="#">SpecialFolder</a> module enables you to access many special folders that are managed by the OS. |
| <b>Examples</b>  | This example displays the absolute path to the Documents folder.  |

```
Dim f as FolderItem  
f=DocumentsFolder  
If f <> Nil then  
    MsgBox f.AbsolutePath  
Else  
    MsgBox "The item does not exist!"  
End if
```

|                 |   |
|-----------------|---|
| <b>See Also</b> | <a href="#">ApplicationSupportFolder</a> , <a href="#">DesktopFolder</a> , <a href="#">FontsFolder</a> , <a href="#">PreferencesFolder</a> , <a href="#">ShutDownItemsFolder</a> , <a href="#">SpecialFolder</a> , <a href="#">StartupItemsFolder</a> , <a href="#">SystemFolder</a> , <a href="#">TemporaryFolder</a> , <a href="#">TrashFolder</a> , functions. |
|-----------------|---|

## Do...Loop Statement

---

Repeatedly executes a series of statements while a specified condition is [True](#).

|               |   |
|---------------|---|
| <b>Syntax</b> | <b>Do [Until condition]</b><br><br><b>[Statements]</b><br><br><b>[Continue]</b><br><br><b>[Exit]</b><br><br><b>[Statements]</b> |
|---------------|---|

**Loop [Until condition]**

| Part                     | Description   |
|--------------------------|---|
| Do [Until]               | Begins the loop. If the optional <b>Until</b> and condition are included, the condition is evaluated to determine if the loop should exit.  |
| Condition                | Any valid <a href="#">Boolean</a> expression. When this expression evaluates to <a href="#">True</a> , the loop will exit.  |
| Statements               | Statements to be executed repeatedly (until condition evaluates to <a href="#">False</a> ).   |
| <a href="#">Continue</a> | If a <a href="#">Continue</a> statement is present, execution will skip over the remaining statements in the loop and resume with a new iteration of the loop. Optional arguments of the <a href="#">Continue</a> statement allow you to specify which loop will iterate next |
| <a href="#">Exit</a>     | If an <a href="#">Exit</a> statement is present, execution of the loop is terminated and resumes with the next line following the loop.   |
| Loop[Until]              | Ends the loop. If the optional <b>Until</b> and condition are included, the condition is evaluated to determine if the loop should exit.  |

**Notes**

The **Do...Loop** statement continues to execute statements repeatedly until condition is [True](#) or an [Exit](#) statement within the loop is executed.

If statements should only be executed if condition is already [True](#), test for condition at the beginning of the loop. If statements should execute at least once, test condition at the end of the loop.

If condition is not used to cause the loop to exit, your logic must call the [Exit](#) statement somewhere inside the **Do...Loop** or the loop will never end.

**Do...Loop** statements can be nested to any level. Each **Loop** statement goes with the previous **Do** statement.

When a loop runs it takes over the interface, preventing the user from interacting with menus and controls. If the loop is very lengthy, you should move the code for the loop to a separate [Thread](#), allowing it to execute in the background. You can also place the DoEvents method of the [Application](#) class in the loop to enable REALbasic to respond to user input, but separate threads for lengthy operations is recommended.

**Example**

This example uses the **Do...Loop** statement to increment a variable. If the variable is greater than 100 before the loop begins, the variable will not be modified.

```
Do Until x>100
x=x*2
Loop
```

## Double Data Type

---

In this example, the variable will be modified at least once because condition is tested at the end of the loop.

```
Do  
x=x*2  
Loop Until x>100
```

### See Also

[Continue](#), [Exit](#), [For...Next](#), [While...Wend](#) statements; [Application](#), [Thread](#) classes.

## Double Data Type

A **Double** is an intrinsic data type in REALbasic. A **Double** is a number that can contain a decimal value, i.e., a real number. It can take on a value between 2.2250738585072013 e-308 and 1.7976931348623157 e+308. In other languages, REALbasic's **Double** may be referred to as a double precision real number. Because **Doubles** are numbers, you can perform mathematical calculations on them. **Doubles** use 8 bytes of memory. The default value of a **Double** is 0.0.

The [VarType](#) function returns a value of 5 when passed a **Double**.

### Example

The [Single](#) and **Double** data types allow you to store and manage floating point numbers.

```
Dim d as Double  
d=3.14159265358979323846264338327950
```

### See Also

[Boolean](#), [Color](#), [Integer](#), [Single](#), [String](#) data types; [\\_](#), [\\_+](#), [\\*](#), [/\\_](#), [<](#), [<=](#), [=](#), [>=](#), [>](#), [>>](#), [\](#), [IsNumeric](#), [Mod](#), [Val](#), [Str](#), [VarType](#) functions; [Dim](#) statement.

# DragItem Class

Used to transfer data being dragged by the user.

**Super Class** [Object](#)

## Properties

| Name                       | Type                        | Description   |
|----------------------------|-----------------------------|---|
| <b>Destination</b>         | <a href="#">Object</a>      | The location of the dropped item when you drop it to the desktop or onto an application. Since <i>Destination</i> is of type <a href="#">Object</a> , it potentially can return any type of REALbasic object. Currently, <i>Destination</i> is set up to work only with the Macintosh Finder and will only return a <a href="#">FolderItem</a> . It requires either Mac OS 8-9 or Mac OS X 10.2 or above. To ensure that your application is compatible with future versions of REALbasic, check the type of object it returns with <a href="#">IsA</a> before recasting it to a <a href="#">FolderItem</a> . On Windows and Linux, dragging items to the Desktop is not supported; therefore this property does not work under that condition. |
| <b>DropHeight</b>          | <a href="#">Integer</a>     | Height of object being dropped.   |
| <b>DropLeft</b>            | <a href="#">Integer</a>     | Distance from the left side of control where the object was dropped.  |
| <b>DropTop</b>             | <a href="#">Integer</a>     | Distance from the top of the control where the object was dropped.  |
| <b>DropWidth</b>           | <a href="#">Integer</a>     | Width of object being dropped.  |
| <b>FolderItem</b>          | <a href="#">FolderItem</a>  | The <a href="#">FolderItem</a> being dragged.   |
| <b>FolderItemAvailable</b> | <a href="#">Boolean</a>     | The <i>FolderItem</i> property contains a <a href="#">FolderItem</a> .  |
| <b>Handle</b>              | <a href="#">Integer</a>     | Returns the Mac OS toolbox handle to the DragItem. Handle returns 0 on Windows and Linux.   |
| <b>MouseCursor</b>         | <a href="#">MouseCursor</a> | The cursor to be displayed during the drag.   |
| <b>Picture</b>             | <a href="#">Picture</a>     | The Picture being dragged.  |
| <b>PictureAvailable</b>    | <a href="#">Boolean</a>     | The <i>Picture</i> property contains a picture.   |
| <b>PrivateRawData</b>      | <a href="#">String</a>      | Parameter is the file type being dragged. Type is a four-character resource type or programmer-defined four-character code. This data cannot be dragged to another application.   |

| Name             | Type                    | Description  |
|------------------|-------------------------|--|
| RawData          | <a href="#">String</a>  | Parameter is the file Type of the data being dragged. Type is a four-character resource type or programmer-defined four-character code. Supported only on Macintosh and within the REALbasic application on Windows. |
| RawDataAvailable | <a href="#">Boolean</a> | If <a href="#">True</a> , if the DragItem contains data of the type specified.   |
| Text             | Text                    | The text being dragged.  |
| TextAvailable    | <a href="#">Boolean</a> | If <a href="#">True</a> , the Text property contains text.   |

## Methods

| Name     | Parameters   | Description  |
|----------|--|--|
| AddItem  | Left as <a href="#">Integer</a> ,<br>Top as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a> | Adds an item to the drag object. The parameters specify the drag rectangle to be displayed.  |
| Drag     |  | Allows the drag to take place.   |
| NextItem |  | Returns <a href="#">True</a> if there is another item that has been dragged and assigns the next item's values to the DragItem properties. |

## Notes

When the user wishes to drag some data from a [window](#) or [control](#), a new **DragItem** object must be created to hold that data in order to transfer it to the object the data will be dropped on. REALbasic implements true drag and drop — which means that the recipient of the data can be anything that supports drag and drop and supports the kind of data being dragged. For example, text can be dragged to the desktop to create a text clipping file or any application that accepts dropped text (like SimpleText or the NotePad). Pictures can be dragged to the Desktop to create picture clipping files. Other types of data can be dragged using the RawData property.

By default, a **DragItem** can contain only one row containing a single entry for each data type. Using the AddItem method, you can add more rows to the **DragItem** allowing multiple entries of the same data type. For example, if you wanted a drag item to contain two separate pictures, you would assign one picture to the Picture property then call the AddItem method to add a row and then assign the second picture to the Picture property.

The RawData and PrivateRawData properties allow you to support drag and drop for other data formats. You specify the data format using a four-character Type, corresponding to a Macintosh resource type. RawData and PrivateRawData return the dragged item as a [string](#). It is up to the control or window that receives the **DragItem** to manage it. Typically, this will mean that you will have to make API calls or use a plug-in that is designed to handle the data format.

You can also use RawData or PrivateRawData to control internal drag and drop. If you make up a four-character Type (not an existing Macintosh resource type), you can prevent a control from receiving dragged text from other sources. See the example of dragging from one [ListBox](#) to another.

For [EditFields](#), drag and drop is supported only if the MultiLine property is [True](#) (checked). There is no need to create a **DragItem** for EditFields as this is handled automatically by REALbasic.

[ListBoxes](#) support the dragging of rows automatically if the EnableDrag property of the [ListBox](#) is [True](#) (checked). Like EditFields, REALbasic creates the **DragItem** for the ListBox automatically but you must populate the **DragItem** in the DragRow event handler. See the notes for the [ListBox](#) control for more information on dragging from ListBoxes.

The DropWidth and DropHeight properties have no meaning in Win32 and will return 0 unless set using [NewDragItem](#).

## Dragging Promised Files

Mac OS X supports the concept of dragging a file before it actually exists in the file system. It could be a new document that hasn't been saved yet or a file that exists on a remote server or on the web.

In these cases, the drag and drop gesture serves the purpose of specifying the location at which to save the new file. When the drag and drop operation is complete, it tells the source where it wants the files saved and the dragging source creates the files. This special type of file drag is called a promise because the drag operation contains a promise from the source to the destination that the source will create the specified files if the drag and drop is accepted.

To drag and drop promised files, call the RectControl or Window's AcceptRawDataDrop method with "phfs" as the data type. When an application that sends file promises is dropped on your application, you will find a FolderItem in the dragged item. Usually this will point to a file in the TemporaryItems folder and you should ordinarily delete it when you are finished with it.

## Examples

This example shows how to implement dragging the picture in a [Canvas](#) control. This example uses the [Me](#) keyword to refer to the [Canvas](#) control.

```
Function MouseDown(X As Integer, Y As Integer) As Boolean
  Dim d as DragItem
  d=NewDragItem(Me.left,Me.top,Me.width,Me.height)
  d.picture=Me.Backdrop
  d.Drag //Allow the drag
```

See also the example for [ImageWell](#), which illustrates drag and drop between two ImageWells and PICT files or clippings on the desktop and the [ListBox](#) control, which has an example of drag and drop between two ListBoxes.

The following example implements a ‘mover’ interface in which a user can drag rows from ListBox1 to ListBox2 (and in the reverse direction), but neither ListBox accepts dragged text from other controls or from outside the REALbasic application. It uses PrivateRawData since it also prevents text from being dragged outside the application.

To permit drag and drop in both directions, the two [ListBoxes](#) are set up in an identical manner.

Each [ListBox](#) has its EnableDrag property set to [True](#). The DragRow event handler is:

```
Function DragRow (drag as DragItem, row as Integer) As Boolean
    Drag.PrivateRawData("text")=Me.List(Row)+EndOfLine //get the text
    Me.RemoveRow(Row)
    Return True //allow the drag
```

To enable each [ListBox](#) to accept the dragged text, the following statement appears in its Open event handler:

```
Me.AcceptRawDataDrop("text")
```

The DropObject event handler is as follows:

```
Sub DropObject (obj as DragItem)
    If obj.RawDataAvailable("text") then
        Me.AddRow(obj.RawData("text"))
    end if
```

The following example allows the user to drag several text files from the desktop to an [EditField](#). The example places the contents of all the text files in the [EditField](#), appending each file’s contents to the Text property of the [EditField](#).

To support several files, the NextItem function is used to determine whether there are any more files remaining to be dropped.

```
Dim textStream as TextInputStream
If obj.folderItemAvailable then
    Do
        textStream=obj.FolderItem.Openastextfile
        Me.text=Me.text+textstream.readall+Chr(13)
    loop until not obj.NextItem
End if
```

The [EditField](#) has the line:

```
Me.AcceptFileDrop("text")
```

in its Open event handler, where “text” is as a File Type of files of Macintosh type TEXT.

**See Also** [Canvas](#), [EditField](#), [ImageWell](#), [ListBox](#) controls; [Picture](#), [FileType](#), [FolderItem](#) classes.

## EasyTCPSocket Class

Communicates with remote REALbasic applications with a proprietary protocol that is implemented over TCP. It is provided so that a user who is new to TCP/IP communications can use a higher-level class than the [TCPSocket](#) class to network their applications.

**Super** [TCPSocket](#)

### Events

| Name            | Parameters   | Description  |
|-----------------|--|--|
| Error           | Code as <a href="#">Integer</a>  | An error occurred. The <i>Code</i> parameter will contain an error code that will correspond to one of the class constants for the <a href="#">SocketCore</a> class. |
| ReceivedMessage | Command as <a href="#">Integer</a> ,<br>Data as <a href="#">String</a> | A message, consisting of an identifying integer, <i>Command</i> , and the text <i>Message</i> , has been received.   |

The DataAvailable event of the [TCPSocket](#) class is not available.

### Methods

| Name              | Parameters   | Description   |
|-------------------|--|---|
| SendMessage       | Command as <a href="#">Integer</a> , Data as <a href="#">String</a>      | Sends a message, consisting of the code <i>Command</i> and the text, <i>Message</i> .   |
| WaitForConnection | Timeout as <a href="#">Integer</a>                                       | Wait for a connection until <i>Timeout</i> elapses. <i>Timeout</i> is in seconds. Returns <a href="#">True</a> if a connection is made; otherwise it returns <a href="#">False</a> .  |
| WaitForMessage    | Command as <a href="#">Integer</a><br>Timeout as <a href="#">Integer</a> | Wait for a message identified by the value of <i>Command</i> . Returns a <a href="#">String</a> , containing the message text. <i>Timeout</i> is the amount of time in seconds that the socket will wait for a message before stopping. |

Even though you have access to the Write, Read and ReadAll methods of the [TCPSocket](#) class, you should **never** call them. Doing so will cause a [RuntimeException](#) to be raised (with an appropriate message set). This is so the internal protocol is enforced.

**Class Constants** These constants indicate which error occurred. They are the same error codes returned by [SocketCore](#). When an Error event occurs, check the value of the passed parameter *Code* against these class constants.

| Error Code | Class Constant      |
|------------|---------------------|
| 100        | OpenDriverError     |
| 102        | LostConnection      |
| 103        | NameResolutionError |
| 105        | AddressInUseError   |
| 106        | InvalidStateError   |
| 107        | InvalidPortError    |
| 108        | OutOfMemoryError    |

### Notes

If you are subclassing a **EasyTCPSocket**, you must call the Super class's constructor in your own subclass's constructor. The subclass will not work or exhibit 'strange' behavior unless this is done.

The *Command* parameter can be used as a code to identify the type of data that is sent. For example, you could send a command ID of 100 to mean that the string data is actually a memory block containing a [FolderItem](#). Or, you could define ID 101 as the the username of a remote application. This message mode is enforced on you in that you cannot use an arbitrary Write command. If you'd like to send arbitrary data, then you can just make up a miscellaneous command ID and send your arbitrary data.

Command ID's less than 0 are reserved for internal use. When you are sending messages, you should not use a command ID less than 0, as it may very well cause issues with other classes.

The **EasyTCPSocket** class allows you to establish connections and communicate via the TCP protocol with a remote machine. The main difference between the **EasyTCPSocket** class and the regular [TCPSocket](#) class is the message-based aspects of the protocol. The connection process is identical to a regular [TCPSocket](#). However, when you want to send data to a remote machine you must use the *SendMessage* method to do so.

When you receive data, you are not given a *DataAvailable* event, as is you are for [TCPSocket](#). Instead, the entire message is passed to you via the *ReceivedMessage* event. Because of this, we do not allow you to read in arbitrary data using the *Read* or *ReadAll* methods of the [TCPSocket](#) class. When you receive a message sent via this protocol, you handle it via the *ReceivedMessage* event handler.

For synchronous communications, there is a *WaitForConnection* method which will synchronously wait a predetermined amount of time for a connection to be established. If the connection is made, it returns [True](#), otherwise, it returns [False](#).

REALbasic calls the *DoEvents* method of the [Application](#) class internally so that your application's user interface will remain responsive during this call. For synchronous

communications, use the `WaitForMessage` method. This method waits for a message to come in with the command ID you specify. Once that message comes in, it will return the string data portion of that message. If a message comes in with a command ID that is different from the one you are expecting, it will drop that message. This method also internally calls the `DoEvents` method of the [Application](#) class, so your user interface will stay responsive.

The **EasyTCPSocket** class is designed only for easy communication among REALbasic applications on the network. It isn't designed to be the basis for custom TCP-based communication protocols. It works only for other applications that implement the **EasyTCPSocket** protocol.

**Examples** Connecting to the network. This example uses an EasyTCPSocket named "Connector".

```
Connector.Address = "localhost"  
Connector.Port = 1270  
Connector.Connect
```

Listening for a message. This example uses an EasyTCPSocket named "Listener".

```
Listener.Port = 1270  
Listener.Listen
```

Sending a message. This example uses the EasyTCPSocket named "Connector". EditField2 contains the value of the *Command* parameter and EditField3 the text of the message.

```
Connector.SendMessage(Val(EditField2.Text), EditField3.Text)
```

**See Also** [AutoDiscovery](#), [EasyUDPSocket](#), [TCPSocket](#) classes.

## EasyUDPSocket Class

Supports communication via the UDP protocol using unicasting, multicasting, or broadcasting. It provides a simple 'wrapper' around the [UDPSocket](#) class that makes easier for beginners to get started with UDP networking.

**Super Class** [UDPSocket](#)

## Events

| Name            | Parameters   | Description  |
|-----------------|--|--|
| Error           | Code as <a href="#">Integer</a>  | An error occurred. This event replaces the <a href="#">TCPsocket</a> error event. It functions the same, but instead this event passes the error code so you don't have to check the LastErrorCode property. |
| ReceivedMessage | FromIP as <a href="#">String</a><br>Command as <a href="#">Integer</a><br>Data as <a href="#">String</a> | A new complete message, consisting of an identifying integer, <i>Command</i> , and the text <i>Message</i> , has been received from the computer with <i>FromIP</i> IP address.                              |

The DataAvailable event of the [UDPSocket](#) class is not available.

## Methods

| Name                    | Parameters  | Description   |
|-------------------------|---|---|
| Bind                    | Port as <a href="#">Integer</a>   | Binds the socket to the specified port.   |
| Register                | GroupName as <a href="#">String</a>   | Registers <i>GroupName</i> as an identifier for messages sent by SendMessageToGroup. <i>GroupName</i> can be any string. REALbasic will map it to a valid Class D IP address. Class D IP addresses are in the range: [224.0.0.0 to 239.255.255.255]. For example, you can call Register("My Cool App"), REALbasic will hash that to a valid Class D IP address. However, if you use a valid Class D IP address, we will use that without modifications. |
| SendMessageToGroup      | GroupName as <a href="#">String</a> ,<br>Command as <a href="#">Integer</a> ,<br>Data as <a href="#">String</a> | Sends message consisting of <i>Command</i> and <i>Data</i> to the group specified by <i>GroupName</i> .   |
| SendMessageToIndividual | IPAddress as <a href="#">String</a> ,<br>Command as <a href="#">Integer</a> ,<br>Data as <a href="#">String</a> | Sends the message consisting of <i>Command</i> and <i>Data</i> to the machine having the IP address specified by <i>IPAddress</i> .   |
| Unregister              | GroupName as <a href="#">String</a>   | Unregisters the <i>GroupName</i> passed.  |

Even though you have access to the Write and Read methods of the [UDPSocket](#) class, you should **never** call them. Doing so will cause a [RuntimeException](#) to be raised (with an appropriate message set). This is so the internal protocol is enforced.

**Notes**

If you are subclassing a **EasyUDPSocket**, you must call the Super class's constructor in your own subclass's constructor. The subclass will not work or exhibit 'strange' behavior unless this is done.

The **EasyUDPSocket** class provides a simple 'wrapper' around the [UDPSocket](#) class that makes easier for beginners to get started with UDP networking. With the **EasyUDPSocket** class, you can send messages either to an individual or to an entire group. One main benefit to the **EasyUDPSocket** class is the way it handles multicasting. It's a nuisance to look up which IP addresses can be used with multicasting.

The *Command* parameter can be used as a code to identify the type of data that is sent. For example, you could send a command ID of 100 to mean that the data is actually a memory block containing a FolderItem. Or, you could define ID 101 as the username of a remote application. This message mode is enforced on you in that you cannot use an arbitrary Write command. If you'd like to send arbitrary data, then you can just make up a miscellaneous command ID and send your arbitrary data.

Command ID's less than 0 are reserved for internal use. When you are sending messages, you should not use a command ID less than 0, as it may very well cause issues with other classes.

To send a message to a group, you pass the group name that you have previously assigned to the group using the Register method. This means that you can pass "MyLocalGroup" as the parameter to the Register method and use that string for calls to SendMessageToGroup. To send a message to an individual, you need the IP address of the user's computer in the SendMessageToIndividual.

You can still pass a valid Class D IP address; REALbasic will use it. One other helper function is the Bind function. This sets up the socket for you properly (so you don't have to set RouterHops or SendToSelf up yourself) and does the bind on the port specified.

One thing to keep in mind is that **EasyUDPSocket** defaults to having SendToSelf on. You are welcome to override this default yourself by setting SendToSelf to [False](#) after the making the Bind call. While you can still call {Join/Leave}MulticastGroup, we suggest that you only use Register and Unregister for your **EasyUDPSocket** application.

**Examples**

Binding to a port. EditField1 contains the user-specified port number:

```
EasyUDPSocket1.Bind(Val\(EditField1.Text\)\)
```

Registering in a group. Everyone on the network should use the same name.

```
EasyUDPSocket1.Register("MyGroup")
```

Sending a message to the group. EditField1 is the text of the message.

```
EasyUDPSocket1.SendMessageToGroup("MyGroup",100,EditField1.Text)
```

Sending a message to an individual in the group. The individual's IP address and port is passed as the first parameter. The second parameter is the value of the Command parameter and, EditField1 contains the text of the message.

```
Dim port as String  
port = Str(EasyUDPSocket1.Port)  
EasyUDPSocket2.SendMessageToIndividual("192.168.1.152" + ":" + port _  
,100, EditField1.Text )
```

Receiving a message in the ReceivedMessage event:

```
Sub ReceivedMessage (fromIP as String, Command as Integer, Data as String)  
MsgBox "Received " + Str(command) + ":" + data + " from " + fromIP _  
+EndOfLine
```

**See Also** [AutoDiscovery](#), [SocketCore](#), [UDPSocket](#) classes.

---

## Edit**able**Movie Class

Used to create, manipulate, and save QuickTime movies.

**Super Class** [Movie](#)

Since **EditableMovie** is subclassed from [Movie](#), you can view an **EditableMovie** using the [MoviePlayer](#) control.

### Properties

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| Duration    | <a href="#">Double</a>  | Duration of the movie (seconds)  |
| EOF         | <a href="#">Boolean</a> | <a href="#">True</a> if the Position is at the end of the movie.   |
| Handle      | <a href="#">Integer</a> | Returns a handle to the <b>EditableMovie</b> , mainly used for API calls that require the handle to a movie.   |
| Picture     | <a href="#">Picture</a> | Picture of the movie at the current position.  |
| Position    | <a href="#">Double</a>  | Current position of the movie (seconds).   |
| Poster      | <a href="#">Picture</a> | Poster frame of the movie.   |
| SaveOnClose | <a href="#">Boolean</a> | If <a href="#">True</a> , the <b>EditableMovie</b> will update and save the file when it goes out of scope (as in prior versions of REALbasic). If <a href="#">False</a> , no updates to the <b>EditableMovie</b> 's file will occur unless you call the <a href="#">CommitChanges</a> method. |
| SelLength   | <a href="#">Double</a>  | The length (seconds) of the selected segment of the <b>EditableMovie</b> .   |

| Name                | Type                       | Description  |
|---------------------|----------------------------|--|
| SelStart            | <a href="#">Double</a>     | The starting position (seconds) of the selected segment of the EditableMovie   |
| <b>TimeDuration</b> | <a href="#">Integer</a>    | Duration of the movie, in TimeScale units.   |
| TimeScale           | <a href="#">Integer</a>    | Defines the timescale of the movie, in 1/TimeScale units per second.   |
| TimeValue           | <a href="#">Integer</a>    | Current position of the movie, expressed in TimeScale units.   |
| <b>TrackCount</b>   | <a href="#">Integer</a>    | Number of tracks in the movie.   |
| <b>UserData</b>     | <a href="#">QTUserData</a> | UserData contains additional information about the movie, including credits, copyright information, etc. The <a href="#">QTUserData</a> class is used to manage the individual pieces of data. |

## Methods

| Name               | Parameters  | Description   |
|--------------------|---|---|
| AppendMovieSegment | SourceMovie As <a href="#">Movie</a> ,<br>SourcePosition As <a href="#">Double</a><br>SourceDuration As <a href="#">Double</a> ,<br>ShowProgress As <a href="#">Boolean</a> | Appends <i>SourceMovie</i> to the end of the calling <b>EditabileMovie</b> . <i>SourcePosition</i> is the beginning (in seconds) of the segment in <i>SourceMovie</i> to be appended to the calling movie. <i>SourceDuration</i> is the length (in seconds) in <i>SourceMovie</i> of the segment to be appended. If the operation is time-consuming, QuickTime will display a modal dialog with a progress indicator if <i>ShowProgress</i> is <a href="#">True</a> . |
| Clear              |   | Clear the <b>EditabileMovie</b> 's current selection. Use the <b>EditabileMovie</b> 's SelStart and SelLength properties to set the selection.  |
| CommitChanges      |   | Returns a <a href="#">Boolean</a> . It returns <a href="#">True</a> if the commit is successful. Commits changes to the <b>EditabileMovie</b> manually. You can also do this using the class's destructor. And, if you set SaveOnClose to <a href="#">True</a> , it will save changes automatically when the <b>EditabileMovie</b> goes out of scope.   |
| Copy               |   | Copies the <b>EditabileMovie</b> 's current selection to the Clipboard. Use the <b>EditabileMovie</b> 's SelStart and SelLength properties to set the selection.  |

## EditableMovie Class

---

| Name               | Parameters  | Description   |
|--------------------|---|---|
| Cut                |   | Cuts the <b>EditableMovie</b> 's selection to the Clipboard. Use the <b>EditableMovie</b> 's SelStart and SelLength properties to set the selection.  |
| <b>ContentSize</b> | StartTime as <a href="#">Integer</a><br>DurationTime as <a href="#">Integer</a>   | Returns an <a href="#">Integer</a> , the number of bytes in the segment specified by StartTime and DurationTime.  |
| Flatten            | Destination As <a href="#">FolderItem</a> ,<br>ResourceFork As <a href="#">Boolean</a> , FastStart as <a href="#">Boolean</a> , ShowProgress As <a href="#">Boolean</a>   | Flattens the <b>EditableMovie</b> and saves it in <i>Destination</i> . Windows will always flatten the data fork. This parameter is ignored in Windows builds. Macintosh users can choose to flatten the ResourceFork by setting ResourceFork to <a href="#">True</a> ; Set <i>FastStart</i> to <a href="#">True</a> to enable QuickTime's "FastStart" feature for internet streaming. If the operation is time-consuming, QuickTime will display a modal dialog with a progress indicator if <i>ShowProgress</i> is <a href="#">True</a> . |
| InsertMovieSegment | sourceMovie As <a href="#">Movie</a> ,<br>sourcePosition As <a href="#">Double</a> , sourceDuration As <a href="#">Double</a> ,<br>destinationPosition As <a href="#">Double</a> ,<br>showProgress As <a href="#">Boolean</a> | Inserts a segment from <i>SourceMovie</i> at the position specified by <i>DestinationPosition</i> . The segment in <i>SourceMovie</i> to be inserted begins at <i>SourcePosition</i> and continuing by <i>SourceDuration</i> . <i>SourcePosition</i> , <i>SourceDuration</i> , and <i>DestinationPosition</i> are in seconds. If the operation is time-consuming, QuickTime will display a modal dialog with a progress indicator if <i>ShowProgress</i> is <a href="#">True</a> .  |
| NewSoundTrack      |   | Creates and returns a <a href="#">QTSoundTrack</a> . You cannot create a new <a href="#">QTSoundTrack</a> with the <a href="#">New</a> function.  |
| NewVideoTrack      | Width as <a href="#">Integer</a><br>Height as <a href="#">Integer</a><br>TimeScale as <a href="#">Integer</a>   | Width and Height are in pixels. TimeScale is in frames per second. Creates a new video track and returns a <a href="#">QTVideotrack</a> .   |

| Name                 | Parameters   | Description   |
|----------------------|--|---|
| Paste                |  | Pastes the contents of the Clipboard (from calling Cut or Copy) to the calling <b>EditableMovie</b> at the SelStart position. Use the SelStart and SelLength properties to set the selection. If SelLength is greater than 0, the contents of the Clipboard will overwrite the selected contents of the <b>EditableMovie</b> .  |
| RestoreRedoEditState |  | Restores the <b>EditableMovie</b> to the state it was in when SetRedoEditState was called.  |
| RestoreUndoEditState |  | Restores the <b>EditableMovie</b> to the state it was in when SetUndoEditState was called.  |
| ScaleMovieSegment    | StartPosition As <a href="#">Double</a> ,<br>OldDuration As <a href="#">Double</a> ,<br>NewDuration As <a href="#">Double</a> ,<br>ShowProgress As <a href="#">Boolean</a> | Change the duration of the selected segment in the <b>EditableMovie</b> . Note: A SelStart of 0 and a SelLength of myEditableMovie.Duration can be used to scale the entire movie. A change in duration will result in a pitch change in the movie as well. For example, making the movie longer causes the pitch to be lowered.<br><br>StartPosition is the beginning of the segment to be scaled.<br>OldDuration is the current duration of the segment and NewDuration is the duration the segment should be scaled to.<br>If the operation is time-consuming, QuickTime will display a modal dialog with a progress indicator if ShowProgress is <a href="#">True</a> . |
| SetRedoEditState     |  | Use this method to store an edit state for the <b>EditableMovie</b> . The <b>EditableMovie</b> can be returned to this state at anytime by calling RestoreRedoEditState, provided the <b>EditableMovie</b> hasn't gone out of scope.  |

## EditableMovie Class

| Name             | Parameters                       | Description   |
|------------------|----------------------------------|---|
| SetUndoEditState |                                  | Use this method to store an edit state for the EditableMovie. The EditableMovie can be returned to this state at anytime by calling EditableMovie.RestoreUndoEditState provided the EditableMovie hasn't gone out of scope. |
| SoundTrack       | Index as <a href="#">Integer</a> | Retrieves the <a href="#">QTSoundTrack</a> corresponding to the index that was passed. Index is 1-based array. Returns a <a href="#">QTSoundTrack</a> .   |
| Track            | Index as <a href="#">Integer</a> | Index is 1-based array. Returns a <a href="#">OTTrack</a> .   |
| VideoTrack       | Index as <a href="#">Integer</a> | Retrieves the video track corresponding to the index that was passed. Index is 1-based array. Returns a <a href="#">OTVideoTrack</a> .  |

### Notes

The TimeScale parameter defines the number of time units that pass each second. For example, a TimeScale value of 60 sets the timescale in sixtieths of a second. The TimeValue and TimeDuration parameters are expressed in TimeScale units.

The Position or the TimeValue properties can be used to position the movie; at that point, the Picture property contains the image at that point in the movie.

### Examples

The following [PushButton](#) Action opens an existing movie as an **EditableMovie**. It updates the [ImageWell](#) and [EditField](#) with the current picture and position of the movie.

```
Dim f as FolderItem
Dim m as EditableMovie
f=GetFolderItem("Honey.Mov")
m=f.OpenEditableMovie
m.position=movieplayer1.position
ImageWell1.image=m.picture
ImageWell2.image=m.poster
EditField1.text=str(m.position)
```

The following example opens an existing movie as an **EditMovie** and displays it in a MoviePlayer control named ThePlayer.

```
Dim f As FolderItem
Dim theEMovie as EditMovie
f=GetOpenFolderItem("video/quicktime") //defined as a file type
If f <> Nil and f.exists then
    theEMovie=f.OpenEditableMovie
    If theEMovie <> Nil then
        ThePlayer.movie=theEMovie
    end if
end if
```

The following example creates a new video track from two pictures, p1 and p2, that have been dragged into the Project Editor.

```
Dim sequence as QTEffectSequence
Dim i,effectID as Integer
Dim theEffect as QTEffect
Dim track as QTVideoTrack
Dim m as EditMovie
Dim f as FolderItem
f=GetFolderItem("Movie")
m=f.CreateMovie
If radiobutton1.value then
    theEffect=GetOTCrossFadeEffect
else
    theEffect=GetOTSMPTEffect(Val)(EffectsList.cell(EffectsList.ListIndex,1))
end if
sequence=New QTEffectSequence(theEffect,p1,p2,96)
track=m.NewVideoTrack(p1.width, p1.height, 32)
```

**See Also** [FolderItem](#), [QTTrack](#), [QTUserData](#), [QTVideoTrack](#) classes.

## EditField Control

The standard editable text field. An **EditField** control may contain one line of text or multiple lines of text, depending on the value of the MultiLine property. Text may either be of only one font, font size, and style or mixed fonts, font sizes, and styles ("Styled Text") depending on the value of the Styled property. Styled text can be printed using the [StyledTextPrinter](#) class. Styled text can also be managed independently of an **EditField** using the [StyledText](#) class.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name              | Type                        | Description   |
|-------------------|-----------------------------|---|
| <b>AcceptTabs</b> | <a href="#">Boolean</a>     | If <a href="#">True</a> , pressing the Tab key will enter a tab into the EditField instead of moving the focus to the next item in the tab order.   |
| <b>Alignment</b>  | <a href="#">Integer</a>     | Sets paragraph alignment for entire contents of EditField. Works for single-line EditFields (Carbon) and multi-line EditFields. You can use either the integer values or the class constants, AlignDefault, AlignLeft, AlignCenter, or AlignRight, to set or get the alignment:<br>AlignDefault (0): Default alignment<br>AlignLeft (1): Left aligned<br>AlignCenter (2): Centered<br>AlignRight (3): Right aligned<br>Use SelAlignment to set paragraph alignments for individual paragraphs.<br>Currently the Default alignment is the same as Left aligned.<br>(Setting the alignment on Linux requires GTK+ 2.4 or greater) |
| <b>BackColor</b>  | <a href="#">Color</a>       | The color of the background (white by default).   |
| <b>Bold</b>       | <a href="#">Boolean</a>     | Applies the bold style to the text.   |
| <b>Border</b>     | <a href="#">Boolean</a>     | If <a href="#">True</a> , draws a border around the object.   |
| <b>DataField</b>  | <a href="#">String</a>      | Relevant only if the EditField is used in conjunction with a <a href="#">DataControl</a> to display the contents of fields in a database table. It is assigned the name of a field ( <a href="#">String</a> ) in the table referenced by the <a href="#">DataControl</a> in the window.   |
| <b>DataSource</b> | <a href="#">DataControl</a> | The <a href="#">DataControl</a> that references a table in a database whose fields are displayed using EditFields ( <i>TableName</i> as <a href="#">String</a> ). If assigned in code, you use the <i>Name</i> property of a <a href="#">DataControl</a> . Use the <i>DataField</i> property to link each field to be displayed to an EditField. In the IDE, a popup menu of field names will appear.   |
| <b>Format</b>     | <a href="#">String</a>      | Formats the contents of the EditField when the EditField loses the focus. It uses the same formatting conventions as the <a href="#">Format</a> function. To turn off formatting, set the Format property to the empty string, "".  |
| <b>Italic</b>     | <a href="#">Boolean</a>     | Applies the italic style to the text.   |

| Name       | Type                    | Description   |
|------------|-------------------------|---|
| LimitText  | <a href="#">Integer</a> | The maximum number of characters allowed in the EditField. The value of zero does not limit text. Works for normal text entry, copy and paste, and drag and drop, but not for text entered programmatically. Applies only if MultiLine is <a href="#">False</a> .   |
| LiveUpdate | <a href="#">Boolean</a> | If <a href="#">True</a> , then the EditField will dynamically update the bound data value. Relevant when the EditField is bound to another control.   |
| Mask       | <a href="#">String</a>  | <p>Entry filter to be used during data entry. Formats the entered value and/or filters out unacceptable values on a character-by-character basis. Uses the same mask characters as Visual Basic. See the section on Input Entry Filters in the Notes section for descriptions of the filter and formatting characters.</p> <p>If the user enters a character that is prohibited by the mask, it causes the ValidationError event to occur.</p> <p>The EditField will autocomplete any literal mask characters for the user. If the user uses the Backspace key he/she can backup the autocompletion but only 1 character at a time.</p> <p>To turn off the Mask, set it to the empty string, " ". You can set the Mask property in the Properties pane. If you use the "#" character as the first character of the mask, there is the remote possibility that it may be confused with the name of a constant. When you use a constant in the Properties pane, you precede its name with the "#" sign. If you want both a constant and a mask to be named "Example", then you should create a new constant named "poundExample" with its value set to "#Example" and assign that in the Properties pane.</p> |
| MultiLine  | <a href="#">Boolean</a> | <p>If <a href="#">True</a>, wrap text to multiple lines and allow more than 32K of text on all platforms. Page Up, Page Down, Home and End keys are supported for Multi-line EditFields. A vertical scroll bar is added automatically when MultiLine is <a href="#">True</a>.</p> <p>MultiLine must be set to <a href="#">True</a> to support styled text and automatic drag and drop. MultiLine EditFields support any standard terminator (Return, Linefeed, or Return + Linefeed) to indicate the end of a paragraph.</p> <p>If MultiLine is off, EditFields have a limit of 32K of text on Mac OS "classic" builds.</p>   |

| Name                | Type                    | Description  |
|---------------------|-------------------------|--|
| Password            | <a href="#">Boolean</a> | If <a href="#">True</a> , bullets appear in field instead of characters typed. The MultiLine property must be <a href="#">False</a> to use the EditField as a Password field. The Cut and Copy Edit menu items are automatically disabled. On Mac OS X 10.3 and above, the Password property enables and disables secure input.  |
| ReadOnly            | <a href="#">Boolean</a> | If <a href="#">True</a> , the text cannot be modified.   |
| ScrollbarHorizontal | <a href="#">Boolean</a> | If <a href="#">True</a> , the EditField has a horizontal scrollbar and text does not wrap at the right edge of the EditField.  |
| ScrollbarVertical   | <a href="#">Boolean</a> | If <a href="#">True</a> , the EditField includes a vertical scrollbar. Called "Scrollbar" in earlier releases of REALbasic.  |
| ScrollPosition      | <a href="#">Integer</a> | Zero-based index of the topmost visible line. Read ScrollPosition to determine the first visible line; write to ScrollPosition to scroll the EditField. Setting ScrollPosition to the last line or beyond the last line will simply move the vertical scrollbar's thumb to the bottom, not necessarily displaying the last line at the top of the EditField. ScrollPosition is 0 if the MultiLine property is <a href="#">False</a> . Can be set in the IDE. |
| ScrollPositionX     | <a href="#">Integer</a> | Scrolls the EditField horizontally to a position X pixels from the left edge of the EditField.   |
| SelAlignment        | <a href="#">Integer</a> | Controls paragraph alignment of the selected text. Class constants and values are shown below<br>AlignDefault(0): Default alignment<br>AlignLeft(1): Left aligned<br>AlignCenter (2): Centered<br>AlignRight (3): Right aligned<br>-1: Mixed—Selection spans multiple paragraphs with different alignments.<br>Currently, the Default alignment is the same as Left alignment.   |
| SelBold             | <a href="#">Boolean</a> | Used to get and set the bold style of the highlighted text.  |
| SelCondense         | <a href="#">Boolean</a> | Used to get and set the condensed style of the highlighted text. Mac OS "classic" only.  |
| SelExtend           | <a href="#">Boolean</a> | Used to get and set the extended style of the highlighted text. Mac OS "classic" only.   |
| SelItalic           | <a href="#">Boolean</a> | Used to get and set the italic style of the highlighted text.  |
| SelLength           | <a href="#">Integer</a> | The number of highlighted characters. A SelLength of 0 means an insertion point rather than a selection. A value greater than the number of characters in the EditField means that the selection is from SelStart to the end of the EditField.   |

| Name         | Type                       | Description  |
|--------------|----------------------------|--|
| SelOutline   | <a href="#">Boolean</a>    | Used to get and set the outline style of the highlighted text. Mac OS "classic" only.  |
| SelShadow    | <a href="#">Boolean</a>    | Used to get and set the shadow style of the highlighted text. Mac OS "classic" only.   |
| SelStart     | <a href="#">Integer</a>    | The position of the insertion point. A value of zero means that the insertion point is before the first character. Use SelStart to set the position of the insertion point in the text. Use SelStart in conjunction with SelLength to select a portion of the text in the EditField, beginning with <i>SelStart</i> and extending for <i>SelLength</i> characters. |
| SelText      | <a href="#">String</a>     | The currently highlighted text.  |
| SelTextColor | <a href="#">Color</a>      | Used to get and set the text color of the highlighted text. When used in an unstyled EditField, it changes the color of all the text in the EditField.   |
| SelTextFont  | <a href="#">String</a>     | Used to get and set the text font of the highlighted text. When used in an unstyled EditField, it changes the font of all the text in the EditField.   |
| SelTextSize  | <a href="#">Integer</a>    | Used to get and set the font size of the highlighted text. When used in an unstyled EditField it changes the font size of all the text in the EditField.   |
| SelUnderline | <a href="#">Boolean</a>    | Used to get and set the underline style of the highlighted text.   |
| Styled       | <a href="#">Boolean</a>    | If <a href="#">True</a> , styled text is allowed. The MultiLine property must also be set to <a href="#">True</a> to permit styled text. If Styled is <a href="#">False</a> , then the EditField cannot have a <a href="#">StyledText</a> object.  |
| StyledText   | <a href="#">StyledText</a> | Enables you to access the properties and methods of the <a href="#">StyledText</a> class for the text in the EditField. The EditField must have its MultiLine and Styled properties set to <a href="#">True</a> . See the example for the <a href="#">StyledText</a> class.  |

## EditField Control

---

| Name          | Type                    | Description   |
|---------------|-------------------------|---|
| Text          | <a href="#">String</a>  | The text displayed. Assigning a <a href="#">string</a> to the Text property replaces the entire contents of the EditField. The font, size, and color are uniform and match the values last set with the TextFont, TextSize, and TextColor properties, or if these haven't been used, the settings in the IDE. If the EditField contains more than one fontsize, style, or font, you will lose these text attributes when assigning a new string to the Text property. You should instead use the SelText, SelStart, and SelLength to make changes to substrings within a styled EditField. (Using SelTextFont, etc. may change the entire appearance of the text — always true if the field is not styled — but does not change the "default" settings used when the entire text is replaced with the Text property.) |
| TextColor     | <a href="#">Color</a>   | The color of the text (black by default).   |
| TextFont      | <a href="#">String</a>  | Name of the font used to display the text. Assigning another font to TextFont changes the font of all the text in the EditField. It also resets the default which will be used if the Text property is subsequently used to reset the contents of the text in the EditField.  |
| TextSize      | <a href="#">Integer</a> | Size of the font used to display the text.  |
| TextStyleData | <a href="#">String</a>  | The style data for Styled EditField. This data can be stored in a 'styl' resource and can be set with the SetTextAndStyle method.   |
| Underline     | <a href="#">Boolean</a> | Applies the underline style to the text.  |
| UseFocusRing  | <a href="#">Boolean</a> | If <a href="#">True</a> , the control indicates that it has the focus with a ring around its border; if <a href="#">False</a> , the appearance of the control does not change when it has the focus.  |

## Events

| Name      | Parameters                       | Description  |
|-----------|----------------------------------|--|
| GotFocus  |                                  | The cursor has moved into the EditField.   |
| KeyDown   | Key as<br><a href="#">String</a> | The user has pressed the Key passed while the EditField has the focus. Returns a <a href="#">Boolean</a> . Returning <a href="#">True</a> means that no further processing is to be done with Key. |
| LostFocus |                                  | The cursor has left the EditField.   |

| Name            | Parameters  | Description  |
|-----------------|---|--|
| MouseDown       | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>                      | The mouse button was pressed inside the control's region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown. Returning <a href="#">True</a> prevents the EditField from handling the mouse click.  |
| MouseUp         | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>                      | The mouse button was released inside the control's region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |
| SelChange       |   | The range of characters highlighted has changed.   |
| TextChange      |   | The text property of the EditField has been changed.   |
| ValidationError | InvalidText as <a href="#">String</a> ,<br>StartPosition as <a href="#">Integer</a> | The user has tried to enter a character that is prohibited by the EditField's Mask property. <i>InvalidText</i> is the entire character string up to and including the invalid text. <i>StartPosition</i> is the starting character position in which the actual invalid text was entered. The first character is numbered 1. If no code is provided in this event and a validation error occurs, REALbasic plays the system beep. |

## Methods

| Name             | Parameters   | Description   |
|------------------|--|---|
| AppendText       | text as <a href="#">String</a>                               | Appends the passed text to the current Text.  |
| BindValue        | value as <a href="#">StringProvider</a>                      | Used in implementing custom object bindings.  |
| CharPosAtLineNum | LineNumber as <a href="#">Integer</a>                        | Returns as an <a href="#">Integer</a> the character position in the Editfield of the first character on the LineNumber line. Characters are numbered consecutively from the start until the end of the EditField. The first character is numbered 1. The first line is numbered zero. |
| CharPosAtXY      | X as <a href="#">Integer</a><br>Y as <a href="#">Integer</a> | Returns as an <a href="#">Integer</a> the character position for pixel coordinates X, Y relative to the EditField. Characters are numbered consecutively from the start until the end of the EditField. The first character is numbered 1.  |
| Copy             |  | Copies the selected text in the EditField to the Clipboard, including the styled text information.  |

| Name                     | Parameters   | Description   |
|--------------------------|--|---|
| LineNumAtCharPos         | CharPosition as <a href="#">Integer</a>                                  | Returns (as an <a href="#">Integer</a> ) the line number in which the CharPosition character appears. Characters are numbered consecutively with the first character numbered 1. The first line is numbered zero. Used with ScrollPosition, this lets you scroll the EditField to a particular place in the text. |
| Paste                    |  | Pastes the styled or unstyled text on the Clipboard into the EditField at the insertion point, adding the text to the existing text.  |
| SetFocus                 |  | Gives the EditField the focus, sending all keydown events to the EditField and moving the cursor there.   |
| SetTextAndStyle          | Text as <a href="#">String</a> , TextStyleData as <a href="#">String</a> | Places the text into the Text property of the EditField and applies the styles to it based on the data in <i>TextStyleData</i> . <i>TextStyleData</i> is compatible with the style resource.  |
| StyledTextPrinter        | g as <a href="#">Graphics</a> , Width as <a href="#">Integer</a>         | Used to create a <a href="#">StyledTextPrinter</a> object for printing the EditField's text property as styled text. The EditField's MultiLine property must be <a href="#">True</a> to support styled text printing. Returns a StyledTextPrinter Object. Width (pixels) is width of printable area.              |
| ToggleSelectionBold      |  | Reverses the bold style of the highlighted text.  |
| ToggleSelectionCondense  |  | Reverses the condensed style of the highlighted text. Mac OS "classic" only.  |
| ToggleSelectionExtend    |  | Reverses the extended style of the highlighted text. Mac OS "classic" only.   |
| ToggleSelectionItalic    |  | Reverses the italic style of the highlighted text.  |
| ToggleSelectionOutline   |  | Reverses the outline style of the highlighted text. Mac OS "classic" only.  |
| ToggleSelectionShadow    |  | Reverses the shadow style of the highlighted text. Mac OS "classic" only.   |
| ToggleSelectionUnderline |  | Reverses the underline style of the highlighted text.   |

## Notes

### Working With Styled Text in EditFields

In order to display styled text in an **EditField**, the **EditField**'s **Styled** property must be checked. The **SelBold**, **SelTextColor**, **SelCondense**, **SelExtend**, **SelItalic**, **SelOutline**, **SelShadow** and **SelUnderline** properties can be used to both set the style of the selected text and get the style of the selected text (The Outline, Shadow, Condense, and Extend styles are available only on Mac OS 8-9). These properties are set automatically by REALbasic at runtime when text is selected in an **EditField**. To set the style, simply assign **True** to the property. For example, if you want to add the bold style to the selected text in **EditField1**, use this syntax:

```
EditField1.SelBold=True
```

You can determine if the selected text is in a particular style using the same property. The following code plays a beep sound if the selected text is bold:

```
If EditField1.SelBold Then
    Beep
End If
```

The **SelBold**, **SelCondense**, **SelExtend**, **SelItalic**, **SelOutline**, **SelShadow** and **SelUnderline** will be **True** if all of the selected characters are in the style being checked. Otherwise, these properties will be **False**.

Getting and setting the selected text font and font size work the same way using the **SelTextFont** and **SelTextSize** properties. To set the font of the selected text, set the **SelTextFont** property to the name of the font. If the selected text uses more than one font, **SelTextFont** will be an empty string. If the selected text has more than one font size, the **SelTextSize** property will be zero.

If the selected text has more than one style, font, or size, you can select individual characters using the **SelStart** and **SelLength** properties to determine which styles, fonts, and sizes are in use. Please note that if an **EditField** contains more than one style, font, or font size and you reassign the value of the **Text** property, you will lose the styled formatting of the text substrings. All the text will then be styled uniformly. To update the text in a styled **EditField**, it is safest to use the **SelStart**, **SelLength**, and **SelText** properties to update text selections.

You can also get and set the color of the selected text using the **SelTextColor** property. For example, the following code checks to see if the selected text is white and if it is, sets it to black:

```
Dim White, Black as Color
White=RGB(255,255,255)
Black=RGB(0,0,0)
If EditField1.SelTextColor=White Then
    EditField1.SelTextColor=Black
End If
```

To print styled text in an **EditField**, use the `StyledTextPrinter` method to create a `StyledTextPrinter` object and then use the `StyledTextPrinter`'s `DrawBlock` method. See the description of `StyledTextPrinter` for an example. There is also an example of styled text printing in the Tutorial lesson on styled text.

- MacOS "classic" Text Styles** The following styles are nonstandard and are supported only on Mac OS "classic": Outline, Shadow, Condensed, and Extended.
- Single-line EditFields** Single-line EditFields on Mac OS "classic" have a limit of 32K of text. They also support character codes above ASCII 127 and the Symbol font.
- Using the StyledText Class** With the `StyledText` class, you can manage styled text independently of an **EditField**. The styled text is contained in the `Text` property of the `StyledText` object and its style attributes can be set via its `Bold`, `Italic`, `Underline`, `Font`, `Size`, and `TextColor` methods. To display the styled text, you can assign the styled text object to the **EditField**'s `StyledText` property. Any programmatic changes to style attributes that you make while the styled text is displayed will update immediately. Since the **EditField** has its own properties for getting and setting style attributes for selected text, those attributes are respected by the `StyledText` object while it is displayed in the **EditField**. For more information, see the `StyledText` class.

- Input Entry Filters** Use the `Mask` property to filter user input on a character-by-character basis and add formatting characters. For example, a mask for a Telephone number field can add parentheses, spaces, and dashes as literals, that are used for formatting, and the digit mask symbol '#' to restrict entry to numbers only.

The following table shows the characters that you can use to define a mask.

| Mask Character | Description  |
|----------------|--|
| #              | Any single digit placeholder. The user can type only a digit character in this position.   |
| .              | Decimal placeholder. The decimal placeholder that is actually used is specified in the user's International settings. The character is treated as a literal (formatting) character for masking purposes. |
| ,              | Thousands separator. The thousands separator that is actually used is specified in the user's International settings. The character is treated as a literal (formatting) character for masking purposes. |
| :              | Time separator. The time separator that is actually used is specified in the user's International settings. The character is treated as a literal (formatting) character for masking purposes.           |
| /              | Date separator. The date separator that is actually used is specified in the user's International settings. The character is treated as a literal (formatting) character for masking purposes.           |

| Mask Character | Description  |
|----------------|--|
| \              | Mask escape character: Treat the next character in the mask as a literal. The escape character enables you to use the '#', '&', 'A', '?' (and so on) characters in the mask. The escaped character is treated as a literal (formatting) character. |
| &              | Character placeholder. Valid values are the ASCII characters 32-126 and the non-ASCII characters 128-255.  |
| >              | Convert all the characters that follow to uppercase. Uppercasing works beyond the ASCII range where appropriate, e.g., ü becomes Ü.  |
| <              | Convert all the characters that follow to lowercase. Lowercasing works beyond the ASCII range where appropriate, e.g., Ü becomes ü.  |
| A              | Alphanumeric character placeholder, where entry is mandatory. For example, the spec "AAA" specifies three alphabetic characters.   |
| a              | Alphanumeric character placeholder, where entry is optional.   |
| 9              | Digit placeholder where entry is optional.   |
| C              | Character or space placeholder, where entry is optional. It operates like the '&' placeholder.   |
| ?              | Alphabetic placeholder.  |
| Any literal    | All other symbols are displayed as literals for formatting purposes.   |
| ~              | Reserved for future use. If you use "~" it will trigger an exception error. Use \~ instead.  |

Here are some examples:

```
EditField1.Mask="###-##-####" //US Social Security number
EditField1.Mask="(###) ###-###" //US Phone number, with area code
```

If the user tries to enter a character that is prohibited by the mask, a `ValidationEvent` event occurs. The character that the user attempted to enter and the character position is passed to the `ValidationEvent` event, where you can handle the keystroke as you like.

To cancel the Mask, set it to the empty string:

```
EditField1.Mask=" "
```

## Adding Text to an EditField

When appending text to an **EditField**, you may notice some flicker as REALbasic redraws the **EditField** to show the new text. This will happen if you appended the `Text` property of the **EditField** like this:

```
EditField1.Text=EditField1.Text+"my new text"
```

This occurs because the entire contents of the **EditField** has to be redrawn. To avoid this flicker, position the cursor at the end of the **EditField** and append the text using the **SelText** property like this:

```
EditField1.SelStart=Len(EditField1.Text)  
EditField1.SelText="my new text"
```

### Text Encoding

EditFields store all text internally in Unicode, which is able to represent a mixture of characters from different writing systems. When you extract the text via the **Text** or **SelText** properties, this text is returned in UTF-8.

### Using Class Constants

The following class constants are available to set paragraph alignments. Set the alignment of the entire contents of the **EditField** by assigning a constant to the **Alignment** property.

| Value | Class Constant |
|-------|----------------|
| 0     | AlignDefault   |
| 1     | AlignLeft      |
| 2     | AlignCenter    |
| 3     | AlignRight     |

For example, the following code in the **Action** event of a control array sets the alignment of the text in an **EditField**. The **Action** event is passed an index parameter that indicates which control was clicked.

```
Sub Action (Index as Integer)  
Select Case Index  
Case 0  
EditField1.Alignment=EditField.AlignDefault  
Case 1  
EditField1.Alignment=EditField.AlignLeft  
Case 2  
EditField1.Alignment=EditField.AlignCenter  
Case 3  
EditField1.Alignment=EditField.AlignRight  
End Select
```

### See Also

[Font](#), [FontCount](#) functions; [Paragraph](#), [Range](#), [StyleRun](#), [StyledText](#), [StyledTextPrinter](#) classes.

---

## Element3D Class

Used as the base class for the 3D objects [Light3D](#), [Group3D](#), [Object3D](#), and [Trimesh](#). These objects are displayed in an [Rb3DSpace](#).

Super Class [Object](#)

## Properties

| Name        | Type                       | Description   |
|-------------|----------------------------|---|
| Bounds      | <a href="#">Bounds3D</a>   | Stores the bounding volume for the <b>Element3D</b> . The default value is <a href="#">Nil</a> . If you assign a <a href="#">Bounds3D</a> here, it will be automatically updated when the object is rendered or when you call <code>UpdateBounds</code> . |
| Orientation | <a href="#">Quaternion</a> | Angle of the object in 3D space. Can also be controlled by the Pitch, Roll, and Yaw methods.  |
| Position    | <a href="#">Vector3D</a>   | The position (i.e., x, y, z coordinates) of the object in 3D space.   |
| Scale       | <a href="#">Double</a>     | Determines the size of the model in 3D space; a scale of 1.0 means that it is drawn at the same size it was when it was created.  |
| ScaleVector | <a href="#">Vector3D</a>   | Used to scale the object by a non-uniform factor. The default X, Y, and Z values are all 1.0  |
| Visible     | <a href="#">Boolean</a>    | If <a href="#">True</a> , the object is visible.  |

## Events

| Name   | Parameters                            | Description  |
|--------|---------------------------------------|--|
| Render | ViewObject as <a href="#">Integer</a> | Allows you to make a subclass that can render itself into the given view (a <code>TQ3DViewobject</code> ). This event is available only when you subclass <b>Element3D</b> directly, not one of its predefined subclasses. |

## Methods

| Name                   | Parameters                         | Description  |
|------------------------|------------------------------------|--|
| ComputeBounds          | type as <a href="#">Integer</a>    | Returns a <a href="#">Bounds3D</a> . May return <a href="#">Nil</a> for invisible elements.  |
| MoveForward            | Distance as <a href="#">Double</a> | Moves the object forward the distance passed.  |
| ObjectToWorldTransform | pt as <a href="#">Vector3D</a>     | Returns a <a href="#">Vector3D</a> . Takes a vector specifying a point relative to the object, and returns the coordinates of that point in the 3D world, taking into account the object's current scale, orientation, and position. |
| Pitch                  | radians as <a href="#">Double</a>  | Rotates the object in the vertical plane.  |
| Roll                   | radians as <a href="#">Double</a>  | Rotates the object clockwise or counterclockwise.  |

## EmailAttachment Class

---

| Name                   | Parameters                        | Description   |
|------------------------|-----------------------------------|---|
| UpdateBounds           |                                   | Updates the <a href="#">Bounds3D</a> object attached to the object via its Bounds property. In the case of a <a href="#">Group3D</a> , this also updates the bounds of any objects contained in the group. This is done automatically to the <a href="#">Rb3DSpace</a> .Objects group when you call Update. |
| WorldToObjectTransform | pt as <a href="#">Vector3D</a>    | Takes a vector specifying a point in the 3D world, and returns the coordinates of that point relative to the object, taking into account the object's current scale, orientation, and position. Returns a <a href="#">Vector3D</a> .  |
| Yaw                    | radians as <a href="#">Double</a> | Rotates the object in the horizontal plane.   |

### Notes

The **Element3D** class represents anything that can go into a [Group3D](#) object. This includes all objects that can be displayed Objects property of the [RB3DSpace](#) control. All such objects have a position, orientation, and scale. The [RB3DSpace](#)'s Camera property is an **Element3D**, rather than an [Object3D](#).

### See Also

[Bounds3D](#), [ColorList](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

---

## EmailAttachment Class

Used to hold attachments to an email.

### Super Class

[Object](#)

### Properties

| Name            | Type                   | Description   |
|-----------------|------------------------|---|
| ContentEncoding | <a href="#">String</a> | Content encoding of the attachment.                           |
| Data            | <a href="#">String</a> | When receiving an attachment, the contents of the attachment. |
| MacCreator      | <a href="#">String</a> | Four-character Macintosh Creator code of the attached file.   |
| MacType         | <a href="#">String</a> | Four-character Macintosh Type code.                           |
| MIMEType        | <a href="#">String</a> | MIME type of the attachment.                                  |
| Name            | <a href="#">String</a> | Name of the attached file.                                    |

**Methods**

| Name         | Parameters                            | Description   |
|--------------|---------------------------------------|---|
| LoadFromFile | File as<br><a href="#">FolderItem</a> | Attaches the passed <i>File</i> .   |
| SaveToFile   | File as<br><a href="#">FolderItem</a> | Saves the attachment to <i>File</i> . Returns a <a href="#">Boolean</a> , <a href="#">True</a> if successful. |

**Example**

The following example takes a path to a file and adds it as an email attachment. The path has been entered into the [EditField](#) named fileFld.

```
Dim file as EmailAttachment
Dim mail as EmailMessage
If fileFld.text <> " " then
    file = New EmailAttachment
    file.loadFromFile GetFolderItem(fileFld.text)
    mail.Attachments.Append file
end if
```

**See Also**

[EmailMessage](#), [EmailHeaders](#), [POP3Socket](#), [POP3SecureSocket](#), [SMTPSocket](#), [SMTPSecureSocket](#), [SocketCore](#), [TCPSocket](#) classes.

## EmailHeaders Class

Used to hold the headers for an email.

**Super Class** [InternetHeaders](#)

Please refer to [InternetHeaders](#) for descriptions of this class's methods.

**Methods**

| Name         | Parameters   | Description   |
|--------------|--|---|
| AppendHeader | Name as<br><a href="#">String</a> ,<br>Value as <a href="#">String</a>   | Appends a new header.   |
| Count        |  | Returns the number of headers as an <a href="#">Integer</a> .         |
| Delete       | Name as <a href="#">String</a><br>OR<br>Name as<br><a href="#">String</a> ,<br>Index as<br><a href="#">Integer</a> | Deletes the specified email header.                                   |
| Name         | Index as<br><a href="#">Integer</a>  | Returns as a <a href="#">String</a> the name of the specified header. |

## EmailMessage Class

---

| Name       | Parameters  | Description  |
|------------|---|--|
| SetHeader  | Name as <a href="#">String</a> , Value as <a href="#">String</a>  | Sets the name and value of a header. Emails that have attachments include a MIME-version header.   |
| Source     |   | Returns a <a href="#">String</a> containing the raw source text of the headers.  |
| Value      | Name as <a href="#">String</a><br>OR<br>Name as <a href="#">String</a> , Index as <a href="#">Integer</a><br>OR<br>Index as <a href="#">Integer</a> | Returns a <a href="#">String</a> containing the value of the specified header.   |
| ValueCount | Name as <a href="#">String</a>  | Returns an <a href="#">Integer</a> . Gets the number of header values that have the name specified by <i>Name</i> . This is included because it is possible to have more than one header with the same name. |

### Example

The following sets the header of an outgoing email.

```
Dim mail as EmailMessage
mail=New EmailMessage
mail.headers.AppendHeader "X-Mailer", "REALbasic SMTPSocket Demo"
```

### See Also

[EmailAttachment](#), [EmailMessage](#), [InternetHeaders](#), [POP3SecureSocket](#), [POP3Socket](#), [SMTPSecureSocket](#), [SMTPSocket](#), [SocketCore](#), [SSLocket](#), [TCPSocket](#) classes.

---

## EmailMessage Class

Used to hold the contents of an email message and its enclosures, if any.

Super Class [Object](#)

### Properties

| Name        | Type                            | Description   |
|-------------|---------------------------------|---|
| Attachments | <a href="#">EmailAttachment</a> | The array of EmailAttachments that are attached to this email message.                              |
| BCCAddress  | <a href="#">String</a>          | Returns an comma-separated list of <a href="#">Strings</a> . Gets the BCC addresses for this email. |

| Name          | Type                         | Description   |
|---------------|------------------------------|---|
| BodyEnriched  | <a href="#">String</a>       | The body text of the email message in enriched format. Some clients such as Mail use this format when sending styled emails.  |
| BodyHTML      | <a href="#">String</a>       | The body text of the email as HTML.   |
| BodyPlainText | <a href="#">String</a>       | The body text of the email as plain text. Clients that do not support HTML or enriched format will display the message as plain text using the contents of this property. |
| CCAddress     | <a href="#">String</a>       | Returns an comma-separated list of <a href="#">Strings</a> . Gets the CC addresses for this email.  |
| FromAddress   | <a href="#">String</a>       | Returns a <a href="#">String</a> containing the address the email is coming from.   |
| FromAddress   | <a href="#">String</a>       | Sets the From address for this email message.   |
| Headers       | <a href="#">EmailHeaders</a> | The headers for this email message.   |
| Source        | <a href="#">String</a>       | Returns a <a href="#">String</a> , containing the raw source of the email message.  |
| Subject       | <a href="#">String</a>       | Returns a <a href="#">String</a> . Gets the subject of the email.   |
| Subject       | <a href="#">String</a>       | Sets the subject of this email.   |
| ToAddress     | <a href="#">String</a>       | Returns an comma-separated list of <a href="#">Strings</a> . Gets the addresses for this email.   |

## Methods

| Name            | Parameters                          | Description                        |
|-----------------|-------------------------------------|------------------------------------|
| AddBCCRecipient | Recipient as <a href="#">String</a> | Adds a BCC recipient to the email. |
| AddCCRecipient  | Recipient as <a href="#">String</a> | Adds a CC recipient to the email.  |
| AddRecipient    | Recipient as <a href="#">String</a> | Adds a recipient to the email.     |

## Example

The following example sets the values of the From Address, Subject, Body, and Headers from the values of the Text properties of EditFields in a window. The EditFields named fromAddressFld, subjectFld, bodyFld, and htmlFld contain this information:

```
Dim mail as =New EmailMessage
mail.fromAddress=fromAddressFld.text
mail.subject=subjectFld.text
mail.bodyPlainText = bodyFld.text
mail.bodyHTML = htmlFld.text
```

See also the examples for the [SMTPSocket](#) and [POP3Socket](#) classes.

## EnableMenuItems Method

---

### See Also

[EmailAttachment](#), [EmailHeaders](#), [HTTPSecureSocket](#), [HTTPSocket](#), [POP3SecureSocket](#), [POP3Socket](#), [SMTPSecureSocket](#), [SMTPSocket](#), [SocketCore](#), [SSLocket](#), [TCPSocket](#) classes.

---

## EnableMenuItems Method

Forces an update of the menu bar and executes the EnableMenuItems event handler.

### Syntax

#### EnableMenuItems

### Notes

In most cases, REALbasic's standard menu bar updating will be sufficient. There are cases, however, when it's not. For example, if the SelChange event handler of an [EditField](#) control causes all of the menu items in a particular menu to be disabled, the menu itself should be disabled. Unfortunately, it won't be because the user has not yet clicked in the menu bar. Under circumstances like this, you can call the **EnableMenuItems** method to force REALbasic to check the condition of the menus and redraw them if necessary.

### See Also

[MenuItem](#) class.

---

## EncodeBase64 Function

Converts a [String](#) to Base64 format.

### Syntax

**result=EncodeBase64(str [,lineWrap])**

| Part            | Type                    | Description   |
|-----------------|-------------------------|---|
| <i>result</i>   | <a href="#">String</a>  | The Base64 representation of <i>str</i> .   |
| <i>str</i>      | <a href="#">String</a>  | The <a href="#">String</a> expression to be converted to Base64 format.   |
| <i>lineWrap</i> | <a href="#">Integer</a> | Optional parameter. Specifies the maximum number of characters per line. The default value is 76, which is the line length specified by the Base-64 Content Transfer Encoding used in MIME. If you specify a value of 0, EncodeBase64 will not insert any linebreaks. |

### Notes

Base64 is used for email file attachments. SMTP was designed around the ASCII encoding, so it assumes that characters below ASCII 32 are control codes and it doesn't expect codes above ASCII 127. **Base64** lets you encode arbitrary binary data into a 64-character alphabet composed of the printable ASCII characters. Three bytes of input become four characters of output. **Base64** is also used in authentication schemes as a way to send an encryption key or hash value through a link that expects text.

The [DecodeBase64](#) function performs the reverse operation, taking a Base64 expression and converting it to the original data.

**Example**

The following example encodes UTF-8 text for transmission without line breaks.

```
Dim encodedText as String
encodedText = EncodingBase64(utf8text, 0)
```

**See Also**

[DecodeBase64](#) function.

## EncodeQuotedPrintable Function

Converts a [String](#) into quoted-printable format.

**Syntax**

**result=EncodeQuotedPrintable(str)**

| Part   | Type                   | Description                            |
|--------|------------------------|--|
| result | <a href="#">String</a> | The result of processing <i>str</i> .  |
| str    | <a href="#">String</a> | The string expression to be converted. |

**Notes**

**EncodeQuotedPrintable** converts unprintable characters (i.e., control characters, returns and tabs) into hexadecimal. It is designed for handling email or usenet messages.

The [DecodeQuotedPrintable](#) function performs the function in reverse.

**See Also**

[DecodeQuotedPrintable](#) function.

## EncodeURLComponent Function

Encodes the components of a URL.

**Syntax**

**result=EncodeURLComponent(str)**

| Part   | Type                   | Description               |
|--------|------------------------|---------------------------|
| result | <a href="#">String</a> | The encoded string.       |
| str    | <a href="#">String</a> | The string to be encoded. |

**Notes**

A valid URL is a series of components that are separated by component separators. They are the “.”, “/”, “;”, “&”, and “?”. EncodeURLComponent works with each component part of the URL. It assumes that any component separators in a component represent text and must be encoded. The reverse operation is done by [DecodeURLComponent](#).

## Encoding Function

---

**Example** Here is an example of how an embedded component separator is encoded.

```
Dim s as String  
s=EncodeURLComponent("www.Bob&Ray.com") //returns "www.bob%26ray.com"
```

**See Also** [DecodeURLComponent](#).

## Encoding Function

---

Returns the text encoding of the passed [String](#).

**Syntax** ***result=Encoding(str)***

OR

***result=str.Encoding***

| Part   | Type                         | Description   |
|--------|------------------------------|---|
| result | <a href="#">TextEncoding</a> | The text encoding of <i>str</i> .   |
| str    | <a href="#">String</a>       | The <a href="#">string</a> whose <a href="#">TextEncoding</a> will be returned. |

### Notes

Use the **Encoding** function to determine the encoding of a [string](#). This is useful for text that you have read in from an external source, for example, with [TextInputStream](#) and you don't know the encoding. For text that is generated within REALbasic, the encoding is known to REALbasic.

If the string's encoding is unknown, **Encoding** returns [Nil](#). Test whether the [TextEncoding](#) object is [Nil](#) or include an [Exception](#) block if there is a chance the string's encoding would not be known at runtime.

### Example

The following example gets the encoding of the text stored in the local variable, *s*. The [Exception](#) block handles the call to one of the [TextEncoding](#) object's properties if the encoding cannot be determined or the file cannot be opened.

```
Dim f As FolderItem  
Dim t As TextInputStream  
Dim s As String  
Dim enc as TextEncoding  
f=GetOpenFolderItem("text") //file type defined via the FileType class  
t=f.OpenAsTextFile  
s=t.ReadAll  
t.Close  
enc=s.Encoding  
Exception err as NilObjectException  
MsgBox err.message
```

**See Also**

[TextConverter](#), [TextEncoding](#) classes; [ConvertEncoding](#), [DefineEncoding](#), [GetTextEncoding](#) functions; [Encodings](#) object.

## Encodings Object

Returns the specified [TextEncoding](#).

**Syntax**

**result=Encodings.EncodingsName**

| Part          | Type                         | Description   |
|---------------|------------------------------|---|
| result        | <a href="#">TextEncoding</a> | The text encoding specified by <i>EncodingsName</i> .   |
| EncodingsName | <a href="#">String</a>       | The name of a <a href="#">TextEncoding</a> . Use the Code Editor's AutoComplete feature to show all possible values of <i>EncodingsName</i> . |

**Notes**

The **Encodings** object makes it easy to obtain a specified [TextEncoding](#). Any text encoding can be obtained via the **Encodings** object. Some of the most useful are UTF8, UTF16, ASCII, MacRoman, MacJapanese, and WindowsLatin1. Use the Autocomplete feature of the Code Editor to view the complete list.

**Example**

Use the Chr method of the [TextEncoding](#) class to get a specific character in any encoding scheme. You use the **Encodings** object to first get the desired encoding. For example:

```
Dim s as String
s=Encodings.UTF8.Chr(169)
```

When you read a string that was created outside REALbasic, you should specify its encoding so that REALbasic can interpret the byte string correctly. Use the **Encodings** object to get the encoding and pass it to the Encoding property of the [TextInputStream](#) class. This example specifies the MacCentralEurRoman encoding.

```
Dim f As FolderItem
Dim t As TextInputStream
f=GetOpenFolderItem("text")
If f <> Nil then
  t=f.OpenAsTextFile
  t.Encoding=Encodings.MacCentralEurRoman
  EditField1.text=t.ReadAll
  t.Close
End if
```

You can also specify the encoding of text using the optional parameter of the Read, ReadLine, or ReadAll methods.

## End Quote Missing Error

---

**See Also**      [TextEncoding](#) class; [DefineEncoding](#), [ConvertEncoding](#), [Encoding](#) functions.

## End Quote Missing Error

An end quote was missing from a literal string.

**Example**    An end quote is missing from the following [String](#) passed to the [MsgBox](#) command:

```
MsgBox "Hello world
```

**See Also**    [""](#) literal.

## EndOfLine Class

Returns an end-of-line terminator.

**Super Class** [Object](#)

**Syntax**      **result=EndOfLine**

| Part   | Type                   | Description   |
|--------|------------------------|---|
| result | <a href="#">String</a> | The end of line <a href="#">String</a> for the platform being compiled. |

or

**result=EndOfLine.*EndOfLineType***

| Part          | Type                   | Description  |
|---------------|------------------------|--|
| result        | <a href="#">String</a> | The end of line <a href="#">String</a> specified by <i>EndOfLineType</i> .                               |
| EndofLineType |                        | The end of line <a href="#">String</a> being requested. The choices are:<br>Windows<br>Macintosh<br>Unix |

## Properties

| Name      | Type                   | Description  |
|-----------|------------------------|--|
| Macintosh | <a href="#">String</a> | The Macintosh line ending, <a href="#">Chr(13)</a> .       |
| Windows   | <a href="#">String</a> | The Windows line ending, <a href="#">Chr(13)+Chr(10)</a> . |
| Unix      | <a href="#">String</a> | The Unix line ending, <a href="#">Chr(10)</a> .            |

## Notes

As indicated by the syntax, **EndOfLine** can be used in either of two ways. You can call one of its properties to get the line ending for that platform or you can call it without

accessing any of its properties. In that case, it returns the **EndOfLine** character or character string for the platform being compiled. For console applications on Mac OS X, the **EndOfLine** function returns EndOfLine.Unix.

Use the [ReplaceLineEndings](#) function to convert the line endings of text from one line ending to another.

Some applications on Mac OS X require a Unix line ending because Mac OS X is based on BSD Unix.

**Example** The following example specifies Unix line endings.

```
Dim cr as String
cr=EndOfLine.Unix
Editfield1.Text ="Hello world"+cr+"Such as it is."
```

**See Also** [ReplaceLineEndings](#) function.

## ExcelApplication Class

Used to automate Microsoft Excel.

**Super Class** [OLEObject](#)

**Notes** The language that you use to automate Microsoft Office applications is documented by Microsoft and numerous third-party books on Visual Basic for Applications (VBA). Microsoft Office applications provide online help for VBA. This is your primary reference for REALbasic office automation.

To access the online help, choose Macros from the Tools Menu of your MS Office application, and then choose Visual Basic Editor from the Macros submenu. When the Visual Basic editor appears, choose Microsoft Visual Basic Help from the Help menu. The help is contextual in the sense that it provides information on automating the Office application from which you launched the Visual Basic editor.

If VBA Help does not appear, you will need to install the VBA help files. On Windows Office 2003, Office prompts you to install the VBA help files when you first request VBA help. You don't need the master CD. On Macintosh, Office v.X does not install the VBA help files as part of the full install. Quit out of Office and locate your master CD. Open the "Value Pack" folder and double-click the Value Pack installer. In the Value Pack installer dialog, scroll down to the Programmability topic, select it, and click Continue. The installer will then add the VBA help files and examples to your Office installation. When the install finishes, the VBA help files will be available to the Visual Basic editor within all your Office X applications.

## Exception Block

---

Microsoft has additional information on VBA at <http://msdn.microsoft.com/vbasic/> and have published their own language references on VBA. One of several third-party books on VBA is “VB & VBA in a Nutshell: The Language” by Paul Lomax (ISBN: 1-56592-358-8).

### Example

The following example transfers the information in a [ListBox](#) to Excel and tells Excel to compute and format a column total. The code is in a [PushButton's](#) Action event handler and it assumes that a two-column [ListBox](#), ListBox1, is in the window. It contains the following data.

| Item    | Price |
|---------|-------|
| Apples  | 5     |
| Oranges | 5     |
| Bananas | 5     |

```
Dim excel as ExcelApplication
Dim book as ExcelWorkbook
Dim sheet as ExcelWorksheet
Dim i as Integer

excel = New ExcelApplication
excel.Visible = True
book = excel.Workbooks.Add
excel.ActiveSheet.Name = "Expenses Report"
For i = 0 to Listbox1.listcount - 1
    excel.Range("A" + Str(i + 1), "A" + Str(i + 1)).value = listbox1.cell(i, 0)
next
excel.Range("A4", "A4").Value = "Total"
excel.Range("B1", "B3").Value = "5"
excel.Range("B1", "B3").Style = "Currency"
excel.Range("B4", "B4").Value = "=SUM(B1:B3)"

Exception err as OLEException
Msgbox err.message
```

### See Also

[Office](#), [OLEException](#), [OLEObject](#), [PowerPointApplication](#), [WordApplication](#) classes.

---

## Exception Block

Used to handle [RuntimeException](#) errors.

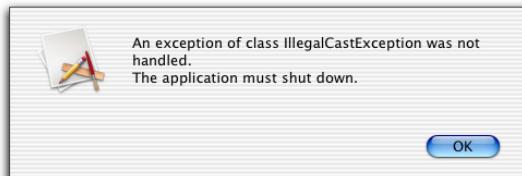
**Syntax****Exception [[*ErrorParameter*] [As *ErrorType*]]**

| Part           | Description   |
|----------------|---|
| ErrorParameter | Optional, used to determine the type of runtime exception.  |
| ErrorType      | Optional, if used, it must be used with ErrorParameter. Used to 'catch' a particular type of runtime error. If used, the Exception block will handle only that type of runtime error. |

**Notes**

One or more **Exception** blocks can be inserted after the last “regular” line of code to catch and handle runtime exceptions that may occur anywhere within the method. Local variables that are declared inside an **Exception** block exist only within the block’s scope rather than inside the entire method’s scope. This means that multiple **Exception** blocks at the same level can use the same exception variable name.

If a runtime exception occurs in a built application and is not handled, REALbasic will display a generic runtime error message box and quit the application. An example message box is shown below:



Exception blocks provide a means of handling the error more gracefully.

Exception blocks always appear at the end of a method (not where you think the error might occur) because every line after the **Exception** line is considered part of the exception block. In the Code Editor, the **Exception** line has the same level of indentation as the [Sub](#) or [Function](#) line.

You can use **Exception** alone if you wish to handle any type of exception in the Exception block, as shown below:

```
Sub...
.
.
Exception
MsgBox "Something really bad happened, but I don't know what."
```

The example shown above is sufficient to prevent the application from quitting, but the message is not very informative because you don’t have a clue what type of exception occurred.

## Exception Block

---

The way to determine which type of runtime error occurred is to use *ErrorParameter* and, in some way, test its type. *ErrorParameter* can be any of the following error types.

| TypeError                                    | Description  |
|--|--|
| <a href="#">FunctionNotFoundException</a>    | A function declared using the <a href="#">Declare</a> statement's "Soft" keyword could not be loaded.  |
| <a href="#">IllegalCastException</a>         | You attempted to recast an object and sent it a message its real class can't accept.   |
| <a href="#">InvalidParentException</a>       | You tried to get the parent of a control using the Parent property of the <a href="#">Control</a> class, but its parent is in a different window.  |
| <a href="#">KeyChainException</a>            | A method of the <a href="#">KeyChain</a> or <a href="#">KeyChainItem</a> classes failed.   |
| <a href="#">KeyNotFoundException</a>         | You tried to access an item in a <a href="#">Dictionary</a> using a key that is not in the <a href="#">Dictionary</a> .  |
| <a href="#">NilObjectException</a>           | An attempt was made to access an object that does not exist.   |
| <a href="#">NoOpenTransportException</a>     | Open Transport is not installed on the computer.   |
| <a href="#">OLEException</a>                 | An OLE-related runtime error occurred.   |
| <a href="#">OutOfBoundsException</a>         | An attempt was made to read from or write to a value, character, or element outside the bounds of the object or data type, i.e., you tried to access an array element that doesn't exist.    |
| <a href="#">OutOfMemoryException</a>         | Raised in certain cases when you run out of memory, for example when there isn't enough memory to complete an operation.   |
| <a href="#">RegExException</a>               | You used an incorrect syntax in a regular expression.  |
| <a href="#">RegExSearchPatternException</a>  | You specified an incorrect search pattern in a regular expression.   |
| <a href="#">RegistryAccessErrorException</a> | Occurs if you try to use the <a href="#">RegistryItem</a> class without proper access privileges or try to use it under any Macintosh OS or Linux. Registry items is a Windows-only feature. |
| <a href="#">ServiceNotAvailableException</a> | Occurs when a requested service is not available.  |
| <a href="#">ShellNotRunningException</a>     | You tried to pass a string to a <a href="#">Shell</a> that is not running.   |
| <a href="#">ShellNotRunningException</a>     | You tried to access an interactive shell when it wasn't running, or tried to change the context of the script while it was running.  |
| <a href="#">SpotlightException</a>           | An error related to a <a href="#">SpotlightQuery</a> was encountered, such as an invalid query.  |

| ErrorType                                     | Description  |
|---|--|
| <a href="#">StackOverflowException</a>        | When one routine (method/event handler/menu handler) calls another, memory is used to keep track of the place in each routine where it was called along with the values of its local variables. The purpose of this is to return (when the routine being called finishes) to the previous routine with all local variables as they were before. The memory set aside for tracking this is called the Stack (because you are "stacking" one routine on top of another). If your application runs out of stack space, a <a href="#">StackOverflowException</a> will occur. |
| <a href="#">ThreadAlreadyRunningException</a> | You tried to set the stack size of a thread that is already running or tried to call the Run method for a thread that is already running.  |
| <a href="#">TypeMismatchException</a>         | You tried to assign to an object the wrong data type. Occurs only when using an object that REALbasic cannot type at compile-time.   |
| <a href="#">UnsupportedFormatException</a>    | You used an incorrect syntax in an expression, for example for column widths in a <a href="#">ListBox</a> , or tried to use an unsupported format in saving and opening pictures   |

One way to test ErrorParameter is with an [If](#) statement in the Exception block:

```
Sub...
.

Exception err
If err IsA TypeMismatchException then
  MsgBox "Tried to retype an object!"
elseif err IsA NilObjectException then
  MsgBox "Tried to access a Nil object!"
.

End if
```

Instead of using multiple [If](#) statements, you can use multiple **Exception** blocks, each of which handles a different runtime exception type:

```
Sub...
.

Exception err as TypeMismatchException
  MsgBox "Tried to retype an object!"
Exception err as NilObjectException
  MsgBox "Tried to access a Nil object!"
```

## Exception Objects Must be of Type RuntimeException Error

---

The [Try](#) block is an alternative to the Exception block. Unlike Exception, you can have nested Try blocks. If the innermost Try block does not handle the exception, it will be passed to the next block, and so forth. If both [Try](#) and **Exception** blocks are used together, the [Try](#) block will catch errors prior to the **Exception** block.

### Examples

See the examples for the [NilObjectException](#), [IllegalCastException](#), [TypeMismatchException](#), [OutOfBoundsException](#), and [StackOverflowException](#) errors.

### See Also

[Function](#), [Raise](#), [Sub](#) statements; [IllegalCastException](#), [KeyNotFoundException](#), [NilObjectException](#), [NoOpenTransportException](#), [OLEException](#), [OutOfBoundsException](#), [OutOfMemoryException](#), [RegExException](#), [RegExSearchPatternException](#), [RuntimeException](#), [ShellNotRunningException](#), [StackOverflowException](#), [TypeMismatchException](#), [UnsupportedFormatException](#) classes; [Nil](#) keyword; [Try](#) block.

---

## Exception Objects Must be of Type RuntimeException Error

You declared an exception variable as something other than a [RuntimeException](#) or a subclass of [RuntimeException](#).

### Example

The following declaration produces this error:

Exception error as [Double](#)

---

## Exit Statement

The **Exit** statement causes control to continue at the statement following the end of a loop without the loop conditions being satisfied.

### Syntax

**Exit**

### Notes

In contrast, the [Continue](#) statement in a loop continues execution with the next iteration of the loop while skipping over the statements between Continue and the end of the loop.

If you use **Exit** to exit a function, the [function](#) will return the default value for the data type of the returned variable.

**Example**

This example increments a counter variable. If the user presses Esc on Windows or Linux or Command-Period on Macintosh, the [UserCancelled](#) function will return [True](#) and the loop will exit. Otherwise, the loop will continue until the Stop variable is [True](#):

```
Dim counter as Integer
Do
    counter=counter+1
    If UserCancelled then
        Exit
    End if
    Loop Until Stop
```

If you use the DoEvents method of the [Application](#) class inside a loop such as this, then the [UserCancelled](#) function does not detect the Esc key or the Command-period sequence. When you use DoEvents, the user interface remains responsive while the loop is executing, so you can add a button that the user can click to stop the loop.

**See Also**

[Continue](#), [Do...Loop](#), [For...Next](#), [GOTO](#), [While...Wend](#) statements.

## Exp Function

Returns “e” to the power of the value specified.

**Syntax**

**result=Exp (value)**

| Part   | Type                   | Description                            |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The exponential of value.              |
| value  | <a href="#">Double</a> | The value you want the exponential of. |

**Notes**

The **Exp** function returns “e” to the power of the value passed to it.

**Examples**

This example uses the **Exp** function to return the exponential of a number.

```
Dim d as Double
d=Exp(10) //returns 22026.4657948
```

**See Also**

[^](#), [\\*](#) operators.

## ExportPicture Function

Exports an object of type [Picture](#).

## ExportPicture Function

---

### Syntax

**result=ExportPicture(pic)**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | <a href="#">True</a> if the picture was saved successfully; <a href="#">False</a> if the picture was not saved. |
| pic    | <a href="#">Picture</a> | A picture.  |

### Notes

When **ExportPicture** is called, it displays a save-file dialog box so the user can save the picture in any directory. The user can also use the dialog box to choose a picture format. If QuickTime is installed (Windows and Macintosh), the available choices are: BMP, PICT, PhotoShop, JPEG, PNG, SGI, TGA, TIFF, and QuickTime image. When the user chooses a format, REALbasic adds the appropriate suffix to the filename.

ExportPicture returns [True](#) if the picture was saved; if the user clicks cancel, ExportPicture returns [False](#).

### Windows

The BMP format is supported on Windows. For all other formats, QuickTime is required.

### Macintosh

The PICT format is supported on Macintosh. For all other formats, QuickTime is required.

### Linux

The BMP format is supported on Linux.

### Example

The following code allows the user to save a picture. The file type "bmp" has already been defined as a File Type using either the [FileType](#) class or the File Types Sets Editor in the IDE.

```
Dim p as Picture
Dim r as Boolean
Dim f as FolderItem
f=GetOpenFolderItem("bmp")
If f {} Nil then
  p=f.OpenAsPicture
  r=ExportPicture(p)
  If r then
    MsgBox "Your picture was saved."
  else
    MsgBox "Your picture was not saved."
  end if
End if
```

### See Also

[Picture](#), [FolderItem](#) classes.

# Extends Keyword

Used in a method declaration to indicate that the method is to be called using the object's syntax, i.e., as a method belonging to the object.

## Syntax

### Extends parameter As Object

| Part      | Type                  | Description  |
|-----------|-----------------------|--|
| parameter |                       | The name of the first parameter.   |
| Object    | Any valid Object type | The Object type from which the new method is to be called. If Object is <a href="#">Nil</a> , the Extends method will automatically not be called. |

## Notes

The **Extends** keyword enables you to call a user-defined method as a method belonging to a class. You use the Extends keyword only for the first parameter in the method declaration. **Extends** indicates that this parameter is to be used on the left side of the dot operator ("."). The remaining parameters in the declaration, if any, are normal parameters.

It is possible to set a default value for a parameter in a call that uses **Extends** as long as the default value is not set for the **Extends** parameter itself.

Methods declared in this way are sometimes called "class extension methods." You can use **Extends** only for methods in a module.

**Extends** cannot be used to override another method, merely to add methods to the class. Extension methods are not virtual since they are not actually part of the class.

## Example

This example adds a method called SelectAll to the [EditField](#) class. The new method selects all the text in the [EditField](#).

In a module, create the method with the declaration:

```
SelectAll (Extends e as EditField)
```

Although e appears to be a parameter that is passed to the method, it isn't. Since the **Extends** keyword is used, it merely indicates that the method will be added to the [EditField](#) class. When you call the SelectAll method, you don't pass any parameters.

The code for the SelectAll method is:

```
e.SelStart = 0
e.SelLength=Len(e.Text)
```

---

## Extends can only be used on the first parameter Error

---

To call the new method, simply use it as if it was built into the REALbasic language. For example, you could add a Select All menu item to the Edit menu and then use the following line as its menu handler:

```
EditField1.SelectAll
```

This assumes that the window has an [EditField](#) named EditField1.

To use the SelectAll method in other applications, simply copy the module that contains it to another project.

**See Also** [Extends can only be used on the first parameter](#) Error.

---

## Extends can only be used on the first parameter Error

You used the [Extends](#) keyword more than once and/or on a parameter other than the first parameter.

### Example

```
Sub myNewMethod(x as Integer, Extends f as FolderItem)
```

Should be rewritten as:

```
Sub myNewMethod(Extends f as FolderItem, x as Integer, )
```

**See Also** [Extends](#) keyword.

---

## External Functions Cannot Use Objects as Parameters Error

This error occurs if you [Declare](#) an external function and try to specify that one of its parameters is an object.

When passing a [MemoryBlock](#) to a [Declare](#), use Ptr instead. When passing a window to a [Declare](#), use WindowPtr instead.

**See Also** [Declare](#) statement; [MemoryBlock](#) class.

# External Functions Cannot Use String Data Types as Parameters Error

Occurs if one of your [Declare](#) statement's parameters is an ordinary REALbasic string. Use CString or PString instead.

**See Also** [Declare](#) statement.

---

## False Keyword

Used to set [Boolean](#) variables or properties. False is the result of a comparison of two objects that are not equal.

**Syntax** ***expression=False***

Expression is any valid [Boolean](#) expression.

**See Also** [True](#) keyword, [And](#), [Or](#), and [Not](#) operators.

---

## FigureShape Class

Used to draw vector graphic shapes composed of lines and curves. It is similar to a polygon except that a 'side' may be curved. A figure consists of a set of [CurveShapes](#), with the end point of one curve automatically joined to the starting point of the next.

**Super Class** [Object2D](#)

### Properties

| Name  | Type                       | Description   |
|-------|----------------------------|---|
| Count | <a href="#">Integer</a>    | The number of <a href="#">CurveShapes</a> .                                       |
| Item  | <a href="#">CurveShape</a> | Parameter is Index as <a href="#">Integer</a> . Gets or sets the specified curve. |

### Methods

| Name     | Parameters   | Description   |
|----------|--|---|
| AddCubic | X as <a href="#">Integer</a><br>Y as <a href="#">Integer</a><br>X2 as <a href="#">Integer</a><br>Y2 as <a href="#">Integer</a><br>CX as <a href="#">Integer</a><br>CY as <a href="#">Integer</a><br>CX2 as <a href="#">Integer</a><br>CY2 as <a href="#">Integer</a> | Adds a cubic curve to the figure. The starting and ending points are x,y and x2, y2, respectively. This is a shortcut for adding a <a href="#">CurveShape</a> . CX, CY, CX2, and CY2 are the control points, Control X(i) and ControlY(i) in <a href="#">CurveShape</a> .     |
| AddLine  | X as <a href="#">Integer</a><br>Y as <a href="#">Integer</a><br>X2 as <a href="#">Integer</a><br>Y2 as <a href="#">Integer</a>   | Adds a straight line to the figure. X and Y are the starting coordinates; X2 and Y2 are the end coordinates.  |
| AddQuad  | X as <a href="#">Integer</a><br>Y as <a href="#">Integer</a><br>X2 as <a href="#">Integer</a><br>Y2 as <a href="#">Integer</a><br>CX as <a href="#">Integer</a><br>CY as <a href="#">Integer</a>   | Adds a quadratic curve to the figure. The starting and ending points are x,y and x2, y2, respectively. This is a shortcut for adding a <a href="#">CurveShape</a> . CX, CY, CX2, and CY2 are the control points, Control X(i) and ControlY(i) in <a href="#">CurveShape</a> . |
| Append   | Curve as <a href="#">CurveShape</a>  | Adds the passed <a href="#">CurveShape</a> to the figure.   |
| Insert   | Index as <a href="#">Integer</a><br>Curve as <a href="#">CurveShape</a>  | Inserts the passed <a href="#">CurveShape</a> at the position indicated by <i>Index</i> .   |
| Remove   | Index as <a href="#">Integer</a> OR<br>Curve as <a href="#">CurveShape</a>   | Removes a curve, specified either by its index or by reference.   |

### Notes

A figure is formed by drawing each curve it contains in order, joining the endpoint of one to the starting point of the next. The figure is degenerate if doing so does not enclose any area. This will be the case for a figure containing only one line, for example. The appearance of a degenerate figure is undefined.

AddLine, AddCubic, and AddQuad are convenience methods that make it easier to add curves to the figure. You could instead create your own [CurveShapes](#), and add them with the Append or Insert methods.

### Example

The following example draws a triangle. The code is placed in the Paint event of a [Window](#) or [Canvas](#).

```
Sub Paint (g as Graphics)
  Dim fx as New FigureShape
  fx.AddLine 0, 100, 50, 0
  fx.AddLine 50, 0, -50, 0
  fx.Border = 100 // opaque border
  fx.BorderColor = &cFF0000 // red border
  fx.FillColor = &cFFFF00 // yellow interior
  g.DrawObject fx, 100,100
```

**See Also**

[ArcShape](#), [CurveShape](#), [FolderItem](#), [Graphics](#), [Group2D](#), [Object2D](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

## FileType Class

Represents a REALbasic file type. It provides the functionality of the File Type Sets Editor in the IDE.

**Super Class** [Object](#)

### Properties

| Name       | Type                    | Description   |
|------------|-------------------------|---|
| All        | <a href="#">String</a>  | Contains all of the file types in the File Type Set. See “Using File Type Sets” in the Notes.   |
| Extensions | <a href="#">String</a>  | The file extension for the file type, or several extensions separated by semicolons, e.g., “jpg;jpeg”. A period before the file type is allowed, but it is not necessary. |
| Icon       | <a href="#">Picture</a> | The icon associated with the file type in the IDE, if any. This property does not apply to dynamically created file types.  |
| MacCreator | <a href="#">String</a>  | The four-byte string used to represent the application that owns the file in the Mac OS.  |
| MacType    | <a href="#">String</a>  | The four-byte string used to identify the type of file in the Mac OS.   |
| Name       | <a href="#">String</a>  | The name by which this file type will be known to other parts of the REALbasic framework. This is what is shown in some open/save file dialogs.                           |

### Notes

Use the **FileType** class to programmatically create REALbasic file types, name them, and specify their MacType, MacCreator, file extensions, and icon (if needed).

The **FileType** class has a built-in conversion to [String](#) operator. This enables you to use a **FileType** object anywhere you would normally use a [String](#) to indicate the file type. It also has built-in addition operators that let you add two **FileTypes**, or a **FileType** and a [String](#), to get a [String](#). The result is the name of the **FileType** joined with the other one (or the [String](#)) with a semicolon. This is the format required by functions like [GetOpenFolderItem](#). These built-in operators make it possible to work with **FileType** objects as if they were strings. The example illustrates the operators. You can, of course, specify the file type as a string by passing its name.

### Using File Type Sets

In the IDE, each file type is shown as one row in the File Types set table. A File Type set has the [String](#) property “All” that returns all of the file types in the set. Use All instead of multiple file types when you want to refer to the group of file types. Suppose you create a File Type Set called ImageTypes in which you specify all of the valid image

## FileType Class

---

types that your application can open. You can specify the entire list of image types with a line such as:

```
f=GetOpenFolderItem(ImageTypes.All)
```

The other way to do so is with the “+” operator, i.e.,

```
f=GetOpenFolderItem(ImageTypes.JPG + ImageTypes.MacPICT)
```

assuming that the file types JPEG and MacPICT are the members of the ImageTypes set. The additional advantage of using All is that you can modify the members of the ImageTypes set and you won't need to update your code.

### Example

The following example dynamically creates two FileTypes, one for JPEG files and one for PNG files. It then displays an Open File dialog box that allows the selection of only those two file types.

```
Dim jpegType As New FileType
jpegType.Name = "image/jpeg"
jpegType.MacType = "JPEG"
jpegType.Extensions = "jpg;jpeg"

Dim pngType As New FileType
pngType.Name = "image/png"
pngType.MacType = "PNG"
pngType.Extensions = "png"

Dim f As FolderItem

//using the addition and conversion operators...
f = GetOpenFolderItem( jpegType + pngType )
```

### See Also

[FolderItem](#), [FolderItemDialog](#), [GetOpenFolderItem](#), [GetSaveFolderItem](#), [OpenDialog](#), [SaveAsDialog](#) classes.

# FillColor Function

The operating system's currently selected background color.

## Syntax

**result=FillColor**

| Part   | Type                  | Description  |
|--------|-----------------------|--|
| result | <a href="#">Color</a> | The color used for drawing the background of controls and windows. |

## Notes

This value is useful when you are using [Canvas](#) controls to create custom controls. When drawing objects, use this color to fill the background.

This value can be changed by the user, so you should access this value during your Paint event handler rather than storing the value.

## See Also

[DarkBevelColor](#), [DarkTingeColor](#), [FrameColor](#), [LightBevelColor](#), [LightTingeColor](#), [HighlightColor](#), [TextColor](#) functions; [Color](#) data type.

---

# Finally Block

Contains code that will run after a method or function is finished, even if a [RuntimeException](#) has occurred.

## Syntax

```
Try
  //REALbasic statements
Catch
[Finally]
  //code that executes even if runtime exceptions were raised
End [Try]
```

## Notes

Sometimes a method needs to do some cleanup work whether it is finishing normally or aborting early because of an [exception](#). The optional **Finally** block at the end of a method or function runs after its exception handlers, if it has any. Code in this block will be executed even if an exception has occurred, whether the exception was handled or not. However, the code in the **Finally** block does not execute if the **Try** block contains a [Return](#). For example, you can write:

```
Sub SomeFunction()
  //some RB code here...
  Finally
  //some more code that will always execute
```

## Floor Function

---

See the entries for the [Sub](#) and [Function](#) statements.

**Example** This simple example shows a **Finally** block that executes even after an exception.

```
Try
Raise New RuntimeException()
Finally
//Executes even though the exception is not handled
MsgBox ("Finally")
End Try
```

**See Also** [Catch](#), [Function](#), [Sub](#) statements; [Exception](#), [Try](#) blocks.

## Floor Function

Returns the value specified rounded down to the nearest [Integer](#).

**Syntax** **result=Floor (value)**

| Part   | Type                   | Description                      |
|--------|------------------------|----------------------------------|
| result | <a href="#">Double</a> | The floor of <i>value</i> .      |
| value  | <a href="#">Double</a> | The value you want the floor of. |

**Notes** The **Floor** function returns the value passed to it rounded down to the nearest [Integer](#).

**Example** This example uses the **Floor** function to return floor of a number.

```
Dim d as Double
d=Floor(1.234) //returns 1
```

**See Also** [Ceil](#), [Round](#) functions.

## FolderItem Class

**FolderItem** class objects represent files, applications, or folders. They are created by calling a method such as [GetFolderItem](#) or via the Parent or Item properties of other **FolderItem** objects.

**Super Class** [Object](#)

## Properties

| Name             | Type                       | Description  |
|------------------|----------------------------|--|
| AbsolutePath     | <a href="#">String</a>     | The full path to the <b>FolderItem</b> . On Windows, the pathname ends with a trailing backslash; on Macintosh, the pathname ends with a trailing colon; on Linux the pathname ends with a forward slash.<br>When accessing a drive on Windows that you do not have permissions for or does not exist, AbsolutePath has no trailing slash. For example, "a:" is a floppy drive with no disk, "a:" has a disk.  |
| Alias            | <a href="#">Boolean</a>    | Returns <a href="#">True</a> if the item is an alias.  |
| Count            | <a href="#">Integer</a>    | The number of items in the <b>FolderItem</b> if it is a directory/folder.  |
| CreationDate     | <a href="#">Date</a>       | The creation date of the <b>FolderItem</b> .   |
| DesktopFolder    | <a href="#">FolderItem</a> | On Macintosh, it returns a <b>FolderItem</b> that references the Desktop Folder belonging to the parent volume. On Windows and Linux, it returns <a href="#">Nil</a> .   |
| Directory        | <a href="#">Boolean</a>    | <a href="#">True</a> if the <b>FolderItem</b> is a directory (a folder).   |
| DisplayName      | <a href="#">String</a>     | The name of the <b>FolderItem</b> as it should be seen by the user. It is usually the same as the Name property. Under Mac OS X 10.1, it could be the name without the extension, depending on whether the user has file extensions hidden for that <b>FolderItem</b> . Under Mac OS X, use DisplayName rather than Name when displaying the name of the item to the user.   |
| ExtensionVisible | <a href="#">Boolean</a>    | Allows you to tell whether a given file has its file extension hidden or visible, or toggle that status. Note that when you change this setting, the operating system's Desktop Manager or subsequent Navigation dialogs may not reflect the change right away. When you create a new file under Mac OS X, REALbasic sets the "hide extension" flag according to the current value of this property. On Windows, hiding the file extension is a global property, so if it is true for one <b>FolderItem</b> on the user's computer it is true for all of them. |
| Exists           | <a href="#">Boolean</a>    | Indicates whether or not the AbsolutePath property points to a file or directory that exists.  |

| Name          | Type                    | Description  |
|---------------|-------------------------|--|
| Group         | <a href="#">String</a>  | Gets or sets the name of the owning Group of the <b>FolderItem</b> . Supported on Unix-based operating systems only (Mac OS X and Linux). Use the <b>FolderItem</b> 's <b>Permissions</b> property or the <a href="#">Permissions</a> class to get and set permissions for the owning Group. The <b>Owner</b> property gets and sets the <b>FolderItem</b> 's <b>Owner</b> .   |
| IsReadable    | <a href="#">Boolean</a> | <a href="#">True</a> if it is possible to read from the <b>FolderItem</b> .  |
| IsWriteable   | <a href="#">Boolean</a> | <a href="#">True</a> if it is possible to write to the <b>FolderItem</b> .   |
| LastErrorCode | <a href="#">Integer</a> | <p>Contains the error code for the last supported operation on the <b>FolderItem</b>. <b>LastErrorCode</b> will be set for any error.</p> <p>The following error codes may be returned:</p> <ul style="list-style-type: none"> <li>100: Destination does not exist. You will get this error only on <b>CopyFileTo</b> and <b>MoveFileTo</b>.</li> <li>101: File not found.</li> <li>102: Access denied</li> <li>103: Out of memory</li> <li>104: File in use</li> <li>105: Invalid name</li> </ul> <p>You can test whether an error occurred by comparing the value in <b>LastErrorCode</b> to one of the Error Constants. See the section "FolderItem Error Constants", in the Notes section.</p> <p><b>LastErrorCode</b> may also return operating system-specific error codes if the error does not map to one of the above errors.</p> |
| Length        | <a href="#">Integer</a> | The size of the file's data fork. For directories, the size will be zero.  |
| Locked        | <a href="#">Boolean</a> | The <b>FolderItem</b> is locked or resides on a locked volume.   |
| MacCreator    | <a href="#">String</a>  | For documents, the 4 character code that identifies the application that would be launched and would open the <b>FolderItem</b> if it were opened from the desktop. Assign the 4-character string to this property. If the <b>FolderItem</b> refers to a directory on Mac OS, MacCreator returns "MACS". MacCreator is Macintosh-specific. MacRoman encoding is used.  |
| MacDirID      | <a href="#">Integer</a> | Mac directory ID. Macintosh only.  |
| MacType       | <a href="#">String</a>  | The 4-character code that identifies the type of file (Macintosh only). For example, "APPL" is the type of all applications. "TEXT" is the type of most text documents. Assign the 4-character string to this property. If the <b>FolderItem</b> refers to a directory on Mac OS, MacType returns "fold". MacRoman encoding is used.   |

| Name             | Type                       | Description  |
|------------------|----------------------------|--|
| MacVRefNum       | <a href="#">Integer</a>    | Macintosh volume reference number (Macintosh only).  |
| ModificationDate | <a href="#">Date</a>       | The modification date of the <b>FolderItem</b> .   |
| Name             | <a href="#">String</a>     | The name of the <b>FolderItem</b> . Changing this name will change the name of the file or folder. It will rename files if the user has permission to rename them, they are not in use, and it is not a folder or temporary file.  |
| Owner            | <a href="#">String</a>     | Gets or sets the Owner of the <b>FolderItem</b> under Unix-based operating systems (Mac OS X and Linux). Use the <b>FolderItem</b> 's <b>Permissions</b> property or the <a href="#">Permissions</a> class to get and set permissions for the Owner. The <b>Group</b> property gets and sets the <b>FolderItem</b> 's <b>Group</b> .   |
| Parent           | <a href="#">FolderItem</a> | Returns the <b>FolderItem</b> object for the parent of this item in the file hierarchy. It returns <a href="#">Nil</a> if this item is the root.   |
| Permissions      | <a href="#">Integer</a>    | <p>Gets and sets the permissions of the <b>FolderItem</b> on Unix-based operating systems only (Mac OS X and Linux). You can get and set the permissions as an octal <a href="#">Integer</a> or you can get and set the permissions via the properties of the <a href="#">Permissions</a> class.</p> <p>Permissions is represented as a three-digit numeric code, in which each digit ranges from 0 to 7. The digits correspond to the permissions of the <b>FolderItem</b> owner, the owning group, and other users not in the owning group, in that order. The code for each digit is computed using the following values:</p> <ul style="list-style-type: none"> <li>4: Read permissions</li> <li>2: Write permissions</li> <li>1: Execute permissions</li> </ul> <p>For each digit, the permissions are expressed by adding up the values. Each digit can take on the following values:</p> <ul style="list-style-type: none"> <li>0 = No permissions</li> <li>1 = Execute permissions</li> <li>2 = Write permissions</li> <li>3 = Write and Execute permissions</li> <li>4 = Read permissions</li> <li>5 = Read and Execute permissions</li> <li>6 = Read and Write permissions</li> <li>7 = Read, Write, and Execute permissions</li> </ul> <p>For example, the code "764" is interpreted as follows:</p> <ul style="list-style-type: none"> <li>Owner: 7 = Read, Write, and Execute permissions</li> <li>Group: 6 = Read and Write permissions</li> <li>Others: 4 = Read permissions</li> </ul> |

## FolderItem Class

| Name               | Type                          | Description  |
|--------------------|-------------------------------|--|
| ResourceForkLength | <a href="#">Integer</a>       | The number of bytes in the file that make up the resource fork. Access to the resourcefork is supported only on Macintosh.   |
| SharedTrashFolder  | <a href="#">FolderItem</a>    | Returns a <a href="#">FolderItem</a> that references the shared trash folder belonging to the parent volume (Macintosh). Returns <a href="#">Nil</a> on Windows and Linux.   |
| ShellPath          | <a href="#">String</a>        | Gets the shell path of the <a href="#">FolderItem</a> . On Windows, this is the short path. On Mac OS X and Linux, this is the POSIX path. On Mac OS 8-9, it is the same as the AbsolutePath. It is not always possible to get the ShellPath of a <a href="#">FolderItem</a> , for example for a non-existent <a href="#">FolderItem</a> on Windows. If REALbasic is unable to get the ShellPath, ShellPath will contain the AbsolutePath. |
| Temporary Folder   | <a href="#">FolderItem</a>    | Returns a <a href="#">FolderItem</a> that references the temporary folder belonging to the parent volume. Returns <a href="#">Nil</a> on Windows and Linux.  |
| TrashFolder        | <a href="#">FolderItem</a>    | Returns a <a href="#">FolderItem</a> that references the shared trash folder belonging to the parent volume.   |
| Type               | <a href="#">String</a>        | If the <a href="#">FolderItem</a> is not a directory, this property contains the file type of the <a href="#">FolderItem</a> . If the <a href="#">FolderItem</a> is a directory (Mac OS), it will return a matching file type if you have created one.   |
| URLPath            | <a href="#">String</a>        | Returns a URL for the <a href="#">FolderItem</a> that can be passed to ShowURL and other methods that require a valid URL.   |
| VirtualVolume      | <a href="#">VirtualVolume</a> | If the <a href="#">FolderItem</a> is in a <a href="#">VirtualVolume</a> , it returns the <a href="#">VirtualVolume</a> that stores the file. If it is a real file, it returns <a href="#">Nil</a> .  |
| Visible            | <a href="#">Boolean</a>       | <a href="#">True</a> if the <a href="#">FolderItem</a> is visible and <a href="#">False</a> if it is not.  |

## Methods

| Name             | Parameters                     | Description   |
|------------------|--------------------------------|---|
| AppendToTextFile |                                | Returns a <a href="#">TextOutputStream</a> . Appends the text to the text file. If the file does not exist, it is created.  |
| Child            | Name as <a href="#">String</a> | Child returns a <a href="#">FolderItem</a> that represents a file or directory within this <a href="#">FolderItem</a> . Shortcuts or aliases are resolved on all platforms. |

| Name                | Parameters                          | Description  |
|---------------------|-------------------------------------|--|
| CopyFileTo          | Item as <a href="#">FolderItem</a>  | If <i>Item</i> is a folder, then the <b>FolderItem</b> is copied into <i>Item</i> . If Item is a file and the file already exists, the copy is aborted. You need to delete the existing file first. If there is an error, the <b>LastErrorCode</b> property contains an error code.  |
| CreateAsFolder      |                                     | Creates a folder at the location specified by the <b>FolderItem</b> properties.  |
| CreateBinaryFile    | FileType As <a href="#">String</a>  | Creates a binary file of type <i>FileType</i> . Returns a <a href="#">BinaryStream</a> . <i>FileType</i> can be the name of a file type that was previously created using the <a href="#">FileType</a> class or you can pass an empty string if you are not concerned about file types.  |
| CreateMovie         |                                     | Creates a new <a href="#">EditableMovie</a> . Returns an <a href="#">EditableMovie</a> object.   |
| CreateResourceFork  | FileType As <a href="#">String</a>  | Macintosh only. Creates a new resource fork. If file exists, then existing resource fork is emptied, otherwise new file is created with FileType passed. FileType is the name of a file type that was created via the <a href="#">FileType</a> class or in the IDE using the File Types Sets Editor.   |
| CreateTextFile      |                                     | Creates a text file at the location specified by the <b>FolderItem</b> . Returns a <a href="#">TextOutputStream</a> .  |
| CreateVirtualVolume |                                     | Creates and formats a new <a href="#">VirtualVolume</a> at the location specified by the <b>FolderItem</b> . Returns a <a href="#">VirtualVolume</a> .   |
| Delete              |                                     | Deletes the file or directory specified by the <b>FolderItem</b> . This method permanently removes the file or directory from the volume it was stored on. If you are deleting a directory, it must be empty. If there is an error, the <b>LastErrorCode</b> property contains an error number.  |
| GetRelative         | SavelInfo as <a href="#">String</a> | Returns a <b>FolderItem</b> . Returns a <b>FolderItem</b> based on the string passed to it. If the string indicates a relative path, the current <b>FolderItem</b> is considered its reference point. If the string passed to it is absolute, then the current <b>FolderItem</b> is ignored when resolving the path. GetRelative returns <a href="#">Nil</a> only if there is not sufficient information in <i>SavelInfo</i> to construct a <b>FolderItem</b> (e.g., using a relative path that causes the parsing to descend below root level). |

## FolderItem Class

| Name             | Parameters  | Description  |
|------------------|---|--|
| GetSaveInfo      | RelativeTo<br>as<br><b>FolderItem</b> ,<br>Mode as<br><a href="#">Integer</a>         | <p>Allow saving <b>FolderItem</b> references without relying on the absolute path. Returns a <a href="#">String</a>. The returned string indicates a relative path with respect to the folder passed in <i>RelativeTo</i>.</p> <p>The Class Constants and values (in parentheses) for Mode are:</p> <ul style="list-style-type: none"><li>SaveInfoDefaultMode (0): The <b>FolderItem</b> should be open in any way REALbasic can possibly create it. This is the Default value.</li><li>SaveInfoRelativeMode (1): The <b>FolderItem</b> should only be opened using relative path information.</li><li>SaveInfoAbsoluteMode (2): The <b>FolderItem</b> should only be opened using absolute path information.</li></ul> <p>Note: The returned string is not intended to be human-readable and any modifications may render it useless.</p> |
| Item             | Index as<br><a href="#">Integer</a>   | If this <b>FolderItem</b> is a directory, Index is an element in a one-based array of <b>FolderItems</b> in this directory. If the <b>FolderItem</b> is an alias, Item automatically resolves the alias and returns a <b>FolderItem</b> for the original file, folder, or application.   |
| Launch           | Parameters<br>as <a href="#">String</a><br>[,Activate<br>as <a href="#">Boolean</a> ] | If the <b>FolderItem</b> is an application, the application is launched. If the <b>FolderItem</b> is a document, the document is opened (and the application is launched if necessary). <i>Parameters</i> is the application's parameters to be passed to the launched application (Windows and Linux). The optional parameter <i>Activate</i> specifies whether the application should be launched in the foreground or background. The default value is <a href="#">True</a> (Foreground). If you specify <a href="#">False</a> , REALbasic will attempt to launch the application in the background, but this may not work with certain applications.   |
| MoveFileTo       | Destination<br>as<br><b>FolderItem</b>  | Moves <b>FolderItem</b> to path specified by Destination. It does a move rather than a copy even when the source file is on another volume. If there is an error, the LastErrorCode property contains an error code.   |
| OpenAsBinaryFile | [ReadWrite<br>as <a href="#">Boolean</a> ]  | Opens the <b>FolderItem</b> passed to be read as a binary file. If ReadWrite is <a href="#">True</a> , the file is opened in read/write mode, otherwise it is opened in read-only mode. Returns a <a href="#">BinaryStream</a> .   |

| Name                | Parameters | Description   |
|---------------------|------------|---|
| OpenAsMovie         |            | Opens the <b>FolderItem</b> to be read as a QuickTime movie and returns its movie as a <a href="#">Movie</a> object. <a href="#">Nil</a> is returned if the movie can't be read.  |
| OpenAsPicture       |            | Opens the <b>FolderItem</b> to be read as a picture and returns its image as a <a href="#">Picture</a> object. <a href="#">Nil</a> is returned if the image can't be read. If QuickTime is installed, OpenAsPicture will open any kind of graphics file QuickTime will open (JPEG, GIF, etc.). QuickTime is not required for opening PICT files on Macintosh or JPEG, GIF, and BMP files on Windows. OpenAsPicture supports loading BMP, GIF, TIFF, and XBM images on Linux. OpenAsPicture does not open JPEG images on Linux. The Windows IDE requires QuickTime to build Macintosh applications. OpenAsPicture uses the full resolution of the image, rather than assuming 72 dpi.  |
| OpenAsSound         |            | Opens the <b>FolderItem</b> to be read as a Macintosh System 7 sound file and returns its contents as a sound object. On Windows, OpenAsSound recognizes any sound format that QuickTime can play. WAV files are played as DirectSound. This allows individual pan and volume settings for each sound. You can also play multiple sounds simultaneously. <a href="#">Nil</a> is returned if the sound can't be read or isn't a sound file at all.   |
| OpenAsTextFile      |            | Opens the <b>FolderItem</b> to be read as a text file. Returns a <a href="#">TextInputStream</a> .  |
| OpenAsVectorPicture |            | Opens the <b>FolderItem</b> to be read as a vector <a href="#">Picture</a> . It will do its best to convert a PICT file (on Macintosh) or an .emf file (Windows) to a Picture composed of <a href="#">Object2D</a> objects. The original file may contain elements that do not have an equivalent <a href="#">Object2D</a> object. REALbasic will do its best to map these objects, but there may be some loss of information, depending on the characteristics of the original file. PICTs support unrotated Rectangles, Lines, Ellipses, RoundRects, Polygons, Text, Pixmaps, and Arcs. .emf files support unrotated Rectangles, Lines, Ellipses, RoundRects, Polygons, Text, Pixmaps and Arcs. .emf files are displayed as actual size. For the most part this is huge; you probably will want to scale them down before viewing (pic1.objects.scale = scalingFactor). We find that a scale factor of 0.045 is a good value. |

## FolderItem Class

---

| Name                | Parameters   | Description  |
|---------------------|--|--|
| OpenAsVirtualVolume |  | Attempts to open the <b>FolderItem</b> as a <a href="#">VirtualVolume</a> . It returns <a href="#">Nil</a> if the <b>FolderItem</b> is not a <a href="#">VirtualVolume</a> . It is opened as Read/Write unless it is locked at the operating system level. Returns a <a href="#">VirtualVolume</a> .   |
| OpenEditableMovie   |  | Opens the <b>FolderItem</b> as an <a href="#">EditableMovie</a> . <a href="#">Nil</a> is returned if the movie can't be read.  |
| OpenResourceFork    |  | Opens the resource fork of the <b>FolderItem</b> . Access to the resourcefork is supported only on Macintosh.  |
| OpenResourceMovie   | ResID as <a href="#">Integer</a>   | Opens the movie specified by ResID as a movie (Macintosh only). Used only if the movie is stored as a Moov resource. Use OpenAsMovie for QT 4.0 (and greater) files.   |
| OpenStyledEditField | Field As <a href="#">EditField</a>   | Places the styled text from the <b>FolderItem</b> into the Field passed.   |
| SaveAsJPEG          | Picture as <a href="#">Picture</a>   | Saves the picture in JPEG format. Appends ".jpg" to the filename. Requires QuickTime to be installed on the user's computer.   |
| SaveAsPicture       | Picture as <a href="#">Picture</a><br>[Format as <a href="#">Integer</a> ] | Saves the picture in PICT format on Macintosh or BMP format on Windows and Linux and appends ".bmp" to filename (Windows and Linux). QuickTime is not required. Macintosh PICTs are "type 2" PICTs that are saved with the full resolution of the image. The optional second parameter specifies the format: Specifying the proper format allows users to save vector data in a vector format rather than converting to bitmaps. See the Notes section for a description of the possible values of Format. |
| SaveStyledEditField | Field As <a href="#">EditField</a>   | Saves the styled text from the Field passed to the <b>FolderItem</b> . If you have defined a File Type of 'styledText' or 'text' then it will use those creator/type values for the saved file.  |
| TrueChild           | Pathname as <a href="#">String</a>   | If Pathname is a directory, TrueChild returns a <b>FolderItem</b> object for the directory with the name passed. TrueChild returns the actual <b>FolderItem</b> , even if it is an alias.  |
| TrueItem            | Index as <a href="#">Integer</a>   | If this <b>FolderItem</b> is a directory, Index is an element in a one-based array of <b>FolderItems</b> in this directory. TrueItem returns the actual <b>FolderItem</b> , even if it is an alias.  |

## Notes

### Specifying Pathnames

Use the [Volume](#) function, the Parent property of the **FolderItem** class, and the Child method of the **Folderitem** class to specify pathnames. The [Volume](#) function returns a reference to any volume on the user's computer. Pass it a number that indicates the desired volume. Zero is the boot volume. You can get the number of volumes with the [VolumeCount](#) function. For example, to get a **FolderItem** for Microsoft Word in the Program Files folder on the boot volume, you can use the following line of code (The line continuation keyword, `_`, is used to split the line into two printed lines).

```
Dim f as FolderItem
f= Volume(0).Child("Program Files").Child("Microsoft Office")._
    Child("OFFICE11").Child("WINWORD.EXE")
```

The [GetFolderItem](#) function can be used to get a **FolderItem** for an item in the current directory (the directory that contains the application or the REALbasic IDE if you are debugging the application). Simply pass it the name of the item. For example, the following returns a **FolderItem** for the directory "MyTemplates" in the current folder:

```
Dim f as FolderItem
f= GetFolderItem("MyTemplates")
```

If the document or directory does not exist, the Exists property of the **FolderItem** is [False](#).

If you pass the empty string to [GetFolderItem](#), it returns the **FolderItem** for the directory that contains the application.

```
Dim f as FolderItem
f= GetFolderItem("")
```

The Parent property of the **FolderItem** class enables you to navigate one level up in the hierarchy. For example, the following gives you the **FolderItem** for the directory that contains the directory that contains the application:

```
Dim f as FolderItem
f= GetFolderItem("").Parent
```



Mac OS X is based on BSD Unix which uses "/" as the separator. However, because REALbasic and all applications made with REALbasic are Carbon applications on Mac OS X, they continue to use ":" as the separator. The one exception is when accessing files via the [Shell](#) class. In this case, you would use the same separator that you use on the command line.

## FolderItem Class

---

**FolderItem Constructor** When you create a FolderItem with the [New](#) command, you can pass the full or relative path to the new FolderItem as an optional parameter. For example:

```
Dim f as FolderItem  
f=New FolderItem("myDoc.txt")
```

specifies the name of the new FolderItem and it is located in the same folder as the REALbasic application (if you're running in the IDE) or the same folder as the built application.

**Copy Constructor** You can create a copy of a **FolderItem** by passing the **FolderItem** to be copied to the constructor. The result is a copy of the passed **FolderItem** rather than a reference to it. For example:

```
Dim f, f2 as FolderItem  
f = DesktopFolder.Child("MyDocument")  
f2 = New FolderItem(f)
```

If you pass a [Nil](#) **FolderItem**, you will raise a [NilObjectException](#).

**Shell Paths and Regular Paths** If you pass the optional parameter for path, you can also pass an optional second parameter indicating whether the path is a ShellPath, a “regular” path, or a path in the form of a URL. **FolderItem** has three class constants that you use to indicate this, PathTypeAbsolute, PathTypeURL, and PathTypeShell. For example:

```
Dim f as FolderItem  
f=New FolderItem("/home/shr/mytextdoc.txt",FolderItem.PathTypeShell)
```

You cannot pass a non-absolute Shell path. Attempting to do so will result in an [UnsupportedFormatException](#).

If you use PathTypeURL, the URL must begin with “file:///”.

You can also create a **FolderItem** without passing any parameters. It works the same as passing an empty text string.

**Picture Formats** The SaveAsPicture method has an optional second parameter that enables you to specify the format in which the picture will be saved. Formats in the range of 0-99 are reserved for meta-formats. Meta-formats map to a various concrete formats based on the target, the data being saved, or other criteria (The mappings may change in future versions of REALbasic).

- The numbers 100-199 are reserved for formats that are supported on multiple platforms (currently none).
- 200-299 are reserved for formats that are supported on the Macintosh.
- 300-399 are reserved for formats supported on Windows.

- x00-x49 are vector formats, while x50-x99 are raster formats.

### MetaFormats

| Value | Class Constant       | Compatibility  | Description   |
|-------|----------------------|----------------|---|
| 0     | SaveAsMostCompatible | MostCompatible | Most widely-used format for the platform<br>Mac = PICT<br>Win32 = BMP)  |
| 1     | SaveAsMostComplete   | MostComplete   | Format most likely to retain all vector info<br>Mac = PICT<br>Win32 = EMF)                                    |
| 2     | SaveAsDefault        | Default        | DefaultVector or DefaultRaster, depending on picture data Mac = PICT<br>Win32 vector=EMF<br>Win32 raster= BMP |
| 3     | SaveAsDefaultVector  | DefaultVector  | Platform's standard vector format<br>Mac = PICT<br>Win32 = EMF)   |
| 4     | SaveAsDefaultRaster  | DefaultRaster  | Platform's standard raster format<br>Mac = Raster PICT<br>Win32 = BMP)  |

### Macintosh Formats

| Value | Class Constant            | Format     | Description                         |
|-------|---------------------------|------------|-------------------------------------|
| 100   | SaveAsMacintoshPICT       | PICT       | Includes simple vector data.        |
| 250   | SaveAsMacintoshRasterPICT | RasterPICT | Flattens all vector data to pixels. |

### Win32 Formats

| Value | Class Constant   | Format | Description                                     |
|-------|------------------|--------|---|
| 300   | SaveAsWindowsWMF | WMF    | Windows Metafile format (old vector format).    |
| 301   | SaveAsWindowsEMF | EMF    | Extended Metafile format (newer vector format). |
| 350   | SaveAsWindowsBMP | BMP    | Windows bitmap format.                          |

### FolderItem Error Constants

To determine which error code was returned in the LastErrorCode property, you can test it against one of the FolderItem error constants. They are given in the following table:

| Error Code | Constant              | Description  |
|------------|-----------------------|--|
|            | NoError               | No error occurred.   |
| 100        | DestDoesNotExistError | Destination does not exist. You will get this error only on CopyFileTo and MoveFileTo. |

## FolderItem Class

---

| Error Code | Constant                 | Description               |
|------------|--------------------------|---------------------------|
| 101        | FileNotFoundException    | The File was not found.   |
| 102        | AccessDeniedException    | Access was denied.        |
| 103        | NotEnoughMemoryException | You ran out of memory.    |
| 104        | FileInUseException       | The file is in use.       |
| 105        | InvalidNameException     | You used an Invalid name. |

|                               |  |
|-------------------------------|--|
| <b>Cross Platform Formats</b> | No cross-platform formats are currently implemented.<br><br>Unrecognized formats or formats not supported for the built target will result in an <a href="#">UnsupportedFormatException</a> . The Message property of the exception will contain additional information in as to what went wrong.  |
| <b>Aliases</b>                | If a <b>FolderItem</b> is actually an alias to a <b>FolderItem</b> , the alias is automatically resolved when the <b>FolderItem</b> is accessed unless you use TrueChild and TrueItem (Aliases for Win32 do not get resolved with Item and Child). They return the item itself, even if it is an alias. Use the Alias property to determine whether the <b>FolderItem</b> is an alias.<br><br>For more information on File Types, see the chapter called “Working With Files” in the <i>User’s Guide</i> . |
| <b>Movies</b>                 | The <a href="#">OpenResourceMovie</a> method allows you to get a movie stored in a MooV resource inside a file or application (Macintosh only). Prior to QuickTime 4.0, the actual movie in a QuickTime movie file was stored in a MooV resource. In QuickTime 4.0 (and above), the movie is stored in the data fork.  |
| <b>Examples</b>               | This example puts the names of all the items on the Desktop that are stored on the boot volume into <a href="#">ListBox1</a> .   |

```
Dim i,n as Integer
Dim f as FolderItem
f=DesktopFolder
n=f.count
If n>0 then
  For i=1 to n
    ListBox1.addrow f.item(i).name
  Next
End if
Exception err as NilObjectException
MsgBox "File not Found"
```

This example uses MoveFileTo. The source file will be deleted and moved into the destination folder.

```
Dim f,g as FolderItem
g=Volume(0).Child("targetFolder")
f=Volume(0).Child("ReleaseNotes")
If f <> Nil and g <> Nil then
  f.MoveFileTo(g)
  MsgBox "success!"
else
  MsgBox "Files not found"
end if
```

The following example creates a text file, changes the Creator from the default creator of “R\*ch” to “ttxt”, and writes some data to the file.

```
Dim f as FolderItem
Dim stream as TextOutputStream
f=GetSaveFolderItem("TEXT","Daily Planet Staff")
stream=f.CreateTextFile
f.MacCreator="ttxt"
Stream.WriteLine ("Perry White")
Stream.WriteLine ("Lois Lane")
Stream.WriteLine ("Jimmy Olsen")
Stream.Close
```

This example displays an open-file dialog box that lets the user select a QuickTime movie. The QuickTime movie is then copied into the Movie property of a [MoviePlayer](#) control.

```
Dim f As FolderItem
f=GetOpenFolderItem("video/quicktime")
If f <> Nil then
  MoviePlayer1.movie=f.OpenAsMovie
End if
```

## FolderItem Class

---

This example copies all the files in a particular folder. The following code is a button's Action:

```
Dim origin, destination as FolderItem
origin=SelectFolder
If origin <> Nil then
    destination=SelectFolder
    If destination <> Nil then
        CopyFileOrFolder origin, destination
        MsgBox "Copy complete!"
    end if
end if
```

The CopyFileorFolder method is as follows:

```
Sub CopyFileorFolder (source as FolderItem, destination as FolderItem)
    Dim i as Integer
    Dim newFolder as FolderItem

    If source.directory then //it's a folder
        newFolder=destination.child(source.name)
        newFolder.createAsFolder
        For i=1 to source.count //go through each item
            If source.item(i).directory then
                //it's a folder
                CopyFileOrFolder source.item(i), newFolder
                //recursively call this
                //routine passing it the folder
            else
                source.item(i).CopyFileTo newFolder
                //it's a file so copy it
            end if
        next
    else //it's not a folder
        source.CopyFileTo destination
    end if
```

Using GetRelative

```
Dim s as String
Dim f,g as FolderItem
f=New FolderItem
g=f.GetRelative(f.GetSaveInfo(Volume(0).Child("Documents"),0))
Statictext2.text=g.Absolutepath
```

### See Also

[GetFolderItem](#), [GetOpenFolderItem](#), [GetSaveFolderItem](#), [SelectFolder](#), [Volume](#), [VolumeCount](#) functions; [BinaryStream](#), [FolderItemDialog](#), [OpenDialog](#), [SaveAsDialog](#), [SelectFolderDialog](#), [TextInputStream](#), [TextOutputStream](#) classes.

# FolderItemDialog Class

Base class for the [OpenDialog](#), [SaveAsDialog](#), and [SelectFolderDialog](#) classes, which allow you to create customized open, save, and select folder dialog boxes.

**Super Class** [Object](#)

## Properties

| Name                | Type                       | Description   |
|---------------------|----------------------------|---|
| ActionButtonCaption | <a href="#">String</a>     | Text of label for the Action button (e.g., Choose, Save, Open, etc., depending on context). Not necessarily the default button for the dialog.  |
| CancelButtonCaption | <a href="#">String</a>     | Text of label for the Cancel button.  |
| Filter              | <a href="#">String</a>     | One or more File Types, separated by semicolons, previously defined via the <a href="#">FileType</a> class or in the File Type Sets Editor in the IDE. Filter controls which files within <i>InitialDirectory</i> are visible.  |
| InitialDirectory    | <a href="#">FolderItem</a> | Full or relative path to directory whose contents are displayed when the dialog first appears. The Filter property controls which files within this directory are visible. On Windows, this defaults to the My Documents directory if no <a href="#">FolderItem</a> is specified. |
| Left                | <a href="#">Integer</a>    | Distance (pixels) of the left side of the dialog from the left side of the main screen.   |
| PromptText          | <a href="#">String</a>     | Help text that appears within the dialog. See illustrations for <a href="#">OpenDialog</a> , <a href="#">SaveAsDialog</a> , and <a href="#">SelectFolderDialog</a> .  |
| Result              | <a href="#">FolderItem</a> | Holds the result of calling ShowModal() or ShowModalWithin. If the user validates the dialog, Result contains the <a href="#">FolderItem</a> corresponding to the selection; otherwise, Result is <a href="#">Nil</a> .   |
| SuggestedFileName   | <a href="#">String</a>     | Default name of the file; appears as the default text in the filename enterable area.   |
| Title               | <a href="#">String</a>     | String that appears in the Title bar. This property is ignored on Macintosh.  |
| Top                 | <a href="#">Integer</a>    | Distance (pixels) of the top of the dialog from the top of the main screen.   |

## Methods

| Name        | Parameters | Description   |
|-------------|------------|---|
| ShowModal() |            | Opens the <a href="#">FolderItemDialog</a> . Returns a <a href="#">Folderitem</a> in Result if the user validates the dialog; otherwise Result is <a href="#">Nil</a> . |

## FolderItemDialog Class

| Name            | Parameters                       | Description   |
|-----------------|----------------------------------|---|
| ShowModalWithin | Parent as <a href="#">Window</a> | Displays the <b>FolderItemDialog</b> as a Sheet window on Mac OS X within the window specified by <i>Parent</i> . If the application is not running under Mac OS X, ShowModalWithin is equivalent to ShowModal. |

### Notes

Not all properties are available for all three subclasses of **FolderItemDialog** and some properties are not supported on Windows. The descriptions for each of the three subclasses have illustrations of the dialogs on each of the three platforms (Windows, Mac OS X, and Linux).

The following table clarifies the situation.

| Property            | Windows     |               |               | Macintosh   |               |               | Linux       |               |               |
|---------------------|-------------|---------------|---------------|-------------|---------------|---------------|-------------|---------------|---------------|
|                     | Open Dialog | SaveAs Dialog | Select Folder | Open Dialog | SaveAs Dialog | Select Folder | Open Dialog | SaveAs Dialog | Select Folder |
| Top & Left          | ✓           | ✓             | ✓             | ✓           | ✓             | ✓             | ✓           | ✓             | ✓             |
| PromptText          |             |               | ✓             | ✓           | ✓             | ✓             |             |               |               |
| Title               | ✓           | ✓             | ✓             | ✓           | ✓             | ✓             | ✓           | ✓             | ✓             |
| ActionButtonCaption |             |               | ✓             | ✓           | ✓             | ✓             |             |               |               |
| CancelButtonCaption |             |               |               | ✓           | ✓             | ✓             |             |               |               |
| SuggestedFileName   | ✓           | ✓             |               |             | ✓             |               |             | ✓             |               |
| Filter              | ✓           | ✓             |               | ✓           |               |               | ✓           | ✓             |               |
| InitialDirectory    | ✓           | ✓             | ✓             | ✓           | ✓             | ✓             | ✓           | ✓             | ✓             |

### Examples

See the examples and illustrations for the [OpenDialog](#), [SaveAsDialog](#), and [SelectFolderDialog](#) classes.

### See Also

[OpenDialog](#), [SaveAsDialog](#), [SelectFolderDialog](#) classes.

# Font Function

Used to access the names of fonts installed on the user's computer.

## Syntax

**result=Font(index)**

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">String</a>  | The name of the font whose index number is passed. <i>Result</i> is in the appropriate WorldScript encoding. |
| index  | <a href="#">Integer</a> | The number of the font.  |

## Notes

Fonts are accessed in alphabetical order where font 0 is the first font. Use the [FontCount](#) function to determine the number of fonts.

## Examples

This is an example of a function that determines if the font named passed is installed on the user's computer:

```
Function FontAvailable(FontName As String) as Boolean
Dim i,n as Integer
n=FontCount-1
For i=0 to n
If Font(i)=FontName Then
    Return True
End If
Next
Return False
```

This example dynamically creates a Font menu using a menu item array named FontFontName:

```
Dim m as MenuItem
Dim i,n as Integer
n=FontCount-1
FontFontName(0).Text=Font(0)
For i=1 to n
m=New FontFontName
m.Text=Font(i)
Next
```

## See Also

[FontCount](#), [FontsFolder](#) functions.

# FontCount Function

Used to determine the number of fonts installed on the user's computer.

## FontsFolder Function

---

### Syntax

**result=FontCount**

| Part   | Type                    | Name                           |
|--------|-------------------------|--------------------------------|
| result | <a href="#">Integer</a> | The number of fonts installed. |

### Notes

**FontCount** is useful when you need to build a list of available fonts or need to determine if a specific font is installed.

### Examples

See the example for the [Font](#) function.

### See Also

[Font](#), [FontsFolder](#) functions.

---

## FontsFolder Function

Used to access the Fonts folder.

### Syntax

**result=FontsFolder**

| Part   | Type                       | Description  |
|--------|----------------------------|--|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the Fonts folder. |

### Notes

Use the **FontsFolder** function to access the Fonts folder.

The **FontsFolder** function provides a way to access the Fonts folder that will work under different language systems, and will continue to work if the operating system is reorganized.

The [SpecialFolder](#) module provides access to many other special folders managed by the OS.

### Windows

On Windows, **FontsFolder** returns a reference to the Windows\Fonts directory.

### Macintosh

On Macintosh, **FontsFolder** returns a reference to the System/Library/Fonts directory.

### Linux

On Linux, **FontsFolder** returns [Nil](#).

### Examples

This example places all the names of the items in the Fonts folder in a [ListBox](#).

```
Dim i as Integer
Dim f as FolderItem
f=FontsFolder
For i=1 to f.count
    ListBox1.addrow f.item(i).name
Next
```

**See Also**

[ApplicationSupportFolder](#), [DesktopFolder](#), [Font](#), [FontCount](#), [PreferencesFolder](#), [StartupItemsFolder](#), [SystemFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

## For Each...Next Statement

Loops through the elements of a one-dimensional array.

**Syntax**

**For Each element [As datatype] In array**

[statements]

**Next**

| Part     | Description   |
|----------|---|
| element  | A variable of the same data type as <i>array</i> that refers to an element of <i>array</i> . The loop processes each value in <i>array</i> .  |
| datatype | Optional: The datatype of the array element. It can be any one-dimensional array. If you declare the datatype with the optional AS clause, you do not have to do so with a <a href="#">Dim</a> statement. The datatype must match <i>array</i> 's datatype. |
| array    | A one-dimensional array.  |

**Examples**

The following function adds up the numbers in the array passed to it.

```
Function SumStuff(values() as Double) as Double
    Dim sum, element as Double
    For Each element In values
        sum=sum+element
    Next
    Return Sum
```

To call the function, pass an array of [doubles](#) in a statement such as:

```
s=SumStuff(MyNumbers)
```

where s is declared as a [Double](#) and MyNumbers is an array of [Doubles](#).

**See Also**

[For...Next](#) statement, [Dim](#) statement.

# For...Next Statement

Executes a series of statements a specified number of times. The [For Each...Next](#) statement is a variation that executes once for each element of a one-dimensional array.

## Syntax

For loops have two syntaxes:

**For counter [AS datatype] = start To | DownTo end [Step value]**

**[Statements]**

**[Continue]**

**[Exit]**

**[Statements]**

**Next [counter]**

| Part                     | Description  |
|--------------------------|--|
| counter                  | A local variable that will be incremented each time the loop executes. It can be an <a href="#">Integer</a> , <a href="#">Single</a> , or <a href="#">Double</a> .   |
| datatype                 | Optional: Datatype of the counter. It will be <a href="#">Integer</a> , <a href="#">Single</a> , or <a href="#">Double</a> . If you use the optional AS clause, you do not have to declare the counter variable with a <a href="#">Dim</a> statement. When you use the AS clause instead of a <a href="#">Dim</a> statement, the counter variable goes out of scope at the end of the <b>For</b> statement rather than at the end of the method or function. |
| start                    | Initial value of counter.  |
| end                      | Final value of counter.  |
| value                    | Amount to increment or decrement counter each time statements are executed. Value is 1 unless otherwise specified. "To" indicates increment; "DownTo" decrement. If the value is negative, it causes the counter to decrement. It provides the same functionality as the optional DownTo keyword.  |
| statements               | Statements to be executed the specified number of times (until counter is greater than end, or less than end if DownTo is used).   |
| <a href="#">Continue</a> | If a <a href="#">Continue</a> statement is present, execution skips over the remaining statements in the <b>For</b> loop and resumes with a new iteration of the loop. Optional parameters of the <a href="#">Continue</a> statement allow you to specify which loop will iterate next, in the case of nested loops.   |
| <a href="#">Exit</a>     | If an <a href="#">Exit</a> statement is present, execution skips over the remaining statements in the loop and resumes with the statement following the Next statement.  |

**Notes**

The counter variable in a **For** statement can be declared inside the **For** statement rather than externally, as a local variable. For example, the code:

```
Dim i as Integer
For i=0 to 10
    //your code goes here
Next
```

can be rewritten as:

```
For i as Integer = 0 to 10
    //your code goes here
Next
```

In the second instance, the counter variable goes out of scope after the last iteration of the **For** loop. If you wanted to read the value of the counter after the end of the loop, you couldn't because it would already be deleted from memory. In the first instance, the counter variable remains in scope for the entire method in which it is declared. You could get or set the value of the counter variable anywhere in the method.

If **start**, **end**, or **step** is an expression that must be evaluated, the For loop does the evaluation each time the loop is executed, even if the expression evaluates to the same value for every iteration. Therefore, the loop:

```
For i=0 to FontCount-1
    .
    .
Next
```

will run more slowly than coding:

```
Dim nFonts as Integer
nFonts=FontCount-1
For i=0 to nFonts
    .
    .
Next
```

The size of the difference, of course, depends on the speed of the computer and the time required to evaluate the expression.

You can either increment or decrement the counter. If **step** is positive or zero, the loop executes until **counter** is greater than **end**.

To decrement the counter, use the second syntax using **DownTo** and a positive value of **step**.

Counter is incremented or decremented when the **Next** statement is reached.

Counter must be a local variable and must be an **integer**.

## Format Function

---

The [Exit](#) statement can be used to force the loop to stop prematurely. The [Exit](#) statement causes control to resume at the statement that follows **Next**.

**For...Next** statements can be nested to any level. However, each **For...Next** statement must have a different *counter*.

When a For loop runs, it takes over the interface, preventing the user from interacting with menus and controls. Ordinarily, this is of no concern. If, however, the loop is very lengthy, you can move the code for the loop to a separate [Thread](#), allowing it to execute in the background.

### Examples

This example uses the **For...Next** statement to test the values in an array and change those that are "Today" to "Tomorrow".

```
Dim i,last As Integer
last=Ubound(aDays)
For i=1 to last
  If aDays(i)="Today" Then
    aDays(i)="Tomorrow"
  End If
Next
```

The following example uses the **DownTo** keyword to decrement the counter:

```
Dim i as Integer
For i=5 DownTo 1 Step 1
  Beep
Next
```

### See Also

[Continue](#), [Do...Loop](#), [Exit](#), [For...Each](#), [While...Wend](#) statements.

---

## Format Function

Returns as a [string](#) a formatted version of the number passed based on the parameters specified. The **Format** function is similar to the way spreadsheet applications format numbers. Format will use the information based on the user's locale even if the user's locale is a Unicode-only locale.

### Syntax

**result=Format(number, formatSpec)**

| Part       | Type                   | Description  |
|------------|------------------------|--|
| result     | <a href="#">String</a> | The formatted number.                                      |
| number     | <a href="#">Double</a> | The number to be formatted.                                |
| formatSpec | <a href="#">String</a> | Defines the formatting to be applied to the number passed. |

**Notes**

The *formatSpec* is a string made up of one or more special characters that control how the number will be formatted:

| Character  | Description  |
|------------|--|
| #          | Placeholder that displays the digit from the value if it is present.   |
| 0          | Placeholder that displays the digit from the value if it is present. If no digit is present, 0 (zero) is displayed in its place. |
| .          | Placeholder for the position of the decimal point.   |
| ,          | Placeholder that indicates that the number should be formatted with thousands separators.  |
| %          | Displays the number multiplied by 100.   |
| (          | Displays an open paren.  |
| )          | Displays a closing paren.  |
| +          | Displays the plus sign to the left of the number if the number is positive or a minus sign if the number is negative.            |
| -          | Displays a minus sign to the left of the number if the number is negative. There is no effect for positive numbers.              |
| E or e     | Displays the number in scientific notation.  |
| \character | Displays the character that follows the backslash.   |

The absolute value of the number is displayed. You must use the + or - signs if you want the sign displayed.

Although the special formatting characters are U.S. characters, the actual characters that will appear are based on the current operating system settings. For example, Windows uses the settings in the user's Regional and Language Options Control Panel. Formatting characters are specified in similar ways on other operating systems.

The *formatSpec* can be made up of up to three formats separated by semicolons. The first format is the format to be used for positive numbers. The second format is the format to be used for negative numbers and the third format is the format to be used for zero.

**Examples**

The following are several examples that use the various special formatting characters.

| Format     | Number   | Formatted String |
|------------|----------|------------------|
| #.##       | 1.784    | 1.78             |
| #.0000     | 1.3      | 1.3000           |
| 0000       | 5        | 0005             |
| #%         | 0.25     | 25%              |
| ###,###.## | 145678.5 | 145,678.5        |
| #.##e+     | 145678.5 | 146e+5           |
| -#.##      | -3.7     | -3.7             |
| +#.##      | 3.7      | +3.7             |

## FrameColor Function

---

| Format              | Number | Formatted String |
|---------------------|--------|------------------|
| #.##;(#.##);\z\el\o | 3.7    | 3.7              |
| #.##;(#.##);\z\el\o | -3.7   | (3.7)            |
| #.##;(#.##);\z\el\o | 0      | zero             |

### Example

The following example returns the number 3560.3 formatted as \$3,560.30.

```
Dim s as String  
s=Format(3560.3, "\$###,##0.00")
```

---

## FrameColor Function

The currently selected color for outlining objects, as set by the operating system.

### Syntax

**result=FrameColor**

| Part   | Type                  | Description   |
|--------|-----------------------|---|
| result | <a href="#">Color</a> | The color used for drawing the outline of a <a href="#">RectControl</a> . |

### Notes

This value is useful when you are using [Canvas](#) controls to create custom controls.

When drawing objects, use this color for the object's frame. [ListBoxes](#), for example, use this color to draw the dark frame around the [ListBox](#).

This value can be changed by the user, so you should access this value in your Paint Event handler rather than storing the value.

### See Also

[DarkBevelColor](#), [DarkTingeColor](#), [FillColor](#), [HighlightColor](#), [LightBevelColor](#), [LightTingeColor](#), [TextColor](#) functions; [Color](#) data type.

---

## Function Statement

Declares the name, parameters, returned value, and code that form the body of a function (method).

### Syntax

**Function *name*((*parameterList*)) As *type***

**[local variable declarations]**

**[statements]**

**[Return]**

[statements]

[exception handlers]

[Finally]

| Part          | Description  |
|---------------|--|
| name          | Required. The name of the function (method); follows standard variable naming conventions.   |
| parameterList | Optional. List of values representing parameters that are passed to the function when it is called. Multiple parameters are separated by commas. |
| type          | The data type of the value returned by the function.   |

## Notes

The **Function** statement is used to define a method that can be used on the right side of an expression just like the built-in functions such as [Abs](#), [Len](#), etc. All executable code must be in a [Sub](#) or **Function** statement. A **Function** differs from a [Sub](#) in that a **Function** can be used on the right side of an expression and a [Sub](#) cannot. A function cannot be defined inside another [Sub](#) or function. A function executes each line of code from the top down. Once the last line of code is executed, control returns to the line that called the function.

You call a function by using its name followed by its parameters in parentheses.

All variables used in a function must be declared before they are used in a statement. Variables and parameters used in a function are local. Properties can be accessed from within any [Sub](#) or function. Local variables are variables that are created each time the function is run and destroyed when the function finishes. Consequently, they can only be accessed by the statements within the function. They are created by using the [Dim](#) statement from within a [Sub](#) or function. A [Dim](#) statement can be placed anywhere within the function.

The [Return](#) statement can be used to immediately return control to the statement that called the function and to pass back a value to the left side of the statement. If the [Return](#) statement is not called, the null value of the data type returned by the function is returned.

Exception handlers are statements that handle errors. See the [RuntimeException](#) class and the [Try](#) and [Exception](#) blocks for more information.

If your method does not need to return a value, you declare it as a [Sub](#). A [Sub](#) is a method that does not return a value. See the [Sub](#) statement for more information.

The variable that is returned by a function can be a “regular” single-element variable or an array. When you want to return one value, simply enter the data type of that variable in the Return Type field. To declare the variable as an array, place empty parentheses after the data type. For example, if you want to return an array of integers instead of only one integer, write “Integer()” instead of “Integer” as the function’s Return Type field.

## Function Statement

Here is a function declaration that returns an array.



If you need to return several values but not in the form of an array, you can use the [ByRef](#) keyword when you define the routine's parameters. Using [ByRef](#), you can return the results into the parameters.

A **Function** method can call itself, resulting in recursion. Too much recursion can lead to stack overflow errors.

Sometimes a function needs to do some cleanup work whether it is finishing normally or aborting early because of an exception. The optional [Finally](#) block at the end of the function serves this purpose. Code in this block will be executed even if an exception has occurred, whether the exception was handled or not.

**Examples** This example is a function that calculates area based on the length and height passed.

```
Function Area(Length as Double, Height as Double) As Double
    Dim theArea As Double
    theArea=Length*Height
    Return theArea
```

This example shows the Area function above written in a simpler form (without the extra local variable).

```
Function Area(Length as Double, Height as Double) As Double
    Return Length*Height
```

This example shows the Area function above being called and its returned value assigned to variable.

```
Dim a As Double
a=Area(10,10) //returns 100
```

**See Also** [Break](#), [Catch](#), [Exit](#), [Finally](#), [Raise](#), [Return](#), [Sub](#) statements; [RuntimeException](#) class.

# FunctionNotFoundException

A function declared using the [Declare](#) statement's "Soft" keyword could not be loaded.

**Super Class** [RuntimeException](#)

**Notes** A **FunctionNotFoundException** is thrown when a Soft [Declare](#) statement such as this:

```
Soft Declare Function getpid Lib "libc" () as Integer
```

fails to find the specified function or library and it cannot be loaded. If a "hard" [Declare](#) fails to load the function, the application will not run at all.

To determine whether the function can be loaded without actually using the Soft [Declare](#), use the [System](#) object's `IsFunctionAvailable` method. It returns a [Boolean](#) indicating whether the function can be loaded on the user's machine.

**See Also** [Declare](#) statement, [RuntimeException](#) class, [Exception](#), [Try](#) blocks; [System](#) object.

---

## GamelInputDevice Class

Manages a specific game input device. Not supported on Linux.

### Properties

| Name                | Type                    | Description  |
|---------------------|-------------------------|--|
| <b>Connected</b>    | <a href="#">Boolean</a> | If <a href="#">True</a> , the device is currently connected. <a href="#">False</a> indicates that the device is not connected. |
| <b>ElementCount</b> | <a href="#">Integer</a> | The number of elements the device has.   |
| <b>Index</b>        | <a href="#">Integer</a> | The index of this device in the <a href="#">GameInputManager</a> .   |
| <b>Name</b>         | <a href="#">String</a>  | The human-readable unique name of this device.   |

### Methods

| Name           | Parameters                       | Description  |
|----------------|----------------------------------|--|
| <b>Element</b> | Index as <a href="#">Integer</a> | Returns the <a href="#">GameInputElement</a> specified by <i>Index</i> . |

**Notes**

A **GamelInputDevice** is a specialized input device used for gaming, such as a joystick. REALbasic also reads the keyboard and the standard mouse as GameInputDevices. The Device and DeviceCount properties of the [GameInputManager](#) class enable you to read the user's GameInputDevices and the Name property of the **GamelInputDevice** class gives you the name of each device.

Each of the device's controls is a [GameInputElement](#).

## GameInputElement Class

---

**See Also** [GameInputElement](#), [GameInputManager](#) classes.

## GameInputElement Class

Manages an element of a [GameInputDevice](#). Not supported on Linux.

**Super Class** [Object](#)

### Properties

| Name          | Type                            | Description                                   |
|---------------|---------------------------------|---|
| <b>Name</b>   | <a href="#">String</a>          | The human-readable name of the input element. |
| <b>Device</b> | <a href="#">GameInputDevice</a> | The device that owns this input element.      |
| <b>Value</b>  | <a href="#">Integer</a>         | The current value of this element.            |

### Notes

A **GameInputElement** is a control on an input device used for gaming (or general user input via the keyboard and mouse). For example, the Fire button on a joystick control is an element, as are the standard mouse button and each of the keys on a keyboard. The Element method of the [GameInputDevice](#) class give you access to the elements of a specified device. You use the WaitForElement method of the [GameInputManager](#) class to detect input from an element.

**Example** See the examples for [GameInputManager](#).

**See Also** [GameInputDevice](#), [GameInputManager](#) classes.

---

## GameInputManager Class

Manages all the game input devices connected to the computer.

**Super Class** [Object](#)

### Methods

| Name        | Parameters                       | Description  |
|-------------|----------------------------------|--|
| DeviceCount |                                  | Returns as an <a href="#">Integer</a> the number of input devices.                     |
| Device      | Index as <a href="#">Integer</a> | Gets the device at the specified index and returns a <a href="#">GameInputDevice</a> . |

| Name           | Parameters                           | Description  |
|----------------|--------------------------------------|--|
| WaitForElement | timeout as<br><a href="#">Single</a> | Returns the <a href="#">GameInputElement</a> the user pressed. Waits for an element to change on any input device. <i>Timeout</i> is in seconds. |

**Notes**

Use the [GameInputManager](#), [GameInputDevice](#), and [GameInputElement](#) classes to manage input devices, such as joysticks, used for gaming. Each such device is a [GameInputDevice](#) and each of a device's controls, such as its buttons and joystick axes, is a [GameInputElement](#).

The system requirements for each platform are as follows:

**Windows**

DirectX 8 (or above) must be installed in order to access any input devices (including the keyboard).

**Mac "classic"**

Only the keyboard and the main (or only) mouse button are supported.

**Mac OS X**

The keyboard and mouse button are supported, as on Mac OS 8-9. In addition, you need to install the file "HID.Bundle" in the same folder as your application to access other game input devices, such as joysticks.

**Linux**

The GameInputManager class is not supported on Linux.

To support game input devices, create a single instance of the [GameInputManager](#) class. Use the Device and DeviceCount properties of the [GameInputManager](#) class to get the list of input devices. Each such device is a [GameInputDevice](#) and it has one or more [GameInputElements](#), such as a Fire button, scroll wheel, joystick axes, and so forth. You can get the list of devices by the Name property of the [GameInputDevice](#) class and each device's list of elements.

Use the WaitForElement method to get input from any element.

**Examples**

The following example determines which element the user is using as the Fire key. It assumes that there is a global property, mManager as [GameInputManager](#), and mFireButton as a [GameInputElement](#).

```
If mManager =Nil then
  mManager=New GameInputManager
End if
mFireButton=mManager.WaitForInput(3)
If mFireButton <> Nil then
  MsgBox "You're using "+mFireButton.Name+" as the Fire button."
End if
```

## GameInputManager Class

---

The following example handles the Fire button:

```
If mFireButton <> Nil and mFireButton.Value <> 0 then  
    //take action here  
End if
```

The first example shows how you can allow the user to configure his game devices and how actions on those devices will correspond to your game's actions. The second example shows how you can use a [GameInputElement](#) to determine whether it is time to do its corresponding game action. Note that in the second example, we are polling for the current state of that element (instead of getting an old value out of it).

The following example loads the names of the input devices into a [PopupMenu](#) control. It also assumes a global property mManager as **GameInputManager**.

```
Dim gCount,i as Integer  
If mManager=Nil then  
    mManager=New GameInputManager  
end if  
gCount=mManager.DeviceCount  
For i =0 to gCount-1  
    PopupMenu1.AddRow mManager.device(i).Name  
Next
```

The following example loads the names of the elements of a device into a [PopupMenu](#) control called ElementPop. It is in the Change event handler of the [PopupMenu](#) containing the list of devices.

```
Dim i, Maxi As Integer  
Dim device As GameInputDevice  
  
ElementPop.DeleteAllRows  
Device = mManager.Device(Me.ListIndex) //selected device in Device popup  
If Device <> Nil then  
    Maxi = Device.ElementCount  
    For i = 0 to Maxi-1  
        ElementPop.AddRow Device.Element(i).Name  
    next  
end if  
  
mElement = Nil
```

**See Also** [GameInputDevice](#), [GameInputElement](#) classes.

## GetAppleEventTarget Function

Used to bring up a dialog box that allows the user to choose the target application on his computer or a networked computer. The dialog box shows computers on the left and applications on the right. The applications list is labelled with the parameter *ListLabel*.

### Syntax

**result=GetAppleEventTarget (Prompt, ListLabel)**

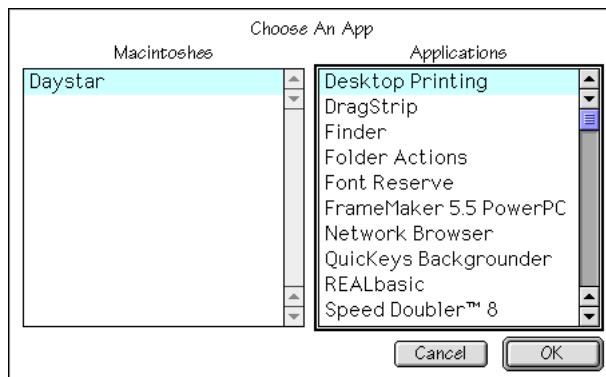
| Part      | Type                             | Description   |
|-----------|----------------------------------|---|
| Result    | <a href="#">AppleEventTarget</a> | Reference to target application.                              |
| Prompt    | <a href="#">String</a>           | Prompt text that will appear as the header of the dialog box. |
| ListLabel | <a href="#">String</a>           | Label for the list of applications in the dialog box.         |

### Notes

This function works under Mac OS “classic” only. In particular, it does not work under Carbon or Mac OS X.

### Example

The following example displays the Choose an App dialog box shown below.



```
Dim AE as AppleEventTarget
AE=GetAppleEventTarget("Choose An App","Applications")
```

## GetFolderItem Function

Used to access an item in a folder (directory).

## GetFolderItem Function

### Syntax

**result=GetFolderItem(path[,pathMode])**

| Part     | Type                       | Description  |
|----------|----------------------------|--|
| result   | <a href="#">FolderItem</a> | Represents the file that was opened.   |
| path     | <a href="#">String</a>     | Empty string or path. Ordinarily, <i>path</i> is the name of a document.   |
| pathMode | Class Constant             | <a href="#">FolderItem</a> class constant, PathTypeAbsolute, PathTypeShell, or PathTypeURL, indicating whether the path is a Shell path, an “ordinary” path, or path in the form of a URL. If you pass a Shell path, it must be an absolute path. If not, an <a href="#">UnsupportedFormatException</a> will result. See the ShellPath property of the <a href="#">FolderItem</a> class for information about shell paths. If you pass <a href="#">FolderItem.PathTypeURL</a> , it must begin with “file:///”. |

### Notes

The **GetFolderItem** function creates a [FolderItem](#) object for a specific item (application, document, or folder). **GetFolderItem** can be passed a file name for a specific item or an empty string. If you want to access an item via an absolute path, use the [Volume](#) function and the Child method of the [FolderItem](#) class.

If you intend to open an existing file with **GetFolderItem**, you should check the Exists property of the [FolderItem](#) to be sure that the file actually exists before accessing that [FolderItem's](#) properties.

Passing an empty string returns a [FolderItem](#) representing the folder the project is in. If you haven't saved the project, it returns the folder the REALbasic application is in.

For Mac OS X Mach-O applications, **GetFolderItem** returns the [FolderItem](#) for the directory containing the bundle instead of a [FolderItem](#) inside of the bundle.

**GetFolderItem** automatically resolves aliases when *filename* represents an alias. To prevent this, use [GetTrueFolderItem](#).

### Examples

This example displays the name of the folder that contains the REALbasic project in a message box.

```
Dim currentFolder as FolderItem  
currentFolder=GetFolderItem(" ")  
MsgBox currentFolder.name
```

The following example uses the Child method of the [FolderItem](#) class to get an item within the current directory:

```
Dim f as FolderItem  
f=GetFolderItem("Project Templates")
```

The following example uses the Parent property of the [FolderItem](#) class to get the parent directory for the directory that contains the application:

```
Dim f as FolderItem
f=GetFolderItem("").Parent
```

The following example opens a TIFF file in the same folder as the application (or the same folder as REALbasic if run from the IDE) and uses it as the background image ("backdrop") for a [Canvas](#) control:

```
Dim f as FolderItem
f=GetFolderItem("Zippy.tif")
If f.exists then
  Canvas1.backdrop=f.OpenAsPicture
end if
```

To get a reference using an absolute path, construct it using the [Volume](#) function and the Child method of the [FolderItem](#) class:. The example uses a [Try](#) block to handle the exception if the path is invalid.

```
Dim f as FolderItem
Try
  f=Volume(0).Child("Documents").Child("Schedule")
Catch Err as NilObjectException
  MsgBox "The path is invalid!"
End Try
```

## See Also

[FolderItem](#), [FolderItemDialog](#), [RuntimeException](#) classes; [GetTrueFolderItem](#) function; [NilObjectException](#) error; [Nil](#) object.

## GetFontTextEncoding Function

Used to get the text encoding to be used as the output encoding when you are going to display the text in that font. Most fonts will have the same encoding but fonts like Symbol have a different encoding.

### Syntax

**result=GetFontTextEncoding(FontName)**

| Part     | Type                         | Description   |
|----------|------------------------------|---|
| result   | <a href="#">TextEncoding</a> | Text encoding of FontName   |
| FontName | <a href="#">String</a>       | Name of a font. It returns the system default encoding if you pass one of the names (e.g., "System", "SmallSystem", "System.UTF8", etc.). |

## GetIndexedObjectDescriptor Function

---

**Notes** In REALbasic, the font names “System” and “SmallSystem” can be used in place of the names of real fonts when you specify the value of the TextFont property for visible controls. When you use these names, REALbasic will choose the default system or small system font for the operating system on which the application is running. Using System and SmallSystem avoids the need to specify different fonts for different operating systems. Use this feature in conjunction with setting font size equal to zero to tell REALbasic to choose the font size for each platform.

**Example** The following code returns the text encoding for the Symbol font.

```
Dim t as TextEncoding  
t=GetFontTextEncoding("Symbol")
```

**See Also** [TextEncoding](#) class.

---

## GetIndexedObjectDescriptor Function

Returns an [AppleEventObjectSpecifier](#) that is within another [AppleEventObjectSpecifier](#). The [AppleEventObjectSpecifier](#) is found by class and index.

**Syntax** `result=GetIndexedObjectDescriptor(DesiredClass, Object, index)`

| Part         | Type                                      | Description   |
|--------------|---|---|
| result       | <a href="#">AppleEventObjectSpecifier</a> | The object that was found.  |
| DesiredClass | <a href="#">String</a>                    | Indicates the class of <a href="#">AppleEvent</a> object you are looking for.   |
| Object       | <a href="#">AppleEventObjectSpecifier</a> | The source object. Pass <a href="#">Nil</a> to search at the application level. |
| Index        | <a href="#">Integer</a>                   | The number of the object you want (starting at 1).                              |

**See Also** [AppleEvent](#) class; [GetNamedObjectDescriptor](#), [GetOrdinalObjectDescriptor](#), [GetPropertyObjectDescriptor](#), [GetRangeObjectDescriptor](#), [GetStringComparisonObjectDescriptor](#), [GetTestObjectDescriptor](#) functions.

---

## GetInternetTextEncoding Function

When decoding mail or web pages you would use this function to get the text encoding rather than [GetTextEncoding](#). This function is the inverse of

[TextEncoding](#).InternetName. It returns the encoding associated with a standard name for that encoding as used on the Internet.

**Syntax**

**result=GetInternetTextEncoding(*InternetEncoding*)**

| Part             | Type                         | Description                        |
|------------------|------------------------------|------------------------------------|
| result           | <a href="#">TextEncoding</a> | Text encoding of InternetEncoding. |
| InternetEncoding | <a href="#">String</a>       | Internet text.                     |

**See Also**

[GetTextEncoding](#), [TextEncoding](#) functions.

## GetNamedObjectDescriptor Function

Returns an [AppleEventObjectSpecifier](#) that is within another [AppleEventObjectSpecifier](#). The [AppleEventObjectSpecifier](#) is found by class and name.

**Syntax**

**result=GetNamedObjectDescriptor(*DesiredClass*, *Object*, *Name*)**

| Part         | Type                                      | Description   |
|--------------|---|---|
| result       | <a href="#">AppleEventObjectSpecifier</a> | The object that was found.  |
| DesiredClass | <a href="#">String</a>                    | Indicates the class of <a href="#">AppleEvent</a> object you are looking for.   |
| Object       | <a href="#">AppleEventObjectSpecifier</a> | The source object. Pass <a href="#">Nil</a> to search at the application level. |
| Name         | <a href="#">String</a>                    | The name of the object you are looking for.                                     |

**See Also**

[AppleEvent](#) class, [GetIndexedObjectDescriptor](#), [GetOrdinalObjectDescriptor](#), [GetPropertyObjectDescriptor](#), [GetRangeObjectDescriptor](#), [GetStringComparisonObjectDescriptor](#), [GetTestObjectDescriptor](#) functions.

## GetOpenFolderItem Function

Used to access a document or application selected by the user via the standard open-file dialog box.

**Syntax**

**result=GetOpenFolderItem(*filter*)**

| Part   | Type                       | Description  |
|--------|----------------------------|--|
| result | <a href="#">FolderItem</a> | Represents the file that was opened by the user. Result will be <a href="#">Nil</a> if the user clicked the Cancel button. |

## GetOpenFolderItem Function

| Part   | Type                   | Description  |
|--------|------------------------|--|
| filter | <a href="#">String</a> | Semi-colon separated list of file types the user can open. The file types must be defined as REALbasic file types, either in the IDE in a File Types Set or with the <a href="#">FileType</a> class. |

### Notes

The **GetOpenFolderItem** function displays the standard open-file dialog box for the platform on which the REALbasic application is running. The [FolderItemDialog](#) class has the same purpose but allows for some customization.

The filter parameter is used to limit the types of files that the user can open to one or more of the file types defined via the [FileType](#) class or in the File Type Sets Editor in the IDE. The *filter* is a semi-colon separated list of file type names. For example, if you wanted the user to be able to open only text files and postscript files when making a particular call to the **GetOpenFolderItem** function, you would define two file types using either the File Type Sets Editor or the [FileType](#) class. You would then pass “application/text; application/postscript” (or the name you chose) as the filter to the **GetOpenFolderItem** function.

Only files whose type matches one of the file types passed in the *filter* will be displayed in the open file dialog box. If you want to display all files, you will need to add a file type to the project that uses “????” as its Macintosh Type.

The **GetOpenFolderItem** function returns a [FolderItem](#) that represents the file the user selected. You can then use the [FolderItem](#) to access various data about the file such as its name, full path, etc. See the [FolderItem](#) class for more information.

If the user clicks the Cancel button in the open file dialog box, the [FolderItem](#) will be [Nil](#). You can test for this by comparing the [FolderItem](#) with the [Nil](#) value. Accessing a [Nil FolderItem](#) will cause a [NilObjectException](#) error.

### Examples

This example illustrates how to use the [FileType](#) class do specify the types of files that can be opened by **GetOpenFolderItem**.

```
Dim TextTypes as New FileType
Dim ImageTypes as New FileType

TextTypes.Name= "Text Files"
TextTypes.Extensions= ".txt;.rtf;.doc"

ImageTypes.Name= "Image Files"
ImageTypes.Extensions = ".bmp;.jpg;.gif"

Dim f as FolderItem = GetOpenFolderItem(textTypes + ImageTypes)
If f <> Nil then
  MsgBox f.AbsolutePath
End if
```

**See Also**

[GetSaveFolderItem](#), [SelectFolder](#) functions; [FileType](#), [FolderItem](#), [FolderItemDialog](#) classes.

## GetOrdinalObjectDescriptor Function

Returns an [AppleEventObjectSpecifier](#) that is within another [AppleEventObjectSpecifier](#). The [AppleEventObjectSpecifier](#) is found by class and an ordinal key.

**Syntax**

*result=GetOrdinalObjectDescriptor(DesiredClass, Object, OrdinalKey)*

| Part         | Type                                      | Description   |
|--------------|---|---|
| result       | <a href="#">AppleEventObjectSpecifier</a> | The object that was found.  |
| DesiredClass | <a href="#">String</a>                    | Indicates the class of <a href="#">AppleEvent</a> object you are looking for.   |
| Object       | <a href="#">AppleEventObjectSpecifier</a> | The source object. Pass <a href="#">Nil</a> to search at the application level. |
| OrdinalKey   | <a href="#">String</a>                    | The scope to use when searching.  |

**Notes**

The ordinal keys are:

| Key  | Description |
|------|-------------|
| firs | First       |
| last | Last        |
| midd | Middle      |
| any  | Any         |
| all  | All         |

Ordinal keys are four characters so the “any” and “all” keys have a space following them.

**Example**

This example asks the Finder for a count of the currently running processes:

```
Dim a as AppleEvent
Dim i, count as Integer
a = NewAppleEvent("core", "cntr", "MACS")
a.MacTypeParam("kocl") = "prcs"
a.ObjectSpecifierParam("----") = GetOrdinalObjectDescriptor("prcs", nil, "all ")
If a.Send then
  count = Val(a.ReplyString)
End if
```

## GetPropertyObjectDescriptor Function

---

**See Also** [AppleEvent](#) class; [GetIndexedObjectDescriptor](#), [GetNamedObjectDescriptor](#), [GetPropertyObjectDescriptor](#), [GetRangeObjectDescriptor](#), [GetStringComparisonObjectDescriptor](#), [GetTestObjectDescriptor](#) functions.

## GetPropertyObjectDescriptor Function

Returns an [AppleEventObjectSpecifier](#) that refers to a property of the [AppleEventObjectSpecifier](#) passed.

**Syntax** *result=GetPropertyObjectDescriptor(Object, Name)*

| Part   | Type                                      | Description   |
|--------|---|---|
| result | <a href="#">AppleEventObjectSpecifier</a> | An object that refers to the Name property of Object.                           |
| Object | <a href="#">AppleEventObjectSpecifier</a> | The source object. Pass <a href="#">Nil</a> to search at the application level. |
| Name   | <a href="#">String</a>                    | The name of the property of Object to be returned.                              |

**See Also** [AppleEvent](#) class; [GetIndexedObjectDescriptor](#), [GetNamedObjectDescriptor](#), [GetOrdinalObjectDescriptor](#), [GetRangeObjectDescriptor](#), [GetStringComparisonObjectDescriptor](#), [GetTestObjectDescriptor](#) functions.

## GetQTCrossFadeEffect Function

Creates a cross-fade effect. Use [QTEffectSequence](#) to create an effect sequence using the cross-fade effect.

**Syntax** *result=GetQTCrossFadeEffect*

| Part   | Type                     | Description       |
|--------|--------------------------|-------------------|
| result | <a href="#">QTEffect</a> | Cross-fade effect |

**Example** This example creates a QuickTime movie from two pictures, p1 and p2, using the cross-fade effect.

```
Dim theEffect as QTEffect
Dim sequence as QTEffectSequence
theEffect=GetQTCrossFadeEffect
sequence=New QTEffectSequence(theEffect,p1,p2,96)
```

**See Also** [QTTrack](#), [QTVideoTrack](#), [QTEffect](#), [QTEffectSequence](#) classes; [GetQTSMPTEEffect](#) function.

## GetQTGraphicsExporter Function

Creates an [QTGraphicsExporter](#) object that uses the specified graphics format.

**Syntax** ***result=GetQTGraphicsExporter(Format)***

| Part   | Type                               | Description  |
|--------|------------------------------------|--|
| result | <a href="#">QTGraphicsExporter</a> | Exporter object.   |
| Format | <a href="#">String</a>             | Format used to save the graphic.<br>BMPf - BMP file format<br>PNGf - PNG file format<br>PNTG - MacPaint<br>8BPS - Photoshop<br>.SGI - SGI<br>TPIC - Targa<br>JPEG - Jpeg |

**Example** See the example for [QTGraphicsExporter](#).

**See Also** [QTGraphicsExporter](#) class.

## GetQTSMPTEEffect Function

Gets an SMPTEE effect.

**Syntax** ***result=GetQTSMPTEEffect(ID)***

| Part   | Type                     | Description                                  |
|--------|--------------------------|--|
| result | <a href="#">QTEffect</a> | The selected SMPTE effect.                   |
| ID     | <a href="#">Integer</a>  | ID of SMPTE effect (see the following table) |

Table 1: SMPTE Effects

| ID | Type | Effect           |
|----|------|------------------|
| 1  | Wipe | Slide horizontal |
| 2  | Wipe | Slide vertical   |
| 3  | Wipe | Top left         |
| 4  | Wipe | Top right        |

Table 1: SMPTE Effects (Continued)

| ID  | Type | Effect   |
|-----|------|--|
| 5   | Wipe | Bottom right   |
| 6   | Wipe | Bottom left  |
| 7   | Wipe | Four corner  |
| 8   | Wipe | Four box   |
| 21  | Wipe | Barn vertical  |
| 22  | Wipe | Barn horizontal  |
| 23  | Wipe | Top center   |
| 24  | Wipe | Right center   |
| 25  | Wipe | Bottom center  |
| 26  | Wipe | Left center  |
| 41  | Wipe | Diagonal left down   |
| 42  | Wipe | Diagonal right down  |
| 43  | Wipe | Vertical bow tie   |
| 44  | Wipe | Horizontal bow tie   |
| 45  | Wipe | Diagonal left out  |
| 46  | Wipe | Diagonal right out   |
| 47  | Wipe | Diagonal cross   |
| 48  | Wipe | Diagonal box   |
| 61  | Wipe | Filled V   |
| 62  | Wipe | Filled V right   |
| 63  | Wipe | Filled V bottom  |
| 64  | Wipe | Filled V left  |
| 65  | Wipe | Hollow V   |
| 66  | Wipe | Hollow V right   |
| 67  | Wipe | Hollow V bottom  |
| 68  | Wipe | Hollow V left  |
| 71  | Wipe | Vertical zig zag   |
| 72  | Wipe | Horizontal zig zag   |
| 73  | Wipe | Vertical barn zig zag  |
| 74  | Wipe | Horizontal barn zig zag  |
| 501 | Wipe | Random wipe — one of the 34 SMPTE wipe effects is chosen at random |
| 101 | Iris | Rectangle  |
| 102 | Iris | Diamond  |
| 103 | Iris | Triangle   |
| 104 | Iris | Triangle right   |

**Table 1: SMPTE Effects (Continued)**

| <b>ID</b> | <b>Type</b> | <b>Effect</b>  |
|-----------|-------------|--|
| 105       | Iris        | Triangle upside down   |
| 106       | Iris        | Triangle left  |
| 107       | Iris        | Arrowhead  |
| 108       | Iris        | Arrowhead right  |
| 109       | Iris        | Arrowhead upside down  |
| 110       | Iris        | Arrowhead left   |
| 111       | Iris        | Pentagon   |
| 112       | Iris        | Pentagon upside down   |
| 113       | Iris        | Hexagon  |
| 114       | Iris        | Hexagon side   |
| 119       | Iris        | Circle   |
| 120       | Iris        | Oval   |
| 121       | Iris        | Oval side  |
| 122       | Iris        | Cat eye  |
| 123       | Iris        | Cat eye slide  |
| 124       | Iris        | Round rect   |
| 125       | Iris        | Round rect side  |
| 127       | Iris        | 4 point star   |
| 128       | Iris        | 5 point star   |
| 129       | Iris        | 6 point star   |
| 130       | Iris        | Heart  |
| 131       | Iris        | Keyhole  |
| 502       | Iris        | Random iris — one of the 26 SMPTE iris effects is chosen at random |
| 201       | Radial      | Rotating top   |
| 202       | Radial      | Rotating right   |
| 203       | Radial      | Rotating bottom  |
| 204       | Radial      | Rotating left  |
| 205       | Radial      | Rotating top bottom  |
| 206       | Radial      | Rotating left right  |
| 207       | Radial      | Rotating quadrant  |
| 211       | Radial      | Top to bottom 180°   |
| 212       | Radial      | Right to left 180°   |
| 213       | Radial      | Top to bottom 90°  |
| 214       | Radial      | Right to left 90°  |
| 221       | Radial      | Top 180°   |

Table 1: SMPTE Effects (Continued)

| ID  | Type   | Effect   |
|-----|--------|--|
| 222 | Radial | Right 180°   |
| 223 | Radial | Bottom 180°  |
| 224 | Radial | Left 180°  |
| 225 | Radial | Counter rotating top bottom  |
| 226 | Radial | Counter rotating left right  |
| 227 | Radial | Double rotating top bottom   |
| 228 | Radial | Double rotating left right   |
| 231 | Radial | V Open top   |
| 232 | Radial | V Open right   |
| 233 | Radial | V Open bottom  |
| 234 | Radial | V Open left  |
| 235 | Radial | V Open top bottom  |
| 236 | Radial | V Open left right  |
| 241 | Radial | Rotating top left  |
| 242 | Radial | Rotating bottom left   |
| 243 | Radial | Rotating bottom right  |
| 244 | Radial | Rotating top right   |
| 245 | Radial | Rotating top left bottom right   |
| 246 | Radial | Rotating bottom left top right   |
| 251 | Radial | Rotating top left right  |
| 252 | Radial | Rotating left top bottom   |
| 253 | Radial | Rotating bottom left right   |
| 254 | Radial | Rotating right top bottom  |
| 261 | Radial | Rotating double center right   |
| 262 | Radial | Rotating double center top   |
| 263 | Radial | Rotating double center top bottom                                      |
| 264 | Radial | Rotating double center left right                                      |
| 503 | Radial | Random radial — one of the 39 SMPTE radial effects is chosen at random |
| 301 | Matrix | Horizontal matrix  |
| 302 | Matrix | Vertical matrix  |
| 303 | Matrix | Top left diagonal matrix   |
| 304 | Matrix | Top right diagonal matrix  |
| 305 | Matrix | Bottom right diagonal matrix   |
| 306 | Matrix | Bottom left diagonal matrix  |
| 310 | Matrix | Clockwise top left matrix  |

**Table 1: SMPTE Effects (Continued)**

| ID  | Type   | Effect   |
|-----|--------|--|
| 311 | Matrix | Clockwise top right matrix   |
| 312 | Matrix | Clockwise bottom right matrix  |
| 313 | Matrix | Clockwise bottom left matrix   |
| 314 | Matrix | Counterclockwise top left matrix                                       |
| 315 | Matrix | Counterclockwise top right matrix                                      |
| 316 | Matrix | Counterclockwise bottom right matrix                                   |
| 317 | Matrix | Counterclockwise bottom left matrix                                    |
| 320 | Matrix | Vertical start top matrix  |
| 321 | Matrix | Vertical start bottom matrix   |
| 322 | Matrix | Vertical start top opposite matrix                                     |
| 323 | Matrix | Vertical start left matrix   |
| 324 | Matrix | Horizontal start left matrix   |
| 325 | Matrix | Horizontal start right matrix  |
| 326 | Matrix | Horizontal start left opposite   |
| 327 | Matrix | Horizontal start right opposite matrix                                 |
| 328 | Matrix | Double diagonal top right matrix                                       |
| 329 | Matrix | Double diagonal bottom right matrix                                    |
| 340 | Matrix | Double spiral top matrix   |
| 341 | Matrix | Double spiral bottom matrix  |
| 342 | Matrix | Double spiral left matrix  |
| 343 | Matrix | Double spiral right matrix   |
| 344 | Matrix | Quad spiral vertical matrix  |
| 345 | Matrix | Quad spiral horizontal matrix  |
| 350 | Matrix | Vertical waterfall left matrix   |
| 351 | Matrix | Vertical waterfall right matrix  |
| 352 | Matrix | Horizontal waterfall left matrix                                       |
| 353 | Matrix | Horizontal waterfall right matrix                                      |
| 504 | Matrix | Random matrix — one of the 34 SMPTE matrix effects is chosen at random |
| 409 |        | Random effect — one of the 133 SMPTE effects is chosen at random       |

**Example** See the example for [QTVVideoTrack](#).

**See Also** [GetQTCrossFadeEffect](#) function; [QTEffect](#), [QTEffectSequence](#), [QTTrack](#), [QTVVideoTrack](#) classes.

## GetRangeObjectDescriptor Function

Returns an [AppleEventObjectSpecifier](#) containing a range of objects contained in the [AppleEventObjectSpecifier](#) passed.

### Syntax

**result=GetRangeObjectDescriptor(*DesiredClass*, *Object*, *RangeStart*, *RangeEnd*)**

| Part         | Type                                      | Description   |
|--------------|---|---|
| result       | <a href="#">AppleEventObjectSpecifier</a> | The object containing the range of objects specified.                           |
| DesiredClass | <a href="#">String</a>                    | The class of <a href="#">AppleEvent</a> object you are looking for.             |
| Object       | <a href="#">AppleEventObjectSpecifier</a> | The source object. Pass <a href="#">Nil</a> to search at the application level. |
| RangeStart   | <a href="#">AppleEventObjectSpecifier</a> | Indicates the start of the range.   |
| RangeEnd     | <a href="#">AppleEventObjectSpecifier</a> | Indicates the end of the range.   |

### See Also

[AppleEvent](#) class, [GetIndexedObjectDescriptor](#), [GetNamedObjectDescriptor](#), [GetOrdinalObjectDescriptor](#), [GetPropertyObjectDescriptor](#), [GetStringComparisonObjectDescriptor](#), [GetTestObjectDescriptor](#) functions.

---

## GetSaveFolderItem Function

Used to present the standard Save As dialog box to the user.

### Syntax

**result=GetSaveFolderItem(*filter*, *default file name*)**

| Part              | Type                       | Description  |
|-------------------|----------------------------|--|
| result            | <a href="#">FolderItem</a> | Represents the file the user wants to create.  |
| filter            | <a href="#">String</a>     | The file type to be assigned to the file, as defined in the File Type Sets Editor or via the <a href="#">FileType</a> class. |
| default file name | <a href="#">String</a>     | The name that should appear by default in the Save As dialog box.  |

### Notes

The **GetSaveFolderItem** function displays the standard Save As file dialog box, allowing the user to choose a location and enter a name for the file to be saved. The [SaveAsDialog](#) class provides the same functionality but allows for customization.

The **GetSaveFolderItem** function does not create the file. It simply returns a [FolderItem](#) that represents the potential file. To create the actual file, you will need to call the CreateBinaryFile or CreateTextfile method for the [FolderItem](#).

The filter should either be an empty string or the name of a file type as defined in the File Type Sets Editor or via the [FileType](#) class.

On Mac OS X, a Hide Filename Extension checkbox appears in the save-file dialog. The [FolderItem](#) returned has its ExtensionVisible property set according to the user's use of this checkbox.

## Examples

This example displays the save as file dialog box. A text file is then created and the text properties of three EditFields are written to the new file. Finally the file is closed.

```
Dim file As FolderItem
Dim fileStream as TextOutputStream
file=GetSaveFolderItem(" ","My Info")
If file<>Nil then
    fileStream=file.CreateTextFile
    fileStream.WriteLine namefield.text
    fileStream.WriteLine addressfield.text
    fileStream.WriteLine phonefield.text
    fileStream.Close
end if
```

## See Also

[GetOpenFolderItem](#), [SelectFolder](#) functions; [FileType](#), [FolderItem](#), [FolderItemDialog](#), [SaveAsDialog](#) classes.

# GetStringComparisonObjectDescriptor Function

Returns the [AppleEventObjectSpecifier](#) that matches the string comparison passed.

## Syntax

**result=GetStringComparisonObjectDescriptor (ComparisonKey, ComparisonForm, Field, Value)**

| Part           | Type                                      | Description  |
|----------------|---|--|
| result         | <a href="#">AppleEventObjectSpecifier</a> | The object that was found.   |
| ComparisonKey  | <a href="#">String</a>                    | The type of comparison to be performed. This string must be four characters in length.                                       |
| ComparisonForm | <a href="#">String</a>                    | The form of the comparison. For example, "prop" indicates that the comparison will be checking a property.                   |
| Field          | <a href="#">String</a>                    | The field the comparison will be performed on. The ComparisonForm parameter controls how the field parameter is interpreted. |

## GetTemporaryFolderItem Function

---

| Part  | Type                   | Description  |
|-------|------------------------|--|
| Value | <a href="#">String</a> | The value that the comparison will be performed against. |

### Notes

Valid comparison keys are:

| Comparison Key | Description              |
|----------------|--------------------------|
| =              | Equals                   |
| >              | Greater than             |
| >=             | Greater than or equal to |
| <              | Less than                |
| <=             | Less than or equal to    |

Comparison keys must be four characters. You must include spaces after the characters in the comparison key so that the string passed is four characters in length.

### See Also

[AppleEvent](#) class, [GetIndexedObjectDescriptor](#), [GetNamedObjectDescriptor](#), [GetOrdinalObjectDescriptor](#), [GetPropertyObjectDescriptor](#), [GetRangeObjectDescriptor](#), [GetTestObjectDescriptor](#) functions.

---

## GetTemporaryFolderItem Function

The **GetTemporaryFolderItem** function creates a [FolderItem](#) object in the current Temporary Folder.

### Syntax

*result*=GetTemporaryFolderItem

| Part   | Type                       | Description  |
|--------|----------------------------|--|
| result | <a href="#">FolderItem</a> | <a href="#">FolderItem</a> object that refers to a document in the Temporary Folder. |

### Example

The following code creates [FolderItem](#) in the active Temporary folder and displays its absolute pathname.

```
Dim f as FolderItem
f=GetTemporaryFolderItem
MsgBox f.absolutePath
```

### See Also

[FolderItem](#) class, [GetFolderItem](#), [TemporaryFolder](#) functions.

# GetTestObjectDescriptor Function

Returns an [AppleEventObjectSpecifier](#) built to perform some kind of test.

## Syntax

**result=GetTestObjectDescriptor(*DesiredClass*, *Object*, *Comparison*)**

| Part         | Type                                      | Description   |
|--------------|---|---|
| result       | <a href="#">AppleEventObjectSpecifier</a> | The object that was created.  |
| DesiredClass | <a href="#">String</a>                    | The class of <a href="#">AppleEvent</a> object you are looking for.             |
| Object       | <a href="#">AppleEventObjectSpecifier</a> | The source object. Pass <a href="#">Nil</a> to search at the application level. |
| Comparison   | <a href="#">AppleEventObjectSpecifier</a> | The object describing the actual test to be performed.                          |

## Notes

At present, [GetStringComparisonObjectDescriptor](#) is the only function that returns an [AppleEventObjectSpecifier](#) describing a test, but this will probably be extended in the future.

## See Also

[AppleEvent](#) class; [GetIndexedObjectDescriptor](#), [GetNamedObjectDescriptor](#), [GetOrdinalObjectDescriptor](#), [GetPropertyObjectDescriptor](#), [GetRangeObjectDescriptor](#), [GetStringComparisonObjectDescriptor](#) functions.

---

# GetTextConverter Function

Creates a [TextConverter](#) object for converting between two encodings. It also works if the input encoding and output encoding are the same.

## Syntax

**result=GetTextConverter(*inputEncoding*, *outputEncoding*)**

| Part           | Type                          | Description  |
|----------------|-------------------------------|--|
| result         | <a href="#">TextConverter</a> | Object of type <a href="#">TextConverter</a> , used to perform conversion from <i>InputEncoding</i> to <i>OutputEncoding</i> . |
| inputEncoding  | <a href="#">TextEncoding</a>  | Input text encoding.   |
| outputEncoding | <a href="#">TextEncoding</a>  | Output text encoding.  |

## Notes

In REALbasic, every string may internally have a record of its encoding as well as the bytes that constitute its actual text. In many cases, the encoding is unknown, but any string returned by a [TextConverter](#) will have a known encoding, and so will be treated properly by [EditFields](#) and [Graphics](#).DrawString. This is why it may sometimes be useful to get a [TextConverter](#) where the input and output encodings are the same; it provides a way to make sure that REALbasic knows what encoding a string represents.

## GetTextEncoding Function

---

### Example

The following example converts the text in an [EditField](#):

```
Dim c as TextConverter  
c=GetTextConverter( GetTextEncoding(&h500), getTextEncoding(0))  
EditField2.text=c.convert(EditField1.text)
```

### See Also

[ConvertEncoding](#), [DefineEncoding](#), [Encoding](#), [GetFontTextEncoding](#), [GetInternetTextEncoding](#), [GetTextEncoding](#) functions; [TextConverter](#), [TextEncoding](#) classes; [Encodings](#) object.

## GetTextEncoding Function

Returns a [TextEncoding](#) object based on user-specified parameters.

### Syntax

***result=GetTextEncoding (Base [,Variant, Format])***

| Part    | Type                         | Description  |
|---------|------------------------------|--|
| result  | <a href="#">TextEncoding</a> | TextEncoding object.   |
| Base    | <a href="#">Integer</a>      | The type of encoding.  |
| Variant | <a href="#">Integer</a>      | Specific variation of Base.  |
| Format  | <a href="#">Integer</a>      | Format—only used by Unicode for defining which format of Unicode you wish to use |

### Notes

See the [TextConverter](#) class for the Base and Variant codes. The codes for Format are as follows:

| Name               | Code |
|--------------------|------|
| Default Format     | 0    |
| Unicode16BitFormat | 0    |
| UnicodeUTF7Format  | 1    |
| UnicodeUTF8Format  | 2    |
| Unicode32BitFormat | 3    |

### Example

See the example for [GetTextConverter](#).

### See Also

[ConvertEncoding](#), [DefineEncoding](#), [Encoding](#), [GetTextConverter](#), [GetFontTextEncoding](#), [GetInternetTextEncoding](#) functions; [Encodings](#) object; [TextEncoding](#), [TextConverter](#) classes.

## GetTrueFolderItem Function

Returns a [FolderItem](#) that refers the true path (without resolving any aliases) of a file or folder.

### Syntax

**result=GetTrueFolderItem(*path[,pathMode]*)**

| Part     | Type                       | Description  |
|----------|----------------------------|--|
| result   | <a href="#">FolderItem</a> | <a href="#">FolderItem</a> that refers to the item referenced by <i>path</i> .   |
| path     | <a href="#">String</a>     | Pathname to the file or folder.  |
| pathMode | Class Constant             | <a href="#">FolderItem</a> class constant, PathTypeAbsolute, PathTypeURL, or PathTypeShell, indicating whether the path is a Shell path, an "ordinary" path, or a path in the form of a URL. If you pass a shell path, it must be an absolute path. If not, an <a href="#">UnsupportedFormatException</a> will result. See the ShellPath property of the <a href="#">FolderItem</a> class for information about shell paths. If you pass the PathTypeURL constant, the URL must begin with "file:///". |

### Note

If the item referenced by *path* does not exist, **GetTrueFolderItem** returns [Nil](#).

### Example

The following example compares **GetTrueFolderItem** with [GetFolderItem](#). While both functions are passed the same absolute path to an alias, f refers to the alias, while g refers to the actual application.

```
Dim f, g as FolderItem
f=GetTrueFolderItem("Hard Disk:Photoshop")
g=GetFolderItem("Hard Disk:Photoshop")
EditField1.text=f.Name //the alias to the app
EditField2.text=g.Name //the application
```

### See Also

[GetFolderItem](#) function; [FolderItem](#) class.

## GetUniqueIDObjectDescriptor Function

Returns a unique AppleEvent Object ID.

### Syntax

**result=GetUniqueIDObjectDescriptor(*DesiredClass, Object, ID*)**

| Part         | Type                                      | Description                 |
|--------------|---|-----------------------------|
| result       | <a href="#">AppleEventObjectSpecifier</a> | AppleEvent object specifier |
| DesiredClass | <a href="#">String</a>                    | AppleEvent Class            |
| Object       | <a href="#">AppleEventObjectSpecifier</a> | AppleEvent Object           |

## GOTO Statement

---

| Part | Type                   | Description |
|------|------------------------|-------------|
| ID   | <a href="#">String</a> | ID          |

**See Also** [AppleEventObjectSpecifier](#) class.

---

## GOTO Statement

Jumps to a statement label.

**Syntax** **GOTO** *label*

| Part  | Type                   | Description             |
|-------|------------------------|-------------------------|
| label | <a href="#">String</a> | Label to which to jump. |

**Note** The **GOTO** statement is included in REALbasic only for compatibility with other implementations of BASIC. Historically, **GOTO** was used extensively in unstructured BASIC to manage flow-of-control. At that time, a program consisted of a single master code segment, with flow-of-control managed by numerous **GOTO** statements that jumped execution around from one section of code to another. In time, such programs proved very hard to maintain because the underlying logic of the program was difficult to follow—especially after the original programmer left the project. The inability to maintain unstructured programs led to the development of event-driven, object-oriented programming environments such as REALbasic.

**Example** The following example displays a warning message if Checkbox1 is checked.

```
If checkbox1.value then
    GOTO myCode
End If

//Return used to keep RB from executing the labelled statement anyway
Return
myCode: //label to jump to
MsgBox "Unstructured programming is bad"
```

In place of all this, it is recommended that you use:

```
If Not CheckBox1.Value then Return
```

**See Also** [Exit](#), [Return](#) statements.

# GOTO Target Not Found Error

You used a label in a [GOTO](#) statement that is not used in the method. Therefore, the jump could not be executed.

## Example

The example from the [GOTO](#) keyword is modified so that there is no code that is identified by the label used in the [GOTO](#) statement.

```
If checkbox1.value then
    GOTO myCode
End If

//Return used to keep RB from executing the labelled statement anyway
Return
myLabel: //does not match label in GOTO
MsgBox "Unstructured programming is bad"
```

## See Also

[GOTO](#) statement.

# Graphics Class

**Graphics** class objects are used for drawing text, lines, rectangles, ovals, and pictures. Normally you use a graphics object in response to a Paint event, but you can also perform direct drawing by using the **Graphics** property of a [Canvas](#) control. Alternatively, you can create vector graphics using the subclasses of the [Object2D](#) class. You cannot create a meaningful **Graphics** object via the [New](#) command.

**Super Class** [Object](#)

## Properties

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| <b>Bold</b>      | <a href="#">Boolean</a> | If <a href="#">True</a> , text will appear in bold when using the <a href="#">DrawString</a> method.  |
| <b>Copies</b>    | <a href="#">Integer</a> | The value in the Copies field of the Print dialog box. This property is only meaningful when the graphics object was created by the <a href="#">OpenPrinterDialog</a> function. |
| <b>FirstPage</b> | <a href="#">Integer</a> | The value in the From field of the Print dialog box. This property is meaningful only when the graphics object was created by the <a href="#">OpenPrinterDialog</a> function.   |
| <b>ForeColor</b> | <a href="#">Color</a>   | The currently selected color for the Graphics object. This color will be used for the various drawing methods.  |

| Name              | Type   | Description  |
|-------------------|--|--|
| <b>Handle</b>     | <a href="#">Integer</a>  | Parameter is Type as <a href="#">Integer</a> . Gets the OS Handle for the passed Type. Pass a Graphics class constant to specify the Type. The class constants are as follows:<br>1-HandleTypeHDC. Gets the HDC on Windows.<br>2-HandleTypeCGrafPtr. Gets the QuickDraw CGrafPtr on Macintosh.<br>3-HandleTypeGdkDrawablePtr. Gets the GdkDrawable * on Linux.<br>4-HandleTypeGdkGCPtr. Gets the GdkGC * on Linux.<br>Handle will return zero if the requested handle Type is not available or is not supported. |
| <b>Height</b>     | <a href="#">Integer</a>  | The height in pixels of the parent object, typically a <a href="#">Canvas</a> control or a <a href="#">Window</a> . For example, if the graphics class object is in a <a href="#">Canvas</a> control, <i>Height</i> returns the height of the control.   |
| <b>Italic</b>     | <a href="#">Boolean</a>  | If <a href="#">True</a> , text will appear in italic when using the DrawString method.   |
| <b>LastPage</b>   | <a href="#">Integer</a>  | The value in the To field of the Print dialog box. This property is meaningful only when the graphics object was created by the <a href="#">OpenPrinterDialog</a> function.  |
| <b>PenHeight</b>  | <a href="#">Integer</a>  | The height in pixels used when drawing lines, ovals, and rectangles.   |
| <b>PenWidth</b>   | <a href="#">Integer</a>  | The width in pixels used when drawing lines, ovals, and rectangles.  |
| <b>Pixel</b>      | X as<br><a href="#">Integer</a> ,<br>Y as<br><a href="#">Integer</a> | Used to get and set the <a href="#">color</a> of a particular pixel.   |
| <b>TextAscent</b> | <a href="#">Integer</a>  | Returns the ascent of a line of text drawn with the current font.  |
| <b>TextFont</b>   | <a href="#">String</a>   | Used to get and set the current font.  |
| <b>TextHeight</b> | <a href="#">Integer</a>  | Contains the height of a line of text drawn with the current font.   |
| <b>TextSize</b>   | <a href="#">Integer</a>  | Used to get and set the current font size.   |
| <b>Underline</b>  | <a href="#">Boolean</a>  | If <a href="#">True</a> , text will appear in underline when using the DrawString method.  |

| Name           | Type                    | Description  |
|----------------|-------------------------|--|
| UseOldRenderer | <a href="#">Boolean</a> | If <a href="#">True</a> , rendering under Mac OS X is done via QuickDraw rather than Quartz and text drawing is done via QuickDraw in Mac OS X. The QuickDraw rendering engine is generally faster than Quartz, but Quartz provides the 'native' Mac OS X look. If you need faster drawing and are not concerned about Mac OS X anti-aliasing, consider using UseOldRenderer. The property can be modified at runtime. The default for a new object is <a href="#">False</a> . This property may also be useful if you need to make your graphics look exactly the same on both the OS 9 and Mac OS X operating systems. |
| Width          | <a href="#">Integer</a> | The width in pixels of the parent object, typically a <a href="#">Canvas</a> control or a <a href="#">Window</a> . For example, if the graphics class object is in a <a href="#">Canvas</a> control, Width returns the width of the control.   |

## Methods

The X and Y parameters of the following methods are the horizontal and vertical coordinates of the left-top corner of the object being drawn or cleared. The origin (0,0) is the top-left corner of the control or window in which the drawing is being done. For example 50,50 is 50 pixels to the right and 50 pixels down from the top-left of the window or control.

| Name            | Parameters  | Description  |
|-----------------|---|--|
| ClearRect       | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a> | Clears the rectangle described by the parameters passed by filling it with the background color of the parent window.                                |
| DrawCautionIcon | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a>  | Draws the operating system's Caution icon at the coordinates specified.  |
| DrawLine        | X1 as <a href="#">Integer</a> ,<br>Y1 as <a href="#">Integer</a> ,<br>X2 as <a href="#">Integer</a> ,<br>Y2 as <a href="#">Integer</a>      | Draws a line from X1, Y1 to X2, Y2 in the current color.   |
| DrawObject      | Object as <a href="#">Object2D</a> ,<br>[x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> ]                                   | Draw the passed <a href="#">Object2D</a> object. The optional parameters X and Y are offsets from the top-left corner of the <b>Graphics</b> object. |
| DrawOval        | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a> | Draws the outline of an oval in the current color.   |

| Name          | Parameters  | Description  |
|---------------|---|--|
| DrawPicture   | Image as Picture,<br>X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a><br>[,DestWidth as<br><a href="#">Integer</a> ,<br>DestHeight as<br><a href="#">Integer</a> ,<br>SourceX as<br><a href="#">Integer</a> ,<br>SourceY as<br><a href="#">Integer</a> ,<br>SourceWidth as<br><a href="#">Integer</a> ,<br>SourceHeight as<br><a href="#">Integer</a> ] | Draws the picture at the specified location. The optional parameters (surrounded by []) are used to copy a portion of the picture and for scaling the picture. DestWidth and DestHeight are used to change the scaling of the picture when it is drawn and default to the Width and Height of the picture. SourceX and SourceY default to 0 and are used to determine the upper-left coordinate you wish to copy from. SourceWidth and SourceHeight default to the Width and Height of the picture and are used to indicate the portion of the picture you wish to copy. |
| DrawPolygon   | Points() as<br><a href="#">Integer</a>  | Draws a polygon using the values in the 1-based array passed to as the x and y coordinates. Odd numbered array elements are X coordinates and even numbered array elements are Y coordinates.  |
| DrawNoteIcon  | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a>  | Draws the operating system's Note icon at the coordinates specified.   |
| DrawRect      | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a>   | Draws the outline of a rectangle in the current color. X and Y are the coordinates of the top-left corner.   |
| DrawRoundRect | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a> ,<br>OvalWidth as<br><a href="#">Integer</a> ,<br>OvalHeight as<br><a href="#">Integer</a>  | Draws the outline of a rounded rectangle in the current color.   |
| DrawStopIcon  | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a>  | Draws the operating system's Stop icon at the coordinates specified.   |
| DrawString    | Text as <a href="#">String</a> ,<br>X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a><br>[,WrapWidth as<br><a href="#">Integer</a> ]<br>[,Condense as<br><a href="#">Boolean</a> ]   | Draws the text at the specified location. The Y parameter specifies the baseline for the text. The optional <i>WrapWidth</i> parameter specifies the width (in pixels) at which text should wrap. If the optional <i>Condense</i> property is True, DrawString tries to kern the string to fit into the space specified by <i>WrapWidth</i> . If it can't fit it in, it will use an ellipsis ("...") to indicate that there is additional text that is not shown.  |

| Name            | Parameters   | Description   |
|-----------------|--|---|
| FillOval        | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a>  | Draws an oval filled with the current color.  |
| FillPolygon     | Points() as <a href="#">Integer</a>  | Draws a polygon using the values in the 1-based array passed as the x and y coordinates. Odd numbered array elements are X coordinates and even numbered array elements are Y coordinates. The polygon is filled with the current ForeColor.  |
| FillRect        | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a>  | Draws a rectangle filled with the current color.  |
| FillRoundRect   | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Width as <a href="#">Integer</a> ,<br>Height as <a href="#">Integer</a> ,<br>OvalWidth as <a href="#">Integer</a> ,<br>OvalHeight as <a href="#">Integer</a> | Draws a rounded rectangle filled with the current ForeColor.  |
| NextPage        |  | When printing a graphics object, this method will cause the current page to be printed and a new page to be created.  |
| StringDirection | Text as <a href="#">String</a>   | Returns an <a href="#">Integer</a> that indicates the direction in which the text is written. This is useful for non-Roman systems, especially Middle-Eastern languages. If you pass an empty string, it returns the system default string direction. If REALbasic doesn't support this function on the user's system, it will return -1; otherwise it will return either 0 for left to right or 1 or right to left. There are three class constants that you can use to test the string direction:<br>TextDirectionUnknown: -1<br>TextDirectionLeftToRight: 0<br>TextDirectionRightToLeft: 1 |
| StringHeight    | Text as <a href="#">String</a><br>WrapWidth as <a href="#">Integer</a>   | Returns the height of the text (based on the current font and fontsize) and a wrapWidth (pixels). The <i>WrapWidth</i> parameter specifies the width (in pixels) at which text should wrap.   |
| StringWidth     | Text as <a href="#">String</a>   | Returns the width of the text (based on the current font and font size) in pixels.  |

### Notes

By default, REALbasic uses the Quartz graphics engine on Mac OS X when you call methods of the **Graphics** class such as DrawLine, DrawRect, DrawOval, and so forth. The Quartz graphics engine uses anti-aliasing to make lines look smoother. The disadvantage is that it is slower than line drawing using the older Apple technology. For most applications, the Quartz engine will be fast enough and it does produce more attractive results on Mac OS X. However, if you need more speed, you can improve the performance of the drawing methods under Mac OS X by turning off the Quartz graphics engine and using the QuickDraw engine instead. Do this by setting the UseOldRenderer property of the **Graphics** class to [True](#) before you begin drawing.

### Examples

This example uses the Paint event handler of a [Canvas](#) control to draw the text “The quick brown fox” in Helvetica bold, italic, 18 point, 50 pixels from the top of and 10 pixels from the left side of the control.

```
Sub Paint(g As Graphics)
g.Bold=True
g.Italic=True
g.TextFont="Helvetica"
g.TextSize=18
g.DrawString "The quick brown fox", 10, 50
```

This example assigns the color of a particular pixel in a [Canvas](#) control to a variable.

```
Dim c as Color
c=Canvas1.Graphics.Pixel(10,10)
```

This example sets a particular pixel of a [Canvas](#) control to a specific [RGB](#) value.

```
Canvas1.Graphics.Pixel(10,10)=RGB(100,105,225)
```

This example draws a triangle in a [Canvas](#) control. It is placed in the Paint event. The parameter g as **Graphics** is passed into this event.

```
Dim Points(6) as Integer
Points(1)=10 //X of Point 1
Points(2)=10 //Y of Point 1
Points(3)=75 //X of Point 2
Points(4)=30 //Y of Point 2
Points(5)=10 //X of Point 3
Points(6)=125 //Y of Point 3
g.ForeColor=RGB(100,200,255)
g.FillPolygon Points
```

The following example assumes that you have imported a resource file into the Project Editor that contains a PICT resource whose ID is 129. The following line of code in the

[Canvas](#) control's Open event handler assigns the picture to the control's Backdrop property:

```
Me.backdrop=app.resourceFork.getpicture(129)
```

**Note:** Access to the resourcefork is supported only on Macintosh. Check the value of the [TargetMacOS](#) constant before attempting to access the resourcefork.

**See Also** [Canvas](#) control.

## Group2D Class

Used to group [Object2D](#) objects.

**Super Class** [Object2D](#)

### Properties

| Name  | Type                     | Description  |
|-------|--------------------------|--|
| Count | <a href="#">Integer</a>  | The number of <a href="#">Object2D</a> objects the group contains.   |
| Item  | <a href="#">Object2D</a> | Parameter is <i>Index</i> as <a href="#">Integer</a> . Returns the <a href="#">Object2D</a> object specified by <i>Index</i> . |

### Methods

| Name         | Parameters   | Description  |
|--------------|--|--|
| Append       | Object as <a href="#">Object2D</a>                                     | Appends the object passed to it.                           |
| Insert       | Index as <a href="#">Integer</a><br>Object as <a href="#">Object2D</a> | Inserts the object passed to it at the specified location. |
| Remove<br>OR | Index as <a href="#">Integer</a>                                       | Removes an object specified by its index.                  |
| Remove       | Object as <a href="#">Object2D</a>                                     | Removes an object specified by its reference.              |

### Notes

Use the Objects property of the [Picture](#) class to associate a **Group2D** object with a picture and the DrawObject method of the [Graphics](#) class to draw the object.

A **Group2D** is a container for [Object2D](#) shapes (including other **Group2Ds**). When a **Group2D** is rotated, translated, or scaled, all of its contents are updated accordingly. This means that all objects use the same coordinate system, but you can quickly transform an entire set of objects by simply transforming their group.

## Group3D Class

---

**Example** This method adds a [PixmapShape](#) and a [StringShape](#) to the **Group2D** object and displays it in the window. The picture, "h1", has been added to the Project Editor.

```
Dim px as PixmapShape
Dim s as StringShape
Dim d as New Group2D

px=New PixmapShape(h1)
d.append px

s=New StringShape
s.y=70
s.Text="This is what I call a REAL car!"
s.TextFont="Helvetica"
s.Bold=true
d.append s
graphics.drawobject d,100,100
```

**See Also** [ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Graphics](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

---

## Group3D Class

Used to group [Element3D](#) objects.

**Super Class** [Element3D](#)

### Methods

| Name   | Parameters                                 | Description   |
|--------|--|---|
| Append | obj as Element3D                           | Adds the <i>obj</i> object to the grouped object.   |
| Count  |  | The number of <a href="#">Element3D</a> objects in the grouped object. Returns an <a href="#">Integer</a> .   |
| Item   | Index as Integer                           | the <i>Index</i> item in the grouped object. The first item is numbered 0. Returns an <a href="#">Element3D</a> .   |
| Remove | Index as Integer<br>OR<br>obj as Element3D | Removes an object from the group. Removes the <i>Index</i> object or the object specified by <i>obj</i> . Note that since the Remove method is overloaded, you must always use parentheses around its argument. |
| Update |  | Updates the position and orientation of the <b>Group3D</b> object.  |

**Notes** The **Group3D** class is used to group 3D objects, but it is also an [Element3D](#) object, which allows you to use all the positioning and orienting methods as any other

[Element3D](#) object. When you add an object to a group, it moves and rotates with the group, but this is somewhat expensive, so using groups in this way should be done sparingly. The position and orientation of the grouped objects are updated to match the group when you call the Update method of the [RB3DSpace](#) control or when the group is drawn in a view. You can remove an object from a group either by its index or by the object reference itself.

**See Also**

[Bounds3D](#), [ColorList](#), [Element3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [RB3DSpace](#) control.

## GroupBox Control

Draws a group box control and typically used as a parent of other controls.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name      | Type                    | Description  |
|-----------|-------------------------|--|
| Bold      | <a href="#">Boolean</a> | Applies the bold style to the GroupBox caption.        |
| Caption   | <a href="#">String</a>  | The GroupBox's text.                                   |
| Italic    | <a href="#">Boolean</a> | Applies the italic style to the GroupBox caption.      |
| TextFont  | <a href="#">String</a>  | Name of the font used to display the GroupBox caption. |
| TextSize  | <a href="#">Integer</a> | Size of the font used to display the GroupBox caption. |
| Underline | <a href="#">Boolean</a> | Applies the underline style to the GroupBox caption.   |

### Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| MouseDown | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the GroupBox at the location passed in to x,y. Returns a <a href="#">Boolean</a> . <a href="#">Return True</a> if you are going to handle the MouseDown. The coordinates x and y are local to the control. Point 0,0 is the top-left corner of the entire control (to the left of the label and above the box that is drawn). |
| MouseDrag | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the GroupBox and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.  |

## GuessJapaneseEncoding Function

---

| Name      | Parameters   | Description  |
|-----------|--|--|
| MouseDown | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was released inside the GroupBox at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |

### Notes

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. This is done to make applications on all platforms behave consistently. The ampersand is used to denote the keyboard accelerator on the Windows platform.

### Example

See the [RadioButton](#) control.

### See Also

[RadioButton](#) control; [RectControl](#) class.

---

## GuessJapaneseEncoding Function

Guesses the text encoding used for a sample of Japanese text (2022\_JP, EUC, S-JIS, or plain US\_ASCII). If it is given a string whose encoding is already known to REALbasic, it returns that encoding. If the encoding is unknown, it assumes the encoding is some Japanese script, and attempts to guess which one.

### Syntax

**result=GuessJapaneseEncoding(text)**

| Part   | Type                         | Description                                   |
|--------|------------------------------|---|
| result | <a href="#">TextEncoding</a> | The guessed TextEncoding.                     |
| text   | <a href="#">String</a>       | The text whose text encoding will be guessed. |

### See Also

[ConvertEncoding](#), [DefineEncoding](#), [Encoding](#) functions; [TextEncoding](#) class, [Encodings](#) object.

---

## Hex Function

Returns as a [String](#) the hexadecimal version of the number passed.

### Syntax

**result=Hex(value)**

| Part   | Type                    | Description                                |
|--------|-------------------------|--|
| result | <a href="#">String</a>  | Value converted to hexadecimal.            |
| value  | <a href="#">Integer</a> | The number to be converted to hexadecimal. |

### Notes

If the value is not a whole number, the decimal value will be truncated.

You can specify binary, hex, or octal numbers by preceding the number with the & symbol and the letter that indicates the number base. The letter b indicates binary, h indicates hex, and o indicates octal.

VB Compatibility Note: VB rounds the value to the nearest whole number so the **Hex** function will probably be changed in a future release to do this as well.

**Examples**

Below are examples of various numbers converted to hex:

```
Dim hexVersion As String
hexVersion=Hex(5) //returns "5"
hexVersion=Hex(75) //returns "4B"
hexVersion=Hex(256) //returns "100"
```

**See Also**

[&b](#), [&h](#), [&o](#) literals; [Bin](#) and [Oct](#) functions.

## HighlightColor Function

Returns the operating system's highlight color.

**Syntax**

**result=HighlightColor**

| Part   | Type                  | Description                        |
|--------|-----------------------|------------------------------------|
| result | <a href="#">Color</a> | The user's chosen highlight color. |

**Notes**

On Windows, the highlight color can be modified via the Display Control Panel. In Mac OS X, the highlight color is set in the General System Preferences panel. On Mac OS “classic,” the highlight color is set using the Appearance Control Panel. On Linux, the highlight color is affected by the user's Window Manager and themes.

**See Also**

[DarkBevelColor](#), [DarkTingeColor](#), [FillColor](#), [FrameColor](#), [LightBevelColor](#), [TextColor](#) functions; [Color](#) data type.

## HSV Function

Returns a [Color](#) object based on the HSV (hue, saturation, value) color model.

**Syntax**

**result=HSV(*hue, saturation, value*)**

| Part   | Type                  | Description   |
|--------|-----------------------|---|
| result | <a href="#">Color</a> | An object that represents the color based on the hue, saturation, and value values. |

| Part       | Type                   | Description                           |
|------------|------------------------|---------------------------------------|
| hue        | <a href="#">Double</a> | The value of Hue in the color.        |
| saturation | <a href="#">Double</a> | The value of Saturation in the color. |
| value      | <a href="#">Double</a> | The value of Value in the color.      |

### Notes

The **HSV** function returns a color object based on the amounts of Hue, Saturation, and Value passed. These amounts are represented by doubles between 0 and 1.

Colors can also be created using the [RGB](#) or [CMY](#) models. You can also use the [&c](#) literal to specify a color using the RGB model.

### Examples

This example uses the **HSV** function to assign a [Color](#) to the FillColor property of a [Rectangle](#) control.

```
Rectangle1.FillColor=HSV(.8,.5,.75)
```

### See Also

[Color](#) data type, [CMY](#), [RGB](#), [SelectColor](#) functions; [&c](#) literal.

---

## HTMLViewer Control

Renders HTML and provides basic navigation features.

**Super Class** [RectControl](#)

### Properties

| Name                        | Type                    | Description  |
|-----------------------------|-------------------------|--|
| <a href="#">IsAvailable</a> | <a href="#">Boolean</a> | If <a href="#">True</a> , it indicates that the required support libraries required by the control were loaded. On Windows, HTML viewer uses Internet Explorer, on Mac OS X, it uses WebKit, the technology on which Safari is based; on Linux, it uses Mozilla. |

### Events

| Name                             | Parameters                    | Description   |
|----------------------------------|-------------------------------|---|
| <a href="#">CancelLoad</a>       | URL as <a href="#">String</a> | Returns a <a href="#">Boolean</a> . Return <a href="#">True</a> to cancel loading the page. This event fires before <a href="#">DocumentBegin</a> . |
| <a href="#">DocumentBegin</a>    | URL as <a href="#">String</a> | Fires when the HTML page is starting to load.   |
| <a href="#">DocumentComplete</a> | URL as <a href="#">String</a> | Fires when the HTML page is finished loading.   |

| Name                    | Parameters  | Description  |
|-------------------------|---|--|
| DocumentProgressChanged | URL as <a href="#">String</a> , percentageComplete as <a href="#">Integer</a>   | Fires when the progress has been updated. If <i>percentageComplete</i> is -1, the percentage cannot be determined.   |
| Error                   | errorNumber as <a href="#">Integer</a> , ErrorMessage as <a href="#">String</a> | Fires when an error occurs.  |
| NewWindow               |   | Returns an <b>HTMLViewer</b> . Fires when a new window should open up. Return a new instance of an <b>HTMLViewer</b> to load the page in that viewer or return <a href="#">Nil</a> when you do not want to allow the window to open. |
| SecurityChanged         | IsSecure as <a href="#">Boolean</a>   | Fires when the document's security has changed. For example, entering a secure web site will cause this event to fire. This event is not available on Linux.   |
| StatusChanged           | NewStatus as <a href="#">String</a>   | Fires when the Status text has changed for the window.   |
| TitleChanged            | newTitle as <a href="#">String</a>  | Fires when the Title of the Web page has been determined or has changed.   |

## Methods

| Name     | Parameters  | Description  |
|----------|---|--|
| Cancel   |   | Cancels any current operations.  |
| LoadPage | Source as <a href="#">String</a> , RelativeTo as <a href="#">FolderItem</a> | Loads the passed source HTML. Any links will be resolved relative to the passed <i>RelativeTo</i> <a href="#">FolderItem</a> . |
| LoadPage | File as <a href="#">FolderItem</a>  | Loads the passed source HTML file.   |
| LoadURL  | URL as <a href="#">String</a>   | Loads the passed URL.  |

## Example

To implement a simple web browser, create a window with a large **HTMLViewer** control. Set its LockLeft, LockTop, LockRight, and LockBottom properties so that it resizes as the user resizes the window. Use an [EditField](#) as the URL entry area and a [PushButton](#) as the browser's Go button. That is, the user clicks Go to display the URL entered into the [EditField](#).

In this example, the HTMLViewer is named “HTML”. The following code is in the Action event of the [PushButton](#):

```
HTML.LoadURL EditField1.text
```

You can use a [ProgressBar](#) to indicate that the web page is loading. In the **HTMLViewer**’s DocumentBegin event, initialize and show the [ProgressBar](#) with the code:

```
ProgressBar1.Value = 0  
ProgressBar1.Visible = True
```

In the DocumentProgressChanged event, increment the value of the [ProgressBar](#). This event handler is passed the value of PercentageComplete (0 to 100).

```
if percentageComplete = -1 then //if it cannot be determined  
  ProgressBar1.Maximum = 0 //display indeterminate progress  
Else  
  ProgressBar1.Maximum = 100  
End if  
  
ProgressBar1.Value = percentageComplete
```

In the Document Complete event handler, hide the [ProgressBar](#) with the line:

```
ProgressBar1.Visible = False.
```

When the title of the web page has been received, you can display it in the window’s Title bar using the **HTMLViewer**’s TitleChanged event. It is passed the new title in the [String](#) parameter newTitle. Update the window title with the line:

```
Title=newTitle
```

Use a second [EditField](#) to display the status of the load process. In the **HTMLViewer**’s StatusChanged event handler, set the value of this [EditField](#). This event handler is passed the current status in the [String](#) parameter, NewStatus. Display this string with the following line in the StatusChanged event:

```
EditField2.Text=NewStatus
```

If a new browser window is supposed to open, you need to insert some code to handle this event. For example, the user clicks a link that is supposed to display the new page

in another window. Use the NewWindow event handler to create the window. The following code assumes that the browser is contained in a window called MainWindow.

```
Dim w as New MainWindow
Title = "new Window" //Title property of new window
w.Show
Return w.HTML
```

## HTTPSecureSocket Class

Used to do secure transmissions via the HTTP protocol using the SSL or TLS protocols.

**Super Class** [SSLSocket](#)

**Properties** None

### Methods

| Name                | Parameters   | Description   |
|---------------------|--|---|
| ClearRequestHeaders |  | Clears all set request headers. See the description of the SetRequestHeaders method for information on headers.   |
| EncodeFormData      | Form as <a href="#">Dictionary</a>                                     | Returns a <a href="#">String</a> . Takes the key/value pairs from the specified Dictionary and formats them into a URL-encoded <a href="#">String</a> . The result can then be used in the URL to send data to a form with a GET action or as content for a POST.                               |
| Get                 | URL as <a href="#">String</a><br>[File as <a href="#">FolderItem</a> ] | Requests the content from the specified URL. The PageReceived event will execute when the data has been retrieved. If a <a href="#">FolderItem</a> is passed, the data will be downloaded to that file. The DownloadComplete event will execute when the response data has been retrieved.      |
| Post                | URL as <a href="#">String</a><br>[File as <a href="#">FolderItem</a> ] | Issues a POST command to the web server. The PageReceived event will execute when the response data has been retrieved. If a <a href="#">FolderItem</a> is passed, the data will be downloaded to that file. The DownloadComplete event will execute when the response data has been retrieved. |

| Name             | Parameters   | Description   |
|------------------|--|---|
| SetFormData      | Form as <a href="#">Dictionary</a>   | Takes the key/value pairs from the <a href="#">Dictionary</a> and converts it to a URL-encoded <a href="#">String</a> . This method also sets the action to POST. After using this method, the form can be sent to the server with the POST action.   |
| SetPostContent   | Content as <a href="#">String</a> ,<br>ContentType as <a href="#">String</a> | Sets the content data and content type to be sent to the server for a POST command.   |
| SetRequestHeader | Name as <a href="#">String</a> ,<br>Value as <a href="#">String</a>          | Sets the value of a request header. Request headers are sent to the server along with the action (GET, POST, etc.). They can contain data that the server might use to process your request. Some examples of common request headers are for things such as referring pages, the client name, OS CPU, and so forth. |

## Events

| Name             | Parameters  | Description   |
|------------------|---|---|
| Connected        |   | Executes when the connection is established with the web server.  |
| DownloadComplete | URL as <a href="#">String</a> ,<br>HTTPStatus as <a href="#">Integer</a> ,<br>Headers as <a href="#">Dictionary</a> ,<br>File as <a href="#">FolderItem</a> | Executes when a download is complete after using the Get method. URL contains the URL that was downloaded. File is the FolderItem the data was downloaded to, and Headers is a dictionary of the HTTP headers that were returned by the server. The HTTPStatus code is also passed to this event. These codes are used for messages such as "Page not found" (error 404), and so forth. |
| Error            | Code as <a href="#">Integer</a>   | Executes when an error occurs. All the standard <a href="#">TCPSocket</a> error codes are passed in Code. An additional error code is:<br>1- Download file could not be created.  |

| Name            | Parameters   | Description   |
|-----------------|--|---|
| PageReceived    | URL as <a href="#">String</a> ,<br>HTTPStatus as <a href="#">Integer</a> ,<br>Headers as <a href="#">Dictionary</a> ,<br>Content as <a href="#">String</a> | Executes when a new page has been retrieved from the server. It provides the URL that was requested and the HTTP headers from the web server. The HTTPStatus code is also passed to this event. These codes are used for messages such as "Page not found" (error 404), and so forth. |
| ReceiveProgress | BytesReceived as <a href="#">Integer</a> ,<br>TotalBytes as <a href="#">Integer</a>  | Executes periodically during the download process. It provides the current number of bytes received and the total number of bytes (if available).   |

**Notes**

Use the **HTTPSecureSocket** class to perform secure communications via the HTTP protocol. It is identical to the [HTTPSocket](#) class except that it is derived from the [SSLocket](#) class instead of the [TCPSocket](#) class. This means that you can use the Secure property of [SSLocket](#) to switch to HTTPS. When you set Secure to [True](#), the default port changes from 80 to 443.

If you use a constructor in a subclass of **HTTPSecureSocket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

**Example**

The following example posts a simple form:

```

Dim form as Dictionary
Dim socket1 as HTTPSecureSocket
Socket1.Secure=True

// create and populate the form object
form = New Dictionary
form.value("firstname") = "Bob"
form.value("lastname") = "Brown"

// setup the socket to POST the form
socket1.setFormData form
socket1.post "https://www.myformlocation.com/form.php"

```

**See Also**

[HTTPSocket](#), [SSLocket](#), [SocketCore](#), [TCPSocket](#) classes.

# HTTPSocket Class

Used to send and receive data via the HTTP protocol.

Super Class [TCPSocket](#)

## Properties

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| HTTPProxyAddress | <a href="#">String</a>  | The proxy address; required only if you are connecting via proxy.   |
| HTTPProxyPort    | <a href="#">Integer</a> | The proxy port; required only if you are connecting via proxy.  |
| HTTPStatusCode   | <a href="#">Integer</a> | Returns the resulting code from the server. This function can provide the status code when the socket is used in synchronous mode.  |
| Yield            | <a href="#">Boolean</a> | When you are using synchronous HTTP, your application may need to wait a noticeable amount of time to get or post a page. Setting <i>Yield</i> to <a href="#">True</a> allows for events and other code to execute while waiting for a page. <i>Yield</i> is relevant when you pass the optional <i>Timeout</i> parameter with <i>Get</i> , <i>Post</i> , or <i>GetHeaders</i> . Set <i>Yield</i> to <a href="#">True</a> just prior to a call to any of these methods. |

## Methods

| Name                | Parameters                         | Description   |
|---------------------|------------------------------------|---|
| ClearRequestHeaders |                                    | Clears all set request headers. See the description of the <i>SetRequestHeaders</i> method for information on headers.  |
| EncodeFormData      | Form as <a href="#">Dictionary</a> | Returns a <a href="#">String</a> . Takes the key/value pairs from the specified Dictionary and formats them into a URL-encoded <a href="#">String</a> . The result can then be used in the URL to send data to a form with a GET action or as content for a POST. |

| Name        | Parameters  | Description  |
|-------------|---|--|
| Get         | URL as <a href="#">String</a><br>[File as <a href="#">FolderItem</a> ]<br>[Timeout as <a href="#">Integer</a> ] | Requests the content from the specified URL. The PageReceived event will execute when the data has been retrieved. If a <a href="#">FolderItem</a> is passed, the data will be downloaded to that file. The DownloadComplete event will execute when the response data has been retrieved.<br>If the optional <i>Timeout</i> parameter (seconds) is used, REALbasic will wait the specified number of seconds for the page. When using the <i>Timeout</i> parameter, the page will be retrieved in a synchronous manner. If <i>Timeout</i> is set to zero, then there will be no timeout period and the socket will wait until it receives the page or gets an error. Set the Yield property to <a href="#">True</a> to allow for background activities while waiting. |
| GetHeaders  | URL as <a href="#">String</a><br>[Timeout as <a href="#">Integer</a> ]  | Requests only the page headers from the specified URL. If the optional <i>Timeout</i> parameter (seconds) is used, REALbasic will wait the specified number of seconds for the headers. When using the <i>Timeout</i> parameter, the headers will be retrieved in a synchronous manner. If <i>Timeout</i> is set to zero, then there will be no timeout period and the socket will wait until it receives the headers or gets an error. Set the Yield property to <a href="#">True</a> to allow for background activities while waiting.   |
| PageHeaders |   | Provides access to the headers received from the server. This can be used to access the headers in synchronous mode. Returns a <a href="#">String</a> .  |
| Post        | URL as <a href="#">String</a><br>[File as <a href="#">FolderItem</a> ]<br>[Timeout as <a href="#">Integer</a> ] | Issues a POST command to the web server. The PageReceived event will execute when the response data has been retrieved. If a <a href="#">FolderItem</a> is passed, the data will be downloaded to that file. The DownloadComplete event will execute when the response data has been retrieved.<br>If the optional <i>Timeout</i> parameter (seconds) is used, REALbasic will wait the specified number of seconds for a response. When using the <i>Timeout</i> parameter, the page will be posted in a synchronous manner. Set the Yield property to <a href="#">True</a> to allow for background activities while waiting.  |

| Name             | Parameters   | Description   |
|------------------|--|---|
| SetFormData      | Form as <a href="#">Dictionary</a>   | Takes the key/value pairs from the <a href="#">Dictionary</a> and converts it to a URL-encoded <a href="#">String</a> . This method also sets the action to POST. After using this method, the form can be sent to the server with the POST action.   |
| SetPostContent   | Content as <a href="#">String</a> ,<br>ContentType as <a href="#">String</a> | Sets the content data and content type to be sent to the server for a POST command.   |
| SetRequestHeader | Name as <a href="#">String</a> ,<br>Value as <a href="#">String</a>          | Sets the value of a request header. Request headers are sent to the server along with the action (GET, POST, etc.). They can contain data that the server might use to process your request. Some examples of common request headers are for things such as referring pages, the client name, OS CPU, and so forth. |

## Events

| Name                   | Parameters   | Description   |
|------------------------|--|---|
| AuthenticationRequired | Headers as <a href="#">InternetHeaders</a><br><a href="#">ByRef</a> Name as <a href="#">String</a><br><a href="#">ByRef</a> Password as <a href="#">String</a> | Executes when a site that requires authentication connects.   |
| Connected              |  | Executes when the connection is established with the web server.  |
| DownloadComplete       | URL as <a href="#">String</a> ,<br>HTTPStatus as <a href="#">Integer</a> ,<br>Headers as <a href="#">Dictionary</a> ,<br>File as <a href="#">FolderItem</a>    | Executes when a download is complete after using the Get or Post methods and passing the optional parameter. URL contains the URL that was downloaded. File is the <a href="#">FolderItem</a> the data was downloaded to, and Headers is a dictionary of the HTTP headers that were returned by the server. The HTTPStatus code is also passed to this event. These codes are used for messages such as "Page not found" (error 404), and so forth. |

| Name            | Parameters   | Description   |
|-----------------|--|---|
| Error           | Code as <a href="#">Integer</a>  | Executes when an error occurs. All the standard <a href="#">TCPSocket</a> error codes are passed in <i>Code</i> . An additional error code is:<br>1- Download file could not be created.  |
| ErrorCode       |  | Returns an <a href="#">Integer</a> error code. The error codes are the same as those for the <a href="#">TCPSocket</a> control, with the addition of the code -1: The socket timed out waiting for a page.  |
| HeadersReceived | Headers as <a href="#">internetHeaders</a><br>HTTPStatus as <a href="#">Integer</a>  | Executes when the headers have been retrieved from the server.  |
| PageReceived    | URL as <a href="#">String</a> ,<br>HTTPStatus as <a href="#">Integer</a> ,<br>Headers as <a href="#">Dictionary</a> ,<br>Content as <a href="#">String</a> | Executes when a new page has been retrieved from the server. It provides the URL that was requested and the HTTP headers from the web server. The HTTPStatus code is also passed to this event. These codes are used for messages such as "Page not found" (error 404), and so forth. |
| ReceiveProgress | BytesReceived as <a href="#">Integer</a> ,<br>TotalBytes as <a href="#">Integer</a>  | Executes periodically during the download process. It provides the current number of bytes received and the total number of bytes (if available).   |
| SendProgress    | BytesSent as <a href="#">Integer</a> ,<br>BytesLeft as <a href="#">Integer</a>   | Executes periodically during the upload process. It provides the number of bytes sent and the number of bytes left to send.   |

## Notes

Use the **HTTPSocket** class to send and receive data via the HTTP protocol. Since this is a subclass of the [TCPSocket](#) class, all the standard socket APIs are also available. To perform secure communications via HTTP, use the [HTTPSecureSocket](#) class instead. It is otherwise identical, except that it is subclassed from [SSLocket](#) instead of [TCPSocket](#). This makes all the options for encrypted communications available to [HTTPSecureSocket](#) objects.

If you use a constructor in a subclass of **HTTPSocket**, you must call the super class's constructor in your subclass's constructor. The subclass fail with error 107 when you try to download.

When using the optional Timeout parameter of the Get, GetHeaders, or Post methods the page will be received or posted in a synchronous manner. That is, REALbasic will

## If...Then...Else Statement

---

wait until the page has been received or posted before it proceeds. If Timeout is set to zero, the socket will wait indefinitely or until it receives an error.

To use synchronous mode, simply pass the optional parameter. For example,

```
Dim http as New HTTPSocket  
MsgBox http.Get("www.realsoftware.com",30)
```

When using synchronous mode, you can set the Yield property to [True](#) to tell the socket to yield time to other events and code to execute while waiting. Simply insert a statement such as

```
http.Yield=True
```

to allow background processes to execute while waiting.

### Example

The following example retrieves the specified URL:

```
Dim socket1 as New HTTPSocket  
socket1.get "http://www.realsoftware.com/"
```

The following example posts a simple form:

```
Dim form as Dictionary  
Dim socket1 as New HTTPSocket  
  
// create and populate the form object  
form = New Dictionary  
form.value("firstname") = "Bob"  
form.value("lastname") = "Brown"  
  
// setup the socket to POST the form  
socket1.setFormData form  
socket1.post "http://www.myformlocation.com/form.php"
```

### See Also

[HTTPSecureSocket](#), [SocketCore](#), [TCPSocket](#) classes.

---

## If...Then...Else Statement

Conditionally executes a group of statements, depending on the value of a [boolean](#) expression.

### Syntax

If *condition* Then

*statements*

[**Elself condition-n Then**

**elseifStatements**]...

[**Else**

**elseStatements**]

**End [If]**

**OR**

**If condition Then statement [Else] [statement]**

| Part             | Description   |
|------------------|---|
| condition        | Required. A <a href="#">boolean</a> , numeric, or <a href="#">string</a> expression that evaluates to <a href="#">True</a> or <a href="#">False</a> . |
| statements       | Optional. One or more statements that are executed if <i>condition</i> is <a href="#">True</a> .  |
| condition-n      | Optional. Same as <i>condition</i> .  |
| elseifStatements | Optional. One or more statements executed if the associated condition-n is <a href="#">True</a> .   |
| elseStatements   | Optional. One or more statements executed if no previous condition or condition-n expression is <a href="#">True</a> .                                |

## Notes

When executing an **If** statement, the condition is tested. If condition is [True](#), the statements associated with the **If** statement following the **Then** statement are executed. If condition is [False](#) and an **Else** clause follows, its statements will be executed. If condition is [False](#) and there is no **Else** clause or it is preceded by an **Elself** statement, the condition following the **Elself** statement is tested. After executing the statements following **Then**, **Elself** or **Else** execution continues with the statement that follows **End If**.

When writing an **If** statement, you can use a couple of shortcuts. First, you can write only the statements that are executed if the *condition* is [True](#). Then select the statements, display the Code Editor's contextual menu, and choose Wrap in If...End if. The Code Editor will then add an **If** statement above the selected lines and an End If statement below them. All you need to do is write the *condition*.

The other shortcut is this: Write the "If *condition*" portion of the statement and then press **Ctrl+Shift+Enter**. The Code Editor will then add the "Then" and the "End if" line.

## IllegalCastException Runtime Error

---

An **If** statement can be written on one line, provided the code that follows the **Then** and **Else** statements can be written on one line. Using this syntax, you omit the **End if** statement. For example, the following examples are valid:

```
If error=123 Then MsgBox "An error occured."  
If error=123 Then MsgBox "An error occured." Else MsgBox "Never mind!"
```

The following is not valid because the Then clause requires two lines:

```
If error=123 Then Beep MsgBox "An error occured."
```

Use the syntax shown in the first example.

**Examples** This example shows an **If** statement.

```
If error=-123 Then  
Beep  
MsgBox "Whoops! An error occured."  
End If
```

This example shows an **If** statement that includes the use of **ElseIf** and **Else** clauses.

```
Dim theNumber As Integer  
Dim digits As Integer  
theNumber=33  
If theNumber<10 Then  
    digits=1  
ElseIf theNumber<100 Then  
    digits=2  
ElseIf theNumber<1000 Then  
    digits=3  
Else  
    digits=4  
End If
```

**See Also** [Select Case](#) statement.

---

## IllegalCastException Runtime Error

Caused by an attempt to recast an object and then sending it a message that its original type cannot handle.

**Super Class** [RuntimeException](#) class

**Notes**

An **IllegalCastException** error occurs when you attempt to cast an object to a different type and call a method or access a property belonging only to the original type. For example, if you recast a [BevelButton](#) as a [PushButton](#) (see the example below) and then call the Pushbutton's Push method, an **IllegalCastException** will occur because a [BevelButton](#) isn't a [PushButton](#) and happens not to have a Push method.

**Example**

This example attempts to cast a [BevelButton](#) as a [PushButton](#) then call the Push method of the [PushButton](#) class. This example assumes that there is a [BevelButton](#) called "BevelButton1" in the same window where this code is being called:

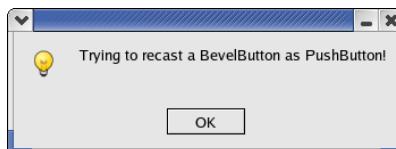
```
Dim c as Control
c = BevelButton1
PushButton(c).Push
```

When you test this code in the IDE, execution will stop at the last line and display an **IllegalCastException** error in your Code editor. If you test the code in a standalone application, the application will display a generic error message and quit when the user clicks OK.

You can handle the error by adding an [Exception](#) block to your code:

```
Dim c as Control
c = BevelButton1
PushButton(c).Push
Exception err
If err isA IllegalCastException then
  MsgBox "Trying to recast a BevelButton as PushButton!"
End if
```

When you test this code in a standalone application, the message box displays a more informative message.



The key difference is that, when the user accepts this dialog box, the application does not quit.

**See Also**

[RuntimeException](#) class; [Function](#), [Raise](#) statements; [Nil](#) keyword; [Exception](#), [Try](#) blocks.

# ImageWell Control

Used to display an image in a window.

## Super Class [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name  | Type                    | Description                         |
|-------|-------------------------|-------------------------------------|
| Image | <a href="#">Picture</a> | Picture displayed in the ImageWell. |

**Notes** If QuickTime is installed on the user's machine, the additional picture formats supported by QuickTime are also supported. This pertains to Windows and Macintosh only. Native support for the following formats is provided:

**Windows** On Windows, an ImageWell can display a BMP or PNG image.

**Macintosh** On Macintosh, an ImageWell can display a PICT image.

**Linux** On Linux, an ImageWell can display a BMP or PNG image.

**Example** The following line of code in the ImageWell's Open event displays a picture of a prancing stallion in the ImageWell. The image had been added to the Project Editor.

```
me.image=Ferrari
```

The following example implements drag and drop between two ImageWells, a BMP file dragged from the desktop to either ImageWell, and from one ImageWell to the other. For Macintosh, use the PICT file type instead of BMP.

In the Open event handler of the ImageWells, the two statements tell the controls to accept either a dragged picture or a dragged file of type BMP. The file type "image/x-bmp" was defined previously in the File Type Sets Editor or via the [FileType](#) class.

```
Me.AcceptPictureDrop  
Me.AcceptFileDrop\("image/x-bmp"\)
```

The DropObject event handler is:

```
Sub DropObject(Obj as DragItem)
If Obj.PictureAvailable then
    Me.Image=Obj.Picture
elseif Obj.FolderItemAvailable then
    Me.Image=Obj.FolderItem.OpenAsPicture
End if
```

The MouseDown event handler is:

```
Dim d as DragItem
d=NewDragItem(Me.left,Me.top,Me.width,Me.height)
d.picture=Me.Image
d.Drag //Allow the drag
```

**See Also** [RectControl](#) class.

## IndexOf Method

Used to search for a value in an array. Returns the index of the array element that contains the matching value.

### Syntax

**result=array.IndexOf(TargetValue[,StartingIndex])**

| Part                 | Type                      | Description   |
|----------------------|---------------------------|---|
| <b>result</b>        | <a href="#">Integer</a>   | Index ( <a href="#">Integer</a> ) of the array element that contains the value <i>targetValue</i> . If no match is found, <i>result</i> is set to -1.   |
| <b>array</b>         | Any valid data type       | The array that you want to search.  |
| <b>TargetValue</b>   | Same type as <i>array</i> | The value in the array that you want to find. It must be of the same data type as the array.  |
| <b>StartingIndex</b> | <a href="#">Integer</a>   | Optional. Array element to begin the search. If omitted, the search starts at the first array element. If you specify an array element outside the range of elements in <i>array</i> , an <a href="#">OutOfBoundsException</a> is raised. |

## Input Method

---

- Example** This example uses a [ComboBox](#) that has five items in its pop-up menu that have the names of the workdays of the week. The selected day is in its Text property. This example returns the index of the array element that matches the menu selection.

```
Dim days() as String  
days() = Array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")  
Dim dayNumber as Integer  
dayNumber = days.IndexOf(ComboBox1.Text)  
If dayNumber >= 0 then  
    MsgBox "You entered day number " + Str(dayNumber)  
else  
    MsgBox "You didn't enter the name of any weekday."  
end if
```

- See Also** [Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [Array](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

---

## Input Method

Retrieves a line from the terminal in console applications.

**Syntax** **result=Input**

| Part   | Type                   | Description  |
|--------|------------------------|--|
| result | <a href="#">String</a> | The line retrieved from the terminal. Input does not return the Newline character as part of <i>result</i> . |

**Notes** **Input** is equivalent to a call to the [ReadLine](#) method of the [StandardInputStream](#) class.

**See Also** [ConsoleApplication](#), [StandardInputStream](#), [StandardOutputStream](#) classes; [Print](#), [StdIn](#) methods; [TargetHasGUI](#) constant.

---

## Insert Method

Inserts a new element into an array.

**Syntax** **array.Insert index, value**

| Part  | Type                         | Description                                      |
|-------|------------------------------|--|
| array | array of any valid data type | The array in which to insert the new element.    |
| index | <a href="#">Integer</a>      | The position in array to insert the new element. |

| Part  | Type               | Description  |
|-------|--------------------|--|
| value | Same as array type | The value to be assigned to the new array element. |

**Notes**

All items below and including *index* are moved down one position to make room for the new element.

The **Insert** method works with one-dimensional arrays only.

**Examples**

This example inserts a new element at position 5 into the array *aNames* and assigns it the string "Bill".

```
aNames.Insert 5, "Bill"
```

**See Also**

[Append](#), [Array](#), [IndexOf](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [Dim](#), [Redim](#) statements; [UBound](#) function.

## InStr Function

Returns the position of the first occurrence of a *string* inside another *string*. The first character is numbered 1.

**Syntax**

*result*=InStr([*start*,] *source*, *find*)

OR

*result*=*stringVariable*.InStr([*start*,] *find*)

| Part                  | Type                    | Description   |
|-----------------------|-------------------------|---|
| <i>result</i>         | <a href="#">Integer</a> | The position of the first occurrence of <i>find</i> in <i>source</i> . If the search string cannot be located in <i>source</i> , InStr returns 0. |
| <i>start</i>          | <a href="#">Integer</a> | Optional position from which to begin searching the source string. One is the default if omitted.   |
| <i>source</i>         | <a href="#">String</a>  | Required. <a href="#">String</a> expression being searched.   |
| <i>find</i>           | <a href="#">String</a>  | Required. <a href="#">String</a> expression being sought.   |
| <i>stringVariable</i> | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .   |

**Notes**

If the *find* string is not found within the *source* string, 0 (zero) is returned. If the *find* string is an empty string, then *start* is returned. That is, **InStr**("This", "") returns 1 and **InStr**(3, "This", "") returns 3.

**InStr** is case-insensitive, even with accented Roman characters and non-Roman alphabets.

## InStrB Function

---

If you need to find the byte position of the find string within the source string, use the [InStrB](#) function or you need a case-sensitive function.

**Examples** This example uses the [InStr](#) function to locate a string within another string.

```
Dim first As Integer
first = InStr("This is a test", "t")
//returns 1
first = InStr("This is a test", "is")
//returns 3
first = InStr(4, "This is a test", "is")
//returns 6
first = InStr("This is a test", "tester")
//returns 0
```

**See Also** [Asc](#), [Chr](#), [Left](#), [Len](#), [Mid](#), [NthField](#), [Right](#), [Split](#), [StrComp](#) functions.

## InStrB Function

Returns the byte position of the first occurrence of a [string](#) inside another [string](#). The first character is numbered 1.

**Syntax** *result=InStrB([start] source, find)*

OR

*result=stringVariable.InStrB([start], find)*

| Part           | Type                    | Description  |
|----------------|-------------------------|--|
| result         | <a href="#">Integer</a> | The byte position of the first occurrence of find in source. If the search string cannot be located in source, InStrB returns 0. |
| start          | <a href="#">Integer</a> | Optional byte position from which to begin searching the source string. One is the default if omitted.                           |
| source         | <a href="#">String</a>  | Required. <a href="#">String</a> expression being searched.  |
| find           | <a href="#">String</a>  | Required. <a href="#">String</a> expression being sought.  |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .  |

### Notes

If the find string is not found within the source string, 0 (zero) is returned. **InStrB** is case-sensitive; it treats source as a series of raw bytes. It should be used instead of [InStr](#) when the string represents binary data or when your application will run in a one-byte character set (such as the US system) and you want case-sensitivity.

If you need to find the character position of the find string within the source string, use the [InStr](#) function.

**Examples**

This example uses the **InStrB** function to locate a [string](#) within another string.

```
Dim first As Integer
first = InStrB("This is a test", "T")
//returns 1
first = InStrB("This is a test", "t")
//returns 11
first = InStrB("This is a test", "is")
//returns 3
first = InStrB(4, "This is a test", "is")
//returns 6
first = InStrB("This is a test", "tester")
//returns 0
first = InStrB("This Is a test", "ls")
//returns 6
```

**See Also**

[AscB](#), [ChrB](#), [InStr](#), [LeftB](#), [LenB](#), [NthFieldB](#), [MidB](#), [RightB](#), [SplitB](#), [StrComp](#) functions.

## Int16 Data Type

An **Int16** is an intrinsic data type in REALbasic. It is a signed integer that uses two bytes of storage. It has a range of -32,768 to 32,767. The default value of an **Int16** is 0. REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <b>Int16</b>                                     | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to 2^64-1                     |

**See Also**

[Boolean](#), [Byte](#), [Color](#), [Double](#), [Int8](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt32](#) data types; [\\_](#), [±](#), [\\*](#), [/](#), [≤](#), [≤=](#), [≥](#), [≥=](#), [<](#), [>](#), [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

## Int32 Data Type

An **Int32** is an intrinsic data type in REALbasic. It is a signed integer that uses four bytes of storage. It has a range of -2,147,483,648 to 2,147,483,647. The default value of an **Int32** is 0.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>                            | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to 2^64-1                     |

### See Also

[Boolean](#), [Byte](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt32](#) data types; [-](#), [+](#), [\\*](#), [/](#), [<](#), [<=](#), [=](#), [>=](#), [>](#), [<>](#), [\](#), [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

---

## Int64 Data Type

An **Int64** is an intrinsic data type in REALbasic. It is a signed integer that uses eight bytes of storage. It has a range of range of -9223372036854775808 to +9223372036854775807. The default value of an **Int64** is 0.

### Notes

If either operand of an integer arithmetic expression is a 64-bit value, the result will be a 64-bit value. If both operands are 32 bits or smaller, the result will be a 32-bit value. If either operand of an arithmetic expression is signed, the result will be signed; the result will only be unsigned if both operands are unsigned.

If a numeric literal outside the range of an [Int32](#) are converted to **Int64s**. Prior to the introduction of the **Int64** data type, they were converted to floating point.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type                        | Number of Bytes | Range                           |
|----------------------------------|-----------------|---------------------------------|
| <a href="#">Int8 or Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>            | 2               | -32,768 to 32,767               |
| <a href="#">Int32 or Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>           | 8               | 0 to 2^64-1                     |

**See Also**

[Boolean](#), [Byte](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt32](#), [UInt64](#) data types; [±](#), [±](#), [\\*](#), [/](#), ≤, ≤, =, ≥, ≥, <, >, [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

## Int8 Data Type

An **Int8** is an intrinsic data type in REALbasic. It is a signed integer that uses one byte of storage. It has a range of -128 to 127. The default value of an **Int8** is 0.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types..

| Data Type                        | Number of Bytes | Range                           |
|----------------------------------|-----------------|---------------------------------|
| <a href="#">Int8 or Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>            | 2               | -32,768 to 32,767               |
| <a href="#">Int32 or Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>           | 8               | 0 to 2^64-1                     |

**See Also**

[Boolean](#), [Byte](#), [Color](#), [Double](#), [Int16](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt32](#), [UInt64](#), data types; [±](#), [±](#), [\\*](#), [/](#), ≤, ≤, =, ≥, ≥, <, >, [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

# Integer Data Type

**Integer** is an alias for a signed integer of the target architecture's word size. Currently, this is four bytes or [Int32](#) for all platforms supported by REALbasic. In other programming languages, REALbasic's **Integer** type is a long integer. Because integers are numbers, you can perform mathematical calculations on them. An **Integer** value uses 4 bytes of memory. The default value of an **Integer** is 0.

If either operand of an integer arithmetic expression is a 64-bit value, the result will be a 64-bit value. If both operands are 32 bits or smaller, the result will be a 32-bit value. If either operand of an arithmetic expression is signed, the result will be signed; the result will only be unsigned if both operands are unsigned.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>                            | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to 2^64-1                     |

## See Also

[Boolean](#), [Byte](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt32](#), [UInt64](#) data types; [-](#), [+](#), [\\*](#), [/](#), [≤](#), [≤=](#), [≡](#), [≥=](#), [≥](#), [≤>](#), [\](#), [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

---

# InternetHeaders Class

Used to manage HTTP and Email headers.

Super Class [Object](#)

## Methods

| Name         | Parameters   | Description           |
|--------------|--|-----------------------|
| AppendHeader | Name as <a href="#">String</a> , Value as <a href="#">String</a> | Appends a new header. |

| Name             | Parameters   | Description  |
|------------------|--|--|
| Count            |  | Returns the number of headers as an <a href="#">Integer</a> .  |
| Delete           | Name as <a href="#">String</a><br>OR<br>Name as <a href="#">String</a> ,<br>Index as <a href="#">Integer</a>   | Deletes the specified email header.  |
| DeleteAllHeaders |  | Deletes all headers.   |
| Name             | Index as <a href="#">Integer</a>   | Returns as a <a href="#">String</a> the name of the header specified by its index.   |
| NameCount        | Name as <a href="#">String</a>   | Returns as an <a href="#">Integer</a> the number of headers with the name passed to it. Internet headers can have multiple headers with the same name. |
| SetHeader        | Name as <a href="#">String</a> ,<br>Value as <a href="#">String</a>  | Sets the name and value of a header.   |
| Source           |  | Returns a <a href="#">String</a> containing the raw source text of the headers.  |
| Value            | Name as <a href="#">String</a><br>OR<br>Name as <a href="#">String</a> ,<br>Index as <a href="#">Integer</a><br>OR<br>Index as <a href="#">Integer</a> | Returns a <a href="#">String</a> containing the value of the specified header.   |

**See Also**

[EmailAttachment](#), [EmailHeaders](#), [EmailMessage](#), [HTTPSocket](#), [POP3Socket](#), [SMTPSocket](#) classes.

---

## InvalidParentException Error

**Super Class** [RuntimeException](#)**Notes**

You tried to get the parent of a control using the Parent property of the [Control](#) class, but its parent is in a different window. The parent control must be in the same window as the control or the parent is the containing window, not another control.

**See Also**

Parent property of the [Control](#) class; [Exception](#), [Try](#) blocks; [RuntimeException](#) class.

# IPCSocket Class

Used for interprocess communication between two applications on the same computer. Use [TCPsocket](#) for interprocess communication between applications on different computers.

Super Class [Object](#)

## Properties

| Name            | Type                    | Description  |
|-----------------|-------------------------|--|
| BytesAvailable  | <a href="#">Integer</a> | The number of bytes of data are available in the internal receive buffer.  |
| BytesLeftToSend | <a href="#">Integer</a> | The number of bytes left in the queue remaining to be sent.  |
| Handle          | <a href="#">Integer</a> | This is the IPCSocket's internal descriptor and it can be used with <a href="#">Declare</a> statements. The descriptor is platform-specific. On Windows it is a handle to a file if Handle is less than zero, the descriptor is not available.           |
| IsConnected     | <a href="#">Boolean</a> | Returns <a href="#">True</a> if connected to another application.  |
| LastErrorCode   | <a href="#">Integer</a> | The last error code. To determine which error occurred, check the value of LastErrorCode against the error class constants, as in the example.   |
| Path            | <a href="#">String</a>  | Unix style path to a directory that you have write access to, followed by a file name for your socket.<br>The name and path are completely arbitrary, but typically (and for maximum compatibility) you should use /tmp/myAppName or /var/tmp/myAppName. |

## Methods

| Name    | Parameters   | Description   |
|---------|--|---|
| Close   |  | Closes the connection. Call Close when you want to terminate the session.   |
| Connect |  | Attempts to make the connection using the specified Path.   |
| Listen  |  | Lists for a connection.   |
| Poll    |  | Polls the socket.   |
| Read    | Bytes as <a href="#">Integer</a> [, Encoding as <a href="#">TextEncoding</a> ] | Reads the amount of bytes specified from the internal receive buffer. The optional Encoding parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify a text encoding. |

| Name    | Parameters                                  | Description   |
|---------|---|---|
| ReadAll | [Encoding as <a href="#">TextEncoding</a> ] | Reads all the data from the internal buffer. The optional <i>Encoding</i> parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify a text encoding. |
| Write   | Text as <a href="#">String</a>              | The text that you are sending to the other application.   |

## Events

| Name          | Parameters | Description  |
|---------------|------------|--|
| Connected     |            | Occurs when a connection is established.   |
| DataAvailable |            | Occurs when data is available.   |
| Error         |            | An error occurred. Examine the value of the LastErrorCode property using the class constants, shown below. |

## Class Constants

These constants indicate which error occurred. To determine which error occurred, compare the value of LastErrorCode to these constants. They are the same error codes returned by [SocketCore](#).

| Error Code | Class Constant      |
|------------|---------------------|
| 100        | OpenDriverError     |
| 102        | LostConnection      |
| 103        | NameResolutionError |
| 105        | AddressInUseError   |
| 106        | InvalidStateError   |
| 107        | InvalidPortError    |
| 108        | OutOfMemoryError    |

## Notes

To set up interprocess communications, you first specify the path to the application that you wish to communicate with. This path is a Unix style path to a directory that you have write access to, followed by a the name for your socket. The name and path are completely arbitrary, but typically (and for maximum compatibility) you should use `/tmp/myAppName` or `/var/tmp/myAppName`.

You then try to connect your socket to an application. If there's already an instance of your application running on this machine, then you will connect to the other instance. However, if there are no other instances of this application running, then you will get a 103 error in the LastErrorCode property. In the example, the code will then attempt to listen, so the next instance of the application can connect to this one.

The example ensures that when you launch two instances of your application, they will automatically connect to one another. You can see this connection by placing a [MsgBox](#) in the Connected event.

When the connection is established, the **IPCSocket** will work like a regular [TCPsocket](#). When you want to terminate the connection, you call the Close method and the connection will be terminated.

The **IPCSocket** class implements the [Readable](#) and [Writeable](#) class interfaces. If you implement either or both of these interfaces, you must provide the methods specified by those class interfaces.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” on page 458 of the *User’s Guide*.

### Example

The following example establishes communications with another copy of the application. It is in the Open event of a window. The **IPCSocket** referred to in the code was added to the window by choosing Add ▶ IPCSocket from the window’s contextual menu.

```
' Pick the path that the IPCSocket should use.  
IPCSocket1.Path = "/tmp/myAwesomeApplication"  
  
' Let's see if we can connect to an application already  
' on this path. If we can, then cool.  
IPCSocket1.Connect  
While True  
    ' the socket  
    IPCSocket1.Poll  
  
    ' Check to see if we're connected, or if  
    ' there is an error  
    If IPCSocket1.IsConnected or IPCSocket1.LastErrorCode <> 0 then  
        Exit  
    end  
  
    ' Update the user interface  
    App.DoEvents  
wend  
  
' Check to see why we exited the loop.  
If IPCSocket1.LastErrorCode = IPCSocket1.NameResolutionError then  
    ' It's because no one else was listening, so let's  
    ' just start listening.  
    IPCSocket1.Listen  
End
```

### See Also

[SocketCore](#), [TCPsocket](#) classes.

## Is Operator

Compares two object references to determine whether they both refer to the same object.

### Syntax

**result=object1 Is object2**

| Part    | Type                    | Description  |
|---------|-------------------------|--|
| result  | <a href="#">Boolean</a> | Any container expecting a <a href="#">boolean</a> value. |
| object1 | <a href="#">Object</a>  | Any object name.   |
| object2 | <a href="#">Object</a>  | Another object name.                                     |

### Notes

The **Is** operator returns [True](#) if *object1* and *object2* actually refer to the same object. It checks identity, not contents, so it is not affected by the presence of a comparison operator. Use it the same way as the [=](#) operator.

### Example

In the following example, the **Is** operator returns [True](#).

```
Dim w1,w2 as Window
w1=Window1
w2=Window1
If w1 Is w2 then
//do something here
Else
//do something else here
End if
```

### See Also

[IsA](#) operator.

## IsA Operator

Used to determine the class of a particular object reference.

### Syntax

**result=object IsA object class**

| Part         | Type                    | Description  |
|--------------|-------------------------|--|
| result       | <a href="#">Boolean</a> | Any container expecting a <a href="#">boolean</a> value. |
| object       | <a href="#">Object</a>  | Any object name.   |
| object class | Object class            | Any object class or class interface.                     |

### Notes

If *object* is of type *object class*, the **IsA** operator returns [True](#); if not, it returns [False](#). If *Object* is [Nil](#), then **IsA** always returns [False](#).

For example, in the following code, **IsA** returns [False](#):

```
Dim t as TCPsocket
If t IsA TCPsocket then //returns False
//do something here
End if
```

You must instantiate the local variable t using the [New](#) operator so that it is non-[Nil](#):

```
Dim t as New TCPsocket
If t IsA TCPsocket then //returns True
//do something here
End if
```

The **IsA** operator returns [True](#) for the object's own class as well as the super class from which that class was derived, all the way to the [Object](#) class. **IsA** will report that all classes ultimately derive from [Object](#).

For example, suppose you create a subclass of [EditField](#) called SecureEditField, that disables the Cut and Copy commands in the Edit menu. If EditField1 is an instance of SecureEditField, the tests:

```
EditField1 IsA EditField
```

and

```
EditField1 IsA SecureEditField
```

both return [True](#).

You use the IsA operator to *cast* objects. With the **IsA** operator, you test whether an object is of a specific subclass and, if it is, cast it as that type to do something specific with it.

Here is an example that uses a [For](#) loop to cycle through all the controls in a window to test whether each control is an [EditField](#). If it is, it casts the control, gets its name and

the value of its Text property, and assigns the contents of the [EditField](#) to the field in a database table named *fieldname*.

```
Dim r as DatabaseRecord
Dim fieldname, fieldContents as String
.
.
For i = 0 to Self.ControlCount-1 //number of controls in window
If Self.control(i) IsA EditField then
    fieldname = EditField(control(i)).DataField //cast it
    //the text property assigned to the contents of that field
    fieldContents = EditField(control(i)).text
    r.column(fieldname)= fieldContents
end if
next
```

As you can see, the code is generic since it uses no references to specific [EditFields](#) or databases.

A second example of casting uses the [Window](#) function to get the frontmost window and then using **IsA** to determine which what kind of window it is (i.e., Window1, Window2) and then casts it as that type to access something specific about the window

```
If Window(0) IsA Window1 then
    Window1(Window(0)).PushButton1.enabled = True
End if
```

You also use the **IsA** operator in connection with a class interface. If a custom class implements a particular class interface, then **IsA** return [True](#). For example, if you create a class interface, myClassInterface, and custom control classes subclassed from the [StaticText](#), [EditField](#), and [Canvas](#) classes all implement myClassInterface, then the **IsA** operator will return [True](#) for each such test.

In this way, you can write generic code that examines all controls in a window to determine whether each control implements a particular class interface. If it does, then you can execute any methods of the class interface.

For example, if you wrote a class interface called FontInterfaceManger that updates the font displayed by a control using a method called UpdateTheFont, you use code such as this to call the method only for the controls in a window that implement the FontInterfaceManager class interface. This following example, which can be the Action event handler of a PushButton in a window, examines all the controls in the window

## IsContextualClick Function

---

and calls the UpdateTheFont method for those controls that implement the class interface FontInterfaceManager.

```
Dim i as Integer
For i=0 to Self.ControlCount-1
  If Self.Control(i) IsA FontInterfaceManager then
    FontInterfaceManager(Self.Control(i)).UpdateTheFont
  End If
Next
```

### Example

This example uses the **IsA** operator to determine if the variable p is a [PushButton](#) object. If it is, the caption property is assigned "OK".

```
If p IsA PushButton Then
  PushButton(p).Caption="OK"
End if
```

### See Also

Please refer to ["Creating Reusable Objects with Classes" on page 419](#) of the User's Guide.

## IsContextualClick Function

Used to display the contextual menu in response to the appropriate mouse gesture. It returns [True](#) if the MouseDown event occurred when the user right+clicked (Control-clicked on Macintosh), indicating that the user wishes to display the contextual menu for the object. It works only for windows and controls that have a MouseDown event handler.

### Syntax

**result=IsContextualClick**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | <a href="#">True</a> if the user right+clicks the object (Control-clicks on Macintosh) and <a href="#">False</a> if it was not. |

### Notes

This function is used to determine if the user wishes to view any available contextual menus for the object they are clicking on. For more information on implementing contextual menus, see the [ContextualMenu](#) Control.

You can also display an existing menu as a contextual menu by calling the Popup method of the [MenuItem](#) class when **IsContextualClick** returns [True](#).

**IsContextualClick** then displays the [MenuItem](#) as a contextual menu. The Popup method returns the selected item as a [MenuItem](#) and fires the selected item's Action event. If the selected item is handled by a MenuHandler that returns [True](#), then Popup will return [Nil](#).

If the user presses the Contextual Menu key on the keyboard (either Shift+F10 or the Contextual Key on some PC keyboards), REALbasic fires a MouseDown event and sets **IsContextualClick** to [True](#).

Contextual menus can also be implemented using the ConstructContextMenu and ContextualMenuAction event handlers of the [Window](#) and [RectControl](#) classes. If you use that approach, there is no need to use **IsContextualClick**. The ConstructContextMenu event fires when conditions are right to display a contextual menu.

## Examples

This example in a MouseDown event handler checks to see if the user has right+clicked, then displays a [contextual menu](#).

```
If IsContextualClick Then
    ContextualMenu1.Open //display the contextual menu
End If
```

## See Also

[ContextMenu](#) control; [MenuItem](#) class.

## IsNumeric Function

Indicates whether the value passed is numeric.

### Syntax

**result=IsNumeric(arg)**

| Part          | Type                    | Description   |
|---------------|-------------------------|---|
| <i>result</i> | <a href="#">Boolean</a> | <a href="#">True</a> if <i>arg</i> is numeric; <a href="#">False</a> otherwise. |
| <i>arg</i>    |                         | Variable, value, or object whose data type is being investigated.               |

### Notes

If the value passed to **IsNumeric** is a [String](#) that contains the string version of a number, then **IsNumeric** returns [True](#). For example,

```
IsNumeric("3.1416")
```

returns [True](#).

One usage of **IsNumeric** is to determine whether the user entered a string that can be converted to a number. For example,

```
IsNumeric( "1234.65e+10" ) //returns True
IsNumeric( "567e-10.5" ) //returns False
```

## See Also

[Boolean](#), [Color](#), [Double](#), [Integer](#), [Single](#), [String](#), [Variant](#) data types.

## Join Function

Assigns a value to a [String](#) variable by concatenating the elements of a [String](#) array.

### Syntax

**result=Join(sourceArray [,delimiter])**

| Part        | Type                         | Description  |
|-------------|------------------------------|--|
| result      | <a href="#">String</a>       | <a href="#">String</a> that results from concatenating all the elements of sourceArray, optionally separated by delimiter. |
| sourceArray | <a href="#">String</a> array | Source array whose elements will be used to created result.  |
| delimiter   | <a href="#">String</a>       | Optional delimiter used in separating the elements of sourceArray when creating result. The default is one space.          |

### Notes

**Join** takes a [String](#) array and concatenates the individual elements into a single [String](#) variable. You can pass an optional delimiter which will be inserted between the fields in the resulting [String](#). If no delimiter is passed, a single space will be used as the delimiter.

The [Split](#) function performs the opposite function. It takes a [String](#) and creates an array by parsing the string into array elements using a specified delimiter.

### Examples

This example concatenates a three-element array into the string "Anthony,Aardvark,Accountant".

```
Dim myArray(2) as String  
Dim s as String  
myArray=Array("Anthony", "Aardvark", "Accountant")  
//specifying the comma delimiter  
s=Join(myArray, ",") //creates "Anthony,Aardvark,Accountant"
```

### See Also

[Dim](#) statement; [Array](#), [NthField](#), [Split](#), [Ubound](#), functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

---

## Keyboard Object

An intrinsic object that represents the current state of the keyboard.

## Properties

| Name                                 | Type                    | Description  |
|--------------------------------------|-------------------------|--|
| <b>AlternateMenuShortCutKey</b>      | <a href="#">Boolean</a> | If <a href="#">True</a> , the alternate menu shortcut modifier is depressed when the current method or event handler began. This checks the Shift key on all platforms.  |
| <b>AltKey</b>                        | <a href="#">Boolean</a> | If <a href="#">True</a> , the Alt key on Windows and Linux and Option key on Macintosh was depressed when the current method or event handler began.   |
| <b>AsyncAlternateMenuShortCutKey</b> | <a href="#">Boolean</a> | If <a href="#">True</a> , the alternate menu shortcut key is depressed. This checks the Shift key on all platforms.  |
| <b>AsyncAltKey</b>                   | <a href="#">Boolean</a> | If <a href="#">True</a> , the Alt key (Windows and Linux) or Option key (Macintosh) is depressed.  |
| <b>AsyncControlKey</b>               | <a href="#">Boolean</a> | If <a href="#">True</a> , the Control key is depressed.  |
| <b>AsyncKeyDown</b>                  | <a href="#">Boolean</a> | Parameter is key as <a href="#">Integer</a> . If <a href="#">True</a> , the key whose KeyCode was passed, is depressed.  |
| <b>AsyncMenuShortcutKey</b>          | <a href="#">Boolean</a> | If <a href="#">True</a> , the menu short cut modifier key is depressed. This property tests the Control key on Windows and Linux and the Command key on Macintosh.   |
| <b>AsyncOSKey</b>                    | <a href="#">Boolean</a> | If <a href="#">True</a> , the OS key is depressed. On Windows and Linux, this is the key with the Windows flag; on Macintosh, it is the Command key.   |
| <b>AsyncShiftKey</b>                 | <a href="#">Boolean</a> | If <a href="#">True</a> , the Shift key is depressed.  |
| <b>ControlKey</b>                    | <a href="#">Boolean</a> | If <a href="#">True</a> , the Control key was depressed when the current method or event handler began.  |
| <b>MenuShortcutKey</b>               | <a href="#">Boolean</a> | If <a href="#">True</a> , the menu short cut modifier key was depressed when the current method or event handler began. This property tests the Control key on Windows and Linux and the Command key on Macintosh. |
| <b>OSKey</b>                         | <a href="#">Boolean</a> | If <a href="#">True</a> , the OAS key was depressed when the current method or event handler began. On Windows and Linux, this is the Windows flag key; on Macintosh, it is the Command key.                       |
| <b>ShiftKey</b>                      | <a href="#">Boolean</a> | If <a href="#">True</a> , the Shift key was depressed when the current method or event handler began.  |

## Keyboard Object

---

**Events**      None

### Methods

| Name    | Parameters                         | Description  |
|---------|------------------------------------|--|
| Keyname | keycode as <a href="#">Integer</a> | Returns a <a href="#">String</a> . The string is the name of the keycode passed. It is the character the key produces. In most cases, special keys will be named as follows: Esc, Tab, CapsLock, Shift, Control, Option, Alt, Command, Win, Return, Enter, Space, Fn, Delete, Backspace, Help, Insert, Home, PageUp, Del, End, PageDown, F1 through F15, PrintScreen, ScrollLock, Pause/Break, Clear, NumLock, KP=, KP/, KP*, KP-, KP0 through KP9, Up, Left, Down, and Right. |

### Notes

The **Keyboard** object is used to determine if particular keys are being pressed. The async version of each keyboard properties tell you the immediate state of the key. If you want to know the state of the key when an event was queued, you should use the non-async version of the properties. These properties are updated constantly during code execution.

### Windows

You can pass raw Win32 key codes to AsyncKeyDown. Add 1 to the raw keyCode and put it in the high word of the [Integer](#) you are passing. For example:

```
Keyboard.AsyncKeyDown((rawKeyCode + 1) * &hFFFF)
```

The ‘regular’ way of detecting a particular keyCode, illustrated by the first example in the Examples section, also works for Windows.

The following illustration shows the keycodes used with the **Keyboard** object. The **AsyncKeyDown** property returns **True** when the keycode passed to it is pressed.



## Examples

The following example tests whether the key for the letter “A” was pressed:

```
If Keyboard.AsyncKeyDown(&h00) then
    //do something with this key here
end if
```

This example monitors the arrow keys and detects when an arrow key is pressed. Place it in the Action event of a [Timer](#) and it will monitor the keyboard continuously.

```
If Keyboard.AsyncKeyDown(123) then
    //do something with the left arrow key
end if
If Keyboard.AsyncKeyDown(124) then
    //do something with the right arrow key...
end if
If Keyboard.AsyncKeyDown(125) then
    //do something with the down arrow key...
end if
If Keyboard.AsyncKeyDown(126) then
    //do something with the Up arrow key...
end if
```

# KeyChain Class

Gives you access to the default Mac OS “classic” and Mac OS X Keychains for your applications. The **KeyChain** class does not provide access to internet passwords.

Super Class [Object](#)

## Constructor

| Name     | Parameters                       | Description   |
|----------|----------------------------------|---|
| KeyChain | Index as <a href="#">Integer</a> | Gives you a reference to any <b>KeyChain</b> known by the KeyChain manager. The index value should be in the range from 0 to <a href="#">System.KeyChainCount</a> . |

## Properties

| Name   | Type                    | Description   |
|--------|-------------------------|---|
| Handle | <a href="#">Integer</a> | Returns a <i>Handle</i> to the <b>KeyChain</b> . Useful only if you want to refer to REALbasic's Keychains with <a href="#">Declare</a> statements. |

## Methods

| Name         | Parameters  | Description   |
|--------------|---|---|
| AddPassword  | kci as <a href="#">KeyChainItem</a><br>Password as <a href="#">String</a> | Adds the password to the <b>Keychain</b> and associates it with a <a href="#">KeyChainItem</a> . If it fails, it generates a <a href="#">KeyChainException</a> .  |
| FindPassword | <a href="#">ByRef</a> kci as <a href="#">KeyChainItem</a>                 | Returns a <a href="#">String</a> . Attempts to find the password for an Item in the given Keychain that matches <i>kci</i> . At the end of a successful call, <i>kci</i> will point to the found KeyChain Item. If it fails, it generates a <a href="#">KeyChainException</a> .   |
| Lock         | LockAll as <a href="#">Boolean</a>  | Locks the Keychain. If LockAll is <a href="#">True</a> , it will lock all the Keychains on the system. Note from Apple: There is usually no need for an application to ever call Lock. Unless your application is directly responding to a user's request for a keychain to be locked, it is recommended that you leave the keychain unlocked so that the user does not have to unlock it again in another application. If it fails, it generates a <a href="#">KeyChainException</a> . |

| Name   | Parameters                         | Description  |
|--------|------------------------------------|--|
| Unlock | Password as <a href="#">String</a> | Unlocks the keychain if the password is the empty string (""). The system displays the standard Unlock dialog, and the passed keychain will appear as the selected item in that dialog's pop-up menu. NOTE: because of the way Apple has implemented this, if you pass an incorrect password once to a particular keychain, the system will require human interaction from then on that keychain until the system is rebooted. If it fails, it generates a <a href="#">KeyChainException</a> . |

## Notes

The *keychain* is a system-wide facility on Mac OS 8.5 and above (including Mac OS X) to store account passwords for applications. By taking advantage of the built-in keychain facility, your users won't have to type their password if their keychain is unlocked. You should always ask the user before storing something in the keychain. An equivalent technology to the Mac OS KeyChain doesn't currently exist on other platforms, so the **KeyChain** class is supported only on Macintosh.

## Examples

The following example adds a [KeyChainItem](#) for an application and assigns a password:

```

Dim newItem as KeyChainItem
If System.KeyChainCount > 0 then

    newItem = New KeyChainItem
    'Indicate the name of the application
    newItem.ServiceName = "MyApplication"

    'Create a new keychain item for the application and assign the password
    System.KeyChain.AddPassword newItem, "SecretPassword"
Else
    Beep
    MsgBox "You don't have a key chain."
End if

Exception err as KeyChainException
MsgBox "Can't add item: " + err.Message

```

## KeyChain Class

---

The following example retrieves the password and displays it in a message box.

```
Dim ItemToFind as KeyChainItem
Dim password As String

ItemToFind = New KeyChainItem

'Indicate the name of the application whose keychain item you wish to find
ItemToFind.ServiceName = "MyApplication"

'get application's password from the system keychain
password = System.KeyChain.FindPassword(ItemToFind)
MsgBox "The password for this item is: " + password

Exception err as KeyChainException
MsgBox "Can't find item: " + err.Message
```

### See Also

[KeyChainItem](#) class; [KeyChainException](#) error; [System](#) object.

# KeyChainException Error

A method of the [KeyChain](#) or [KeyChainItem](#) classes failed. See the error code returned by **KeyChainException** to diagnose the problem.

**Super Class** [RuntimeException](#)

## Properties

| Name        | Type                    | Description                                       |
|-------------|-------------------------|---|
| ErrorNumber | <a href="#">Integer</a> | Indicates the type of error that was encountered. |

## Notes

The following tables shows the values of *ErrorNumber* and the associated text returned in the *Message* property.

| Error Number | Message                             |
|--------------|-------------------------------------|
| -128         | User Cancelled.                     |
| -25291       | Key Chain not available.            |
| -25292       | Key Chain read only.                |
| -25293       | Key Chain Authorization failed.     |
| -25294       | No such Key Chain.                  |
| -25295       | Invalid Key Chain.                  |
| -25296       | Duplicate Key Chain.                |
| -25797       | Key Chain Duplicate Callback.       |
| -25298       | Key Chain Invalid Callback.         |
| -25299       | Key Chain Duplicate Item.           |
| -25300       | Key Chain Item Not Found.           |
| -25301       | Key Chain Buffer Too Small.         |
| -25302       | Key Chain Data Too Large.           |
| -25303       | Key Chain No Such Attribute.        |
| -25304       | Key Chain Invalid Item Reference.   |
| -25305       | Key Chain Invalid Search Reference. |
| -25306       | Key Chain No Such Class.            |
| -25307       | No Default Key Chain.               |
| -25308       | Key Chain Interaction Not Allowed.  |
| -25309       | Key Chain Read Only Attribute.      |
| -25310       | Wrong Key Chain Version.            |
| -25311       | Key Chain Key Size Not Allowed.     |
| -25312       | Key Chain No Storage Module.        |

## KeyChainException Error

| Error Number | Message                          |
|--------------|----------------------------------|
| -25313       | Key Chain No Certificate Module. |
| -25314       | Key Chain No Policy Module.      |
| -25315       | Key Chain Interaction Required.  |
| -25316       | Key Chain Data Not Available.    |
| -25317       | Key Chain Data Not Modifiable.   |
| -25318       | Key Chain Create Chain Failed.   |

### Examples

The following example displays a message box if, for example, you try to create more than one [KeyChainItem](#) for the same application:

```
Dim newItem as KeyChainItem
If System.KeyChainCount > 0 then

    newItem = New KeyChainItem
    'Indicate the name of the application
    newItem.ServiceName = "MyApplication"

    'Create a new keychain item for the application and assign the password
    System.KeyChain.AddPassword newItem, "SecretPassword"
Else
    Beep
    MsgBox "You don't have a key chain."
End if

Exception err as KeyChainException
MsgBox err.Message+. Error Code: "+Str(err.errorNumber)
```

The following example uses an [Exception](#) block to display a message box if the application specified by ServiceName does not have a [KeyChainItem](#).

```
Dim itemToFind as KeyChainItem
Dim password As String

itemToFind = New KeyChainItem

'Indicate the name of the application whose keychain item you wish to find
itemToFind.ServiceName = "MyApplication"

'get application's password from the system keychain
password = System.KeyChain.FindPassword(itemToFind)
MsgBox "The password for this item is: " + password

Exception err as KeyChainException
MsgBox "Can't find item: " + err.Message
```

**See Also** [KeyChain](#), [KeyChainItem](#), [RuntimeException](#) classes; [System](#) object.

## KeyChainItem Class

Refers to a [Keychain](#) item.

**Super Class** [Object](#)

### Properties

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| AccountName | <a href="#">String</a>  | Contains the name of the account (required for adding, can be <a href="#">Nil</a> to find).  |
| Comment     | <a href="#">String</a>  | End user editable string containing comments for this Keychain item.   |
| Description | <a href="#">String</a>  | End user visible string describing this Keychain item.   |
| Handle      | <a href="#">Integer</a> | Contains the <b>KeyChainItem</b> reference, for use with Macintosh toolbox calls.  |
| Label       | <a href="#">String</a>  | End user editable string containing the label for this Keychain item.  |
| ServiceName | <a href="#">String</a>  | Contains the name of the service (required for adding, can be <a href="#">Nil</a> to find). To add a password for an application, set ServiceName to the application's name. |

### Methods

| Name   | Parameters | Description   |
|--------|------------|---|
| Delete |            | Deletes the <b>KeyChainItem</b> . You can change passwords in a <a href="#">KeyChain</a> by deleting the original item using this method and adding another item. |

### Notes

REALbasic **KeyChainItems** can access passwords for applications only, not internet passwords.

## KeyChainItem Class

---

**Examples** The following example adds a [KeyChainItem](#) for an application and assigns a password:

```
Dim newItem as KeyChainItem
If System.KeyChainCount > 0 then

    newItem = New KeyChainItem
    'Indicate the name of the application
    newItem.ServiceName = "MyApplication"

    'Create a new keychain item for the application and assign the password
    System.KeyChain.AddPassword newItem, "SecretPassword"
Else
    Beep
    MsgBox "You don't have a key chain."
End if

Exception err as KeyChainException
MsgBox "Can't add item: " + err.Message
```

The following example retrieves the password that was set in the previous example and displays it in a message box.

```
Dim itemToFind as KeyChainItem
Dim password As String

itemToFind = New KeyChainItem

'Indicate the name of the application whose keychain item you wish to find
itemToFind.ServiceName = "MyApplication"

'get application's password from the system keychain
password = System.KeyChain.FindPassword(itemToFind)
MsgBox "The password for this item is: " + password

Exception err as KeyChainException
MsgBox "Can't find item: " + err.Message
```

**See Also** [KeyChain](#) class; [KeyChainException](#) error; [System](#) object.

# KeyNotFoundException Error

You tried to access an item in a [Dictionary](#) using a key that is not in the [Dictionary](#).

**Super Class** [RuntimeException](#).

**Example** The following [For](#) loop produces a **KeyNotFoundException** error when the loop runs with the counter equal to 2. The value of 2 — which is a [Variant](#), not an [Integer](#) — has just been removed.

```
Dim d as New Dictionary
Dim i as Integer
d.Value(1)="ID"
d.Value(2)="Lois Lane"
d.Value(3)="Reporter"
d.Value(4)=84000
d.Remove(2)
For i=1 to d.Count //fails when i=2
    ListBox1.Addrow d.value(i)
Next
```

To handle the error, add an [Exception](#) block to the end of the method, such as:

```
Exception err as KeyNotFoundException
MsgBox "You tried to access a nonexistent item!"
```

**See Also** [Dictionary](#), [RuntimeException](#) classes; [Exception](#) block.

## Left Function

Returns the first n characters in a source [string](#).

**Syntax** `result=Left(source, count)`

OR

`result=stringVariable.Left(count)`

| Part   | Type                   | Description   |
|--------|------------------------|---|
| result | <a href="#">String</a> | The first count characters of source.               |
| source | <a href="#">String</a> | The source string from which to get the characters. |

## LeftB Function

---

| Part           | Type                    | Description   |
|----------------|-------------------------|---|
| count          | <a href="#">Integer</a> | The number of characters you wish to get from source. If count is greater than the number of characters in source, all characters are returned. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .   |

### Notes

The **Left** function returns characters from the *source string* starting from the left side (as the name implies).

If you need to read bytes rather than characters, use the [LeftB](#) function.

### Example

This example returns the first five characters in a [string](#).

```
Dim s As String  
s=Left("Hello World", 5) //returns "Hello"
```

### See Also

[Asc](#), [Chr](#), [InStr](#), [Len](#), [LeftB](#), [Mid](#), [Right](#) functions.

---

## LeftB Function

Returns the first n bytes in a source [string](#).

### Syntax

**result=LeftB(source, count)**

OR

**result=stringVariable.LeftB(count)**

| Part           | Type                    | Description  |
|----------------|-------------------------|--|
| result         | <a href="#">String</a>  | The first count bytes of source.   |
| source         | <a href="#">String</a>  | The source string from which to get the bytes.   |
| count          | <a href="#">Integer</a> | The number of bytes you wish to get from source. If count is greater than the number of bytes in source, all bytes are returned. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .  |

### Notes

The **LeftB** function returns bytes from the *source string* starting from the left side (as the name implies).

If you need to read characters rather than bytes, use the [Left](#) function.

**Example** This example uses the **LeftB** function to return the first 5 bytes from a string.

```
Dim s As String
s=LeftB("Hello World", 5) //returns "Hello"
```

**See Also** [AscB](#), [ChrB](#), [InStrB](#), [Left](#), [LenB](#), [MidB](#), [RightB](#) functions.

## Len Function

Returns the number of characters in the specified [string](#).

**Syntax** *result=Len(string)*

OR

*result=stringVariable.Len*

| Part           | Type                    | Description                                   |
|----------------|-------------------------|---|
| result         | <a href="#">Integer</a> | The number of characters in string.           |
| string         | <a href="#">String</a>  | Any valid string expression.                  |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> . |

**Notes** If you need the number of bytes in the [string](#) rather than the number of characters, use the [LenB](#) function.

**Examples** This example uses the **Len** function to return the number of characters in a string.

```
Dim n As Integer
n=Len("Hello world") //returns 11
```

**See Also** [Asc](#), [Chr](#), [InStr](#), [LenB](#), [Mid](#), [Right](#) functions.

## LenB Function

Returns the number of bytes in the specified [string](#).

**Syntax** *result=LenB(string)*

OR

**result=stringVariable.LenB**

| Part           | Type                    | Description                                   |
|----------------|-------------------------|---|
| result         | <a href="#">Integer</a> | The number of bytes in string.                |
| string         | <a href="#">String</a>  | Any valid string expression.                  |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> . |

### Notes

**LenB** treats *string* as a series of bytes, rather than a series of characters. It should be used when *string* represents binary data. If you need to know the number of characters in *string* rather than the number of bytes, use the [Len](#) function.

### Examples

This example uses the **LenB** function to return the number of bytes in a string.

```
Dim n As Integer  
n=LenB("Hello world") //returns 11
```

### See Also

[AscB](#), [ChrB](#), [InStrB](#), [Len](#), [MidB](#), [RightB](#) functions.

---

## Light3D Class

Allows you to define your own lights for use in [RB3DSpace](#) worlds.

**Super Class** [Element3D](#)

### Properties

| Name        | Type                     | Description  |
|-------------|--------------------------|--|
| Attenuation | <a href="#">Integer</a>  | Specifies how the brightness of a point light falls off with distance.<br>0: No attenuation at all.<br>1: Brightness falls off linearly with distance.<br>2: Brightness falls off with the square of the distance.<br>The default value is 1 (Linear). |
| Brightness  | <a href="#">Integer</a>  | Amount of brightness expressed as a percentage from 0 (completely off) to 100 (the default). Values greater than 100 are valid if you need extra brightness.   |
| Direction   | <a href="#">Vector3D</a> | If <a href="#">Nil</a> , you get a point light that radiates in all directions. If non-nil, it is the direction for a floodlight. It can be changed from directional to point as long as it is attached to only one <a href="#">RB3DSpace</a> .        |
| LightColor  | <a href="#">Color</a>    | The color of the light. Defaults to white.   |
| Position    | <a href="#">Vector3D</a> | If <a href="#">Nil</a> , you get a floodlight. If non-nil, it is the position of a point light.  |

**Notes**

It is invalid to use a light with Position and Direction either both [Nil](#), or both non-nil. If the Attenuation value is 1 or 2, the effect of the light decreases with distance; so you may need to increase the Brightness substantially or make your objects smaller and closer to the light.

**See Also**

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [RB3DSpace](#) control.

## LightBevelColor Function

The currently selected operating system color for drawing light lines in dividing lines and group boxes.

**Syntax**

**result=LightBevelColor**

| Part   | Type                  | Description   |
|--------|-----------------------|---|
| result | <a href="#">Color</a> | The color used for drawing the light lines in dividing lines and group boxes. |

**Notes**

This value is useful when you are using [Canvas](#) controls to create custom controls. When drawing controls like dividing [lines](#) and [GroupBoxes](#), use this color for the light portions of the object (usually the top and left sides of the object). This value can be changed by the user, so you should access this value in the Paint event handler rather than storing the value.

**See Also**

[DarkBevelColor](#), [DarkTingeColor](#), [FillColor](#), [FrameColor](#), [HighlightColor](#), [LightTingeColor](#), [TextColor](#) functions; [Color](#) data type.

## LightTingeColor Function

The currently selected operating system color for drawing light lines inside frames on the top and left sides.

**Syntax**

**result=LightTingeColor**

| Part   | Type                  | Description   |
|--------|-----------------------|---|
| result | <a href="#">Color</a> | The color used for drawing the light lines inside frames. |

**Notes**

This value is useful when you are using [Canvas](#) controls to create custom controls. When drawing beveled objects, use this color for the light portions of the object

(usually the top and left sides of the object). In a [Checkbox](#) control, this color is used to draw the light lines that give it the beveled appearance just inside the checkbox on the top and left sides.

This value can be changed by the user, so you should access this value in the Paint event handler rather than storing the value.

**See Also** [DarkBevelColor](#), [DarkTingeColor](#), [FillColor](#), [FrameColor](#), [LightBevelColor](#), [TextColor](#) functions; [Color](#) data type.

---

## Line Control

Draws a line.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) class for other properties and events that are common to all [RectControls](#).

### Properties

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| BorderWidth | <a href="#">Integer</a> | The width of the line in pixels.  |
| LineColor   | <a href="#">Color</a>   | The <a href="#">color</a> of the line.                                    |
| Visible     | <a href="#">Boolean</a> | If <a href="#">True</a> , the line will be visible when the window opens. |
| X1          | <a href="#">Integer</a> | The point on the X axis where the line begins.                            |
| X2          | <a href="#">Integer</a> | The point on the X axis where the line ends.                              |
| Y1          | <a href="#">Integer</a> | The point on the Y axis where the line begins.                            |
| Y2          | <a href="#">Integer</a> | The point on the Y axis where the line ends.                              |

### Events

| Name      | Parameters   | Description  |
|-----------|--|--|
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Line at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown.   |
| MouseDrag | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Line and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the Line region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.                                  |

**See Also** [RectControl](#) class.

## ListBox Control

The scrollable **ListBox** control, used to display one or more columns of information. You can add a checkbox and/or a picture to a row. You can control font style on a cell-by-cell basis and set the column alignment. With some programming, you can create hierarchical lists that use disclosure triangles to show nested items.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name                | Type                      | Description  |
|---------------------|---------------------------|--|
| ActiveCell          | <a href="#">EditField</a> | <a href="#">EditField</a> that the ListBox uses for its editable cell operations. You can use this property to set or get the text of the Listbox cell ( <code>SelText</code> ), set the selection, or change other properties of the ListBox's <a href="#">EditField</a> .  |
| Bold                | <a href="#">Boolean</a>   | Applies the bold style to all items in the ListBox.  |
| CellAlignment       | <a href="#">Integer</a>   | Parameters are row, column ( <a href="#">Integers</a> ). The first cell is 0,0. Aligns the specified cell. You specify the alignment via ListBox class constants. They are:<br>AlignDefault (0): Default alignment, same as column type<br>AlignLeft (1): Left<br>AlignCenter (2): Center<br>AlignRight (3): Right<br>AlignDecimal (4): Decimal<br>For example:<br><code>listbox1.CellAlignment(1,1)=ListBox.AlignRight</code><br>Decimal aligns the decimal separator to the right edge of the cell. You need to use <code>CellAlignmentOffset</code> or <code>ColumnAlignmentOffset</code> to move the data into the column. |
| CellAlignmentOffset | <a href="#">Integer</a>   | Parameters are row, column ( <a href="#">Integers</a> ). Value is the distance in pixels from the right edge of the cell. Overrides <code>ColumnAlignmentOffset</code> for that cell.  |

| Name             | Type                    | Description  |
|------------------|-------------------------|--|
| CellBorderBottom | <a href="#">Integer</a> | Parameters are row, column ( <a href="#">Integers</a> ).Sets the bottom border of the cell designated by <i>Row</i> , <i>Column</i> to a rule style. You specify the Border via ListBox class constants: They are:<br>BorderDefault (0): Default<br>BorderNone (1): None<br>BorderThinDotted (2): Thin Dotted<br>BorderThinSolid (3): Thin Solid<br>BorderThickSolid (4): Thick Solid<br>BorderDoubleThinSolid (5): Double Thin Solid<br>For example:<br>lb1.CellBorderBottom(1,1)=ListBox.BorderThinSolid |
| CellBorderLeft   | <a href="#">Integer</a> | Parameters are row, column ( <a href="#">Integers</a> ).Sets the left border of the cell designated by <i>Row</i> , <i>Column</i> to a rule style. You specify the Border via ListBox class constants: They are:<br>BorderDefault (0): Default<br>BorderNone (1): None<br>BorderThinDotted (2): Thin Dotted<br>BorderThinSolid (3): Thin Solid<br>BorderThickSolid (4): Thick Solid<br>BorderDoubleThinSolid (5): Double Thin Solid<br>For example:<br>lb1.CellBorderLeft(1,1)=ListBox.BorderThinSolid     |
| CellBorderRight  | <a href="#">Integer</a> | Parameters are row, column ( <a href="#">Integers</a> ).Sets the right border of the cell designated by <i>Row</i> , <i>Column</i> to a rule style. You specify the Border via ListBox class constants: They are:<br>BorderDefault (0): Default<br>BorderNone (1): None<br>BorderThinDotted (2): Thin Dotted<br>BorderThinSolid (3): Thin Solid<br>BorderThickSolid (4): Thick Solid<br>BorderDoubleThinSolid (5): Double Thin Solid<br>For example:<br>lb1.CellBorderRight(1,1)=ListBox.BorderThinSolid   |

| Name          | Type                       | Description   |
|---------------|----------------------------|---|
| CellBorderTop | <a href="#">Integer</a>    | Parameters are <i>row</i> , <i>column</i> ( <a href="#">Integers</a> ). Sets the top border of the cell designated by <i>Row</i> , <i>Column</i> to a rule style. You specify the Border via ListBox class constants: They are:<br>BorderDefault (0): Default<br>BorderNone (1): None<br>BorderThinDotted (2): Thin Dotted<br>BorderThinSolid (3): Thin Solid<br>BorderThickSolid (4): Thick Solid<br>BorderDoubleThinSolid (5): Double Thin Solid<br>For example:<br>lb1.CellBorderTop(1,1)=ListBox.BorderThinSolid              |
| CellCheck     | <a href="#">Boolean</a>    | Parameters are <i>row</i> , <i>column</i> ( <a href="#">Integers</a> ). The first cell is 0,0. Setting CellCheck to <a href="#">True</a> checks the checkbox.   |
| CellTag       | <a href="#">Variant</a>    | A 'hidden' identifier associated with the cell identified by its parameters, <i>row</i> as <a href="#">Integer</a> and <i>column</i> as <a href="#">Integer</a> .   |
| CellType      | <a href="#">Integer</a>    | Parameters are <i>row</i> , <i>column</i> ( <a href="#">Integers</a> ). The first cell is 0,0.<br>Use the following class constants to set the CellType:<br>TypeDefault (0): Default, same as column type<br>TypeNormal (1): Normal<br>TypeCheckBox (2): Add checkbox<br>TypeEditable (3): Inline editable<br>For example:<br>lb1.CellType(1,1)=ListBox.TypeCheckBox<br>The values of CellType > 0 override ColumnType. For example, if ColumnType is 2, but a cell in the column has CellType set to 1, the cell will be normal. |
| Column        | <a href="#">ListColumn</a> | Parameter is <i>columnNumber</i> as <a href="#">Integer</a> . The first column is numbered zero. Use -1 to refer to all columns. Enables you to access the <a href="#">ListColumn</a> properties of the specified column.   |

| Name                  | Type                    | Description   |
|-----------------------|-------------------------|---|
| ColumnAlignment       | <a href="#">Integer</a> | Parameter is <i>columnNumber</i> as <a href="#">Integer</a> ; the first column is numbered zero. Aligns the specified column. You specify the alignment via ListBox class constants. They are:<br>AlignDefault (0): Default alignment<br>AlignLeft (1): Left<br>AlignCenter (2): Center<br>AlignRight (3): Right<br>AlignDecimal (4): Decimal<br>For example:<br><code>Ibox1.ColumnAlignment(1)=ListBox.AlignRight</code><br>Decimal aligns the decimal separator to the right edge of the column. You need to use ColumnAlignmentOffset to move the data into the column.  |
| ColumnAlignmentOffset | <a href="#">Integer</a> | Parameter is <i>columnNumber</i> as <a href="#">Integer</a> ; the first column is numbered zero. Value is distance in pixels from the right edge of the column. A negative value moves the decimal separator to the left, i.e., into the body of the column. See the example in the Notes subsection "Decimal Alignment."   |
| ColumnCount           | <a href="#">Integer</a> | The number of columns the ListBox contains. The maximum number of visible columns is 64 (columns 0 through 63).   |
| ColumnSortDirection   | <a href="#">Integer</a> | Parameter is <i>columnNumber</i> ( <a href="#">Integer</a> ). The first column is numbered zero. Used to get or set the sort direction, according to the following class constants:<br>SortDescending (-1): Descending<br>SortNone (0): Don't sort<br>SortAscending (1): Ascending (default)<br>For example:<br><code>lb1.ColumnSortDirection(2)=ListBox.SortDescending</code><br>ColumnSortDirection doesn't actually sort the rows; it only establishes the sort direction that is used when the Sort method is called. The sort direction can be set even if there is no header for the ListBox.<br>If you set ColumnSortDirection to Don't Sort, the user can't sort the column by clicking its header. Don't Sort will block the usual calls to the SortColumn and CompareRows events. |

| Name       | Type                    | Description  |
|------------|-------------------------|--|
| ColumnType | <a href="#">Integer</a> | <p>Parameter is column number; the first column is numbered zero. Use the following class constants to set the ColumnType:</p> <p>TypeDefault (0): Default, same as column type<br/>TypeNormal (1): Normal<br/>TypeCheckBox (2): Add checkbox<br/>TypeEditable (3): Inline editable<br/>For example:<br/><code>lb1.ColumnType(1)=ListBox.TypeEditable</code><br/>(Disclosure triangles don't work in hierarchical ListBoxes if Inline Editable is selected.)</p> |

| Name         | Type                   | Description  |
|--------------|------------------------|--|
| ColumnWidths | <a href="#">String</a> | <p>A list of comma separated values, with each value controlling the width of the associated column. A value can be an absolute value (in pixels), a percentage, or a relative length expressed as <math>i*</math> where <math>i</math> is an integer. If you use percentages, you can use non-integer values to specify fractions of a percent, e.g., 43.52%. The percentage value can be greater than 100%.</p> <p>If you use pixels, the last column doesn't grow to the size of the rest of the ListBox. Without any column width specifications, the headers will be divided evenly. If there are fewer column widths specified than the total number of columns, the remaining columns will divide up the remaining width equally. An element with a length of "3*" will be allotted three times the space of an element with length "1*". The value "*" is equivalent to "1*" and can be used to mean "fill the remaining space."</p> <p>You can use a mixture of pixels, percentages, and relative lengths. Column widths specified in pixels are guaranteed to have the specified width. Column widths specified in percentages are guaranteed to have that percentage of the visible width of the ListBox. The column widths specified using the * divide up the remaining width proportionally. For example, if there are four columns, the specification</p> <p>20, 20%, *, 2*</p> <p>gives 20 pixels to the first column 20% of the total to the second column, and the last two columns divide up the remaining width in the ratio of 1:2, with the last column getting any remaining fractional pixels.</p> <p>Unrecognized or unsupported expressions (e.g. '2@') for the column width properties will result in an <a href="#">UnsupportedFormatException</a>. The message of this exception describes in English what went wrong.</p> <p>Resizing a column will resize the value of the expression. If you resize the ListBox, both the percentage and relative lengths recompute their actual widths. There are two resizing "modes"; see notes.</p> <p>Header End caps are added for any additional unused space if headers are used. Header end caps do nothing when clicked.</p> |

| Name                       | Type                        | Description   |
|----------------------------|-----------------------------|---|
| <b>DataField</b>           | <a href="#">String</a>      | Relevant only if the ListBox is used in conjunction with a <a href="#">DataControl</a> to display the contents of fields in a database table. Name of a field in the table referenced by <a href="#">DataControl</a> .  |
| <b>DataSource</b>          | <a href="#">DataControl</a> | The <a href="#">DataControl</a> that references a table in a database whose fields are displayed using the ListBox. Use the DataField property to link the field to be displayed to the ListBox. In the IDE, a popup menu of field names will appear. When set in code, it takes a String, e.g., <code>ListBox1.DataSource=DataControl1.Name</code>   |
| <b>DefaultRowHeight</b>    | <a href="#">Integer</a>     | Determines the height of every row in the ListBox (pixels). To allow REALbasic to determine the height automatically based on the current font size, set DefaultRowHeight to -1.  |
| <b>EnableDrag</b>          | <a href="#">Boolean</a>     | Allows rows to be dragged. See the examples of dragging rows from one ListBox to another ListBox in the Examples section.   |
| <b>EnableDragReorder</b>   | <a href="#">Boolean</a>     | If <a href="#">True</a> , you can reorder rows within the ListBox by dragging rows. An insertion line indicator appears when dragging within the ListBox to provide you with visual feedback as to where the row would be dropped if you release the mouse button.  |
| <b>Expanded</b>            | <a href="#">Integer</a>     | Used to get or set the expanded state of the row passed. The row must have been added with the AddFolder method.  |
| <b>GridLinesHorizontal</b> | <a href="#">Integer</a>     | Draws horizontal rules between rows in one of five styles. You can use the class constants for Borders to set the gridline styles. They are:<br>BorderDefault (0): Default<br>BorderNone (1): None<br>BorderThinDotted (2): Thin Dotted<br>BorderThinSolid (3): Thin Solid<br>BorderThickSolid (4): Thick Solid<br>BorderDoubleThinSolid (5): Double Thin Solid<br>For example:<br><code>lb1.GridLinesHorizontal=ListBox.BorderThinSolid</code> |

| Name              | Type                         | Description  |
|-------------------|------------------------------|--|
| GridLinesVertical | <a href="#">Integer</a>      | Draws vertical rules between columns in one of five styles. You can use the class constants for Borders to set the gridline styles. They are:<br>BorderDefault (0): Default<br>BorderNone (1): None<br>BorderThinDotted (2): Thin Dotted<br>BorderThinSolid (3): Thin Solid<br>BorderThickSolid (4): Thick Solid<br>BorderDoubleThinSolid (5): Double Thin Solid<br>For example:<br>lb1.GridLinesVertical=ListBox.BorderThickSolid   |
| HasHeading        | <a href="#">Boolean</a>      | If <a href="#">True</a> , a row of column headers is added to the ListBox. The user can sort the column by clicking the heading.   |
| Heading           | <a href="#">String</a> array | Zero-based array of column headings.<br>Headings appear only if HasHeading is <a href="#">True</a> .<br>If you assign values to both Heading and InitialValue, the first row of InitialValue is interpreted as the first row of data; otherwise, it is used as the heading and the second row of InitialValue is used as the first row of data.<br>You can set the headings in a multi-column listbox by assigning to Heading(-1) the text of the headings separated by the tab character, e.g.,<br>me.heading(-1)=<br>"FirstName"+ <a href="#">Chr</a> (9)+"LastName"<br>You must use a space if you want an empty header; empty strings will use the default header.<br>ListBox1.heading(-1) = "" sets all headers of ListBox1 to their defaults.<br>ListBox1.heading(5) = "" sets column 5's heading to its default heading<br>ListBox1.heading(5) = "" sets column 5's heading to empty. |
| HeadingIndex      | <a href="#">Integer</a>      | Allows you to get and set the sort column in a ListBox. The first column is numbered zero.<br>This property is obsolete. To sort a ListBox, use the SortedColumn property to set the sort column and the ColumnSortDirection property to set the direction of the sort. Then call the Sort method to do the sort.  |

| Name                | Type                    | Description   |
|---------------------|-------------------------|---|
| Hierarchical        | <a href="#">Boolean</a> | Allows for disclosure triangles for rows added via the AddFolder method. On Windows, plus and minus signs are used instead of disclosure triangles.   |
| InitialValue        | <a href="#">String</a>  | A list of the default items separated by carriage returns. If the ListBox has more than one column (ColumnCount > 1), separate column entries with Tabs within each row. If HasHeading is <a href="#">True</a> , the first row of InitialValue is assumed to be the column headings—unless the Heading array is also specified. |
| Italic              | <a href="#">Boolean</a> | Applies the italic style to all items in the list.  |
| LastIndex           | <a href="#">Integer</a> | The number of the last row added with the AddRow, AddFolder, or InsertRow method.   |
| List                | <a href="#">String</a>  | The list of items. It takes one parameter, the row index. The first row is numbered zero. Use it to get or set the contents of that cell, or, in multi-column ListBoxes, the contents of the first column.  |
| ListCount           | <a href="#">Integer</a> | The number of items in the list.  |
| ListIndex           | <a href="#">Integer</a> | The selected item number.   |
| RequiresSelection   | <a href="#">Boolean</a> | If <a href="#">True</a> , users will not be able to deselect the last row by clicking below the last visible row or by dragging. You can still deselect the last row by setting the ListIndex property to -1.   |
| ScrollBarHorizontal | <a href="#">Boolean</a> | If <a href="#">True</a> , adds a horizontal scroll bar to the ListBox. The position of the thumb is indicated by ScrollPositionX. Used to scroll the ListBox horizontally without a separate <a href="#">ScrollBar</a> control.   |
| ScrollBarVertical   | <a href="#">Boolean</a> | If <a href="#">True</a> , adds a vertical scroll bar to the ListBox. The position of the thumb is indicated by ScrollPosition.  |
| ScrollPosition      | <a href="#">Integer</a> | Zero-based index of the top visible row in the ListBox. Read ScrollPosition to determine the top visible row; write to ScrollPosition to scroll the ListBox. When the scrollbar thumb is scrolled to the bottom, ScrollPosition cannot be incremented any further.  |
| ScrollPositionX     | <a href="#">Integer</a> | Zero-based index of the horizontal position of the ListBox. Used with a <a href="#">ScrollBar</a> control to scroll the ListBox horizontally. See the Notes section.  |

## ListBox Control

| Name          | Type                    | Description   |
|---------------|-------------------------|---|
| SelCount      | <a href="#">Integer</a> | The number of rows highlighted (selected).  |
| Selected      | <a href="#">Boolean</a> | Parameter is Row as <a href="#">Integer</a> . Indicates if the row passed is selected or not. Selected is <a href="#">True</a> if the row passed is selected. This property can be used to determine if the row is selected and to select the row. For example, <code>Listbox1.Selected(1)=True</code> selects the second item in the first column. |
| SelectionType | <a href="#">Integer</a> | Indicates the type of selection allowed. Use the following class constants to set the selection type:<br><a href="#">SelectionSingle</a> (0): Single row selection<br><a href="#">SelectionMultiple</a> (1): Multiple row selection.<br>If multiple row selection is on, a ListBox will handle Edit > Select All menu item commands by default.     |
| SortedColumn  | <a href="#">Integer</a> | Gets or sets the current sort column but doesn't do the sort. Call the Sort method to sort the ListBox based on the values of SortedColumn and ColumnSortDirection. The first column is numbered zero.  |
| Text          | <a href="#">String</a>  | The text of the currently selected item.  |
| TextFont      | <a href="#">String</a>  | Name of the font used to display the list text.   |
| TextSize      | <a href="#">Integer</a> | Size of the font used to display the list text.   |
| Underline     | <a href="#">Boolean</a> | Applies the underline style to all items in the list.   |
| UseFocusRing  | <a href="#">Boolean</a> | If <a href="#">True</a> , the control indicates that it has the focus with a ring around its border; if <a href="#">False</a> , the appearance of the control does not change when it has the focus.  |

## Events

| Name       | Parameters   | Description   |
|------------|--|---|
| CellAction | Row as <a href="#">Integer</a> , Column as <a href="#">Integer</a> | If a cell is editable, a CellAction event occurs when the user edits the cell. "Editing" is defined as exiting the cell after clicking in it. Clicking a checkbox in a checkbox cell also qualifies as "Editing." The user doesn't necessarily have to change the contents. |

| Name                | Parameters  | Description   |
|---------------------|---|---|
| CellBackgroundPaint | g as <a href="#">Graphics</a><br>Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a>                                | <p>The parameter <i>g</i> is a <a href="#">Graphics</a> object that corresponds to the content area of the cell <i>Row</i>, <i>Column</i>. 0,0 is the upper left of the cell. Returns a <a href="#">Boolean</a>.</p> <p><a href="#">True</a> means the user has handled the background paint and no other processing is to be done with the background. In this case the user is responsible for all highlighting.</p> <p><a href="#">False</a> means the user wants REALbasic to help paint the background. REALbasic will overwrite your drawing on the row that needs to be highlighted. REALbasic will attempt to highlight the row or column as appropriate (according to the platform and the hierarchical style).</p> <p>CellBackgroundPaint fires for every visible row in a ListBox, regardless of whether there is an actual row there or not. This enables you to implement the background paint event for the entire ListBox in a consistent way. For example, to do alternating row colors.</p> <p>Because of this, you need to check whether the current row is less than ListCount when accessing the row, for example with the Cell method.</p> <p>Before CellBackgroundPaint fires, some drawing may have taken place. It isn't safe to assume that the background of the cell is white or that the selection hasn't already been drawn. If you need the background to be clear, you need to clear the background yourself and then return <a href="#">True</a>.</p> |
| CellClick           | Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a><br>X as <a href="#">Integer</a><br>Y as <a href="#">Integer</a> | <p>The user has clicked on the <i>Row</i>, <i>Column</i> cell. The parameters <i>X</i> and <i>Y</i> are the <i>x</i> and <i>y</i> coordinates of the mouse click relative to the top-left corner of the cell that was clicked. <i>X</i> and <i>Y</i> are on the same scale of reference as the coordinates used by the <a href="#">Graphics</a> property of the CellBackgroundPaint event.</p> <p>Returns a <a href="#">Boolean</a>. Returning <a href="#">True</a> means that the event will not be processed further (i.e., editable cells won't be editable and ListBox selection won't change).</p>   |
| CellGotFocus        | Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a>   | The user has selected an editable cell of a ListBox. The <i>Row</i> and <i>Column</i> parameters indicate which cell just got the focus.  |

| Name           | Parameters   | Description   |
|----------------|--|---|
| CellKeyDown    | Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a><br>Key as <a href="#">String</a> | The user has pressed a key while a cell in the ListBox is being edited. This cell is identified by the <i>Row</i> and <i>Column</i> parameters. <i>Key</i> is the key that the user pressed.<br>Returns a <a href="#">Boolean</a> . Returning <a href="#">True</a> prevents the text from changing automatically and prevents the CellTextChange event from firing. |
| CellLostFocus  | Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a>                                  | The <i>Row</i> , <i>Column</i> cell has just lost the focus. The user could have clicked on another cell or pressed Return or Escape.   |
| CellTextChange | Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a>                                  | Occurs after the KeyDown event if the KeyDown event returns <a href="#">False</a> . The event passes the <i>Row</i> and <i>Column</i> of the cell being edited.   |

| Name          | Parameters   | Description  |
|---------------|--|--|
| CellTextPaint | g as <a href="#">Graphics</a><br>Row as <a href="#">Integer</a><br>Column as <a href="#">Integer</a><br>x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | <p>The parameter g is a <a href="#">Graphics</a> object that corresponds to the text drawing area of the cell <i>Row</i>, <i>Column</i>. This does not necessarily correspond to the entire cell content area, for example, if you use a row picture in the cell.</p> <p>The parameters x and y are the coordinates of the suggested ideal location to draw text based on the current value of ColumnAlignment or CellAlignment, as well as the cell's font, font size, and font style.</p> <p>Returns a <a href="#">Boolean</a>.</p> <p>The drawing order of the cell is as follows, with the background first:</p> <ul style="list-style-type: none"> <li>Background</li> <li>Disclosure Triangle/Treebox</li> <li>Checkbox</li> <li>RowPicture</li> <li>Text</li> <li>Border</li> </ul> <p>Although the border is painted last, it isn't advisable to change the state of the border in the CellTextPaint event since the area is determined by the size of the border before the cell is painted and so it could leave unpainted areas, or possibly cover up some of the painting you have done.</p> <p><a href="#">True</a> means the user has handled the text paint and no other processing is to be done with the text. In this case, the user is responsible for text highlighting. Text highlighting is currently only done for the hierarchical listbox style. <a href="#">False</a> means the user wants REALbasic to paint the text. REALbasic will attempt to highlight the text as appropriate (according to the platform, and the hierarchical style).</p> |
| Change        |  | The selected item has changed.   |
| CollapseRow   | Row as <a href="#">Integer</a>   | The user has clicked on the disclosure triangle of the expanded Row passed.  |

| Name            | Parameters   | Description  |
|-----------------|--|--|
| CompareRows     | Row1 as <a href="#">Integer</a><br>Row2 as <a href="#">Integer</a><br>Column as <a href="#">Integer</a><br><a href="#">ByRef</a> Result as <a href="#">Integer</a> | <p>Returns a <a href="#">Boolean</a>. The CompareRows event is useful for sorting a column of a ListBox in a manner that is not provided by the default mechanism. The default mechanism sorts lexicographically. If you use the event, it gets called during a ListBox sort, e.g., when a user clicks in the header area.</p> <p>Parameters:</p> <p>Row1: Row number of one of the rows being compared.</p> <p>Row2: Row number of the other row being compared.</p> <p>Column: Number of column being sorted.</p> <p>Result:</p> <ul style="list-style-type: none"> <li>0 – If items in Row1 and Row2 in specified column are equal.</li> <li>1 – Contents of Row1 &gt; Contents of Row2.</li> <li>-1 – Contents of Row2 &gt; Contents of Row1.</li> </ul> <p><a href="#">Return True</a> if the returned <i>Result</i> parameter is accurate for sorting.</p> <p><a href="#">Return False</a> if you want REALbasic to use the default lexicographic sorting of the column.</p> |
| DoubleClick     |  | The user has double-clicked on an item.  |
| DragReorderRows | NewPosition as <a href="#">Integer</a><br>ParentRow as <a href="#">Integer</a>   | <p>Triggered when a row being reordered by dragging is dropped on the ListBox. The parameter <i>newPosition</i> indicates the row at which the dragged row is to be inserted.</p> <p>Returns a <a href="#">Boolean</a>. Returning <a href="#">True</a> prevents the reordering from happening.</p> <p>The ParentRow parameter gives you the row number of the parent folder if you are dragging within a hierarchical ListBox. As is the case for NewPosition, the row numbers are adjusted for the assumption that all selected rows will be deleted. A value of -1 indicates a drop at the root level of the ListBox. That is, there is no parent row.</p> <p>The DragReorderRows event fires, but the user is responsible for reordering the rows. It is the same behavior as returning <a href="#">True</a> in the DragReorderRows event.</p>  |

| Name          | Parameters   | Description   |
|---------------|--|---|
| DragRow       | Drag as <a href="#">DragItem</a> ,<br>Row as <a href="#">Integer</a> | The user is dragging a row. <i>Drag</i> is the <a href="#">DragItem</a> object that is created automatically. Assign the values to the <a href="#">DragItem</a> 's properties that the user should drag. <i>Row</i> is the row of the ListBox that is being dragged. You must return <a href="#">True</a> in this event handler to allow the drag to occur. |
| ExpandRow     | Row as <a href="#">Integer</a>                                       | The user has clicked on the disclosure triangle of the collapsed Row passed.  |
| GotFocus      |  | The ListBox has the focus, i.e., the user has selected the ListBox.   |
| HeaderPressed | Column as <a href="#">Integer</a>                                    | Returns a <a href="#">Boolean</a> . Runs after a ListBox header has been clicked/pressed. This event enables you to specify whether you want REALbasic to sort the column. If you return <a href="#">True</a> , REALbasic does not sort the column and it does not update the SortedColumn property.  |
| KeyDown       | Key as <a href="#">String</a>  | The user has pressed the Key passed while the ListBox has the focus. Returns a <a href="#">Boolean</a> . Returning <a href="#">True</a> means that no further processing is to be done with the Key.  |
| LostFocus     |  | The ListBox has lost the focus.   |
| MouseDown     | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>       | Returns a <a href="#">Boolean</a> . The mouse button was pressed inside the ListBox region at the location passed in to x,y. Returning <a href="#">True</a> in the MouseDown event causes the MouseDrag and MouseUp events to fire and the default ListBox click processing not to fire.  |
| MouseDrag     | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>       | The mouse button was pressed inside the ListBox and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |
| MouseUp       | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>       | The mouse button was released inside the ListBox region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.  |
| SortColumn    | Column as <a href="#">Integer</a>                                    | The user has clicked on the column header to sort that column. <a href="#">Return True</a> if you don't want the ListBox to be sorted.  |

## Methods

| Name                | Parameters   | Description  |
|---------------------|--|--|
| AddFolder           | Item as <a href="#">String</a>   | Appends Item in a new row to the end of the list and adds disclosure triangle only if the Hierarchical property is set to <a href="#">True</a> . For multi-column ListBoxes, Item is always assigned to column zero. In the case of hierarchical ListBoxes, AddFolder appends <i>Item</i> to the subitems of the expanded row when called in the ExpandRow event.  |
| AddRow              | Item as <a href="#">String</a>   | Appends Item in a new row to the end of the list. For multi-column ListBoxes, Item is always assigned to column zero. In the case of hierarchical ListBoxes, AddRow appends <i>Item</i> to the subitems of the expanded row when called in the ExpandRow event. The theoretical maximum number of rows is over 2 billion but the actual number will be lower due to memory constraints.                              |
| Cell                | RowNumber as <a href="#">Integer</a> , ColumnNumber as <a href="#">Integer</a> | Used to read from or write to the cell based on the row and column numbers passed. Passing -1 as either the Row or Column number means all rows or all columns, respectively. For example, the following specifies all columns in the next available row:<br><a href="#"><code>Me.cell(Me.lastindex, -1)</code></a><br>If you set this equal to a tab-delimited array, you can update the row with one line of code. |
| CellBold            | RowNumber as <a href="#">Integer</a> , ColumnNumber as <a href="#">Integer</a> | Used to add or remove the bold style from the text of the specified cell. Assign <a href="#">True</a> to add the bold style and <a href="#">False</a> to remove the bold style.  |
| CellItalic          | RowNumber as <a href="#">Integer</a> , ColumnNumber as <a href="#">Integer</a> | Used to add or remove the italic style from the text of the specified cell. Assign <a href="#">True</a> to add the italic style and <a href="#">False</a> to remove the italic style.  |
| CellUnderline       | RowNumber as <a href="#">Integer</a> , ColumnNumber as <a href="#">Integer</a> | Used to add or remove the underline style from the text of the specified cell. Assign <a href="#">True</a> to add the underline style and <a href="#">False</a> to remove the underline style.   |
| ColumnValueProvider | Column as <a href="#">Integer</a>  | Used in programming custom bindings. Returns a <a href="#">StringProvider</a> . The return object also implements DataNotifier and DataAvailableProvider.  |
| DeleteAllRows       |  | Deletes all rows in the list.  |

| Name           | Parameters   | Description  |
|----------------|--|--|
| EditCell       | Row as <a href="#">Integer</a> ,<br>Column as <a href="#">Integer</a>    | Scrolls the <i>Row</i> , <i>Column</i> cell into view (if necessary) and temporarily makes the cell editable. It sets the focus within the ListBox to the <i>Row</i> , <i>Column</i> cell. See the CellGotFocus event. The editable cell has a focus ring around it.   |
| InsertFolder   | RowNumber as <a href="#">Integer</a> ;<br>Item as <a href="#">String</a> | Creates a new folder at RowNumber (moving the existing rows down). For multicolumn ListBoxes, Item is always assigned to column zero.  |
| InsertRow      | RowNumber as <a href="#">Integer</a> ,<br>Item as <a href="#">String</a> | Creates a new row at RowNumber (moving the existing rows down). For multicolumn ListBoxes, Item is always assigned to column zero.   |
| InvalidateCell | Row as <a href="#">Integer</a> ,<br>Column as <a href="#">Integer</a>    | Redraws the specified cell "from scratch," rather than using the data in the internal buffer. If you pass a -1 as the <i>Row</i> or <i>Column</i> parameter, it will redraw the specified entire row or column, i.e., Invalidate(2,-1) redraws row 2. The CellBackgroundPaint and CellTextPaint events execute when InvalidateCell is called. InvalidateCell should be necessary only if you are using a custom storage mechanism for your ListBox data. If the contents of the ListBox are stored in the ListBox cells, the ListBox will update automatically as needed. This method doesn't have a high overhead if used unnecessarily since nothing will happen if the specified cell is not visible. |
| PressHeader    | Column as <a href="#">Integer</a>  | Causes the ListBox header to be pressed, causing a HeaderPressed event to occur. Calling this method does not update the sort direction.   |
| RemoveRow      | RowNumber as <a href="#">Integer</a>                                     | Deletes the specified row.   |
| RowPicture     | RowNumber as <a href="#">Integer</a>                                     | Adds the picture assigned to the RowNumber passed.   |
| SetFocus       |  | Gives the ListBox the focus sending all keydown events to the ListBox.   |
| Sort           |  | Sorts the rows based on the current values of the SortedColumn and ColumnSortDirection properties. When a hierarchical ListBox is sorted, the rows are sorted at all levels of the hierarchy, not just the base level.   |

### Notes

Items in single-column ListBoxes can be accessed using the List property. The List property is an array. Arrays are zero-based which means that the first row of the List property of a **ListBox** is row number 0 (zero).

### Multi-Column ListBoxes

You can create multi-column ListBoxes by changing the ColumnCount property. The first column in a multi-column **ListBox** is column 0 (zero). This means that the ColumnCount property will always be one more than the number of the last column. The maximum number of visible columns is 64 (columns 0 through 63). You should set ColumnCount to the number of columns that you want to display. If you want to put data in an invisible column, set the column width to zero.

You can use the InitialValue property to set up the initial values of multi-column ListBoxes by separating the column values with tabs and row values with carriage returns.

The widths of columns in multi-column ListBoxes can be set by passing the widths as a list of values separated by commas to the ColumnWidths property. The widths can be passed in pixels or as percentages of the total width of the **ListBox**. If you don't pass widths for all the columns, the remaining columns will be evenly spaced over the remaining space. If too many widths are passed, the additional values are ignored. If the total of the widths passed is greater than the width of the **ListBox**, then the remaining columns will be truncated.

Specific cells in a multi-column **ListBox** can be accessed using the Cell method.

### Decimal Alignment

When you use decimal alignment in a cell or column, you must take into account the fact that REALbasic aligns the decimal separator with the right edge of the column or cell. You must pass a negative number to CellAlignmentOffset or ColumnAlignmentOffset to make room for the numbers to the right of the decimal place. The correct value to pass depends on the number of digits to the right of the decimal place in the column or cell.

The following illustrates the problem and the solution.

| First  | Last   | Num1 | Num2 |
|--------|--------|------|------|
| Rollin | Hand   | 1.52 | 45.  |
| James  | Phelps | .78  | 0.   |

Without ColumnAlignmentOffset

| First  | Last   | Num1 | Num2  |
|--------|--------|------|-------|
| Rollin | Hand   | 1.32 | 45.75 |
| James  | Phelps | .785 | 0.38  |

ColumnAlignmentOffset(2)=-32

ColumnAlignmentOffset(3)=-18

### Hierarchical ListBoxes

Creating a simple hierarchical **ListBox** is more involved than a two-column **ListBox** because you must manage hiding and displaying the sublist data. A simple way to do this is to assign the sublists to a "hidden" column in the **ListBox** and toggle the display of that data when the user double-clicks on a "parent" element.

You create a row with a disclosure triangle using the AddFolder method (rather than the AddRow method) and then set the Hierarchical property to [True](#). See the Example for a simple hierarchical **ListBox** with one level.

## Resizing Columns

There are two “modes” for column resizing. There is no formal mode property. Rather, the “mode” is implicitly set according to whether every column width is specified as an absolute amount. If you specify all columns either in pixels or as a percentage, you will be using the second mode. If you use an asterisk or leave a column width blank, you will be using the first mode.

- A change to one column width affects the width of another column.

If column  $i$  gets bigger, column  $i+1$  gets smaller by the same amount. This mode is great when using a ListBox without a horizontal scrollbar. You turn this mode on when you have at least one column width that is blank, or specified using an asterisk (e.g. "", " ", "\*!", or "4\*!").

NOTE: By design you can't resize the right edge of the last column in this mode. To resize the last column you need to resize the previous column.

- Each column width is independent and can grow or shrink on its own.

You are responsible when the user does this, and you need to provide a horizontal scrollbar so that the user can get to the any headers that have been pushed out of view to the right. You enable this mode by making sure every column width is specified in terms of an absolute pixel width, or a percentage width (e.g. “20”, or “35%”). If you use an asterisk or leave a column width blank, you will automatically be using the first mode.

You can switch between mode 1 and 2 at runtime using the same criteria as above.

The ColumnWidths property (the only one that can be set in the IDE), is equivalent to the concatenation of all of the ColumnWidthExpressions.

ColumnWidthExpressions are strings and they can represent several different types of column width calculations: absolute pixels (e.g., “45”), percentages (e.g. “22.3%”), and asterisk widths (or blanks) (e.g. " ", "4\*!").

ColumnWidthExpressions retain their type even when a column is resized. This means that if you:

- Resize a window to which a ListBox is locked, it will grow or shrink. The columns grow or shrink as well if their expressions were \*-based (unless you use “0\*”), or percentage based (0%). If you want them to stay fixed, you need to express the ColumnWidthExpression as an absolute pixel value.
- If you resize a column by dragging it internally, it will recompute its percentage or asterisk value. This is so that you can, say, start with a two-column ListBox with no column widths specified (each column will take up half the space). Then drag one column to take up 3/4 of the space, then enlarge the ListBox, and now both column widths will enlarge so that their widths remain in a 3/4 to 1/4 ratio.

Changing the pixel value of a column will not change its fundamental type, but will change the value of that type.

Finally, if you want to create columns that won't get resized, change the UserResizable property for each of the columns in question. If you are using mode 1, you will need to change the UserResizable property for both the column and the one to its left.

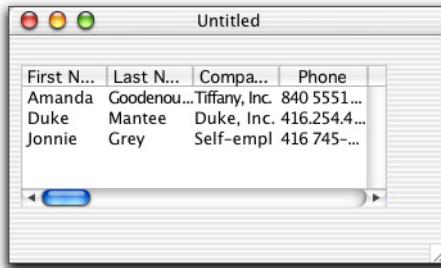
### Databases

A **ListBox** is often used to display the results of [database](#) queries. A **ListBox** can be bound to a [DatabaseQuery](#) control so that it displays the results of the SQL query passed to the [DatabaseQuery](#) control automatically. See the example of object binding in the chapter “Creating Databases with REALbasic” in the *User’s Guide*. Otherwise, a **ListBox** can be populated with the results of a query programmatically. See the example for [DatabaseField](#).

### Scrolling a ListBox Horizontally

The ScrollBarHorizontal property automatically adds a horizontal scrollbar to the **ListBox**. The ScrollPositionX property can be used to get the position of the thumb or scroll the **ListBox** horizontally. This is the easiest way to scroll a **ListBox**.

You can also use the ScrollPositionX property in conjunction with a [ScrollBar](#) control to scroll contents of the **ListBox** horizontally. You place a [ScrollBar](#) control below the **ListBox**, as shown below:



This **ListBox** has nine columns, eight of which have a width of 100 pixels and one has a width of 50 pixels.

The [ScrollBar](#) control has the following code in its Open event handler:

```
Me.maximum=500
Me.minimum=0
Me.lineStep=50
```

The values for maximum and linestep were chosen to match the total width of the **ListBox**’s columns. Its ValueChanged event handler has the following line of code:

```
ListBox1.scrollPositionX=Me.value
```

In this way, the user can scroll the **ListBox** horizontally, bringing all columns into view.

The image displays two separate horizontal scrollable tables. The top table has columns for First Name, Last Name, Company, and Phone. The bottom table has columns for Address, City, State, and Zip. Both tables contain three rows of data and feature scroll bars at the bottom for navigating through the data.

| First Name | Last Name  | Company       | Phone |
|------------|------------|---------------|-------|
| Amanda     | Goodenough | Tiffany, Inc. | 555 5 |
| Duke       | Mantee     | Duke, Inc.    | 416 5 |
| Jonnie     | Gray       | Self-empl.    | 3125  |

| Address        | City     | State | Zip   |
|----------------|----------|-------|-------|
| 1212 Broadway  | New York | NY    | 10001 |
| 1451 S. Kelvin | Chicago  | IL    | 60637 |
| 45 Summit      | Atlanta  | GA    | 54821 |

## Customized Scroll Controls

If you want miniature scrollbars or scrollbars that leave room for status bars or buttons, set `ScrollBarVertical` and/or `ScrollBarHorizontal` to `False` and use separate `ScrollBar` controls in conjunction with whatever other controls that you want to locate in the a portion of the area normally taken up by standard scrollbars. Use the `ScrollPosition` and `ScrollPositionX` properties to control scrolling, as described in the preceding section.

## Horizontal and vertical rules

The following screen shots illustrate the four types of horizontal and vertical rules that can be drawn with the `GridLinesHorizontal` and `GridLinesVertical` properties and the `CellBorderTop`, `CellBorderLeft`, `CellBorderRight`, and `CellBorderBottom` properties.

The Default style is “None.”

This table uses thin dotted horizontal and vertical grid lines. The data consists of five rows and three columns: Name, Phone, and Email. The scroll bar is visible on the right side.

| Name    | Phone    | Email               |
|---------|----------|---------------------|
| Milton  | 555-1210 | milt@fredonia.com   |
| Herbert | 555-1211 | herb@fredonia.c...  |
| Julius  | 555-1212 | julius@fredonia.... |
| Adolph  | 555-1213 | adolph@fredoni...   |

Thin Dotted

This table uses thin solid horizontal and vertical grid lines. The data consists of five rows and three columns: Name, Phone, and Email. The scroll bar is visible on the right side.

| Name    | Phone    | Email               |
|---------|----------|---------------------|
| Milton  | 555-1210 | milt@fredonia.com   |
| Herbert | 555-1211 | herb@fredonia.c...  |
| Julius  | 555-1212 | julius@fredonia.... |
| Adolph  | 555-1213 | adolph@fredoni...   |

Thin Solid

This table uses thick solid horizontal and vertical grid lines. The data consists of five rows and three columns: Name, Phone, and Email. The scroll bar is visible on the right side.

| Name    | Phone    | Email               |
|---------|----------|---------------------|
| Milton  | 555-1210 | milt@fredonia.com   |
| Herbert | 555-1211 | herb@fredonia.c...  |
| Julius  | 555-1212 | julius@fredonia.... |
| Adolph  | 555-1213 | adolph@fredoni...   |

Thick Solid

This table uses double thin solid horizontal and vertical grid lines. The data consists of five rows and three columns: Name, Phone, and Email. The scroll bar is visible on the right side.

| Name    | Phone    | Email               |
|---------|----------|---------------------|
| Milton  | 555-1210 | milt@fredonia.com   |
| Herbert | 555-1211 | herb@fredonia.c...  |
| Julius  | 555-1212 | julius@fredonia.... |
| Adolph  | 555-1213 | adolph@fredoni...   |

Double Thin Solid

With the `CellBorderTop`, ...`Left`, ...`Bottom`, and ...`Right` methods, you can apply these ruling styles to selected cells or even selected cell borders.

## ListBox Control

For example, in this ListBox, a cell containing a phone number is highlighted using Thick Solid borders while the remainder of the ListBox uses Thin Dotted rules:

| Name    | Phone    | Email              |
|---------|----------|--------------------|
| Milton  | 555-1210 | milt@fredonia.com  |
| Herbert | 555-1211 | herb@fredonia.c... |
| Julius  | 555-1212 | julius@fredonia... |
| Adolph  | 555-1213 | adolph@fredoni...  |

**Examples** Adding a row to ListBox1:

```
ListBox1.addrow "October"
```

Inserting a row at row 1 in ListBox1:

```
ListBox1.insertRow 1, "October"
```

Changing all items in the ListBox to bold, underline:

```
ListBox1.bold=True  
ListBox1.underline=True
```

Copying the fifth element of ListBox1 to another variable:

```
Dim e as String  
e=ListBox1.list(4)
```

Adding a column to ListBox1 and setting the widths of the columns to 50 and 65 pixels, respectively:

```
ListBox1.columnCount=2  
ListBox1.columnWidths="50,65"
```

Setting the number of columns of ListBox1 to three and setting the widths of the columns to 60%, 20% and 20% respectively:

```
ListBox1.columnCount=3  
ListBox1.columnWidths="60%,20%,20%"
```

If ListBox1 is 100 pixels wide and has three columns, the following code will set the columns widths as indicated but the last column will only be 10 pixels wide instead of 20:

```
ListBox1.columnWidths="60,30,20"
```

If ListBox1 is 100 pixels wide and has three columns, the following code will set the columns widths but the last column will not be displayed:

```
ListBox1.columnWidths="60,40,20"
```

Copying the fifth row of the third column of ListBox1 to another variable:

```
Dim e as String  
e=ListBox1.cell(4,2)
```

Assigning a value to the fifth row of the third column of ListBox1:

```
ListBox1.cell(4,2)="Bill"
```

Setting the fifth row of the third column of ListBox1 to bold, italic:

```
ListBox1.cellBold(4,2)=True  
ListBox1.cellItalic(4,2)=True
```

Setting the picture for the fifth row of ListBox1 to “MyFolderPicture”:

```
ListBox1.RowPicture(4)=MyFolderPicture
```

Setting up the DragRow event handler to allow the user to drag a value from a ListBox:

```
Function DragRow(Drag as DragItem, Row as Integer) as Boolean  
Drag.Text=ListBox1.List(Row)  
Return True
```

Summing the numeric values of the selected rows:

```
Dim i, total as Integer  
For i=0 to ListBox1.ListCount-1  
If ListBox1.Selected(i) then  
    total=total+Val(ListBox1.List(i))  
End if  
Next
```

This example expands the first row of ListBox1 (if it is collapsed) or collapses it (if it was expanded). The row must have been added with the AddFolder method:

```
ListBox1.Expanded(1)=Not ListBox1.Expanded(1)
```

This example populates a three-column ListBox with headings:

```
ListBox1.heading(0)="ID"  
ListBox1.heading(1)="JobTitle"  
ListBox1.heading(2)="Name"
```

This example sets up a **ListBox** with four visible columns plus one hidden column. Column zero is hidden:

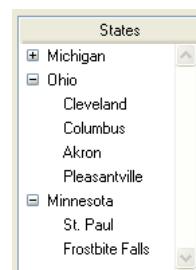
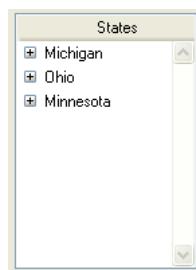
```
Me.columnCount=5  
Me.columnWidths="0,25%,25%,25%,25%"  
Me.heading(0)="ID"  
Me.heading(1)="FirstName"  
Me.heading(2)="LastName"  
Me.heading(3)="Phone"  
Me.heading(4)="Zip"
```

The following line of code displays the value of the hidden column in the selected row:

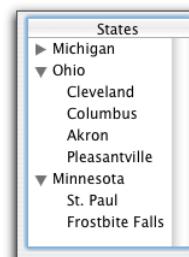
```
MsgBox ListBox1.Cell(ListBox1.ListIndex,0)
```

**Hierarchical  
ListBoxes** The following example creates a single-level hierarchical ListBox.

Windows



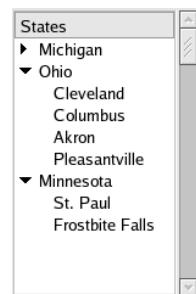
Macintosh



Linux



Collapsed State



An Expanded State

Windows uses plus and minus signs to indicate disclosure; Macintosh and Linux use disclosure triangles. To display these widgets, set the `set` the Hierarchical property of the ListBox to [True](#) in the Properties pane.

The following code, which is in the ListBox's Open event handler, populates the hierarchy: The `s1` string contains the parent level and `sub1` contains the elements that are nested within each of `s1`'s elements. It is a list of comma-delimited lists, with each

list delimited by semicolons. The elements of sub1 are initially hidden because they are stored in a hidden column.

```
Dim i, u as Integer
Dim s1,sub1 as string
Me.columnwidths="150,0"
s1="Michigan,Ohio,Minnesota"
sub1="Grand Blanc,Bad Axe,Flint,Benton Harbor,
Detroit;Cleveland,Columbus,Akron, Pleasantville; St. Paul,Frostbite Falls"
u=CountFields(s1, ", ")
For i=1 to u
If NthField(sub1,";",i)<> " " then
Me.addfolder ""
Me.cell(i-1,1)=NthField(sub1, ";",i)
end if
Me.cell(i-1,0)=NthField(s1, ", ",i)
Next
Me.ColumnCount=1
```

Note that the AddFolder method, rather than AddRow, is used to add the State names. The following line of code in the DoubleClick event handler toggles the expanded state of the row that was double-clicked:

```
Me.expanded(Me.listindex)=Not Me.expanded(Me.listindex)
```

The following code in the ExpandRow event handler runs when the user double-clicks a collapsed element:

```
Dim s1 as string
Dim i,u as Integer
s1=me.cell(row,1)
u=CountFields(s1, ", ")
For i=1 to u
Me.addrow ""
Me.cell(me.lastIndex,0)=NthField(s1, ", ",i)
Next
```

It creates the sublist rows each time the user double-clicks a collapsed state name.

The following code is in the CollapseRow event handler:

```
Dim i,u,NSubRows as Integer
NSubRows=CountFields(Me.cell(row,1), ", ")
u=row+1
For i=row+NSubRows downto u
Me.removerow i
Next
```

## Drag and Drop between ListBoxes

It removes the rows that were created by the ExpandRow event handler.

The following example allows the user to drag one row from ListBox1 to ListBox2. ListBox1 has its EnableDrag property set to `True` and its SelectionType property set to zero (Single). Its DragRow event handler is as follows:

```
Function DragRow (Drag as DragItem, Row as Integer) as Boolean
    drag.text=Me.List(row)
    Return True //allow the drag
```

ListBox2's Open event handler has the line:

```
Me.AcceptTextDrop
```

Its DropObject event handler is this:

```
Sub DropObject (obj as DragItem)
    Me.AddRow obj.text //adds the dropped text as a new row
```

The following example allows the user to drag more than one row from ListBox1 to ListBox2. The dragged rows are added to the end of the list.

ListBox1 has its EnableDrag property set to `True`, enabling items in its list to be dragged, and its SelectionType property set to 1 (Multiple row selection). Its DragRow event handler is as follows:

```
Function DragRow (Drag as DragItem, Row as Integer) as Boolean
    Dim nRows, i as Integer
    nRows=Me.ListCount-1
    For i=0 to nRows
        If Me.Selected(i) then
            Drag.AddItem(0,0,20,4)
            Drag.Text=Me.List(i) //get text
        End if
    Next
    Return True //allow the drag
```

It uses the AddItem method of the `DragItem` to add an additional item to the DragItem each selected row. The DropObject event handler then cycles through all items to retrieve all dragged rows.

ListBox2 has the following line of code in its Open event handler. It permits it to receive dragged text.

```
Me.AcceptTextDrop
```

## ListBox Control

---

Its DropObject event handler checks to see if the dragged object is text; if it is, it adds a row to the end of the list and assigns the text property of the dragged object to the new row: It loops through all items in the DragItem until NextItem returns [False](#).

```
Sub DropObject(obj as DragItem)
Do
  If Obj.TextAvailable then
    Me.AddRow(Obj.Text)
  End if
Loop until Not obj.NextItem
```

You can also drag from ListBox1 to the desktop to get a text clipping or to another application that supports text drag and drop.

### Colors

This example, which is placed in the CellBackgroundPaint event, assigns alternating colors to the rows in a ListBox:

```
If row Mod 2=0 then
  g.foreColor=RGB(158,238,255)
else
  g.foreColor=RGB(172,255,145)
end if
g.FillRect 0,0,g.width,g.height
```

Notes: The CellBackgroundPaint event passes the parameters g ([Graphics](#)), and the row and column numbers (as [Integer](#)). You can assign a color to the ForeColor property by creating it as a constant in the App class or a module and assign the color constant to the ForeColor property.

The following line in the CellTextPaint event tells REALbasic to draw the text in the preceding example in red:

```
g.foreColor=RGB(255,0,0)
```

The CellTextPaint event is passed the coordinates of the suggested starting position to draw text in the parameters x and y. You can use them in a call to the DrawString property to specify the string to draw in a particular cell:

```
If row=4 and column=1 then
  g.foreColor=RGB(255,0,0)
  g.DrawString "Payment Overdue!",x,y
end if
Return True
```

### Sorts

To sort a ListBox, set the column on which the ListBox will be sorted with the SortedColumn property. Specify the sort direction on that column with the ColumnSortDirection property, and then do the sort by calling the Sort method.

The following example sorts a Listbox in descending order on the first column.

```
//first column, descending order
ListBox1.ColumnsortDirection(0)=ListBox.SortDescending
ListBox1.SortedColumn=0 //first column is the sort column
ListBox1.Sort
```

You can also sort a column based on the current value of ColumnSortDirection by calling the PressHeader method. This method programmatically clicks the header for the column passed to it.

**Custom Sorts** The following example uses the CompareRows event to sort columns of numbers. If you rely on default sorting, numbers are not sorted in numerical order, i.e., 2 is greater than 100 and less than 200.

```
Function CompareRows(row1 as Integer, row2 as Integer, column as Integer,
ByRef result as Integer) as Boolean
If Val(Me.Cell(row1,column))> Val(Me.cell(row2,column)) then
    result=1
else
    result=-1
End if
Return True //tells RB to use the custom sort
```

With this code in place, the correct (numerical) sorting is done whenever the user clicks the header area. Test to be sure that the custom sort affects only the numerical columns.

To sort dates, store the TotalSeconds property of the [Date](#) in a column of width zero and sort the rows based on that column.

**See Also** [DatabaseQuery](#), [EditField](#) controls; [DatabaseField](#), [ListColumn](#), [RecordSet](#) classes.

## ListColumn Class

Used to refer to a particular column in a [ListBox](#) via its Column property.

**Super Class** [Object](#)

### Properties

| Name               | Type                    | Description  |
|--------------------|-------------------------|--|
| MaxWidthActual     | <a href="#">Integer</a> | The maximum width of the column (pixels).  |
| MaxWidthExpression | <a href="#">String</a>  | The maximum width of the column as a string expression that can include spaces and the percent sign. |

| Name               | Type                    | Description  |
|--------------------|-------------------------|--|
| MinWidthActual     | <a href="#">Integer</a> | The minimum width of the column (pixels).  |
| MinWidthExpression | <a href="#">String</a>  | The minimum width of the column as a string expression that can include spaces and the percent sign.   |
| UserResizable      | <a href="#">Boolean</a> | <a href="#">True</a> if the column is resizable; the default is <a href="#">False</a> .  |
| WidthActual        | <a href="#">Integer</a> | The width of the column in pixels.   |
| WidthExpression    | <a href="#">String</a>  | Column width as a string expression that can include the percent sign, a decimal point, or blanks. Width can be specified as a value in pixels, a percentage, or a relative length. To obtain the absolute width of a column specified using percent or a relative length, call <a href="#">WidthActual</a> . If you use * or %, the actual width will vary as the width of the <a href="#">ListBox</a> changes. |

**Notes**

Use the UserResizable property to set the property, e.g.,

```
ListBox1.Column(1).UserResizable=False
```

To set column widths, you can use a mixture of pixels, percentages, and relative lengths. Column widths specified in pixels are guaranteed to have the specified width with the user resizes the [ListBox](#) or resizable columns. Column widths specified in percentages are guaranteed to have that percentage of the visible width of the [ListBox](#). The column widths specified using the "\*" divide up the remaining width proportionally. Resizing a column will resize the value of the expression. If you resize the [ListBox](#), both the percentage and relative lengths recompute their actual widths. There are two resizing “modes”; for more information, see the section [“Resizing Columns” on page 347](#) in the documentation for the [ListBox](#) control.

For example, if there are three columns, the specification:

```
Listbox1.Column(0).widthExpression = "3*"
Listbox1.Column(1).widthExpression = "*"
Listbox1.Column(2).widthExpression = "10"
```

allocates a fixed amount of space to Column(2). There remaining width is divided up into four segments and allocates it between Columns 0 and 1 in the ratio of 3 to 1. This means that Column(0) gets the width of the [ListBox](#) minus 10 times 3/4, Column(1) gets the width of the [ListBox](#) minus 10 times 1/4, and Column(2) gets 10 pixels. As the [ListBox](#) changes in size (due to a window resize, for example) columns 0 and 1 will change to reflect that growth, while column 2 will not.

Header End caps are added for any additional unused space if headers are used. Header end caps do nothing when clicked.

If the string expression passed to WidthExpression, MinWidthExpression, or MaxWidthExpression cannot be evaluated to a width or a percentage width, an

[UnsupportedFormatException](#) will occur. The Message property of this exception describes in English what went wrong.

## Example

The following line in the Open event of a [ListBox](#) makes all the columns in the [ListBox](#) resizable:

```
Me.Column(-1).UserResizable=True
```

The following line sets the maximum width of the first column in a [ListBox](#) to 75% of the total width of the ListBox.

```
Listbox1.Column(0).MaxWidthExpression="75%"
```

## See Also

[ListBox](#) class.

# ListSelectionNotificationReceiver Class Interface

Used to program custom object bindings. See the example of custom object bindings in the User's Guide.

## Methods

| Name              | Parameters | Description                |
|-------------------|------------|----------------------------|
| SelectionChanging |            | The selection is changing. |
| SelectionChanged  |            | The selection has changed. |

---

# ListSelectionNotifier Class Interface

Used to program custom object bindings.

## Methods

| Name                                    | Parameters  | Description  |
|---|---|--|
| addListSelectionNotificationReceiver    | receiver as <a href="#">ListSelectionNotificationReceiver</a> | Adds receiver object to custom class interface.      |
| removeListSelectionNotificationReceiver | receiver as <a href="#">ListSelectionNotificationReceiver</a> | Removes receiver object from custom class interface. |

## Log Function

Returns the natural logarithm of the value specified.

**Syntax**

*result=Log (value)*

| Part   | Type                   | Description                                  |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The natural logarithm of value.              |
| value  | <a href="#">Double</a> | The value you want the natural logarithm of. |

**Examples**

This example uses the **Log** function to return the natural logarithm of a number.

```
Dim d As Double
d=Log(10) //returns 2.3025851
```

## Logic Operations require Boolean Values Error

You used a variable that was not of type [Boolean](#) or an expression that did not evaluate to a Boolean where REALbasic was expecting a [Boolean](#).

**See Also**

[Boolean](#) data type.

---

## Lowercase Function

Converts all characters in a [string](#) to lowercase characters.

**Syntax**

*result=Lowercase(value)*

OR

*result=stringVariable.Lowercase*

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | A copy of the original string with all characters converted to their lowercase equivalent. |
| value          | <a href="#">String</a> | The original string.   |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .  |

**Notes**

Returns the value with all alphabetic characters in lowercase.

**Examples** The example below converts the value passed to lowercase.

```
Dim s As String
s=Lowercase("tHe Quick fOX") //returns "the quick fox"
s=Lowercase("THE 5 LAZY DOGS") //returns "the 5 lazy dogs"
```

**See Also** [Titlecase](#), [Uppercase](#) functions.

## LTrim Function

Returns the [string](#) passed with leading (left side) whitespaces removed.

**Syntax** **result=LTrim(SourceString)**

OR

**result=stringVariable.Ltrim**

| Part           | Type                   | Description   |
|----------------|------------------------|---|
| result         | <a href="#">String</a> | SourceString with leading whitespaces removed.                                |
| SourceString   | <a href="#">String</a> | The source, a copy of which, to be returned with leading whitespaces removed. |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .                                 |

**Notes** **LTrim** uses the list of unicode “whitespace” characters at <http://www.unicode.org/Public/UNIDATA/PropList.txt>.

**Examples** This example removes the whitespaces from the left side of the [string](#) passed:

```
Dim s as String
s=LTrim(" Hello World ")
//Returns "Hello World "
```

**See Also** [Asc](#), [Chr](#), [InStr](#), [Left](#), [Len](#), [Mid](#), [Right](#), [RTrim](#), [Trim](#) functions.

## Material Class

Used to wrap an object shader (i.e., a texture and/or a solid color). Used with the [Object3D](#) and [Trimesh](#) classes in an [RB3DSpace](#).

**Super Class** [Object](#)

**Properties**

| Name            | Type                    | Description   |
|-----------------|-------------------------|---|
| DiffuseColor    | <a href="#">Color</a>   | Used for untextured materials when HasDiffuseColor is <a href="#">True</a> . Under certain rendering conditions, it may mix with a color as well. |
| Handle          | <a href="#">Integer</a> | Used to access the underlying data structure (a TQ3AttributeSet, whose reference count is not incremented before returning it to you).            |
| HasDiffuseColor | <a href="#">Boolean</a> | <a href="#">True</a> if the Material has a diffuse color. The DiffuseColor property is used when HasDiffuseColor is <a href="#">True</a> .        |
| Texture         | <a href="#">Picture</a> | The texture of the material. The Texture can be <a href="#">Nil</a> .   |

**See Also**

[Bounds3D](#), [Element3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

---

## Max Function

Returns the largest value passed to it.

**Syntax**

**result=Max(value1, value2[...valueN])**

| Part   | Type                   | Description  |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The maximum value of value1 to valueN.   |
| value1 | <a href="#">Double</a> | The first number for comparison.   |
| value2 | <a href="#">Double</a> | The second number for comparison.  |
| valueN | <a href="#">Double</a> | Optional: Max requires two parameters but optionally can take more than two. N is the number of parameters passed. |

**Notes**

The **Max** function compares two or more numbers and returns the largest of the passed values.

**Examples**

This example uses the **Max** function to return the largest of the values passed.

```
Dim d As Double
d=Max(3.01, 4.05) //returns 4.05
d=Max(3.012, 3.011, 1.56) //returns 3.012
```

**See Also**

[Abs](#), [Ceil](#), [Exp](#), [Floor](#), [Min](#), [Pow](#), [Sqrt](#) functions.

# MD5 Function

Returns the MD5 message-digest value of a [String](#).

## Syntax

**result=MD5(String)**

| Part   | Type                   | Description  |
|--------|------------------------|--|
| Result | <a href="#">String</a> | The MD5 message digest form of the passed <a href="#">String</a> . |
| String | <a href="#">String</a> | The input string to be parsed.                                     |

## Notes

The MD5 algorithm is intended for digital signature applications, where a file must be compressed in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. It is a technology for verifying data integrity and is similar in concept to checksum techniques.

Use the **MD5** function to process the raw data to be processed. For complete information on the MD5 message digest format, see RFC 1321.

The [MD5Digest](#) class implements the same algorithm but is capable of processing a large file in segments. The MD5 function is better suited for short strings that need to be encrypted such as passwords, account numbers, and private transaction strings.

## See Also

[MD5Digest](#) class.

# MD5Digest Class

Processes a [String](#) and returns the message-digest form of the input string.

## Super Class

[Object](#)

## Properties

| Name  | Type                   | Description                          |
|-------|------------------------|--------------------------------------|
| Value | <a href="#">String</a> | Contains the current message digest. |

## Methods

| Name    | Parameters                     | Description   |
|---------|--------------------------------|---|
| Process | Data as <a href="#">String</a> | Data is the text to be processed.   |
| Clear   |                                | Resets the MD5Digest object so that you can start with a new data stream. |

### Notes

The **MD5Digest** class enables you to process a string in segments. Pass each string segment to the Process method. The Value property contains the current message digest and the Clear method clears the **MD5Digest** object so that you can repeat the process.

The MD5 message digest algorithm takes a message of any length and produces a 128-bit “fingerprint” or *message digest* of the input string. The MD5 algorithm is useful for digital signature applications, where a large file must be processed in a secure manner before being encrypted with a secret key under a system such as RSA. See RFC 1321 for complete information.

### See Also

[MD5](#) function.

---

## MDIWindow Class

Used to configure the MDI window on Multiple Document Interface applications (Windows only). Specify the Multiple Document Interface option in the Properties pane for the App class.

Super Class [Object](#)

### Properties

| Name      | Type                    | Description   |
|-----------|-------------------------|---|
| Handle    | <a href="#">Integer</a> | Returns a handle to the MDI window. Any visible Win32 control should have a handle.               |
| Height    | <a href="#">Integer</a> | The height of the MDI window.   |
| Left      | <a href="#">Integer</a> | The distance (in pixels) between the left edge of the screen and the left edge of the MDI window. |
| MinHeight | <a href="#">Integer</a> | The minimum height to which the MDI window can be resized.  |
| MinWidth  | <a href="#">Integer</a> | The minimum width to which the MDI window can be resized.   |
| MaxHeight | <a href="#">Integer</a> | The maximum height to which the MDI window can be resized.  |
| MaxWidth  | <a href="#">Integer</a> | The maximum width to which the MDI window can be resized.   |
| Title     | <a href="#">String</a>  | The MDI window title.   |
| Top       | <a href="#">Integer</a> | The distance (in pixels) between the top edge of the screen and the top edge of the MDI window.   |
| Visible   | <a href="#">Boolean</a> | If <a href="#">True</a> , the MDI window will be visible when it is opened.                       |
| Width     | <a href="#">Integer</a> | The width of the MDI window.  |

## Events

| Name    | Parameters | Description                            |
|---------|------------|--|
| Moved   |            | Occurs when the MDI window moves.      |
| Open    |            | Occurs when the MDI window is created. |
| Resized |            | Occurs when the MDI window resizes.    |

## Methods

| Name     | Parameters | Description               |
|----------|------------|---------------------------|
| Maximize |            | Maximizes the MDI window. |
| Minimize |            | Minimizes the MDI window. |
| Restore  |            | Restores the MDI window.  |

## Notes

You have the option of using the Multiple Document Interface (MDI) for the Windows build of your application. If you select this option, all of your application's windows will be enclosed in a "parent" window called the MDI window. If you don't select the MDI interface, then your application will be a Single Document Interface (SDI) application and your application's windows will be directly on the desktop (since there is no MDI window). This class allows you to set certain properties and behaviors of the MDI window.

You can set the MDIWindow property of the [App](#) class to a new instance of an **MDIWindow** subclass. Use this technique to implement the events for an **MDIWindow**.

## Getting the Handle of a Child Window

There are two handles that a [Declare](#) writer may need when it comes to MDIWindows. The Handle property returns the frame window's handle. If you want the MDICLIENT

handle for doing things like cascading child windows, then you need to get the child window based on the Handle with a code snippet like this:

```
Module MDIWindowExtensions
Private Dim mClientHandle as Integer
Function MDIClientHandle(extends w as MDIWindow) As Integer
    // There's two different handles used for an MDI window. The frame
    // handle (which is MDIWindow.Handle), and the client handle. This
    // gets the client handle, which is used for things like tiling or cascading
    // child windows.
    If mClientHandle <> 0 then return mClientHandle
    #if TargetWin32
        Declare Sub EnumChildWindows Lib "User32" ( parent as Integer,
            proc as Ptr,IParam as Integer )
        mClientHandle = 0
        // Do the enumeration
        EnumChildWindows( w.Handle, AddressOf enumChildProc, 0 )
        // Return the client's handle
        Return mClientHandle
    #endif
End Function
Function enumChildProc(hwnd as Integer, IParam as Integer) As Boolean
    #if TargetWin32
        // We need to figure out what class this window belongs to
        Soft Declare Function GetClassNameW Lib "User32" ( hwnd as Integer,
            name as Ptr, count as Integer ) as Integer
        Soft Declare Function GetClassNameA Lib "User32" ( hwnd as Integer,
            name as Ptr, count as Integer ) as Integer
        Dim classNamePtr as New MemoryBlock( 256 )
        Dim className as String
        if System.IsFunctionAvailable( "GetClassNameW", "User32" ) then
            Dim cnt as Integer = GetClassNameW( hwnd, classNamePtr,
                classNamePtr.Size )
            className = classNamePtr.WString( 0 )
        else
            Dim cnt as Integer = GetClassNameW( hwnd, classNamePtr,
                classNamePtr.Size )
            className = classNamePtr.CString( 0 )
        end if
        // If the name is MDICLIENT, then we're done
        If className = "MDICLIENT" then
            mClientHandle = hwnd
            Return False
        end if
        Return True
    #endif
End Function
End Module
```

**Example**

The following code in the Open event of the App class (which is added to your project by default) sets some properties of the MDI window:

```
App.MDIWindow.Title = "REAL Scheduler"  
App.MDIWindow.MinHeight = 225  
App.MDIWindow.Height = 72
```

**See Also**

[Application](#) class; [App](#) object.

## Me Keyword

**Me** refers to the control that fired the current event. Outside an event handler, it simply refers to the current object ([Self](#)).

**Notes**

When called within an event handler or method of a control on a form, **Me** will always be a reference to the control that owns the event handler or method where **Me** was called. **Me** is only different from [Self](#) inside an event handler for a control on a window.

This means that in [PushButton1](#)'s Action event handler:

```
PushButton1.Enabled = False
```

and

```
Me.Enabled = False
```

mean the same thing. When **Me** is called within the method of a class, **Me** will be a reference to the instance of the class in use. The **Me** keyword is different from the [Self](#) keyword in that [Self](#) always refers to the object's parent and not the object itself.

For code for a control in a window, [Self](#) refers to the window, and **Me** refers to the control.

One big advantage of using the **Me** keyword when referring to a control in one of its event handlers is that if you change the name of the control, the code in its own event handlers will not need to be updated. Also, you can copy the code to another control of the same type and it will work without modification.

## Me Cannot be used in a Method of a Module Error

---

### Examples

Me can be used in place of a control's name inside any of its event handlers. For example, this example in the Open event of a [ListBox](#) sets the column widths and populates the header row:

```
Me.columncount=5  
Me.columnwidths="0,25%,25%,25%,25%"  
Me.heading(0)="ID"  
Me.heading(1)="FirstName"  
Me.heading(2)="LastName"  
Me.heading(3)="Phone"  
Me.heading(4)="Zip"
```

The following example in the DropObject event handler of a [ListBox](#) adds a row for each item being dropped.

```
Sub DropObject(obj as DragItem)  
Do  
If Obj.TextAvailable then  
Me.AddRow(Obj.Text)  
End if  
Loop until Not obj.NextItem
```

The following example in the MouseEnter event handler of a Canvas changes the mouse pointer to an open hand.

```
Me.MouseCursor=System.Cursors.HandOpen
```

The following code in theMouseDown event handler of an [ImageWell](#) creates a [DragItem](#) and enables dragging:

```
Dim d as DragItem  
d=NewDragItem(Me.left,Me.top,Me.width,Me.height)  
d.Picture=Me.image  
d.Drag //Allow the drag
```

### See Also

[Self](#) keyword.

---

## Me Cannot be used in a Method of a Module Error

You can't use [Self](#) or [Me](#) in a method or function in a module because there is no parent window or control. [Self](#) can be used only when there is a parent window for the method and [Me](#) can be used only in a control's event handler.

**See Also** [Self](#), [Me](#) keywords.

## MemoryBlock Class

A **MemoryBlock** object allocates a sequence of bytes in memory and manipulates those bytes directly. A **MemoryBlock** can be passed in place of a [Ptr](#) when used in a [Declare](#) call.

**Super Class** [Object](#)

### Properties

| Name         | Type                    | Description   |
|--------------|-------------------------|---|
| LittleEndian | <a href="#">Boolean</a> | Sets the endianness of a <b>MemoryBlock</b> . The default is the endianness of the platform on which the code is being compiled.  |
| Size         | <a href="#">Integer</a> | The size of the <b>MemoryBlock</b> in bytes. If you assign a value to this property, REALbasic resizes the <b>MemoryBlock</b> , retaining as much of the existing data as possible. Returns an <a href="#">Integer</a> . <b>MemoryBlock</b> has a class constant, <b>SizeUnknown</b> ( <a href="#">Integer</a> ), whose value is -1. <b>Size</b> returns <b>SizeUnknown</b> when the <b>MemoryBlock</b> is of unknown size. You can get this condition with <a href="#">Declare</a> statements. |

### Methods

| Name         | Parameters   | Description  |
|--------------|--|--|
| BooleanValue | Offset as <a href="#">Integer</a>                                    | Get (0= <a href="#">False</a> ; 1= <a href="#">True</a> ) or set (0= <a href="#">False</a> ; 1= <a href="#">True</a> ) a one-byte <a href="#">Boolean</a> value. Returns a <a href="#">Boolean</a> .                               |
| Byte         | Offset as <a href="#">Integer</a>                                    | Returns an <a href="#">integer</a> .   |
| ColorValue   | Offset as <a href="#">Integer</a><br>Bits as <a href="#">Integer</a> | Get or set a <a href="#">Color</a> in one of three formats, selected by Bits. 32: 32-bit color (high byte unused)<br>24: 24-bit color<br>16: 16-bit color (5 bits per color, high bit unused)<br>Returns a <a href="#">Color</a> . |
| CString      | Offset as <a href="#">Integer</a>                                    | Returns a <a href="#">String</a> . The string ends with a null ( <a href="#">Chr(0)</a> ).   |
| DoubleValue  | Offset as <a href="#">Integer</a>                                    | Gets or sets a <a href="#">Double</a> value.   |
| Int16Value   | Offset as <a href="#">Integer</a>                                    | Gets or sets an <a href="#">Int16</a> value.   |
| Int32Value   | Offset as <a href="#">Integer</a>                                    | Gets or sets an <a href="#">Int32</a> value.   |
| Int64Value   | Offset as <a href="#">Integer</a>                                    | Gets or sets an <a href="#">Int64</a> value.   |
| Int8Value    | Offset as <a href="#">Integer</a>                                    | Gets or sets an <a href="#">Int8</a> value.  |

## MemoryBlock Class

---

| Name        | Parameters   | Description   |
|-------------|--|---|
| LeftB       | Length as <a href="#">Integer</a>  | Returns a new <b>MemoryBlock</b> of the specified size using the same endianness as the source <b>MemoryBlock</b> .   |
| Long        | Offset as <a href="#">Integer</a>  | Returns an <a href="#">Integer</a> , four bytes in length.  |
| MidB        | Offset as <a href="#">Integer</a> , [Length as <a href="#">Integer</a> ] | Returns a new <b>MemoryBlock</b> of the specified size using the same endianness as the source <b>MemoryBlock</b> .   |
| PString     | Offset as <a href="#">Integer</a>  | Returns a <a href="#">String</a> , up to 255 characters. The first byte is the length of the string.  |
| Ptr         | Offset as <a href="#">Integer</a>  | Returns a <b>MemoryBlock</b> .  |
| RightB      | Length as <a href="#">Integer</a>  | Returns a new <b>MemoryBlock</b> of the specified size using the same endianness as the source <b>MemoryBlock</b> .   |
| Short       | Offset as <a href="#">Integer</a>  | Returns a signed 16-bit <a href="#">Integer</a> .   |
| SingleValue | Offset as <a href="#">Integer</a>  | Returns a 4-byte representation of a single-precision floating number as a <a href="#">Double</a>   |
| StringValue | Offset as <a href="#">Integer</a> , Length as <a href="#">Integer</a>    | Gets and sets a <a href="#">String</a> of arbitrary size; may contain nulls, and is not prefixed with a length byte. When setting, if Length is greater than the length of the string, it is padded with zeros. |
| UInt16Value | Offset as <a href="#">Integer</a>  | Gets or sets a <a href="#">UInt16</a> value.  |
| UInt32Value | Offset as <a href="#">Integer</a>  | Gets or sets a <a href="#">UInt32</a> value.  |
| UInt64Value | Offset as <a href="#">Integer</a>  | Gets or sets a <a href="#">UInt64</a> value.  |
| UInt8Value  | Offset as <a href="#">Integer</a>  | Gets or sets a <a href="#">UInt8</a> value.   |
| UShort      | Offset as <a href="#">Integer</a>  | Returns an unsigned 16 bit <a href="#">Integer</a> .  |
| WString     |  | Allows you to get and set WStrings (UTF-16 strings) that are null terminated in a <b>MemoryBlock</b> .  |

### Notes

Memoryblocks can be used whenever you need a container for any arbitrary binary data.

Memoryblocks are used when accessing entry points in shared libraries and when making OS calls using the [Declare](#) function.

MemoryBlocks supports creating a zero-length **MemoryBlock**.

The contents of a Memoryblock can be viewed in the REALbasic Debugger.

**Examples**

The following example reads a real number into a memory block and then displays it:

```
Dim m as MemoryBlock
Dim d as Single
m=NewMemoryBlock\(4\)
m.singlevalue(0)=123.456
d=m.singlevalue(0)
MsgBox Str\(d\)
```

The following example stores a string in a MemoryBlock and displays it:

```
Dim m as New MemoryBlock(13)
m.byte(0)=12
m.byte(1)=72
m.byte(2)=101
m.byte(3)=108
m.byte(4)=108
m.byte(5)=111
m.byte(6)=32
m.byte(7)=87
m.byte(8)=111
m.byte(9)=114
m.byte(10)=108
m.byte(11)=100
m.byte(12)=33
MsgBox m.pstring(0)
```

To read the string using CString, add the terminating byte before the call, i.e.,

```
m.byte(13)=0
MsgBox m.CString(1)
```

This example backs a BinaryStream with a MemoryBlock that is declared 0-sized.

```
Dim mb as New MemoryBlock(0)
Dim bs as New BinaryStream(mb)
bs.WriteLong(4)
bs.WriteDouble(3.14)
bs.Close

MsgBox Str\(mb.Long \(0\)\)
```

**See Also**

[NewMemoryBlock](#) function.

# MenuItem Class

An individual menu item.

**Super Class** [Object](#)

## Properties

| Name                | Type                              | Description  |
|---------------------|-----------------------------------|--|
| AutoEnable          | <a href="#">Boolean</a>           | If set to <a href="#">True</a> , the <b>MenuItem</b> is enabled by default, as long as the <a href="#">App</a> object or frontmost window has a menu handler for the menuitem. There is no need to put code in the <a href="#">EnableMenuItems</a> event handler to explicitly enable the menu item. AutoEnable is <a href="#">True</a> by default. When you create dynamic menus by creating a new class based on <b>MenuItem</b> , AutoEnable is also <a href="#">True</a> by default. |
| BalloonHelp         | Message as <a href="#">String</a> | Message that appears when Balloon Help is on and the user holds the mouse over the menu item while the item is enabled. For Mac OS "classic" only.   |
| Checked             | <a href="#">Boolean</a>           | Indicates whether or not the menu item is checked.   |
| DisabledBalloonHelp | Message as <a href="#">String</a> | Message that appears when Balloon Help is on and the user holds the mouse over the menu item while the item is disabled. For Mac OS "classic" only.  |
| Enabled             | <a href="#">Boolean</a>           | Indicates whether or not the menu item is enabled. This property can be set only from an <a href="#">EnableMenuItems</a> event handler. You can enable a menu item either by assigning the Enabled property a value of <a href="#">True</a> or by calling the <a href="#">Enable</a> method.   |
| Index               | <a href="#">Integer</a>           | The number of the MenuItem when it is part of an array.  |
| KeyboardShortCut    | <a href="#">String</a>            | Keyboard shortcut for the <b>MenuItem</b> . In the Menu Editor, you assign the KeyboardShortCut property using the Key, MenuModifier, AlternateMenuModifier, MacControlKey, MacOptionKey, and PCAltKey properties rather than dealing directly with this property. See the section on the keyboard shortcut in the Notes section.  |
| Name                | <a href="#">String</a>            | The name of the menu item.   |
| Submenu             | <a href="#">Boolean</a>           | Indicates that the menu item is a submenu.   |

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Tag     | <a href="#">Variant</a> | A 'hidden' text string associated with the menu item. The tag is accessible via code when the user chooses the menu item but, unlike the Text property, is not displayed in the menu. It works like the RowTag property of a <a href="#">PopupMenu</a> control. |
| Text    | <a href="#">String</a>  | The text of the menu item.  |
| Visible | <a href="#">Boolean</a> | Indicates whether or not the menu item is visible. This property is not supported on Windows.   |

## Events

| Name       | Parameters | Description                               |
|------------|------------|---|
| Action     |            | Fires when a <b>MenuItem</b> is selected. |
| EnableMenu |            | Fires when a user pulls down a menu.      |

## Methods

| Name   | Parameters  | Description  |
|--------|---|--|
| Append | newChild as <b>MenuItem</b>                                     | Appends the passed <b>MenuItem</b> to the menu.  |
| Child  | Name as <a href="#">String</a>                                  | Looks up menu items by name and or by Text and returns a <b>MenuItem</b> .   |
| Close  |   | Removes dynamically created menu items.  |
| Count  |   | Returns as an <a href="#">Integer</a> the number of menu items. For a menu item, it returns the number of submenu items, if any. If there are no submenu items, it returns zero. |
| Enable |   | Sets the Enabled property of the menu item to <a href="#">True</a> . Call Enable only within an EnableMenuItems event handler.   |
| Insert | Index as <a href="#">Integer</a><br>newChild as <b>MenuItem</b> | Inserts newChild as a <b>MenuItem</b> at the position indicated by Index.  |
| Item   | Index as <a href="#">Integer</a>                                | Item returns as a <b>MenuItem</b> the passed item (zero-based). It causes an <a href="#">OutOfBoundsException</a> if the passed index is out of range.                           |

| Name   | Parameters  | Description   |
|--------|---|---|
| Popup  | [x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> ]     | Displays the <b>MenuItem</b> as a contextual menu. If no parameters are passed, the contextual menu appears at the location of the mouse pointer. If you pass the optional parameters, the contextual menu appears at the passed location. The coordinates are global, not just in the object that handles the MouseDown event.<br>Popup returns the selected item as a <b>MenuItem</b> . The selected item's Action event will be fired. If the selected item is handled by a MenuHandler that returns <a href="#">True</a> , then PopUp will return <a href="#">Nil</a> . |
| Remove | Index as<br><a href="#">Integer</a> OR<br>Child as<br><b>MenuItem</b> | Removes the <b>MenuItem</b> specified either by its position (index) or by reference.   |

### Notes

**MenuItem** objects are used to access the properties of menu items. You can change the menu item text using the Text property. You can find out if a menu item is checked, or check or uncheck a menu item with the Checked property. You can also enable or disable a menu item using the Enabled property. The Enabled property should be set only from within an EnableMenuItems event handler. Setting it from anywhere else has no effect.

**MenuItem** has a constructor that lets you set the Text and optionally the Tag of the new MenuItem. Pass the value of the Text property and optionally the Tag when instantiating the **MenuItem**.

Three other classes handle specialized menu items. [QuitMenuItem](#) is designed to manage the File ▶ Quit (File ▶ Exit on Windows and Linux) menu of a built application; it is enabled by default and automatically calls the [Quit](#) method. The [PrefsMenuItem](#) class is designed to handle the Preferences menu item. In Mac OS X, this menu item is supposed to be located under the application's menu, but on other operating systems, this menu does not exist. A menu item derived from the [PrefsMenuItem](#) class automatically appears under the application's menu under Mac OS X; on other operating systems it appears where you put it in the Menu Editor. The [AppleMenuItem](#) class is designed for creating menu items that appear under the Apple menu on Mac OS "classic" systems. Any menu subclassed from [AppleMenuItem](#) will move to the Apple menu for your Mac OS "classic" build but stay where it is for your other builds.

**MenuItems** can be created on the fly using the [New](#) operator. See the [New](#) operator for more information.

### Specifying the keyboard shortcut in the Menu Editor

When you use the Properties pane in the Menu Editor, you specify the menu item's shortcut key by assigning a letter to the Key property and (normally) at least one modifier key. The Menu Editor translates your settings into the value for the

KeyboardShortcut property. The Key property entry and the entries for the modifier keys are described in the table below.

| Name                  | Description  |
|-----------------------|--|
| AlternateMenuModifier | The Shift key on all platforms. If selected, the Shift key must be held down while pressing the value of the Key property to trigger the event handler of the MenuItem.  |
| Key                   | The shortcut key for the menu item. If this key and the selected modifier keys set is held down, the event handler for the menu item will be executed as if the menu item itself were chosen via the mouse pointer.                              |
| MacControlKey         | The Control key on Macintosh. This property is Macintosh-only. If selected, the Control key must be held down while pressing the value of the Key property in order to trigger the event handler of the menuitem.                                |
| MacOptionKey          | The Option key on Macintosh. This property is Macintosh-only. If selected, the Option key must be held down while pressing the value of the Key property to trigger the event handler of the menu item.  |
| MenuModifier          | The Control key on Windows and Linux and the Command key on Macintosh. If this property is selected, the MenuModifier key must be held down while pressing the key specified by the Key property to trigger the event handler for the menu item. |
| PCAltKey              | If selected, the Alt key is must be held down while pressing the value of the Key property in order to trigger the event handler for the menu item. This property is for Windows and Linux only.   |

## Examples

The following example changes the text of the EditPaste menu item to “Paste Special...”:

```
EditPaste.text="Paste Special..."
```

The following example inserts a new item in the Edit menu with the text “Paste Special...” just below the Paste item.

```
Dim EditPasteSpecial as New MenuItem
EditPasteSpecial.text="Paste Special..."
Editmenu.Insert(5>EditPasteSpecial)
```

Using the constructor, you can write:

```
Dim EditPasteSpecial as New MenuItem("Paste Special...")
Editmenu.Insert(5>EditPasteSpecial)
```

## MenuItem Class

---

The following example adds a Select All menu item to the Edit menu.

```
Dim EditSelectAll as New MenuItem
EditSelectAll.text="Select All..."
Editmenu.Append(EditSelectAll)
```

The following example gets the **MenuItem** corresponding to the Cut item on the Edit menu.

```
Dim c as MenuItem
c>EditMenu.child("EditCut")
Msgbox c.text
```

The following example gets the **MenuItem** corresponding to the Cut item on the Edit menu by position:

```
Dim c as MenuItem
c>EditMenu.item(2)
MsgBox c.text
```

The following example removes the fourth dynamically created menu item from a menu item array named WindowItem:

```
WindowItem(3).Close
```

The following example assigns a value to the Tag property of a menu item:

```
SearchFind.Tag="UserSearch"
```

The following example displays the Edit menu as a contextual menu. The code is in the MouseDown event handler of a [RectControl](#). You can get the text of the selected item by accessing the Text property of the returned **MenuItem**.

```
If IsContextualClick Then
Dim m as MenuItem
m>EditMenu.Popup
End if
```

Note that you can also create and display contextual menus using the ConstructContextMenu and ContextualMenuItemAction event handlers of the [Window](#) and [RectControl](#) classes.

## See Also

[AppleMenuItem](#), [PrefsMenuItem](#), [QuitMenuItem](#) classes; [IsContextualClick](#), [EnableMenuItems](#) functions, [New](#) operator.

# MessageDialog Class

Used to design and display customized message dialog boxes. Similar to the [MsgBox](#) function but the **MessageDialog** class is much more flexible.

**Super Class** [Object](#)

## Properties

| Name                  | Type                                | Description   |
|-----------------------|-------------------------------------|---|
| ActionButton          | <a href="#">MessageDialogButton</a> | The button that performs the default action. Its <code>Visible</code> property is <code>True</code> by default. By default, its caption is "OK" or an equivalent in the language used by the application. To be safe, you should set the <code>Caption</code> property explicitly.  |
| AlternateActionButton | <a href="#">MessageDialogButton</a> | The optional button that performs an alternate action. Its <code>Visible</code> property is <code>False</code> by default. It has no default text; if you display it, you should specify its <code>Caption</code> .   |
| CancelButton          | <a href="#">MessageDialogButton</a> | The optional "cancel" button. Its <code>Visible</code> property is <code>False</code> by default. Its default <code>Caption</code> is "Cancel" or the equivalent in the language used by the application. To be safe, you should set the <code>Caption</code> property explicitly. If clicked, its <code>Cancel</code> property is set to <code>True</code> .   |
| Explanation           | <a href="#">String</a>              | The text of a secondary message. On Macintosh only it appears in a smaller font size below <code>Message</code> . On other platforms it appears in the same font size as <code>Message</code> . Use this text to provide a fuller description of the situation, its consequences, and how to get out of it. For example, a warning that an action cannot be undone is an appropriate use of explanation text. |

## MessageDialog Class

---

| Name    | Type                    | Description  |
|---------|-------------------------|--|
| Icon    | <a href="#">Integer</a> | <p>Number indicating the type of icon to be displayed in the dialog box. These icons are supplied by the host operating system. You can use the following class constants to specify the icon:</p> <ul style="list-style-type: none"><li>-1 - GraphicNone</li><li>0 - GraphicNote (The application icon on Mac OS X)</li><li>1 - GraphicCaution triangle (On Mac OS X, the application icon superimposed on the caution triangle)</li><li>2 - GraphicStop (On Mac OS X 10.3 and above, the application icon)</li><li>3 - GraphicQuestion icon (On Mac OS X 10.3 and above, it is the same as 0)</li></ul> <p>See the Notes section in the pdf version of the Language Reference for screen shots of these icons on each platform. If the value of <i>Icon</i> is out of range, an <a href="#">OutOfBoundsException</a> occurs.</p> |
| Message | <a href="#">String</a>  | The text of the message to be displayed. Use this property to present a short summary of the error, condition, or choice. Often, the message is posed as a question. On Macintosh, the text is emphasized in a bold font. Use the Explanation property for a more detailed message.  |
| Title   | <a href="#">String</a>  | Text to be displayed in the Title bar (Windows and Linux only).  |

## Methods

| Name            | Parameters | Description   |
|-----------------|------------|---|
| ShowModal       |            | Displays the <b>MessageDialog</b> window. Returns a <a href="#">MessageDialogButton</a> , which indicates which button was pressed. To find out which button was pressed, compare it to the button instances in the dialog.                                   |
| ShowModalWithin |            | Displays the <b>MessageDialog</b> window as a sheet window (Mac OS X only). Returns a <a href="#">MessageDialogButton</a> , which indicates which button was pressed. To find out which button was pressed, compare it to the button instances in the dialog. |

## Notes

A **MessageDialog** dialog can have up to three buttons, an icon, and main and subordinate text. On Windows and Linux, it can also have text in its title bar. By

default, only the ActionButton's Visible property is [True](#). To use any other buttons, you must set their Visible properties to [True](#).

**Icons**

The four icons supported by MessageDialog are not the same on all platforms. In particular, Mac OS X shows the application icon for the values of 0, 2, and 3.

The following table shows the icons on the three platforms.

| Platform | Value    |             |          |              |
|----------|----------|-------------|----------|--------------|
|          | 0 (Note) | 1 (Caution) | 2 (Stop) | 3 (Question) |
| Windows  |          |             |          |              |
| Mac OS X |          |             |          |              |
| Linux    |          |             |          |              |

**Handling the button click**

After the user has clicked a button in the **MessageDialog**, the ShowModal method returns the [MessageDialogButton](#) that was pressed. You need to check this against the three types of [MessageDialogButtons](#) belonging to the **MessageDialog** to determine which button the user clicked. See the example.

## MessageDialogButton Class

---

### Example

The following example creates and manages a “Save Changes” dialog box without the need to create an instance of the [Window](#) class.

```
Dim d as New MessageDialog //declare the MessageDialog object
Dim b as MessageDialogButton //for handling the result
d.icon=MessageDialog.GraphicCaution //display warning icon
d.ActionButton.Caption="Save"
d.CancelButton.Visible=True //show the Cancel button
d.AlternateActionButton.Visible=True //show the "Don't Save" button
d.AlternateActionButton.Caption="Don't Save"
d.Message="Do you want to save changes to this document"_
    +" before closing?"
d.Explanation="If you don't save, your changes will be lost. "
b=d.ShowModal //display the dialog
Select Case b //determine which button was pressed.
Case d.ActionButton
    //user pressed Save
Case d.AlternateActionButton
    //user pressed Don't Save
Case d.CancelButton
    //user pressed Cancel
End select
```

### See Also

[MsgBox](#) function, [MessageDialogButton](#), [Window](#) classes.

---

## MessageDialogButton Class

A button in a [MessageDialog](#) box. There are three possible MessageDialogButtons: ActionButton, CancelButton, and AlternateActionButton.

Super Class [Object](#)

### Properties

| Name    | Type                    | Description  |
|---------|-------------------------|--|
| Cancel  | <a href="#">Boolean</a> | Set to <a href="#">True</a> to indicate that the button will respond to the Esc key and, on Macintosh, pressing the Command-period sequence. Only one button can be the Cancel button. Some operating systems do not allow one button to be both the Cancel and Default buttons. |
| Caption | <a href="#">String</a>  | The text displayed in the button. Place a single “&” in front of any character to make it the shortcut key for the button. For example “&Don’t Save” makes “D” the shortcut key.   |

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Default | <a href="#">Boolean</a> | Set to <a href="#">True</a> to highlight the button as the default button in the <a href="#">MessageDialog</a> . The Default button will respond to the Enter and Return keys. By default, the ActionButton's Default property is <a href="#">True</a> . Only one button can be the Default button. Some operating systems do not allow one button to be both the Default and Cancel buttons. |
| Visible | <a href="#">Boolean</a> | Set to <a href="#">True</a> to show the button. Only the ActionButton is shown by default. It has no effect on the ActionButton.  |

**Notes**

By default, only the ActionButton has its Visible property set to [True](#). To show either of the other buttons in a [MessageDialog](#), set their Visible properties.

**Example**

See the example for [MessageDialog](#).

**See Also**

[MsgBox](#) function; [MessageDialog](#), [Window](#) classes.

## Microseconds Function

Returns the number of microseconds (1,000,000<sup>th</sup> of a second) that have passed since the user's computer was started.

**Syntax**

**result=Microseconds**

| Part   | Type                   | Description  |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The number of microseconds that have passed since the user's computer was started. On Windows only, the result is accurate only to milliseconds. |

**Note**

Because modern operating systems can stay running for so long, it's possible for the machine's internal counters to "roll over." This means that if you are using this function to determine how much time has elapsed, you may encounter a case where this time is inaccurate.

**Examples**

This example displays in message box the number of minutes the computer has been on.

```
Dim minutes As Integer
minutes=Microseconds/1000000/60
Msgbox "Your computer has been on for "+Str(minutes)+" minutes."
```

# Mid Function

Returns a portion of a [string](#). The first character is numbered 1.

**Syntax** **result=Mid(source, start [,length])**

**OR**

**result=stringVariable.Mid(start[,length])**

| Part           | Type                    | Description   |
|----------------|-------------------------|---|
| result         | <a href="#">String</a>  | The portion of source from start and continuing for length characters or all remaining characters if length is not specified.   |
| source         | <a href="#">String</a>  | Required. The string from which characters are returned.  |
| start          | <a href="#">Integer</a> | Required. The position of the first character to be returned. The first character position is 1. If start is greater than the number of characters in source, an empty string is returned.  |
| length         | <a href="#">Integer</a> | Optional. The number of characters to return from source. If omitted, all characters from start to the end of source are returned. If length + start is greater than the length of source, all characters from start to the end of source will be returned. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .   |

## Notes

To determine the number of characters in a [string](#), use the [Len](#) function.

The **Mid** function works properly with international text.

## Examples

These examples use the **Mid** function to return portions of a [string](#).

```
Dim s As String
s = Mid("This is a test", 6) //returns "is a test"
s = Mid("This is a test", 11, 4) //returns "test"
```

This example converts the text string in EditField1 to hex and writes the result to EditField2:

```
Dim i as Integer
EditField2.text = ""
For i=1 to Len(EditField1.text)
    EditField2.text = EditField2.text + "&h" + Hex(Asc(Mid(EditField1.text,i,1)))
Next
```

## See Also

[Asc](#), [Chr](#), [InStr](#), [Left](#), [Len](#), [Right](#) functions.

# MidB Function

Returns a portion of a [string](#). The first character is numbered 1.

**Syntax** `result=MidB(source, start [,length])`

OR

`result=stringVariable.MidB(start [,length])`

| Part           | Type                    | Description  |
|----------------|-------------------------|--|
| result         | <a href="#">String</a>  | The portion of source from start and continuing for length characters or all remaining characters if length is not specified.  |
| source         | <a href="#">String</a>  | Required. The string from which bytes are returned.  |
| start          | <a href="#">Integer</a> | Required. The position of the first byte to be returned. If start is greater than the number of bytes in source, an empty string is returned.  |
| length         | <a href="#">Integer</a> | Optional. The number of bytes to return from source. If omitted, all bytes from start to the end of source are returned. If length + start is greater than the length of source, all bytes from start to the end of source will be returned. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .  |

## Notes

**MidB** treats source as a series of bytes, rather than a series of characters. **MidB** should be used when source represents binary data. If you need to extract characters rather than bytes, use the [Mid](#) function. To determine the number of bytes in a [string](#), use the [LenB](#) function.

## Examples

These examples use the **MidB** function to return portions of a [string](#).

```
Dim s As String
```

```
s=MidB("This is a test", 6) //returns "is a test"
```

```
s=MidB("This is a test", 11, 4) //returns "test"
```

## See Also

[AscB](#), [ChrB](#), [InStrB](#), [LeftB](#), [LenB](#), [Mid](#), [RightB](#) functions.

# Min Function

Returns the smallest of the numbers passed.

## Mod Operator

---

### Syntax

**result=Min(value1, value2,...ValueN)**

| Part   | Type                   | Description  |
|--------|------------------------|--|
| result | <a href="#">Double</a> | The smallest of passed values.   |
| value1 | <a href="#">Double</a> | The first number for comparison.   |
| value2 | <a href="#">Double</a> | The second number for comparison.  |
| valueN | <a href="#">Double</a> | Optional: Min requires two parameters but optionally accepts additional parameters. N is the total number of parameters. |

### Notes

The **Min** function compares the numbers passed and returns the smallest value.

### Examples

This example uses the **Min** function to return the lesser of the two values passed.

```
Dim d As Double  
d=Min(3.01, 4.05) //returns 3.01  
d=Min(3.012, 3.011) //returns 3.011
```

### See Also

[Abs](#), [Ceil](#), [Exp](#), [Floor](#), [Max](#), [Pow](#), [Sqr](#) functions.

## Mod Operator

---

Returns the remainder of the division of two numbers.

### Syntax

**result= number1 Mod number2**

| Part    | Description   |
|---------|---|
| result  | The remainder (as an <a href="#">Integer</a> ) of number1 divided by number2. |
| number1 | Any number.   |
| number2 | Any number.   |

The **Mod** operator divides number1 by number2 and returns the remainder as result. If number2 is zero, **Mod** returns number1.

The **Mod** operator operates on integers. If either number1 or number2 are non-integers, they are first truncated to integers. For example:

```
Dim r as Integer  
r=5 Mod 2 //returns 1  
r=5 Mod 1.99999 //returns 0
```

**Examples** These examples use the **Mod** operator to divide two numbers and return the remainder.

```
Dim r As Integer
r=10 Mod 3 //Returns 1
r=2 Mod 4 //Returns 2
r=9.3 Mod 2.75 //Returns 1
r=4.5 Mod 1 //Returns 0
r=25 Mod 5 //Returns 0
```

**See Also** [\ operators](#); [Operator\\_Modulo](#) function.

## MouseCursor Class

Used to specify the appearance of the pointer.

**Super Class** [Object](#)

### Notes

The [Cursors](#) object contains a library of standard mouse cursors that you can access by calling [System.Cursors.MouseCursorName](#). See the [Cursors](#) object for the names and descriptions of the available mouse cursors.

See the **MouseCursor** properties of the [Application](#) class, the [Window](#) class, and the [Control](#) class. The relationships among them are as follows:

The [MouseCursor](#) property of a control determines the shape of the mouse pointer when the pointer enters the control's region **only** if the [Application](#) and [Window](#) classes' MouseCursor properties are [Nil](#). Similarly, the Window class's MouseCursor property determines the shape of the pointer when it enters the region of the window **only** if the [Application](#) class's MouseCursor property is [Nil](#). If the [Application](#) class's MouseCursor property is not [Nil](#), then it controls the shape of the pointer and a window's and any control's MouseCursor property values are ignored.

A MouseCursor can be assigned in the following ways:

- You can use a cursor in the [Cursors](#) object.
- On Macintosh, you can add a resource file to the Project Editor that contains a CURS resource and use the [ResourceFork.GetCursor](#) method to obtain a cursor resource.
- On Macintosh, you can open a file's resource fork and open any CURS or crsr resource using the [ResourceFork.GetCursor](#) method.

### Custom Cursors

On Macintosh, you can add custom cursors using the following technique:

Create a separate CURS resource that contains only one cursor for each custom cursor you wish to use. Give each resource file a different name and add them to the Project Editor.

## Movie Class

---

When you do this, each resource file has a pointer icon:

You can access each cursor using the resource's name, e.g.,

```
Self.mousecursor=curC
```

On Windows, you can create cursors with a Windows resource editor and then import the .cur files that it creates into REALbasic. Currently, only 16 x 16 monochrome cursors are supported.

### Example

The following line changes the pointer to the HandOpen cursor when the mouse enters the region of a Canvas control. The line is in the MouseEnter event handler.

```
Me.MouseCursor=System.Cursors.HandOpen
```

Assign a custom cursor to the MouseCursor property of the [Application](#) or a [window](#) in the same way.

### See Also

[Cursors](#) object; MouseCursor properties of the [Application](#) class, the [Window](#) class, and the [Control](#) class; [System](#) class.

---

## Movie Class

A reference to a movie. Movie class objects have no events.

**Super Class** [Object](#)

### Properties

| Name                   | Type                    | Description  |
|------------------------|-------------------------|--|
| <b>BaseMovieHeight</b> | <a href="#">Integer</a> | Actual height of the movie (pixels).   |
| <b>BaseMovieWidth</b>  | <a href="#">Integer</a> | Actual width of the movie (pixels).  |
| Handle                 | <a href="#">Integer</a> | Returns handle to the movie, mainly used for API calls that require the handle to a movie.   |
| MovieHandle            | <a href="#">Integer</a> | Handle to the movie. Provides the same functionality as Handle, but may not be supported in future releases of REALbasic. Obsolescent, use Handle instead. |
| <b>MovieHeight</b>     | <a href="#">Integer</a> | Height of the QuickTime movie in pixels  |
| <b>MovieWidth</b>      | <a href="#">Integer</a> | Width of the QuickTime movie in pixels.  |

### Notes

Movie objects can be accessed via the Movie property of [MoviePlayer](#) control or when calling the OpenAsMovie method of a [FolderItem](#) object.

**Examples**

This example opens a movie on disk, assigns it to the movie property of a MoviePlayer object, and displays its height and width in EditFields.

```
Dim f as FolderItem
Dim m as Movie
f=GetFolderItem("Logo.mov")
m=f.OpenasMovie
moviePlayer1.border=True
moviePlayer1.movie=m
moviePlayer1.controller=2
EditField2.text=Str(m.movieheight)
EditField3.text=Str(m.moviewidth)
```

**See Also**

[OpenURLMovie](#) function; [EditableMovie](#), [QTTrack](#) classes; [MoviePlayer](#) control.

## MoviePlayer Control

Displays a control for playing Quicktime and Windows Media Player movies.

**Super Class**

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

**Properties**

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| AutoPlay         | <a href="#">Boolean</a> | If <a href="#">True</a> , the movie will start playing automatically when it is assigned to the Movie property of the control. The default is <a href="#">False</a> . |
| AutoResize       | <a href="#">Boolean</a> | If <a href="#">True</a> , the MoviePlayer control resizes itself to fit the size of the movie.  |
| Border           | <a href="#">Boolean</a> | Draws a border around the movie area.   |
| Controller       | <a href="#">Integer</a> | 0=no movie controls,<br>1=Badge,<br>2=Full controller   |
| ControllerHeight | <a href="#">Integer</a> | Height of the controller (pixels).  |
| ControllerWidth  | <a href="#">Integer</a> | Width of the controller (pixels).   |
| EditingEnabled   | <a href="#">Boolean</a> | If <a href="#">True</a> , it is possible to edit the movie. A MoviePlayer's methods for editing a movie work when EditingEnabled is <a href="#">True</a> .            |

| Name              | Type                    | Description  |
|-------------------|-------------------------|--|
| HasStep           | <a href="#">Boolean</a> | If <a href="#">True</a> , the controller has forward and reverse arrows to the right of the slider. When set in code, HasStep takes effect the next time the MoviePlayer is assigned a movie.  |
| Looping           | <a href="#">Boolean</a> | Re-plays the movie automatically when it reaches the end.  |
| Movie             | <a href="#">Movie</a>   | The movie that will be played.   |
| Palindrome        | <a href="#">Boolean</a> | Plays the movie in reverse when the movie reaches its end. Looping must be <a href="#">True</a> for Palindrome to work.  |
| PlayerType        | <a href="#">Integer</a> | Indicates the type of movie the MoviePlayer will play.<br>0 - Preferred player<br>1 - QuickTime player<br>2 - Windows Media player<br><br>On Windows, the preferred player is the Windows Media Player, but if the WMP cannot play the movie, REALbasic will attempt to use the QuickTime player. On Macintosh, <i>PlayerType</i> is ignored, as the only type of movie currently supported is QuickTime. If desired, set <i>PlayerType</i> to 2 to prevent REALbasic from trying the QuickTime player. On Linux, these players are not currently available. |
| PlaySelection     | <a href="#">Boolean</a> | If a selection is made (via SelStart and SelLength) and this property is set to <a href="#">True</a> , only the selection will be played. If <a href="#">False</a> , normal playback will occur.   |
| Position          | <a href="#">Double</a>  | Number of seconds from the start of the movie.   |
| QTMovieController | <a href="#">Integer</a> | Returns a pointer that refers to the MoviePlayer control. Of use to API programmers.   |
| QTVRNode          | <a href="#">Integer</a> | The node that is currently being shown in the controller.  |
| QTVRNodeCount     | <a href="#">Integer</a> | The number of nodes in the MoviePlayer's movie.  |
| QTVRPan           | <a href="#">Double</a>  | The pan angle in degrees of MoviePlayer's QTVR movie.  |
| QTVRPanMax        | <a href="#">Double</a>  | The maximum pan angle (in degrees) of the MoviePlayer's QTVR movie.  |
| QTVRPanMin        | <a href="#">Double</a>  | The minimum pan angle (in degrees) of the MoviePlayer's QTVR movie.  |
| QTVRPanTiltSpeed  | <a href="#">Integer</a> | The speed of tilt motion (angle in degrees) of the MoviePlayer's QTVR movie.   |
| QTVRTilt          | <a href="#">Double</a>  | The tilt angle (in degrees) of the MoviePlayer's QTVR movie.   |
| QTVRTiltMax       | <a href="#">Double</a>  | The maximum tilt angle (in degrees) of MoviePlayer's QTVR movie.   |
| QTVRTiltMin       | <a href="#">Double</a>  | The minimum tilt angle (in degrees) of MoviePlayer's QTVR movie.   |

| Name          | Type                    | Description  |
|---------------|-------------------------|--|
| QTVRZoom      | <a href="#">Double</a>  | The desired vertical field of view for the specified movie. This value is constrained by the maximum field of view of the movie. Values that lie outside that limit are clipped to the maximum. Pan and tilt angle values are also clipped if, when combined with the current field of view, they would cause an image to lie outside the current constraints. |
| QTVRZoomMax   | <a href="#">Double</a>  | The maximum zoom field of view.  |
| QTVRZoomMin   | <a href="#">Double</a>  | The minimum zoom field of view.  |
| QTVRZoomSpeed | <a href="#">Integer</a> | The rate at which you can zoom.  |
| Rate          | <a href="#">Double</a>  | Sets the playback rate. For example, the following values have these effects:<br>0 – Paused / stopped<br>0.5 – Half normal playback rate<br>1 – Normal playback rate<br>2 – Double normal playback rate<br>-1 – Reverse normal playback rate<br>Since Rate is a <a href="#">Double</a> , other values, including values outside this range, work.              |
| SelLength     | <a href="#">Double</a>  | Gets or sets the length (in seconds) of the selection.   |
| SelStart      | <a href="#">Double</a>  | Gets or sets the starting position (in seconds) of the selection.  |
| Speaker       | <a href="#">Boolean</a> | Displays the speaker volume control. When set in code, Speaker takes effect the next time the MoviePlayer control is assigned a movie.   |
| Volume        | <a href="#">Integer</a> | Gets or sets the movie's volume. The range is from 0 to 256.   |

## Events

| Name                  | Description  |
|-----------------------|--|
| ControllerSizeChanged | The size of the controller has changed. When a movie first starts this event fires and allows you read the ControllerWidth/Height properties and then resize the MoviePlayer and or window to best accommodate that size. The event also fires if the movie changes size 'on its own'. |
| Play                  | The current movie is about to begin playing.   |
| Stop                  | The current movie is about to stop playing.  |

## Methods

| Name                   | Parameters   | Description   |
|------------------------|--|---|
| Clear                  |  | Clears the movie's current selection. Set the EditingEnabled property to <a href="#">True</a> to enable the user to make a selection using the MoviePlayer's controller. If the movie is not an <a href="#">EditableMovie</a> , no changes can be saved.                                  |
| Copy                   |  | Copies the movie's current selection to the Clipboard. Set the EditingEnabled property to <a href="#">True</a> to enable the user to make a selection using the MoviePlayer's controller. If the movie is not an <a href="#">EditableMovie</a> , no changes can be saved.                 |
| Cut                    |  | Cuts the movie's current selection and places it on the Clipboard. Set the EditingEnabled property to <a href="#">True</a> to enable the user to make a selection using the MoviePlayer's controller. If the movie is not an <a href="#">EditableMovie</a> , no changes can be saved.     |
| Paste                  |  | Pastes the contents of the Clipboard to the Movie's current selection. Set the EditingEnabled property to <a href="#">True</a> to enable the user to make a selection using the MoviePlayer's controller. If the movie is not an <a href="#">EditableMovie</a> , no changes can be saved. |
| Play                   |  | Plays the current movie.  |
| QTVRHotSpotCount       | nodeID As <a href="#">Integer</a>                                  | Returns an <a href="#">Integer</a> , the number of QTVR hot spots in the movie with the node passed in nodeID.  |
| QTVRHotSpotID          | node As <a href="#">Integer</a> , index as <a href="#">Integer</a> | Returns an <a href="#">Integer</a> , the ID of a specific hot spot based on node number and hot spot index passed as parameters.  |
| QTVRNodeTypeObject     | node As <a href="#">Integer</a>                                    | Determines whether the current movie has a QTVR object in it. Returns a <a href="#">Boolean</a> ; <a href="#">True</a> if the movie has a QTVR object.  |
| QTVRNodeTypePanorama   | node as <a href="#">Integer</a>                                    | Determines whether the current movie has a QTVR Panorama in it. Returns a <a href="#">Boolean</a> ; <a href="#">True</a> if the movie has a QTVR Panorama.  |
| QTVRToggleHotSpotNames |  | Toggles whether HotSpot names are shown in the movie controller.  |

| Name               | Parameters           | Description   |
|--------------------|----------------------|---|
| QTVRTriggerHotSpot | hotSpotID as Integer | Navigates to the specified hot spot in the MoviePlayer's QTVR movie.  |
| Stop               |                      | Stops the current movie.  |
| Trim               |                      | Retains the current selection (specified by SelStart and SelLength) and removes all other media. Set the EditingEnabled property to <a href="#">True</a> to enable the user to make a selection using the MoviePlayer's controller. If the movie is not an <a href="#">EditableMovie</a> , no changes can be saved. |
| Undo               |                      | Undo the last operation. This is actually an Undo/Redo toggle. Set the EditingEnabled property to <a href="#">True</a> to enable the user to make a selection using the MoviePlayer's controller. If the movie is not an <a href="#">EditableMovie</a> , no changes can be saved.                                   |

**Notes**

Windows uses the Windows Movie Player to play movies and Macintosh uses the QuickTime player. Neither player is currently available on Linux, so the MoviePlayer is not supported.

On Windows, you can add WMV files to your project and play them via the MoviePlayer. QuickTime movies work on both Macintosh and Windows.

The Controller property dictates how the movie controls (if any) will be displayed. Passing 0 (zero) means that there will be no user controls available. Passing 1 means that a movie icon or badge will be displayed in the lower left corner of the movie area instead of the controller. When this badge is clicked by the user, the badge disappears and the regular movie controls appear at the bottom of the movie frame. Passing 2 displays the regular movie controls.

**Examples**

This example sets the movie "myHomeMovie" as the movie to be played.

```
MoviePlayer1.movie=myHomeMovie
```

This example loads a movie called "MyMovie" from the current directory (folder) into MoviePlayer1 and plays it.

```
Dim f As FolderItem
f=GetFolderItem("MyMovie")
MoviePlayer1.Speaker=True
MoviePlayer1.HasStep=False
MoviePlayer1.movie=f.OpenAsMovie
MoviePlayer1.play
```

## MsgBox Function

---

If you added the movie to your project, you can assign it to the movie property with only one line of code:

```
MoviePlayer1.movie=MyMovie
```

Using object binding, you can add pushbuttons that play or stop the movie. See the section on the **MoviePlayer** control in the *User's Guide*.

**See Also** [OpenURLMovie](#) function; [Movie](#), [EditableMovie](#) classes.

---

## MsgBox Function

Displays message box showing the [string](#) passed. With the exception of simple message boxes, you should use the [MessageDialog](#) class instead.

### Syntax

**result=MsgBox (message[,buttons][,title])**

| Part           | Type                    | Description   |
|----------------|-------------------------|---|
| <i>result</i>  | <a href="#">Integer</a> | Indicates which button was pressed. It returns 1 if the OK button is clicked. If you don't use the optional parameters, <b>MsgBox</b> can be called without handling <i>result</i> .<br>1 – OK Pressed<br>2 – Cancel Pressed<br>3 – Abort pressed<br>4 – Retry pressed<br>5 – Ignore pressed<br>6 – Yes pressed<br>7 – No pressed |
| <i>message</i> | <a href="#">String</a>  | Any valid <a href="#">string</a> expression. This parameter contains the main message of the dialog. If a paragraph break exists in the message, the second and following paragraphs will be de-emphasized on the Macintosh. They appear in the same font size on other platforms.  |
| <i>buttons</i> | <a href="#">Integer</a> | Optional code indicating the icon and choice of buttons displayed in the Message box. You must handle the result if you pass a value for <i>buttons</i> .   |
| <i>title</i>   | <a href="#">String</a>  | Optional text displayed in the Title bar (Windows and Linux). <i>Title</i> is not displayed on Macintosh. If you want to specify a value for <i>Title</i> , you must handle the result of the MsgBox call and also pass a value for <i>button</i> .   |

### Notes

A message can be presented to the user with either the **MsgBox** function or the [MessageDialog](#) class. The **MsgBox** function is recommended for simple informational messages only. For other situations, the [MessageDialog](#) class is more appropriate. It enables you to add up to three buttons, label them in any way, and take any action after

the user clicked a button. **MsgBox** is also ideally suited for porting VisualBasic programs to REALbasic.

The Message box opened by **MsgBox** has a Title bar. On Windows, the dialog also has a Close widget in its Title bar. The dialog can be closed either by clicking OK or clicking the Close widget. On Windows and Linux, the width increases to accommodate the longest paragraph. On Macintosh, the Message box has a fixed width and the text word-wraps to fit the width of the **MsgBox**.

Multiple paragraphs can be passed in the **message** parameter by separating each paragraph with the [EndOfLine](#) function.

## The Buttons Parameter

*Buttons* is an optional parameter that enables you to customize the buttons and icon displayed in the message box to a limited extent. You get a fixed choice of button text and button positions. The value of *Buttons* is the sum of values that describe the number and labelling of the buttons, the icon, and the default button. There are three groups of button values; each group offers choices for a particular feature.

### Group 1: Number and Type of Buttons.

| Value | Description                               |
|-------|---|
| 0     | Display OK button only.                   |
| 1     | Display OK and Cancel buttons.            |
| 2     | Display Abort, Retry, and Ignore buttons. |
| 3     | Display Yes, No, and Cancel buttons.      |
| 4     | Display Yes and No buttons only.          |
| 5     | Display Retry and Cancel buttons.         |

The second group of values specifies the icon to be displayed.

### Group 2: Icon to be Displayed.

| Value | Description            |
|-------|------------------------|
| 0     | No icon.               |
| 16    | Stop sign icon.        |
| 32    | Question icon.         |
| 48    | Caution triangle icon. |
| 64    | Note icon.             |

The third group of values pertains to the selection of the default button. The terms first, second, and third in the following table do not necessarily refer to the physical positions of the buttons, which may vary among platforms. It refers to the ordering in the Group 1 table.

### Group 3: Selection of Default Button.

| Value | Description                                  |
|-------|--|
| 0     | First button of Group 1 list is the default. |

### Group 3: Selection of Default Button.

| Value | Description                                   |
|-------|---|
| 256   | Second button of Group 1 list is the default. |
| 512   | Third button of Group 1 list is the default.  |
| 768   | No button is the default.                     |

Since the value of the *button* parameter is the sum, you make a selection from each of the three tables, add up their values and pass that value. For example, if you want the buttons to be the “Abort, Retry, and Ignore” set, with a Caution icon, and Retry as the default, you would add up  $2 + 48 + 256 = 306$ . This also illustrates why you should be using the [MessageDialog](#) class for a message dialog such as this. In contrast, the MessageDialog class allows you to set button text to any string and set the default property via a [Boolean](#) property.

### Examples

This example displays a one line Message box.

```
MsgBox "Hello World."
```

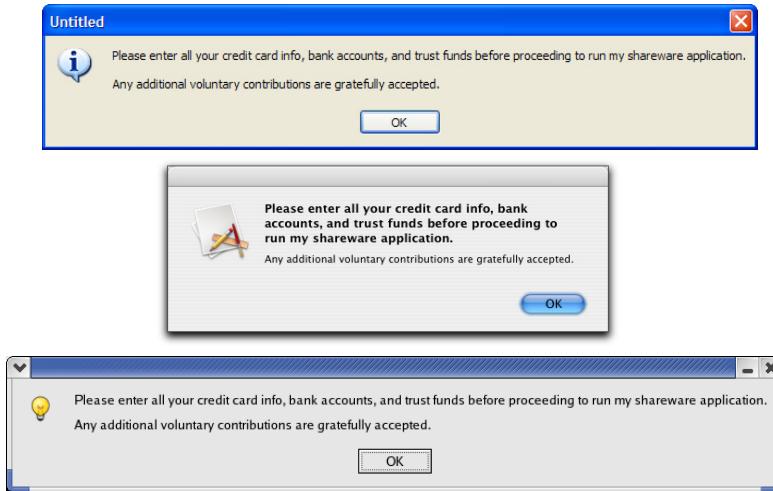


The following example uses the [EndOfLine](#) function to break text to be displayed in a **MsgBox** into two paragraphs. When you use [EndOfLine](#) in this way, the second line will be de-emphasized relative to the first (Macintosh only).

```
MsgBox "Hello world"+EndOfLine+EndOfLine+"Such as it is"
```

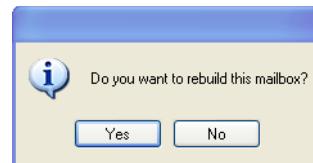
The following example displays a multiline Message box. The text after the two [EndOfLine](#) functions appears in plain type and a smaller font size on Macintosh. On other platforms, the second paragraph appears in the same font size and style as the first. Please note that it is entered as one line in the Code Editor.

```
MsgBox "Please enter all your credit card info, bank accounts, and trust funds before proceeding to run my shareware application." +EndOfLine+EndOfLine+"Any additional voluntary contributions are gratefully accepted."
```



This example displays a Message box with “Yes” and “No” buttons.

```
Dim n as Integer
n=MsgBox("Do you want to rebuild this mailbox",36)
If n=6 then
    //user pressed Yes
elseif n=7 then
    //user pressed No
end if
```



**See Also** [EndOfLine](#), [MessageDialog](#), [MessageDialogButton](#) classes.

## Mutex Class

A type of [CriticalSection](#) that has operating system-wide scope and is visible to other REALbasic applications.

**Super Class** [CriticalSection](#)

## MySQLDatabase Class

---

### Constructor

| Name  | Parameters                     | Description  |
|-------|--------------------------------|--|
| Mutex | Name as <a href="#">String</a> | The parameter is the name of the <b>Mutex</b> . If you pass an empty string, the <b>Mutex</b> will be identical to a <a href="#">CriticalSection</a> ; it will not be visible to other applications. If you pass a name, you can use the name to access the Mutex. |

### Notes

**Mutex** is short for Mutual Exclusion object. It allows several applications to share the same resource, but not simultaneously. The suggested usage of a Mutex is to create a **Mutex** with a unique name when the application launches. When the application needs to use the resource, it calls the Enter or TryEnter methods and calls the Leave method when it is finished. The most common use of a **Mutex** is to determine whether another instance of your application is currently running. You create a named **Mutex** in the application's open event and then check to see if you can get a lock on it. If the lock fails, then you know there's another instance of your application running. You can also use Mutexes to work with resources that are shared between applications (such as a serial port, printer, or some other system device).

You can call Enter or TryEnter multiple times in the same way as with a [CriticalSection](#), including recursive calls.

### See Also

[Application](#), [CriticalSection](#), [Semaphore](#), [Thread](#) classes.

---

## MySQLDatabase Class

Used to open MySQL databases.

### Properties

| Name    | Type                    | Description  |
|---------|-------------------------|--|
| Port    | <a href="#">Integer</a> | The port that the MySQL database will use to connect. The default value is 3306. |
| Timeout | <a href="#">Integer</a> | The connection timeout value (seconds). The default is 15.                       |

### Methods

| Name            | Parameters | Description  |
|-----------------|------------|--|
| GetAffectedRows |            | Returns as an <a href="#">Integer</a> the number of rows that were modified by the most recent SQLExecute statement. |
| GetInsertID     |            | Returns as an <a href="#">Integer</a> the value of the AUTOINCREMENT field for the last row that was inserted.       |

| Name       | Parameters                                | Description   |
|------------|---|---|
| LogPackets | Location as<br><a href="#">FolderItem</a> | A debugging function that logs all network packets between the driver and server to the file specified by <i>Location</i> . |

**Notes**

In order to use this class, you must install the MySQL database plug-in in your plugins folder.

If the plug-in is installed, you can also open a MySQL database in the REALbasic IDE using the Project ▶ Add ▶ Database ▶ SelectMySQLDatabase command. This command presents a dialog box that requests the server, port, database, Username, and Password properties.

The set of database plug-ins is included on the REALbasic CD, but you may find more recent versions at the REAL Software web site, <http://www.realsoftware.com>.

**Example**

This example establishes a connection to a remote database.

```
Dim db as mySQLDatabase
db=New mySQLDatabase
db.host="192.168.1.172"
db.port=3306
db.databaseName="MyOwnmySQLdatabase"
db.userName="Mary"
db.Password="Elton"
If db.Connect then
    //proceed with database operations
else
    MsgBox "Connection failed!"
end if
```

**See Also**

[Database](#), [RecordSet](#) classes.

## Network Object

Enables you to access certain properties of the networking environment. You access the **Network** object from the [System](#) object.

**Methods**

| Name             | Parameters                              | Description  |
|------------------|---|--|
| LookupDNSAddress | IPAddress as<br><a href="#">String</a>  | Looks up the passed IP address and returns its DNS address as a <a href="#">String</a> . |
| LookupIPAddress  | DNSAddress as<br><a href="#">String</a> | Looks up the passed DNS address and returns its IP address.                              |

## NetworkInterface Object

---

| Name        | Parameters                        | Description   |
|-------------|-----------------------------------|---|
| IsConnected |                                   | Returns a <a href="#">Boolean</a> . Returns <a href="#">True</a> if connected to a network. If it returns <a href="#">False</a> , then using sockets may cause a dial-up to occur. This function is not supported on Linux and will always return <a href="#">False</a> . |
| WakeOnLan   | Address as <a href="#">String</a> | Wakes up a remote computer on the LAN, assuming the computer supports this feature. You can pass a broadcast address, an IP address, or a MAC address of the machine you want to wake up.   |

### Notes

The `LookupDNSAddress` and `LookupIPAddress` methods return empty strings on errors but return the string that was passed to the function if the address could not be resolved.

### Examples

The following call returns the IP address for the REALSoftware.com domain name.

```
Dim s as String
s=System.Network.LookupIPAddress("REALSoftware.com")
if s <> "" then
    MsgBox s
else
    MsgBox "An error occurred"
end if
```

### See Also

[System](#) object.

---

## NetworkInterface Object

Used to obtain information about the computer's network interface.

### Properties

| Name       | Type                   | Description                             |
|------------|------------------------|---|
| IPAddress  | <a href="#">String</a> | The IP address of the user's computer.  |
| MacAddress | <a href="#">String</a> | The Mac address of the user's computer. |
| SubnetMask | <a href="#">String</a> | The subnet mask of the user's computer. |

### Notes

Use the `GetNetworkInterface` method of the [System](#) object to instantiate the **NetworkInterface** object. See the example.

REALbasic provides multiple network interface support for Windows, Mac OS X, and Linux. This allows you to write applications that can bind to different network interface

cards on a user's machine. You can use this to write tunneling applications, for example.

To see what interfaces are installed on the user's machine, use the `GetNetworkInterface` method of the [System](#) object and assign the obtained interface object to the `NetworkInterface` property of the [SocketCore](#) class. When you do so, the socket will bind to that network interface.

## Example

The following simple example displays the IP address, Subnet mask, and Mac address for the selected network interface. At start-up the application detects all the network interfaces installed on the user's computer and loads them into a [PopupMenu](#). The user then selects the desired network interface and the values are displayed in [EditFields](#).

The [PopupMenu](#)'s Open event handler is:

```
Dim i as Integer
For i = 0 to System.NetworkInterfaceCount-1
    PopupMenu1.addRow Str(i)
Next
```

The Change event handler for the [PopupMenu](#) is this:

```
Dim n as NetworkInterface
//Get the NetworkInterface object for the selected item
n = System.GetNetworkInterface(Me.listIndex)

//Get the MAC Address
MacAddressField.Text = n.MACAddress
//Get the IP Address
IPAddressField.Text = n.IPAddress
//Get the Subnet Mask
SubnetMaskField.Text = n.SubnetMask
```

## See Also

[System](#) object; [SocketCore](#), [TCPSocket](#) classes.

# New Operator

Used to create new instances of objects at runtime.

## Syntax

The **New** operator has several different syntaxes:

When instantiating new instances of controls:

### ***object reference = New ObjectName***

| Part             | Type            | Description  |
|------------------|-----------------|--|
| object reference | Any object type | Required; variable of type object name.                |
| object name      | Object name     | Required. Any name of an object in the current window. |

When instantiating new instances all other types of objects:

### ***object reference = New ObjectClass | ControlArrayName [(param1, param2,...)]***

| Part             | Type            | Description  |
|------------------|-----------------|--|
| object reference | Any object type | Variable of type object class.                             |
| object class     | Object name     | Name of any object class other than the control classes.   |
| ControlArrayName | Object name     | The name of any control in the current window.             |
| param1...n       | Any             | Optional parameters passed to the object or control array. |

When creating and instantiating instances of all other types of objects:

### ***Dim ObjectReference As New ObjectClass| ControlArrayName [param1, param2,...]***

| Part             | Type            | Description  |
|------------------|-----------------|--|
| ObjectReference  | Any object type | Required; variable of type object class.                           |
| ObjectClass      | Object name     | Required. Name of any object class other than the control classes. |
| ControlArrayName | Object name     | The name of any control in the current window.                     |
| param1...n       | Any             | Optional parameters passed to the object or control array.         |

## Notes

The [constructor](#) for a class is called whenever you create a new instance of it with the **New** statement.

The **New** operator can appear as a modifier on a [Dim](#) statement that creates a new object. This usage takes the place of two lines of code: The [Dim](#) statement that creates the object and a separate **New** statement that instantiates the object.

## Creating new instances of controls or menu items

The **New** operator can only create a new instance of a control that already exists in the window or a menu item that already exists on a menu. The object name acts as a template for creating the new instance of the control or menu item. The **New** operator works if the control or menu item is in a control array. The object reference variable must be defined as type object name. For an example, see the section “Creating New Instances of Controls On The Fly” on page 278 of the *User’s Guide*.

**Creating new instances of all other types of objects**

The **New** operator can be used to create a new instance of any type of object (for controls see the notes above). The object reference must be defined as being of type object class.

If the class's Visible property is set to [True](#), the new instance will appear visibly in the running application.

**Examples**

This example creates a new instance of a pushbutton called "PushButton1" then sets its caption. Add a PushButton to a window and name it PushButton1. Set its Index property to zero, making it the first element in a control array. Set its Caption property to "Original".

In its Action event, create a clone of the pushbutton and enter this code

```
Dim pb As PushButton
pb=New pushbutton1
pb.caption="Clone"
pb.Left=Me.Left+Me.Width+10
```

When you run the application and click the pushbutton, it creates a new pushbutton to its right that is the second element of the control array. You can tell by its caption that it is the new pushbutton that was created in the Action event of the original PushButton.

This example creates and then displays a new instance of a [Window](#) called "AboutBox".

```
Dim w As aboutbox
w=New aboutbox
```

This example creates a new instance of a [MenuItem](#) called "WindowItem1".

```
Dim m As MenuItem
m=New WindowItem1
```

This example creates a new instance of a [MessageDialog](#) box.

```
Dim d as New MessageDialog
```

It is equivalent to the following two lines:

```
Dim d as MessageDialog
d=New MessageDialog
```

**See Also**

[Nil](#) keyword; [NewPicture](#) functions.

## NewAppleEvent Function

Creates a new [AppleEvent](#) class object.

**Syntax**

***result=NewAppleEvent(eventClass, eventID, creatorCode)***

| Part        | Type                       | Description  |
|-------------|----------------------------|--|
| Result      | <a href="#">AppleEvent</a> | The new AppleEvent object.   |
| EventClass  | <a href="#">String</a>     | The four letter <a href="#">AppleEvent</a> class code.                     |
| EventID     | <a href="#">String</a>     | The four letter AppleEvent ID.   |
| creatorCode | <a href="#">String</a>     | The four letter creator of the application the AppleEvent will be sent to. |

**Notes**

Use this function to create a new [AppleEvent](#) to send data to another application. The EventClass and EventID together uniquely define a particular AppleEvent. The EventClass acts as a category for logically grouping events together. While there are many standard (and even some required) AppleEvents, many applications have several custom AppleEvents for performing actions specific to the application. Consult the application's documentation or its author to get information on what custom AppleEvents may be available. If you pass a blank string for the CreatorCode, the event will be "sent to self."

Once you create the [AppleEvent](#) object and populate the necessary parameters with data, you then send the AppleEvent to the target application using the [AppleEvent](#) object's Send method.

**Examples**

See the [AppleEvent](#) class for an example of creating and sending an AppleEvent.

**See Also**

[AppleEvent](#) class.

---

## NewAppleEventTarget Function

Used to specify the target application that will receive an Apple Event.

**Syntax**

***result=NewAppleEventTarget(Name, Computer, Zone, PortType)***

| Part     | Type                             | Description                             |
|----------|----------------------------------|---|
| Result   | <a href="#">AppleEventTarget</a> | Object representing Apple Event Target  |
| Name     | <a href="#">String</a>           | Name of target application.             |
| Computer | <a href="#">String</a>           | Computer on which the app is installed. |
| Zone     | <a href="#">String</a>           | Appletalk zone                          |

| Part     | Type                   | Description   |
|----------|------------------------|---|
| PortType | <a href="#">String</a> | Four character Creator code identifying the target application. |

**Example** This example targets the Mailsmith application running on the Daystar computer.

```
Dim apptarget as AppleEventTarget
apptarget=NewAppleEventTarget("Mailsmith 1.1","Daystar","*","BLTO")
```

**See Also** [AppleEvent](#) class.

## NewMemoryBlock Function

Returns a new [MemoryBlock](#) class object of the size specified. Used for dealing with raw memory on any platform for [Declare](#) statements, for sending information over sockets and serial devices, and miscellaneous uses that require direct access to memory. It can be used with pointer parameters of shared library entry points.

**Syntax** **result=**[NewMemoryBlock](#)(size)

| Part   | Type                        | Description                                     |
|--------|-----------------------------|---|
| result | <a href="#">MemoryBlock</a> | A new memory block based on the size specified. |
| size   | <a href="#">Integer</a>     | The size (in bytes) of the new memory block.    |

**Notes** Use the **NewMemoryBlock** function to create an empty [MemoryBlock](#) class object that can be used to pass data to a shared library entry point or return data from an entry point.

If the application doesn't have sufficient memory to allocate to the memory block, [Nil](#) is returned. Check for [Nil](#) before proceeding.

**Example** The following example creates a 384 byte [MemoryBlock](#), whose bytes are numbered 0 to 383:

```
Dim m as MemoryBlock
m=NewMemoryBlock(384)
If m = Nil then
  Beep
  MsgBox "Insufficient memory to continue."
else
  //proceed normally
end if
```

**See Also** [MemoryBlock](#) class; [Nil](#) keyword.

## NewPicture Function

Returns a picture object of the size and depth specified. Used for off-screen drawing and direct pixel manipulation using the RGBSurface property of the [Picture](#) class.

### Syntax

***result=NewPicture(width, height, depth)***

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Picture</a> | A new picture based on the width, height, and depth specified.  |
| width  | <a href="#">Integer</a> | The width (in pixels) of the new picture.   |
| height | <a href="#">Integer</a> | The height (in pixels) of the new picture.  |
| depth  | <a href="#">Integer</a> | The color depth of the new picture. Acceptable values are 0, 1, 2, 4, 8, 16, and 32. A depth of zero indicates that the picture will be created with no pixel map at all; this option is for creating vector graphics, for example with the subclasses of the <a href="#">Object2D</a> class. |

### Notes

Use the **NewPicture** function to create an empty picture object that can be used to create graphics that you don't want drawn to screen immediately. This is commonly referred to as "drawing to an off-screen area."

Pictures that are created with **NewPicture** with a color depth of 16 or 32 can be manipulated at the pixel level using the RGBSurface property of the [Picture](#) class. This is faster than equivalent operations using the [Graphics](#) class's methods.

If the application doesn't have enough memory to create the new picture, it will be [Nil](#). The way to detect this is to immediately test for a [Nil Picture](#) object immediately after calling **NewPicture**. This is done in the example in the following section.

Alternatively, you can create a new picture using the [New](#) operator instead of using **NewPicture**. The parameters are the same in either case (width, height, depth), but when you use [New](#), an [OutOfMemoryException](#) will be raised if there is insufficient memory to create the requested pixel map.

## Example

The following example creates an animation in an off-screen area and then displays it in a [Canvas](#) control once the animation is complete. This approach produces a flicker-free animation.

```
Dim x,y as Integer
Dim buffer as Picture
'Build a buffer to hold the graphics while they
'are being assembled. Use the canvas Width and
'Height and the color depth on the main screen
Buffer=NewPicture(cvslImage.Graphics.Width,cvslImage.Graphics.Height,
Screen(0).Depth)
If Buffer <> Nil then
'Set the starting x and a random y
'position for the spaceship
x=0
y=Rnd*(cvslImage.Graphics.width-50)

'Loop until the x position of the ship is greater
'than the width of the canvas.
Do
'Draw the background into the buffer
buffer.Graphics.DrawPicture Space,0,0
'Draw the spaceship into the buffer
buffer.Graphics.DrawPicture Ship,x,y
'Increment the ship's position
'Note: I increased the increment because this method takes a little longer.
x=x+2
'Update the canvas with the buffer image
cvslImage.Graphics.DrawPicture buffer,0,0
loop until x>cvslImage.Graphics.Width
else 'buffer is Nil
MsgBox "Insufficient memory to continue"
end if
```

The following code creates the animation directly in the [Canvas](#) control, resulting in flicker as the object is moved:

```
Dim x,y as Integer  
'Set the starting x and a random y  
'position for the spaceship  
x=0  
y=Rnd*(cvslImage.Graphics.width-50)  
  
'Loop until the x position of the ship is greater  
'than the width of the canvas.  
Do  
    'Draw the background  
    cvslImage.Graphics.DrawPicture Space,0,0  
    'Draw the spaceship  
    cvslImage.Graphics.DrawPicture Ship,x,y  
    'Increment the ship's position  
    x=x+1  
    loop until x>cvslImage.Graphics.Width
```

Whenever possible, use **NewPicture** to create smoother animations.

**See Also** [Graphics](#), [Picture](#) classes; [Canvas](#) control; [Nil](#) keyword.

---

## Nil Keyword

Used to determine if an object is nil (no value).

**Syntax** *expression=Nil*

**Notes** **Nil** means no value. When an object is first instantiated, it is automatically initialized to [Nil](#). Some functions will return a **Nil** value under certain circumstances. [GetOpenFolderItem](#) will return **Nil** if the user clicks the Cancel button when [GetOpenFolderItem](#) presents the standard open file dialog box.

If you try to access an object that has a **Nil** value, a [NilObjectException](#) error will be generated. If you don't handle the exception, execution will stop. If you are testing the application in the IDE, you can select Project ▶ Break on Exceptions to halt execution whenever a runtime exception error occurs and open the Debugger.

If the code executes in a built application, the user will see a generic error message before the application quits.

To prevent this, you should always test for **Nil** values, either with an [If](#) statement or by including a [Try](#) or [Exception](#) block at the end of the method.

**Example**

This example uses the **Nil** keyword to determine if the user clicked the Cancel button in the standard open file dialog box presented by the [GetOpenFolderItem](#) function.

```
Dim f as FolderItem
f=GetOpenFolderItem("image/x-pict") //defined as a FileType
If f <> Nil Then
    //the user clicked OK so continue here
End If
```

This example uses an [Exception](#) block to handle [NilObjectExceptions](#).

```
Dim f as FolderItem
Try
    f=GetOpenFolderItem("image/x-pict")

    Catch err as NilObjectException
        MsgBox err.message
```

**See Also**

[RuntimeException](#) class; [Exception](#), [Try](#) blocks; [NilObjectException](#) error.

## NilObjectException Runtime Error

Occurs when code tries to access an object that doesn't exist.

**Super Class** [RuntimeException](#)

**Notes**

A **NilObjectException** occurs when you use a property or variable to access an object that doesn't exist. For example, when the [GetFolderItem](#) function is passed an invalid pathname, it returns **Nil**. Certain functions that are designed to return [FolderItems](#) on one platform may return **Nil** when run on other platforms. If your code attempts to access or write to a property of a [Nil FolderItem](#), a **NilObjectException** will occur.

A function also returns **Nil** if there isn't enough memory to load the object. For example, if you tried to load a large picture into memory but your application didn't have enough RAM left to load the picture, the function used to load the picture would return **Nil**.

Immediately after calling a function that returns an object (like [GetFolderItem](#), for example) you should check the object to see if it is **Nil** or add an [Exception](#) or [Try](#) block to the end of the method that handles [NilObjectExceptions](#). Any function's result that is not a [String](#), [Integer](#), [Single](#), [Double](#), [Color](#), or [Boolean](#) is an object and should be checked.

**Examples**

This example tries to open a PICT file using [GetFolderItem](#) and display it in an [ImageWell](#). If the pathname passed to [GetFolderItem](#) is invalid, it returns **Nil**. The

## NoOpenTransportException Error

---

**NilObjectException** is handled in the [Exception](#) block which is after the last “regular” line of the method.

If the pathname is valid, but the item does not exist, [GetFolderItem](#) returns a valid [FolderItem](#) whose [AbsolutePath](#) property is equal to the pathname. You should check the [Exists](#) property of the [FolderItem](#) to determine whether the file exists before trying to do something with the [FolderItem](#).

This code, in other words, checks for two different problems—invalid pathname and valid path to a nonexistent file.

```
Dim f as FolderItem
f=GetFolderItem("HD:Images:Logo")
If Not f.exists then
    Beep
    MsgBox "The file "+f.getAbsolutePath+" doesn't exist!"
else //document exists
    If f.MacType="PICT" then
        ImageWell1.image=f.OpenAsPicture
    end if
end if
Exception err as NilObjectException
MsgBox "Invalid pathname!"
```

The following example creates a new, empty picture. If there isn’t enough memory to create the picture, [Nil](#) will be returned by the [NewPicture](#) function. The example code checks for this and informs the user:

```
Dim p as Picture
p = NewPicture(100,100,32)
If p = Nil then
    Beep
    MsgBox "There isn't enough memory available to create the picture."
else
    //do something with the picture here
end if
```

### See Also

[RuntimeException](#) class; [Function](#), [Raise](#) statements; [Nil](#) keyword; [Exception](#), [Try](#) blocks.

---

## NoOpenTransportException Error

**Super Class** [RuntimeException](#)

**Notes**

This error can occur if the built application is running on Mac OS “classic” and networking operation that requires OpenTransport is attempted. OpenTransport is not required on any other platform.

**See Also**

[RuntimeException](#), [Exception](#), [Try](#) blocks; [Catch](#) statement.

## Not Operator

Used to perform logical negation of a [Boolean](#) expression.

**Syntax**

***result=Not expression***

| Part       | Description                                   |
|------------|---|
| result     | A <a href="#">boolean</a> value.              |
| expression | Any valid <a href="#">boolean</a> expression. |

**Notes**

**Not** returns the opposite of the [Boolean](#) value **expression** evaluates to. If **expression** evaluates to [True](#), **Not** returns [False](#). If **expression** evaluates to [False](#), **Not** returns [True](#).

**See Also**

[And](#), [Or](#) operators; [Operator\\_Not](#) function.

## NotePlayer Control

Plays musical notes via the QuickTime on Macintosh and the MIDI functions on Windows.

**Super Class**

[Control](#)

Because this is a [Control](#), see the [Control](#) class for other properties and events that are common to all [Control](#) objects.

**Properties**

| Name              | Type                    | Description   |
|-------------------|-------------------------|---|
| <b>Index</b>      | <a href="#">Integer</a> | The number of the control when it is part of a control array.                       |
| <b>Instrument</b> | <a href="#">Integer</a> | The number of the musical instrument to be used to play the note. (see chart below) |
| <b>Left</b>       | <a href="#">Integer</a> | The left side of the control in local coordinates (relative to the window).         |
| <b>Name</b>       | <a href="#">String</a>  | The name of the object.   |
| <b>Top</b>        | <a href="#">Integer</a> | The top of the control in local coordinates (relative to the window).               |

## Methods

| Name     | Parameters   | Description  |
|----------|--|--|
| PlayNote | Pitch as <a href="#">Integer</a> , Velocity as <a href="#">Integer</a> | Plays the note specified by the pitch at the velocity specified. |

## Notes

When calling the PlayNote method, the pitch is represented as a number from 0 to 127 where middle C is 60. Incrementing by 1 raises the pitch by a half step. Decreasing by 1 decreases the pitch by a half step. You can also express the value of pitch as a number from 256 to 32767, i.e.  $\text{pitch} * 256$ . Use this to specify half-steps. For example, if you want a pitch of 60.5, specify  $60.5 * 256 = 15488$  as the value of Pitch.

The velocity parameter is a scale from 0 to 127 that represents how hard the key is pressed. 127 represents a very hard key press. 0 represents the key no longer being pressed. When calling the PlayNote method, the note played will continue to play until it is played again with a velocity of 0.

## Linux

The **NotePlayer** is not supported on Linux.

## Instruments

| Instrument            | Value | Instrument   | Value |
|-----------------------|-------|--------------|-------|
| Acoustic Grand Piano  | 1     | Soprano Sax  | 65    |
| Bright Acoustic Piano | 2     | Alto Sax     | 66    |
| Electric Grand Piano  | 3     | Tenor Sax    | 67    |
| Honky-tonk Piano      | 4     | Baritone Sax | 68    |
| Rhodes Piano          | 5     | Oboe         | 69    |
| Chorused Piano        | 6     | English Horn | 70    |
| Harpsichord           | 7     | Bassoon      | 71    |
| Clavinet              | 8     | Clarinet     | 72    |
| Celesta               | 9     | Piccolo      | 73    |
| Glockenspiel          | 10    | Flute        | 74    |
| Music Box             | 11    | Recorder     | 75    |
| Vibraphone            | 12    | Pan Flute    | 76    |
| Marimba               | 13    | Bottle blow  | 77    |
| Xylophone             | 14    | Shakuhachi   | 78    |
| Tubular bells         | 15    | Whistle      | 79    |
| Dulcimer              | 16    | Ocarina      | 80    |
| Draw Organ            | 17    | Square Lead  | 81    |
| Percussive Organ      | 18    | Saw Lead     | 82    |
| Rock Organ            | 19    | Calliope     | 83    |

| <b>Instrument</b>          | <b>Value</b> | <b>Instrument</b> | <b>Value</b> |
|----------------------------|--------------|-------------------|--------------|
| Church Organ               | 20           | Chiffer           | 84           |
| Reed Organ                 | 21           | Synth Lead 5      | 85           |
| Accordion                  | 22           | Synth Lead 6      | 86           |
| Harmonica                  | 23           | Synth Lead 7      | 87           |
| Tango Accordion            | 24           | Synth Lead 8      | 88           |
| Acoustic Nylon Guitar      | 25           | Synth Pad 1       | 89           |
| Acoustic Steel Guitar      | 26           | Synth Pad 2       | 90           |
| Electric Jazz Guitar       | 27           | Synth Pad 3       | 91           |
| Electric clean Guitar      | 28           | Synth Pad 4       | 92           |
| Electric Guitar muted      | 29           | Synth Pad 5       | 93           |
| Overdriven Guitar          | 30           | Synth Pad 6       | 94           |
| Distortion Guitar          | 31           | Synth Pad 7       | 95           |
| Guitar Harmonics           | 32           | Synth Pad 8       | 96           |
| Wood Bass                  | 33           | Ice Rain          | 97           |
| Electric Bass Fingered     | 34           | Soundtracks       | 98           |
| Electric Bass Picked       | 35           | Crystal           | 99           |
| Fretless Bass              | 36           | Atmosphere        | 100          |
| Slap Bass 1                | 37           | Bright            | 101          |
| Slap Bass 2                | 38           | Goblin            | 102          |
| Synth Bass 1               | 39           | Echoes            | 103          |
| Synth Bass 2               | 40           | Space             | 104          |
| Violin                     | 41           | Sitar             | 105          |
| Viola                      | 42           | Banjo             | 106          |
| Cello                      | 43           | Shamisen          | 107          |
| Contrabass                 | 44           | Koto              | 108          |
| Tremolo Strings            | 45           | Kalimba           | 109          |
| Pizzicato Strings          | 46           | Bagpipe           | 110          |
| Orchestral Harp            | 47           | Fiddle            | 111          |
| Timpani                    | 48           | Shanai            | 112          |
| Acoustic String Ensemble 1 | 49           | Tinkle bell       | 113          |
| Acoustic String Ensemble 2 | 50           | Agogo             | 114          |
| Synth Strings 1            | 51           | Steel Drums       | 115          |
| Synth Strings 2            | 52           | Woodblock         | 116          |
| Aah Choir                  | 53           | Taiko Drum        | 117          |
| Ooh Choir                  | 54           | Melodic Tom       | 118          |
| Synvox                     | 55           | Synth Tom         | 119          |

## NthField Function

---

| Instrument    | Value | Instrument        | Value |
|---------------|-------|-------------------|-------|
| Orchestra Hit | 56    | Reverse Cymbal    | 120   |
| Trumpet       | 57    | Guitar Fret Noise | 121   |
| Trombone      | 58    | Breath Noise      | 122   |
| Tuba          | 59    | Seashore          | 123   |
| Muted Trumpet | 60    | Bird Tweet        | 124   |
| French Horn   | 61    | Telephone Ring    | 125   |
| Brass Section | 62    | Helicopter        | 126   |
| Synth Brass 1 | 63    | Applause          | 127   |
| Synth Brass 2 | 64    | Gunshot Table     | 128   |
|               |       | Drum Kit          | 16385 |

**Examples** The following plays a note on the ‘Church Organ’ instrument:

```
NotePlayer1.Instrument=20  
NotePlayer1.PlayNote(60,60)
```

The following line of code stops the note:

```
NotePlayer1.PlayNote(60,0)
```

**See Also** [Control](#) class.

---

## NthField Function

Returns a field from a row of data. The first field is numbered 1. If you need to parse binary data, use [NthFieldB](#) instead.

**Syntax** *result=NthField(source, separator, fieldNumber)*

**OR**

*result=stringVariable.NthField(separator, fieldNumber)*

| Part           | Type                    | Description  |
|----------------|-------------------------|--|
| result         | <a href="#">String</a>  | The field value desired.   |
| source         | <a href="#">String</a>  | The fields of data.  |
| separator      | <a href="#">String</a>  | The character that separates the columns of data.                              |
| fieldNumber    | <a href="#">Integer</a> | The column number of the field of data desired. The first field is numbered 1. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .                                  |

**Notes**

The **NthField** function returns the field value from the source that precedes the *fieldNumber* occurrence of the **separator** in the source.

If *fieldNumber* is out of bounds, an empty string is returned. **NthField** is not case-sensitive.

**Examples**

This example returns “Smith”.

```
Dim field As String
```

```
field=NthField("Dan*Smith*11/22/69*5125554323*Male","*",2)
```

Using the second syntax:

```
Dim s,field As String
```

```
s="Dan*Smith*11/22/69*5125554323*Male"
```

```
field=s.NthField("*",2)
```

```
MsgBox field
```

See also the example that illustrates how to populate a [PopupMenu](#) control.

**See Also**

[CountFields](#), [NthFieldB](#), [Split](#) functions.

## NthFieldB Function

Returns a field from a row of data. **NthFieldB** is identical to [NthField](#) except that it treats the source data as binary data. The first field is numbered 1.

**Syntax**

*result*=NthFieldB(*source*, *separator*, *fieldNumber*)

OR

*result*=*stringVariable*.NthFieldB(*separator*, *fieldNumber*)

| Part                  | Type                    | Description  |
|-----------------------|-------------------------|--|
| <i>result</i>         | <a href="#">String</a>  | The field value desired.   |
| <i>source</i>         | <a href="#">String</a>  | The fields of data.  |
| <i>separator</i>      | <a href="#">String</a>  | The character that separates the columns of data.                              |
| <i>fieldNumber</i>    | <a href="#">Integer</a> | The column number of the field of data desired. The first field is numbered 1. |
| <i>stringVariable</i> | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .                                  |

**Notes**

The **NthFieldB** function returns the field value from the source that precedes the *fieldNumber* occurrence of the **separator** in the source.

## Object Class

---

If fieldNumber is out of bounds, an empty string is returned. **NthFieldB** is not case-sensitive.

**Examples** This example returns “Smith”.

```
Dim field As String  
field=NthFieldB("Dan*Smith*11/22/69*5125554323*Male","*",2)
```

Using the second syntax:

```
Dim s,field As String  
s="Dan*Smith*11/22/69*5125554323*Male"  
field=s.NthFieldB("*",2)  
MsgBox field
```

See also the example that illustrates how to populate a [PopupMenu](#) control.

**See Also** [CountFieldsB](#), [NthField](#), [SplitB](#) functions.

---

## Object Class

**Object** is the base class of all other classes. Any object can be assigned into a variable of type **Object**. The default value of a new instance of an object is [Nil](#). The **Object** class has no properties, methods, or events.

---

## Object2D Class

The base class of any object that can be drawn in a vector graphics environment.

**Super Class** [Object](#)

### Properties

| Name        | Type                   | Description  |
|-------------|------------------------|--|
| Border      | <a href="#">Double</a> | Opacity of the border, from 0 (completely transparent) to 100 (opaque). The default is 0. Degrees of transparency is currently supported only on Mac OS X. On other platforms, the border is either visible (100%) or invisible. |
| BorderColor | <a href="#">Color</a>  | Color of the border.   |
| BorderWidth | <a href="#">Double</a> | Width of the border, in points. The default is 1.  |

| Name      | Type                   | Description  |
|-----------|------------------------|--|
| Fill      | <a href="#">Double</a> | Opacity of the interior, from 0 (completely transparent) to 100 (opaque). The default is 100. Degrees of transparency is currently supported only on Mac OS X. On other platforms, the fill is either visible (100%) or invisible. |
| FillColor | <a href="#">Color</a>  | Color of the interior of the shape.  |
| Rotation  | <a href="#">Double</a> | Clockwise rotation, in radians, around the X, Y point.   |
| Scale     | <a href="#">Double</a> | Scaling factor relative to the object's original size  |
| X         | <a href="#">Double</a> | Horizontal position of center or main anchor point.  |
| Y         | <a href="#">Double</a> | Vertical position (down from top) position of center or anchor point.  |

**Notes**

**Object2D** is the base class for a group of subclasses that enable you to create vector (as opposed to bitmap) graphics. The subclasses are shown in the following table:

| Class                          | Description  |
|--------------------------------|--|
| <a href="#">ArcShape</a>       | Draws an arc of a circle.  |
| <a href="#">CurveShape</a>     | Draws straight lines or curves using one or more "control points."                             |
| <a href="#">FigureShape</a>    | Draws polygons that can (optionally) have curved sides.  |
| <a href="#">OvalShape</a>      | Draws circles and ovals.   |
| <a href="#">PixmapShape</a>    | Imports a bitmap picture into the image.   |
| <a href="#">RectShape</a>      | Draws a square or rectangle.   |
| <a href="#">RoundRectShape</a> | Draws a square or rectangle with rounded corners (subclassed from <a href="#">RectShape</a> ). |
| <a href="#">StringShape</a>    | Draws text strings in a specified font, font size, and style.                                  |

These basic shapes can be organized into a hierarchy using the [Group2D](#) class.

The [Picture](#) class has a new property, Objects, that enables you to include a vector graphics drawing (a [Group2D](#) object) in a picture and the [Graphics](#) class has a new method, DrawObject, that enables you to draw a [Group2D](#) object within the [Graphics](#) object. Also, the SaveAsPicture method of the [FolderItem](#) class now has an optional parameter that enables you to save a vector graphics picture in a format that retains the vector information. The OpenAsVectorPicture method of the [FolderItem](#) class attempts to open an existing image as a vector graphic and map the contents of the image into Object2D drawing primitives.

**Examples**

See the examples for [ArcShape](#), [CurveShape](#), [FigureShape](#), [OvalShape](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), and [StringShape](#).

**See Also**

[ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Graphics](#), [Group2D](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

# Object3D Class

A drawable 3D object type with zero or more shapes. Each shape can represent the contents of a 3DMF file. For use in an [RB3DSpace](#) control.

**Super Class** [Element3D](#)

## Properties

| Name                 | Type                    | Description  |
|----------------------|-------------------------|--|
| <b>MaterialCount</b> | <a href="#">Integer</a> | Then number of materials in the object.  |
| NullShader           | <a href="#">Boolean</a> | If set to <a href="#">True</a> , causes the object to be rendered without respect to external lighting, as if you had set AmbientLight=100 and FloodLight=0). Useful for objects that should appear to glow from within, such as torches and explosions, or for backdrops or sprites that have lighting pre-calculated as part of the texture. |
| RenderBackFaces      | <a href="#">Boolean</a> | Allows you to decide on an object-by-object basis whether polygons facing away from the camera should be drawn. The default value is <a href="#">False</a> , meaning such polygons should be skipped for faster rendering.   |
| Shape                | <a href="#">Integer</a> | Selects which of several shapes is currently being drawn. The first shape added is numbered 0. Select this shape by setting Shape=0, and so forth. If you try to assign a value to Shape before actually creating a shape, its value is set to -1.   |
| <b>ShapeCount</b>    | <a href="#">Integer</a> | The number of shapes in the object.  |
| <b>TrimeshCount</b>  | <a href="#">Integer</a> | The number of Trimeshes in the object.   |
| Visible              | <a href="#">Boolean</a> | <a href="#">True</a> if visible.   |

## Methods

| Name             | Parameters   | Description   |
|------------------|--|---|
| AddShapeFromFile | File as <a href="#">FolderItem</a> [, insertPos as <a href="#">Integer</a> ] | Loads a model file and adds it to an object in one step. If the file does not exist or does not contain valid model data, then no shape is added. The optional parameter insertPos specifies the 0-based position at which the new shape should appear in the shape list. |

| Name                    | Parameters   | Description  |
|-------------------------|--|--|
| AddShapeFromHandle      | data as <a href="#">Integer</a><br>[, insertPos as <a href="#">Integer</a> ]   | Adds a shape from geometry which has been created through <a href="#">Declare</a> statements.<br>The optional parameter insertPos specifies the 0-based position at which the new shape should appear in the shape list.   |
| AddShapeFromString      | data as <a href="#">String</a> [, insertPos as <a href="#">Integer</a> ]   | Loads a model from 3DMF data in a string.<br>The optional parameter insertPos specifies the 0-based position at which the new shape should appear in the shape list.   |
| AddShapePicture         | image as <a href="#">Picture</a> , scale as <a href="#">Double</a><br>[, insertPos as <a href="#">Integer</a> ]                                    | Creates a flat, rectangular model from a REALbasic picture, with white pixels appearing transparent. The picture is treated as a 16-bit image regardless of the original color depth. A scale of 1.0 means that each pixel of the picture is 1 unit across in the 3D space. The scale value of 0.5 means that each pixel in the original is half a unit wide in the 3D space.<br>Pictures are padded internally to a power of 2 in both width and height. For example, if the original picture is 100 x 20 pixels, this will be padded internally to 128 by 256. To make the most of the available texture memory, use pictures that are a power of 2 in both directions.<br>The optional parameter insertPos specifies the 0-based position at which the new shape should appear in the shape list. |
| AddShapePictureWithMask | image as <a href="#">Picture</a> , mask as <a href="#">Picture</a><br>scale as <a href="#">Double</a><br>[, insertPos as <a href="#">Integer</a> ] | Like AddShapePicture, but specifies a second picture as a mask to create a translucent image.  |
| Clone                   |  | Clones the object and returns another object of the same class as the original, even when the original is a subclass of <b>Object3D</b> . The clone has the same geometry as the original.<br>Returns an <b>Object3D</b> .   |

| Name           | Parameters                         | Description   |
|----------------|------------------------------------|---|
| ComputeBounds  | Type as <a href="#">Integer</a>    | Returns a <a href="#">Bounds3D</a> bounding volume of type <i>Type</i> . Typically used to create and assign the object's bounding volume to the <i>Bound</i> property, |
| GetShapeHandle | Index as <a href="#">Integer</a>   | Returns (as an <a href="#">Integer</a> ) the TQ3GeometryObject of the specified shape.  |
| Material       | Index as <a href="#">Integer</a>   | Returns a <a href="#">Material</a> . Gets or sets the material for the specified shape.   |
| MoveForward    | distance as <a href="#">Double</a> | Moves the object in whatever direction it is facing.  |
| RemoveShape    | Index as <a href="#">Integer</a>   | Removes the shape specified by Index. Index is zero-based.  |
| Trimesh        | Index as <a href="#">Integer</a>   | Gets or sets the passed <a href="#">Trimesh</a> in the Object3D.  |

### Notes

This is the basic three-dimensional object type. For information about the 3D Metafile format used to create 3DMF objects used in [Rb3DSpace](#), see:

<http://developer.apple.com/documentation/QuickTime/QD3D/qd3dmetafile.htm>

An **Object3D** has a position, an orientation, and it may have multiple geometry models. Use *AddShapeFromString* to load a model from 3DMF data in a string, or use *AddShapePicture* to create a flat, rectangular model from a REALbasic picture, with white pixels appearing transparent. The scale parameter determines the size of the model in 3D space; a scale of 1.0 means that each pixel in the picture is 1 unit across in 3D space. You can change its position and orientation by directly modifying its properties, or you can use the Roll, Pitch, and Yaw methods to rotate the object, and *MoveForward* to move it in whatever direction it's facing (considered to be the +Z direction initially).

You can change its scale by modifying the *Scale* property; this defaults to 1.0. If you need more than one copy of an object in the game (as is often the case), use the *Clone* method which returns a new object that shares the same geometry as the original. Because the geometry is shared, this is much more economical than loading separate, duplicate copies of the model.

**Example**

The following loads a 3DMF object from an external .3DMF file. This file type is declared in the File Types Set “FileTypes1”.

```

Dim f As FolderItem
Dim obj As Object3D

//allow only 3DMF files types to be shown
f = GetOpenFolderItem(FileTypes1.All)
If f <> Nil then

    // create an object
    obj = New Object3D

    // give the object a shape specified by the 3DMF
    obj.AddShapeFromFile f

    If obj.Shapecount < 0 then
        MsgBox "No valid 3DMF data found in "+.DisplayName+".
        Return
    End if

    // add the object to this Rb3DSpace
    Me.objects.Append obj
    End if

```

**See Also**

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [RB3DSpace](#) control.

## OLEContainer Control

The generic container for embedding ActiveX controls (Windows only).

**Super Class** [RectControl](#)

**Properties**

| Name      | Type                      | Description   |
|-----------|---------------------------|---|
| Content   | <a href="#">OLEObject</a> | The OLEObject associated with the ActiveX control. You use this to automate the ActiveX control.            |
| ProgramID | <a href="#">String</a>    | The ActiveX control's program ID as stored in the registry. It can also be the GUID (include curly braces). |

## OLEException Error

---

### Methods

| Name              | Parameters | Description   |
|-------------------|------------|---|
| Create            |            | Creates an ActiveX control. Create uses the ProgramID property to create the ActiveX control. Returns a <a href="#">Boolean: True</a> if successful, <a href="#">False</a> if unsuccessful. |
| Destroy           |            | Destroys the currently embedded ActiveX control, so you can reuse your OLEContainer.  |
| ShowPropertyPages |            | Displays the ActiveX control's Property pages, if any. Not all ActiveX controls support Property pages.   |

### Events

| Name           | Parameters   | Description  |
|----------------|--|--|
| EventTriggered | NameOfEvent as <a href="#">String</a><br>Parameters() as array of <a href="#">Variants</a> | Occurs when the ActiveX control receives an event from the user. The event name is passed as the first parameter specifying which event occurred. The second parameter contains the parameters for this event, passed in as an array of variants. The user should check the <a href="#">Ubound</a> of the parameters list. |
| ShowObject     |  | Occurs when the ActiveX object is ready to be displayed.   |

### See Also

[OLEObject](#) class.

---

## OLEException Error

An OLE-related exception occurred. This class handles Microsoft Office Automation exceptions.

### Super Class

[RuntimeException](#)

---

## OLEObject Class

Used to automate COM servers. Use the [WordApplication](#), [ExcelApplication](#), and [PowerPointApplication](#) classes to automate Microsoft Office applications.

### SuperClass

[Object](#)

## Constructors

| Name      | Parameters  | Description  |
|-----------|---|--|
| OLEObject | ProgramID as <a href="#">String</a>   | <i>ProgramID</i> is the COM server's program ID as stored in the registry. It can also be the Class ID (in curly braces). This constructor will try to find a previous instance of the COM server if it's running. Otherwise, it will create a new instance. |
| OLEObject | ProgramID as <a href="#">String</a><br>NewInstance as <a href="#">Boolean</a> | The <i>ProgramID</i> parameter is the same as above. The second parameter specifies whether to create a new instance of the COM server ( <a href="#">True</a> ) or try to use an existing one if it is running ( <a href="#">False</a> ).                    |

## Events

| Name           | Parameters   | Description   |
|----------------|--|---|
| EventTriggered | NameOfEvent as <a href="#">String</a><br>Parameters() as <a href="#">Variant</a> | Occurs when the OLEObject receives an event from the automation server. The event name is passed as the first parameter and the parameters for the event are passed as an array of <a href="#">variants</a> . |

## Methods

| Name   | Parameters   | Description  |
|--------|--|--|
| Invoke | NameOfFunction as <a href="#">String</a>   | Invokes a method of the COM server. Returns a <a href="#">Variant</a> .  |
| Invoke | NameOfFunction as <a href="#">String</a><br>Parameters() as array of <a href="#">Variants</a>                                | Invokes a method of the COM server, and passes the array of parameters to the method. Make sure to correctly dimension the array, as this will determine the number of parameters that get passed to the method. The first parameter begins at 1.  |
| Value  | PropertyName as <a href="#">String</a><br>[,params() as <a href="#">Variant</a> ,]<br>[,ByValue as <a href="#">Boolean</a> ] | Used to get or set a value of the object. The parameter is the name of the property to assign a new value to or to get the value. The value property can now take a list of properties when assigning a value, i.e., OLEObject.Value(PropertyName as String, params() as Variant)=value. If the optional parameter <i>ByValue</i> is <a href="#">True</a> , property assignment is by value. |

## Notes

By default, **OLEObject** will make the property assignment by value. If it encounters an error it will try by reference if the property is an object. If the optional *ByValue* parameter is [True](#), the property assignment is by value (i.e., a copy); otherwise the assignment is by reference (i.e., a pointer copy). In Visual Basic, an assignment by reference is done using the Set command, but since REALbasic doesn't provide that

feature, you will need to use the *ByValue* parameter when you know the assignment should be by reference.

Currency types are treated as [String](#) to preserve precision.

**Example** The following example automates Internet Explorer.

```
Dim obj as OLEObject
Dim v as Variant
Dim params(1) as Variant

obj = New OLEObject("InternetExplorer.Application", True)
obj.property("Visible") = True
params(1) = "http://www.realsoftware.com/"
v = obj.invoke("Navigate", params)

Exception err as OLEException
Msgbox err.message
```

The OLEObject class supports setting indexed properties. For example the Word.Document.Compatibility property is an indexed property. Here is an example.

```
Dim word As New OLEObject("Word.Application")
Dim doc As OLEObject

word.Visible = True
doc = word.Documents.Add

Dim params(1) As Variant
params(1) = Office.wdNoTabHangIndent

doc.Value("Compatibility", params) = True
// or
/doc.Compatibility(Office.wdNoTabHangIndent)=True
```

This example automates Microsoft Word.

```

Dim obj as OLEObject
Dim docs as OLEObject
Dim doc as OLEObject
Dim range as OLEObject
Dim v as Variant

obj = New OLEObject("Word.Application", True)
// make it visible
obj.Property("Visible") = True

v = obj.Property("Documents")
if v.objectvalue IsA OLEObject then
    docs = OLEObject(v.objectValue)
    v = docs.invoke("Add")
    if v.objectValue isa oleobject then
        doc = OLEObject(v.objectValue)
        v = doc.invoke("Range")
        if v.objectValue IsA OLEObject then
            range = OLEObject(v.objectValue)
            range.Property("Text") = "This is a sentence."
        end if
    end if
end if

Exception err as OLEException
MsgBox err.message

```

## See Also

[ExcelApplication](#), [Office](#), [OLEContainer](#), [OLEParameter](#), [PowerPointApplication](#), [WordApplication](#) classes; [OLEException](#) error.

---

## OLEParameter Class

Use to pass an OLE parameter.

### Properties

| Name      | Type    | Description  |
|-----------|---------|--|
| PassByRef | Boolean | If <a href="#">True</a> , the parameter is passed <a href="#">ByRef</a> . The default is <a href="#">False</a> . |

## ODBCDatabase Class

---

| Name     | Type                    | Description  |
|----------|-------------------------|--|
| Position | <a href="#">Integer</a> | Allows you to specify which position (1-based index) that you want the parameter to be passed in. This is akin to named parameters, if you know the position of the parameter within the function that you want to call, you can create an OLEParameter object and set its Position property to achieve the same effect. The default value is 0, which means no positioning. |
| Type     | <a href="#">Integer</a> | Specifies the type of the parameter.   |
| Value    | <a href="#">Variant</a> | The value of the parameter. It accepts one-dimensional arrays  |
| Value2D  |                         | The value of the parameter if the parameter is a two-dimensional array. See Notes.   |

### Notes

Here is an example of the usage of the Value2D property.

```
Dim word as new WordApplication
Dim doc as WordDocument
Dim param as new OLEParameter
Dim pts(3, 1) As Variant
pts(0, 0) = 50
pts(0, 1) = 20
pts(1, 0) = 100
pts(1, 1) = 160
pts(2, 0) = 150
pts(2, 1) = 160
pts(3, 0) = 200
pts(3, 1) = 20

param.Type = OLEParameter.ParamTypeSingle
param.Value2D = pts

doc = word.Documents.Add
doc.Shapes.AddCurve param
word.Visible = True
```

### See Also

[ExcelApplication](#), [OLEContainer](#), [OLEObject](#), [PowerPointApplication](#), [WordApplication](#) classes; [OLEException](#) error.

---

## ODBCDatabase Class

Used to open an ODBC database.

**Super Class** [Database](#)

## Properties

| Name            | Type                    | Description   |
|-----------------|-------------------------|---|
| Attribute       | <a href="#">Integer</a> | Optional attribute. See docs on ODBC for SQLSetConnectAttr. If you don't want to use <i>Attribute</i> , pass zero.  |
| AttributeString | <a href="#">String</a>  | Optional attribute. See docs on ODBC for SQLSetConnectAttr. If you don't want to pass <i>AttributeString</i> , pass the empty string.   |
| DataSource      | <a href="#">String</a>  | The connection string to be used to establish a connection to an ODBC database. Pass an empty string (" ") to the <i>DataSource</i> property to get an open-file dialog box. It will prompt you to choose either a File DSN (Data Source Name) or a User DSN. |

## Notes

In order to use this class, you must install the appropriate database plug-in in your plugins folder.

The set of database plug-ins is included on the REALbasic CD, but you may find more recent versions at the REAL Software web site, <http://www.realsoftware.com>.

The **ODBCDatabase** class also requires a third-party driver manager and driver.

REALbasic supports OpenLink's ODBC Driver manager and drivers ([www.openlinksw.com](http://www.openlinksw.com)) and Merant/Metro Technologies ODBC Manager. OpenLink supports both Mac OS 8/9 and Mac OS X, while Merant currently does not support Mac OS X. A driver manager must be installed as well as the driver for whichever database back end you wish to access. Check REAL Software's web site, <http://www.realsoftware.com>, for updated drivers.

## Example

This example presents an open-file dialog box in which you can select an ODBC database.

```

Dim db as ODBCDatabase
db=New ODBCDatabase
db.DataSource=" "
If db.Connect then
  //proceed with database operations
else
  MsgBox "The connection failed."
end if

```

## See Also

[Database](#), [DatabaseField](#), [DatabaseRecord](#), [RecordSet](#) classes.

## Oct Function

Returns as a [string](#), the octal version of the number passed.

**Syntax**

**result=Oct(value)**

| Part   | Type                    | Description                          |
|--------|-------------------------|--------------------------------------|
| result | <a href="#">String</a>  | The value passed converted to octal. |
| value  | <a href="#">Integer</a> | The number to be converted to octal. |

**Notes**

If the *value* is not a whole number, the decimal value will be truncated.

VB Compatibility Note: VB rounds the value to the nearest whole number so the **Oct** function will probably be changed in a future release to do this as well.

You can specify binary, hex, or octal numbers by preceding the number with the & symbol and the letter that indicates the number base. The letter b indicates binary, h indicates hex, and o indicates octal.

**Examples**

Here are examples of various numbers converted to octal:

```
Dim OctVersion As String  
OctVersion=Oct(5) //returns "5"  
OctVersion=Oct(75) //returns "113"  
OctVersion=Oct(256) //returns "400"
```

**See Also**

[&b](#), [&h](#), [&o](#) literals, [Bin](#), [Hex](#) functions.

---

## Office Class

The class that contains the enums you need for Office Automation.

**Super Class** [Object](#)**Notes**

Office Automation is supported under Windows and Mac OS X.

For Mac OS X, The OLE libraries are stored privately within Office, so you must compile your application and copy it to the Office folder inside the Microsoft Office X parent folder. The path is ..../Applications/Microsoft Office X/Office. For debugging, you need to copy the REALbasic IDE to the Office folder.

The language that you use to automate Microsoft Office applications is documented by Microsoft and numerous third-party books on Visual Basic for Applications (VBA).

Microsoft Office applications provide online help for VBA. To access the online help, choose Macros from the Tools Menu of your MS Office application, and then choose

Visual Basic Editor from the Macros submenu. When the Visual Basic editor appears, choose Microsoft Visual Basic Help from the Help menu. The help is contextual in the sense that it provides information on automating the Office application from which you launched the Visual Basic editor.

If VBA Help does not appear, you will need to install the help files. Windows Office 2003 prompts you to install the VBA help files when you first request VBA help. You don't need the master CD. On Macintosh, Office v.X does not install the VBA help files as part of the full install. Quit out of Office and locate your master CD. Open the "Value Pack" folder and double-click the Value Pack installer. In the Value Pack installer dialog, scroll down to the Programmability topic, select it, and click Continue. The installer will then add the VBA help files and examples to your Office installation. When the install finishes, the VBA help files will be available to the Visual Basic editor within all your Office X applications.

Microsoft has additional information on VBA at <http://msdn.microsoft.com/vbasic/> and have published their own language references on VBA. One of several third-party books on VBA is "VB & VBA in a Nutshell: The Language" by Paul Lomax (ISBN: 1-56592-358-8).

## Office Automation in REALbasic vs. VB

Working from the Application class. There is an implied Application instance when you write VBA code from within Excel, PowerPoint, or Word. For example:

```
Dim pres as Presentation
Dim slide1 as Slide

Set pres = Presentations.Add
' The above is the same as saying:
' Set pres = Application.Presentations.Add

Set slide1 = pres.Slides.Add(1, ppLayoutText)
```

In PowerPoint, this code would run just fine since it knows what a Presentation object is. Obviously if you typed this code in either Word or Excel, it would generate errors. Here's the REALbasic code:

```
Dim PowerPoint as new PowerPointApplication
Dim pres as OLEObject
Dim slide1 as OLEObject

pres = PowerPoint.Presentations.Add
slide1 = pres.Slides.Add(1, Office.ppLayoutText)
```

There's really only two differences here. First, all objects are now OLEObjects, so whether it's a Presentation or Slide, you can create them all as of type OLEObject. All constant values are in the Office module.

### Passing parameters by Name

REALbasic doesn't support passing parameters by name; however you can still achieve this with a bit of work. First of all, you have to understand how to use the OLEObject. Here's an example that you can model after. Let's record a find and replace macro in Word and translate it to REALbasic.

```
Selection.Find.ClearFormatting
Selection.Find.Replacement.ClearFormatting
With Selection.Find
    .Text = "find this"
    .Replacement.Text = "replace with"
    .Wrap = wdFindContinue
    .Format = false
    .MatchCase = false
    .MatchWholeWord = false
    .MatchWildcards = false
    .MatchSoundsLike = false
    .MatchAllWordForms = false
End With
Selection.Find.Execute Replace:=wdReplaceAll
```

Here's what it looks like in REALbasic.

```
Dim word as New WordApplication
Dim find as OLEObject

find = word.Selection.Find

find.ClearFormatting
find.Replacement.ClearFormatting
find.text = "find this"
find.Replacement.Text = "replace with"
find.Wrap = Office.wdFindContinue
find.Format = False
find.MatchCase = false
find.MatchWholeWord = false
find.MatchWildcards = false
find.MatchSoundsLike = false
find.MatchAllWordForms = false

// Now the fun stuff

Dim replaceParam as New OLEParameter

replaceParam.Value = Office.wdReplaceAll
// according to the docs on Find.Execute the Replace parameter is the 11th
replaceParam.Position = 11

find.Execute replaceParam
```

That's all there is too it. Obviously the most painful bit is finding the correct position of that named parameter. That's about the only time when you really need to launch VBA and look it up in their Object Browser.

**Troubleshooting** Here is a procedure to verify that the OLE libraries are installed.

One easy way is to launch Visual Basic Editor (under the Tools and Macros menu), and do some automation with VBA.

Here are the steps you can use to automate PowerPoint from Word:

- 1 Launch Word.**
- 2 Launch Visual Basic Editor in Word.**
- 3 Insert a UserForm.**
- 4 Add a CommandButton to the form.**
- 5 In the click event of the button, put in this code:**

```
Dim obj as Object  
Set obj = CreateObject("PowerPoint.Application")  
obj.Activate
```

- 6 Run the program and click on the button.**

If PowerPoint starts and you don't get any errors, then the OLE libraries are installed.

### Error Handling

Errors come through OLE, so you need to handle [OLEExceptions](#). This will report the last command that failed along with any additional information about the exception.

```
Dim word as New WordApplication  
word.ShowClipboard  
Exception err as OLEException  
Msgbox err.message
```

### See Also

[ExcelApplication](#), [OLEContainer](#), [OLEObject](#), [OLEParameter](#),  
[PowerPointApplication](#), [WordApplication](#) classes.

---

## One of the Interfaces of this Class is not of Type Class Interface Error

In the Properties pane for a Class Interface, you entered the name of a class in the Interfaces field that is not of type Class Interface. This occurs if you use the name of any 'ordinary' class instead of the name of a Class Interface.

## Only Boolean Constants can be Used with #If Error

Only the [Boolean](#) constants [TargetBigEndian](#), [TargetLittleEndian](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetCarbon](#), [TargetWin32](#), [TargetMachO](#), [TargetLinux](#), [DebugBuild](#), [TargetHasGUI](#), and [RBVersion](#) and [RBVersionString](#) can be used in the [#if](#) statement.

### Example

A ‘regular’ [Boolean](#) variable cannot be used as shown here.

```
Dim b as Boolean  
b=True  
#if b then  
    Beep  
#endif
```

### See Also

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetWin32](#) constants; [#If statement](#).

---

## Only One Parameter may use the Assigns Option Error

You used the [Assigns](#) keyword with more than one parameter. You can use [Assigns](#) with only one parameter and it must be the last parameter.

### Example

The following method declaration uses [Assigns](#) twice.

```
Sub myMethod(Assigns a as Integer,Assigns b as Integer)
```

### See Also

[Assigns](#) keyword.

---

## Only One Parameter my use the ParamArray Option Error

You tried to pass more than one parameter using the [ParamArray](#) keyword. You can pass only one parameter that uses the [ParamArray](#) keyword. If there is more than one parameter, the [ParamArray](#) keyword should be used for the last parameter.

**Example** The following method declaration uses the [ParamArray](#) keyword twice.

```
Sub myMethod(ParamArray a As Integer, ParamArray s as String)
```

**See Also** [ParamArray](#) keyword.

## Only String Constants can be used for Declaring Libraries Error

Occurs when you've declared an external function using a constant, but either the constant is not defined or it contains something other than a string.

## Only the First Parameter may use the Extends Option Error

You tried to use the [Extends](#) keyword for any parameter other than the first parameter.

## Only the Last Parameter may use the Assigns Option Error

You used the [Assigns](#) keyword with any parameter other than the last parameter. The [Assigns](#) keyword can only be used for one parameter and it must be the last parameter.

**Example** The following method declaration uses the [Assigns](#) keyword for the first parameter.

```
Sub myMethod(Assigns a As Integer, b As Integer)
```

**See Also** [Assigns](#) keyword.

## OpenBaseDatabase Class

Used to open an OpenBase database.

**Super Class** [Database](#)

## Properties

| Name     | Type                    | Description   |
|----------|-------------------------|---|
| Encoding | <a href="#">Integer</a> | This encoding value is the same value you'd use for <a href="#">GetTextEncoding</a> . For example, <code>&amp;h201</code> is ISOLatin1, <code>&amp;h202</code> is ISOLatin2, and so forth. This is used to convert from your specified encoding to the OpenBase database encoding, and vice versa. This is included for 4.0.2 compatibility and is deprecated in version 4.5 and above. |

## Notes

In order to use this class, you must install the appropriate database plug-in in your plugins folder.

The set of database plug-ins is included on the REALbasic CD, but you may find more recent versions at the REAL Software web site, [www.realsoftware.com](http://www.realsoftware.com).

If you use the [RecordSet](#) class to update a record using edit and update you need to include the primary key column of the table. If you don't, OpenBase won't be able to find the record in order to update it.

Caution: Inserting data into Object fields in OpenBase can corrupt the database. The workaround for this is to insert the record without the object fields and then use the [Update](#) method to add the data for the object fields.

## Example

The following code opens the OpenBase database, "pubs." The Host property can also be an IP address.

```
Dim obdb as OpenBaseDatabase
Obdb = New OpenBaseDatabase
Obdb.host = "opendb.mydomain.com"
Obdb.databasename = "pubs"
Obdb.Username="Sam"
Obdb.Password="Caveman"
If obdb.connect then
    //proceed with database operations
Else
    MsgBox "Failed to connect."
end if
```

## See Also

[Database](#), [DatabaseField](#), [RecordSet](#) classes.

# OpenCSVCursor Function

Opens a comma-delimited text file as a [RecordSet](#). The file must use commas to separate fields, returns to separate records, and double quotes ("") to enclose text fields. Commas are permitted within quoted strings. The following record is valid:

```
22,"Fred","Duesenberg","Engineer"
```

## Syntax

**result=OpenCSVCursor(file)**

| Part   | Type                       | Description   |
|--------|----------------------------|---|
| result | <a href="#">RecordSet</a>  | <a href="#">RecordSet</a> containing the rows and columns in the text file. |
| file   | <a href="#">FolderItem</a> | <a href="#">FolderItem</a> that refers to the text file.                    |

## Note

**OpenCSVCursor** requires the CSV Database plug-in. Place this plug-in in the REALbasic Plugins folder. The REALbasic CD includes all database plug-ins and you can always find the most current versions on REAL Software's web site, [www.realsoftware.com](http://www.realsoftware.com).

## Example

The following example opens the specified text file.

```
Dim db as RecordSet
db=OpenCSVCursor(GetFolderItem("table1.txt"))
If db = Nil then
  Beep
  MsgBox "The database couldn't be opened."
else
  //continue with database operations
end if
```

## See Also

[Database](#), [DatabaseField](#), [RecordSet](#) classes; [OpenDBFCursor](#), [OpenDTFDatabase](#) functions.

# OpenDialog Class

Creates a customizable Open-file dialog box. The non-customizable version of this dialog is displayed by the [GetOpenFolderItem](#) function.

**Super Class** [FolderItemDialog](#).

## Notes

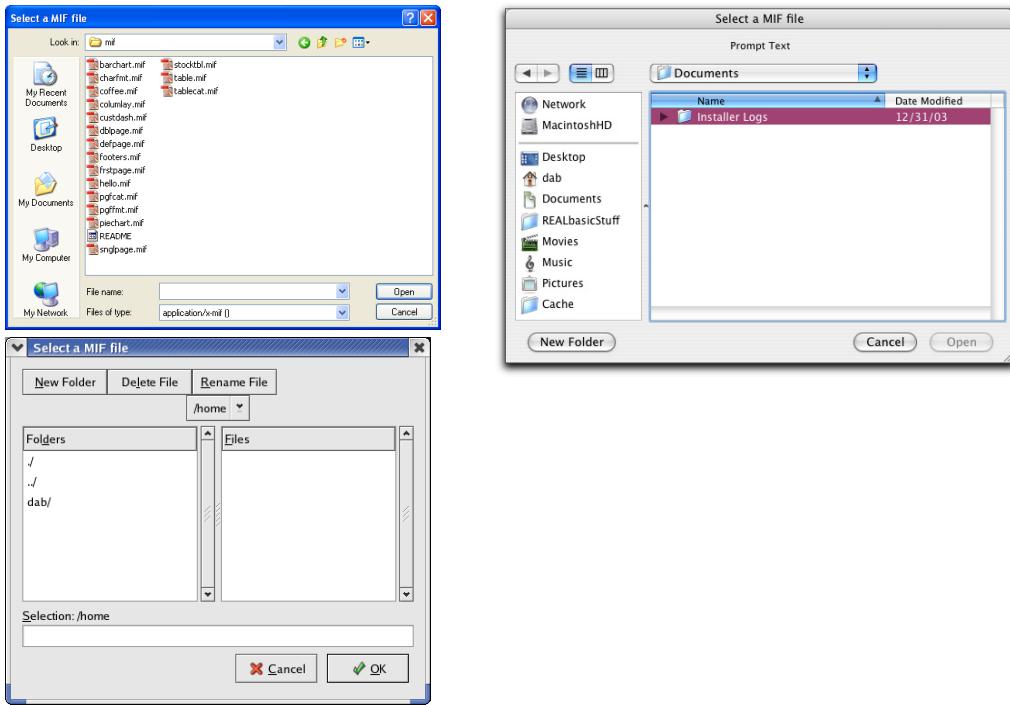
Using the properties of the [FolderItemDialog](#) class, you can customize the following aspects of an open-file dialog box:

## OpenDialog Class

- Position (Left and Top properties)
- Default directory (Initial Directory property)
- Valid file types to show (Filter property)
- Text of the Validate and Cancel buttons (ActionButtonCaption and CancelButtonCaption properties)
- Text that appears in the Title bar of the dialog (Title property)
- Text that appears inside the dialog (PromptText property)

### Example

This is what it looks like on Windows, Macintosh, and Linux:



The following code opens a customizable open-file dialog box and presents the Documents or Home directory in the file browser. Only MIF files are listed. The file

type passed to the Filter parameter was previously defined in a File Type Set called "TextTypes" in the File Type Sets Editor or via the [FileType](#) class.

```
Dim dlg as OpenDialog
Dim f as FolderItem
dlg=New OpenDialog
#If Not (TargetLinux) then
  dlg.InitialDirectory=Volume(0).Child("Documents")
#Else //open Home directory on linux
  dlg.InitialDirectory=Volume(0).Child("home")
#endif
dlg.Title="Select a MIF file"
dlg.Filter=TextTypes.mif
f=dlg.ShowModal()
If f <> Nil then
  //proceed normally
Else
  //User Cancelled
End if
```

**See Also**

[FileType](#), [FolderItem](#), [FolderItemDialog](#), [SaveAsDialog](#), [SelectFolderDialog](#) classes; [GetOpenFolderItem](#) function.

## OpenDBFCursor Function

Opens an xBase format file (i.e., a dbf file from dBase) as a [RecordSet](#).

**Syntax**

**result=OpenDBFCursor(file)**

| Part   | Type                       | Description   |
|--------|----------------------------|---|
| result | <a href="#">RecordSet</a>  | <a href="#">RecordSet</a> containing the rows and columns in the .dbf file. |
| file   | <a href="#">FolderItem</a> | <a href="#">FolderItem</a> that refers to the .dbf file.                    |

**Example**

```
Dim db as RecordSet
db=OpenDBFCursor(GetFolderItem("phone.dbf"))
If db <> Nil then
  //proceed with database operations
else
  Beep
  MsgBox "The database couldn't be opened."
end if
```

**Note** **OpenDBFCursor** requires the DBF Database plug-in. Place this plug-in in the Plugins folder. The REALbasic CD includes all database plug-ins and you can always find the most current versions on REAL Software's web site, [www.realsoftware.com](http://www.realsoftware.com).

**See Also** [Database](#), [DatabaseField](#), [DatabaseRecord](#), [RecordSet](#) classes; [OpenDTFDatabase](#), [OpenCSVCursor](#) functions.

---

## OpenDTFDatabase Function

Opens a dtF database.

**Syntax** **result=OpenDTFDatabase (dbFile,blobFile,username,password)**

| Part     | Type                       | Description                                      |
|----------|----------------------------|--|
| result   | <a href="#">Database</a>   | Database object that refers to the dtF database. |
| dbFile   | <a href="#">FolderItem</a> | All the fixed-length fields in the dtF file.     |
| blobFile | <a href="#">FolderItem</a> | All the blob fields in the dtF file.             |
| username | <a href="#">String</a>     | Username for the dtF database.                   |
| password | <a href="#">String</a>     | Password for the dtF database.                   |

**Notes** **OpenDTFDatabase** requires the dtfSQL Database plug-in. Place this plug-in in the REALbasic Plugins folder. The REALbasic CD includes all database plug-ins and you can always find the most current versions on REAL Software's web site, [www.realsoftware.com](http://www.realsoftware.com).

This function requires the Dtf\_PPC shared library be installed in extensions folder (Macintosh).

### Example

```
Dim db as Database
Dim textdata as FolderItem
Dim blobdata as FolderItem
textdata=GetFolderItem("mydb")
blobdata=GetFolderItem("myblobs")
db=OpenDTFdatabase(textdata, blobdata, "Nancy", "Pavlov")
If db.Connect then
    //continue with database operations
else
    Beep
    MsgBox "The database couldn't be opened."
end if
```

**See Also** [Database](#), [DatabaseField](#), [DatabaseRecord](#), [RecordSet](#) classes; [OpenDBFCursor](#) function.

## OpenPrinter Function

Returns a [Graphics](#) object in order to print to the currently selected printer.

**Syntax** **result=OpenPrinter([PageSetup])**

| Part      | Type                         | Description   |
|-----------|------------------------------|---|
| result    | <a href="#">Graphics</a>     | A <a href="#">graphics</a> object that represents the page to be printed.   |
| PageSetup | <a href="#">PrinterSetup</a> | Optional. A <a href="#">PrinterSetup</a> object whose properties will be used (like page orientation, scale, etc.) when printing. |

**Notes** The **OpenPrinter** function returns a [Graphics](#) object. The various drawing routines can then be used to draw into the [Graphics](#) object and will be sent to the printer for printing.

**Examples** This example prints “Hello World” to the currently selected printer.

```
Dim g As Graphics
g = OpenPrinter()
g.DrawString "Hello World", 100, 100
```

This example uses the Page Setup properties stored in the variable “ps” during printing:

```
Dim g As Graphics
g = OpenPrinter(ps)
g.DrawString "Hello World", 100, 100
```

**See Also** [Graphics](#), [PrinterSetup](#) classes; [OpenPrinterDialog](#) function.

## OpenPrinterDialog Function

Displays the standard Print dialog box and returns a [Graphics](#) object in order to print to the currently selected printer.

**Syntax** **result=OpenPrinterDialog([,PageSetup][,window])**

| Part   | Type                     | Description   |
|--------|--------------------------|---|
| result | <a href="#">Graphics</a> | A <a href="#">Graphics</a> object that represents the page to be printed. |

## OpenURLMovie Function

---

| Part      | Type                         | Description  |
|-----------|------------------------------|--|
| PageSetup | <a href="#">PrinterSetup</a> | Optional. A <a href="#">PrinterSetup</a> object whose properties will be used (like page orientation, scale, etc.) when printing.                                  |
| window    | <a href="#">Window</a>       | Optional. If you want to present the Print dialog as a Sheet window (Mac OS X only), <i>window</i> is the window to which you want to attach the Print dialog box. |

### Notes

The **OpenPrinterDialog** function returns a [Graphics](#) object. The various drawing routines can then be used to draw into the [Graphics](#) object and will be sent to the printer for printing. You can get the values in the Copies, From and To fields of the Print dialog box using the Copies, FirstPage and LastPage properties of the [Graphics](#) object returned by the **OpenPrinterDialog** function.

If the user clicks the Cancel button, the **OpenPrinterDialog** function returns [Nil](#).

Since both parameters are optional, there is the possibility that you will want to pass the second parameter but not the first. If you pass only one parameter, REALbasic will assume that it is the first parameter. If you want to pass only the second parameter, pass [Nil](#) as the value of the first parameter, for example:

```
Dim g As Graphics
g = OpenPrinterDialog(Nil, myWindow)
```

### Examples

This example prints “Hello World” to the currently selected printer.

```
Dim g As Graphics
g = OpenPrinterDialog()
If g<> Nil then
    g.DrawString "Hello World", 100, 100
End if
```

This example uses the Page Setup properties stored in the variable “ps” during printing:

```
Dim g As Graphics
g = OpenPrinterDialog(ps)
g.DrawString "Hello World", 100, 100
```

### See Also

[Graphics](#), [PrinterSetup](#), [StyledTextPrinter](#) classes; [OpenPrinter](#) function.

---

## OpenURLMovie Function

Opens a movie at the specified URL.

**Syntax****result=OpenURLMovie(URL)**

| Part   | Type                   | Description                                   |
|--------|------------------------|---|
| result | <a href="#">Movie</a>  | Movie object                                  |
| URL    | <a href="#">String</a> | URL that specifies the location of the movie. |

**Notes**

On Windows, the movie is played using the Windows Media Player, Macintosh uses QuickTime and, OpenURLMovie is not supported on Linux.

For QuickTime movies, if the user has QuickTime 4 (or above) installed, **OpenURLMovie** supports streaming video. QuickTime is not available on Linux.

**Example**

The following example loads a streaming video and assigns it to the Movie property of a [MoviePlayer](#) control:

```
MoviePlayer1.movie=OpenURLMovie\("http://www.apple.com/trailers/miramax/frida.html"\)
```

Note: Apple Computer, Inc. may remove this movie at their discretion. It is included here only to illustrate the function's syntax.

**See Also**

[MoviePlayer](#) control; [EditableMovie](#), [Movie](#) classes.

## Operator\_Add Function

Used to overload the [+](#) operator, providing custom functionality.

**Notes**

Create an **Operator\_Add** function in a class to specify the functionality of the [+](#) operator for that class.

In the function, the [Self](#) instance is one of the operands and the other operand is passed as a parameter. **Operator\_Add** assumes the [Self](#) instance is on the left and the passed instance is on the right.

**Example**

Suppose you have a class, Vector, that can store a vector of modest size. To define the [+](#) operator for this class, create a method of the Vector class called **Operator\_Add**.

The Vector class has two [integer](#) properties, x and y.

## Operator\_Add Function

---

**Operator\_Add** is defined as:

```
Function Operator_Add (rhs as Vector) as Vector
//rhs stands for right-hand-side, the vector to be added to Self
Dim ret as New Vector  the sum of the two vectors
ret.x=Self.x + rhs.x
ret.y=Self.y + rhs.y

Return ret
```

**See Also** [± operator](#), [Operator\\_AddRight](#) function.

## Operator\_AddRight Function

Used to overload the [±](#) operator, providing custom functionality. **Operator\_AddRight** is the same as [Operator\\_Add](#) but the implicit [Self](#) instance is on the right instead of the left in the addition.

**Notes** Create an **Operator\_AddRight** function in a class to specify the functionality of the [±](#) operator for that class and the [Self](#) instance is on the right.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

**Example** Suppose you have a class, Vector, that can store a vector of modest size. To define the [±](#) operator for this class, create a method of the Vector class called **Operator\_AddRight**.

The Vector class has two [integer](#) properties, x and y.

**Operator\_AddRight** is defined as:

```
Function Operator_AddRight (lhs as Vector) as Vector
//lhs stands for left-hand-side, the vector to be added to Self
Dim ret as New Vector  the sum of the two vectors
ret.x=lhs.x + Self.x
ret.y=lhs.y + Self.y

Return ret
```

**See Also** [± operator](#), [Operator\\_Add](#) function.

## Operator\_And Function

Used to overload the [And](#) function, providing custom functionality.

**Notes**

Create an **Operator\_And** function in a class to specify the functionality of the [And](#) operator for that class. Whenever you use the [And](#) operator in your code to operate on two instances of the class, your **Operator\_And** function will be called.

In the function, the [Self](#) instance is one of the operands and the other operand is passed as a parameter. [Self](#) is assumed to be on the left.

**Example**

Suppose you define a class myResult with the property a as [Boolean](#). **Operator\_And** is defined as:

```
Function Operator_And(rhs as MyResult) as Boolean
Return Self.a And rhs.a
```

**See Also**

[And](#) operator, [Operator\\_AndRight](#) function.

## Operator\_AndRight Function

Used to overload the [And](#) function, providing custom functionality.

**Notes**

Create an **Operator\_AndRight** function in a class to specify the functionality of the [And](#) operator for that class. **Operator\_AndRight** is the same as [Operator\\_And](#) except that the [Self](#) instance is on the right.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

**Example**

Suppose you define a class myResult with the property a as [Boolean](#). **Operator\_AndRight** is defined as:

```
Function Operator_AndRight(lhs as MyResult) as Boolean
Return lhs.a And Self.a
```

**See Also**

[And](#) operator, [Operator\\_And](#) function.

## Operator\_Compare Function

Used to overload the intrinsic comparison operators, providing custom functionality.

**Notes**

If **Operator\_Compare** is defined, it is called whenever any of the comparison operators are used to compare the [Self](#) instance of the class with another instance. The following comparison operators are covered: [=](#), [<](#), [>](#), [≤](#), [≥](#), [<=](#).

## Operator\_Convert Method

---

**Operator\_Compare** returns an [integer](#) whose meaning is as follows: <0 means that [Self](#) is less than the passed parameter, 0 means that [Self](#) is equal to the passed parameter, and >0 means that [Self](#) is greater than the passed parameter.

### Example

This example defines a comparison between two Vectors which have been declared with [integer](#) properties x and y (See [Operator\\_Add](#)). The function compares the square lengths of the vectors and then reports on which one is longer.

```
Function Operator_Compare(rhs as Vector) as Integer
Dim a, b as Integer
a=Self.x^2 + Self.y^2
b=rhs.x^2 + rhs.y^2

If a>b then Return 1
If a=b then Return 0
If a<b then Return -1
```

### See Also

[=](#), [≤](#), [≥](#), [<](#), [≥](#), [<](#) operators.

## Operator\_Convert Method

Used to convert from one type to another, providing custom functionality.

### Notes

The following example converts to a Vector from an Integer (The Vector class is defined in [Operator\\_Add](#).) The **Operator\_Convert** method is added to the Vector class.

```
Sub Operator_Convert(rhs as Operator)
Self.x=rhs
Self.y=rhs
```

You use this definition of the **Operator\_Convert** method like this:

```
Dim v as Vector
v = 10 //both elements of V get the value 10
```

Note that the [New](#) operator is not needed here. This is because the assignment operator is returning a new instance of Vector. If you were to write `v=New Vector`, it would create one instance of Vector but that would be overwritten by the assignment statement.

The second type of conversion is a ‘to’ conversion. Use this when you want to convert the class to another type. In the following example, the Vector is converted to a string that represents its square length.

```
Function Operator_Convert as String
  Return Str(Self.x^2 + Self.y^2)
```

After defining both forms of **Operator\_Convert**, you can use them with code like this:

```
Dim v as Vector =10
MsgBox v
```

The first line invokes the data type conversion in the first example and assigns 10 to both elements of the Vector object. The second line computes the square length and returns it as a String.

## Operator\_Divide Function

Used to overload the / operator, providing custom functionality.

### Notes

Create an **Operator\_Divide** function in a class to specify the functionality of the / operator for that class.

In the function, the Self instance the left operands and the other operand is passed as a parameter.

### Example

Using the Vector class (see [Operator\\_Add](#)) with two Integer elements, x and y, we define an **Operator\_Divide** function that divides the sum of the Self instance with the sum of the passed instance.

```
Function Operator_Divide(rhs as Vector) as Double
  Dim a, b as Integer
  a=Self.x + Self.y
  b=rhs.x + rhs.y
  Return a / b
```

### See Also

/ operator; [Operator\\_DivideRight](#) function.

## Operator\_DivideRight Function

Used to overload the `/` operator, providing custom functionality. **Operator\_DivideRight** is the same as [Operator\\_Divide](#), except that the intrinsic `Self` is assumed to be the one on the right in the division operation.

### Notes

Create an **Operator\_DivideRight** function in a class to specify the functionality of the `/` operator for that class.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

### Example

Using the Vector class (see [Operator\\_Add](#)) with two `Integer` elements, `x` and `y`, we define an **Operator\_DivideRight** function that divides the sum of the `Self` instance with the passed instance.

```
Function Operator_DivideRight(lhs as Vector) as Double
Dim a, b as Integer
a=Self.x + Self.y
b=lhs.x + lhs.y

Return b / a
```

### See Also

`/` operator; [Operator\\_Divide](#) function.

---

## Operator\_IntegerDivide Function

Used to overload the `\` operator, providing custom functionality.

### Notes

Create an **Operator\_IntegerDivide** function in a class to specify the functionality of the `\` operator for that class.

In the function, the `Self` instance is the left operand and the other operand is passed as a parameter. Whenever you use the `\` operator in your code to operate on two instances of the class and the `Self` instance is on the left, your **Operator\_IntegerDivide** function will be called.

**Example**

Using the Vector class (see [Operator\\_Add](#)) with two [Integer](#) elements, x and y, we define an **Operator\_IntegerDivide** function that divides the sum of the [Self](#) instance with the passed instance.

```
Function Operator_IntegerDivide(rhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=rhs.x + rhs.y

Return a \ b
```

**See Also**

\ operator; [Operator\\_IntegerDivideRight](#) function.

## Operator\_IntegerDivideRight Function

Used to overload the \ operator, providing custom functionality. **Operator\_IntegerDivideRight** is the same as [Operator\\_IntegerDivide](#), except that the intrinsic [Self](#) is assumed to be the one on the right in the division operation

**Notes**

Create an **Operator\_IntegerDivideRight** function in a class to specify the functionality of the \ operator for that class.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

**Example**

Using the Vector class (see [Operator\\_Add](#)) with two [Integer](#) elements, x and y, we define an **Operator\_IntegerDivideRight** function that divides the sum of the passed instance by the sum of the [Self](#) instance.

```
Function Operator_DivideRight(lhs as Vector) as Double
Dim a, b as Integer
a=Self.x + Self.y
b=lhs.x + lhs.y

Return b \ a
```

**See Also**

\ operator; [Operator\\_IntegerDivide](#) function.

## Operator\_Lookup Function

Used to specify a lookup value for an otherwise undefined property of a class.

## Operator\_Modulo Function

---

**Notes** Use **Operator\_Lookup** to define a function that will be called if the called item cannot otherwise be found in the class.

**Operator\_Lookup** overloads a class's lookup operator. The method's first parameter must be a [String](#) which will receive the name of the desired member. The method can have any combination of other parameters and can have any return type. The **Operator\_Lookup** operator is called whenever you use the *foo.bar* syntax to look up a member of an object and the compiler can't find anything named "bar". Before it returns an Undefined Identifier error, the compiler will try to call a lookup operator and pass "bar" as the first parameter. Overloading works for the lookup operator just as it does for any other method.

Please be advised that this can be a dangerous thing to do because you may, for example, inadvertently call the **Operator\_Lookup** function with a misspelling and you don't get an error message.

**Example** This example defines "SquareLength" in the Vector class (see [Operator\\_Add](#)), a class with two [integer](#) properties, x and y.

```
Function Operator_Lookup (Name as String) as Double
  If Name = "SquareLength" then Return Self.x^2 + Self.y^2
```

When you write:

```
Dim myDouble as Double
myDouble=myVector.SquareLength
```

it will return the result of the calculation. However, you can also write:

```
myDouble=myVector.UndefinedFunction //does not exist
```

and it will compile, returning zero. This can lead to difficult to track down errors.

---

## Operator\_Modulo Function

Used to overload the [Mod](#) function, providing custom functionality.

**Notes** Create an **Operator\_Modulo** function in a class to specify the functionality of the [Mod](#) operator for that class.

In the function, the [Self](#) instance the left operand and the other operand is passed as a parameter.

- Example** Using the Vector class (see [Operator\\_Add](#)) with two [Integer](#) elements, x and y, we define an **Operator\_Modulo** function that returns the remainder of the division of the sum of the [Self](#) instance by the passed instance.

```
Function Operator_Modulo (rhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=rhs.x + rhs.y

Return a Mod b
```

- See Also** [Mod](#) operator; [Operator\\_ModuloRight](#) function.

## Operator\_ModuloRight Function

Used to overload the [Mod](#) function, providing custom functionality.

- Notes** Create an **Operator\_ModuloRight** function in a class to specify the functionality of the [Mod](#) operator for that class. The instance of [Self](#) is on the right side of the [Mod](#) expression. The other instance is passed as a parameter.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

- Example** Using the Vector class (see [Operator\\_Add](#)) with two [Integer](#) elements, x and y, we define an **Operator\_ModuloRight** function that returns the remainder of the division of the sum of the Self instance by the passed instance.

```
Function Operator_ModuloRight (lhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=lhs.x + lhs.y

Return b Mod a
```

- See Also** [Mod](#) operator; [Operator\\_Modulo](#) function.

## Operator\_Multiply Function

Used to overload the [\\*](#) operator, providing custom functionality.

- Notes** Create an **Operator\_Multiply** function in a class to specify the functionality of the [\\*](#) operator for that class.

## Operator\_Multiply Function

---

In the function, the [Self](#) instance is the left operand and the other operand is passed as a parameter.

### Example

Using the Vector class (see [Operator\\_Add](#)) with two [Integer](#) elements, x and y, we define an **Operator\_Multiply** function that multiplies the sum of the [Self](#) instance with the passed instance.

```
Function Operator_Multiply(rhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=rhs.x + rhs.y

Return a*b
```

### See Also

[\\*](#) operator; [Operator\\_MultiplyRight](#) function.

## Operator\_MultiplyRight Function

Used to overload the [\\*](#) operator, providing custom functionality.

### Notes

Create an **Operator\_Multiply** function in a class to specify the functionality of the [\\*](#) operator for that class.

In the function, the [Self](#) instance is the operand on the right and the other operand is passed as a parameter.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

### Example

Using the Vector class (see [Operator\\_Add](#)) with two [Integer](#) elements, x and y, we define an **Operator\_MultiplyRight** function that multiplies the sum of the passed instance with the [Self](#) instance.

```
Function Operator_MultiplyRight(lhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=lhs.x + lhs.y

Return b*a
```

### See Also

[\\*](#) operator; [Operator\\_Multiply](#) function.

## Operator\_Negate Function

Used to overload the `_` operator when used for negation, providing custom functionality.

### Notes

Create an **Operator\_Negate** function in a class to specify the functionality of the `_` operator for that class.

In the function, the definition of negation is given for the `Self` instance only. No parameters are passed.

### Example

In this example, the negation of the Vector class (see [Operator\\_Add](#)), a class with two `Integer` properties, `x` and `y`, is defined as the negative of the square length of the vector.

```
Function Operator_Negate as Integer
Dim a as Integer
a=Self.x^2 + Self.y^2
Return -a
```

### See Also

`_` operator.

## Operator\_Not Function

### Notes

Create an **Operator\_Not** function in a class to specify the functionality of the `Not` operator for that class. Whenever you use the `Not` operator in your code to operate on two instances of the class, your **Operator\_Not** function will be called.

In the function, the `Self` instance is the only operand. No parameters are passed.

### Example

Suppose you define a class `myResult` with the property `a` as `Boolean`. **Operator\_Not** is defined as:

```
Function Operator_Not as Boolean
Return Not Self.a
```

### See Also

`Not` operator.

## Operator\_Or Function

Used to overload the `Or` operator, providing custom functionality.

## Operator\_Or Function

---

|              |   |
|--------------|---|
| <b>Notes</b> | Create an <b>Operator_Or</b> function in a class to specify the functionality of the <a href="#">Or</a> operator for that class. Whenever you use the <a href="#">Or</a> operator in your code to operate on two instances of the class, your <b>Operator_Or</b> function will be called.<br><br>In the function, the <a href="#">Self</a> instance is the left operand and the other operand is passed as a parameter. |
|--------------|---|

|                |  |
|----------------|--|
| <b>Example</b> | Suppose you define a class myResult with the property a as <a href="#">Boolean</a> . <b>Operator_Or</b> is defined as: |
|----------------|--|

```
Function Operator_Or(rhs as MyResult) as Boolean  
Return Self.a Or rhs.a
```

|                 |   |
|-----------------|---|
| <b>See Also</b> | <a href="#">Or</a> operator, <a href="#">Operator_OrRight</a> function. |
|-----------------|---|

## Operator\_OrRight Function

---

Used to overload the [Or](#) operator, providing custom functionality.

|              |   |
|--------------|---|
| <b>Notes</b> | Create an <b>Operator_OrRight</b> function in a class to specify the functionality of the <a href="#">Or</a> operator for that class. <b>Operator_OrRight</b> assumes that the <a href="#">Self</a> instance of the class is on the right and the passed instance is on the left.<br><br>In the function, the <a href="#">Self</a> instance the right operand and the other operand is passed as a parameter.<br><br>The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version. |
|--------------|---|

|                |   |
|----------------|---|
| <b>Example</b> | Suppose you define a class myResult with the property a as <a href="#">Boolean</a> . <b>Operator_OrRight</b> is defined as: |
|----------------|---|

```
Function Operator_OrRight(lhs as MyResult) as Boolean  
Return lhs.a Or Self.a
```

|                 |  |
|-----------------|--|
| <b>See Also</b> | <a href="#">Or</a> operator, <a href="#">Operator_Or</a> function. |
|-----------------|--|

## Operator\_Power Function

---

Used to overload the [^](#) operator, providing custom functionality.

**Notes**

Create an **Operator\_Power** function in a class to specify the functionality of the  $\wedge$  operator for that class. When you use the  $\wedge$  operator in your code to operate on two instances of the class, your **Operator\_Power** function will be called.

In the function, the **Self** instance the left operand and the other operand is passed as a parameter. The **Self** instance is raised to the power specified by the passed instance.

**Example**

Using the Vector class (see [Operator\\_Add](#)) with two **Integer** elements, x and y, we define an **Operator\_Power** function that raises the sum of the **Self** instance to the power of the passed instance.

```
Function Operator_Power(rhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=rhs.x + rhs.y

Return a  $\wedge$ b
```

**See Also**

$\wedge$  operator; [Operator\\_PowerRight](#) function.

## Operator\_PowerRight Function

Used to overload the  $\wedge$  operator, providing custom functionality.

**Notes**

Create an **Operator\_PowerRight** function in a class to specify the functionality of the  $\wedge$  operator for that class when the **Self** instance is on the right.

In the function, the other operand is passed as a parameter and is raised to the power specified by the **Self** instance.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

**Example**

Using the Vector class (see [Operator\\_Add](#)) with two **Integer** elements, x and y, we define an **Operator\_Power** function that raises the sum of the **Self** instance to the power of the passed instance.

```
Function Operator_PowerRight(lhs as Vector) as Integer
Dim a, b as Integer
a=Self.x + Self.y
b=lhs.x + lhs.y

Return b  $\wedge$ a
```

**See Also**

$\wedge$  operator; [Operator\\_Power](#) function.

## **Operator\_Subtract Function**

Used to overload the `_` operator when used for subtraction, providing custom functionality.

### **Notes**

Create an **Operator\_Subtract** function in a class to specify the functionality of the `_` operator for that class.

In the function, the `Self` instance is the left operand and the other operand is passed as a parameter.

### **Example**

Suppose you have a class, Vector, that can store a vector. To define the `_` operator for this class, create a method of the Vector class called **Operator\_Subtract**.

The Vector class has two `integer` properties, `x` and `y`.

**Operator\_Subtract** is defined as:

```
Function Operator_Subtract (rhs as Vector) as Vector
//rhs stands for right-hand-side, the vector to be added to Self
Dim ret as New Vector //the differenct between the two vectors
ret.x=_Self.x - rhs.x
ret.y=_Self.y - rhs.y

Return ret
```

### **See Also**

`_` operator, [Operator\\_SubtractRight](#) function.

---

## **Operator\_SubtractRight Function**

Used to overload the `_` operator, providing custom functionality.

**Operator\_SubtractRight** is the same as [Operator\\_Subtract](#) but the implicit `Self` instance is on the right instead of the left in the subtraction.

### **Notes**

Create an **Operator\_SubtractRight** function in a class to specify the functionality of the `+` operator for that class.

The ordinary methods are always preferred; REALbasic uses the Right versions only if there is no legal left version.

### **Example**

Suppose you have a class, Vector, that can store a vector. To define the `_` operator for this class, create a method of the Vector class called **Operator\_SubtractRight**.

The Vector class has two `integer` properties, `x` and `y`.

**Operator\_SubtractRight** is defined as:

```
Function Operator_SubtractRight (lhs as Vector) as Vector
  //lhs stands for left-hand-side, the vector to be added to Self
  Dim ret as New Vector //the difference between the two vectors
  ret.x=lhs.x + Self.x
  ret.y=lhs.y + Self.y

  Return ret
```

**See Also** [-](#) operator, [Operator\\_Subtract](#) function.

## Optional Keyword

Used in parameter declarations to indicate that a parameter is optional.

### Syntax *Optional parameter*

| Part      | Type          | Description                 |
|-----------|---------------|-----------------------------|
| parameter | Any Data Type | Parameter that is optional. |

**Notes** When you use the **Optional** keyword in a parameter declaration, you make the parameter optional without specifying a default value. The other way of specifying an optional parameter is to give it a default value in the declaration. If you omit the third parameter when you call such a method, the omitted parameter gets the default value you specified in the declaration.

The **Optional** keyword precedes the parameter name in the declaration. If the calling statement omits this parameter, it will receive the standard initial value for its type.

**Example** If the method declaration is:

```
myMethod (a as Integer, b as Integer, Optional c as Integer)
```

The call can omit the third parameter and the last parameter, c, will get the default value of 0.

The other way of making the parameter optional is to give it a default value in the declaration, such as:

```
myMethod (a as Integer, b as Integer, c as Integer = 10)
```

## Or Operator

Used to perform a logical comparison between two [Boolean](#) expressions.

### Syntax

**result=expression1 Or expression2**

| Part        | Description                                   |
|-------------|---|
| result      | A <a href="#">Boolean</a> value.              |
| expression1 | Any valid <a href="#">Boolean</a> expression. |
| expression2 | Any valid <a href="#">Boolean</a> expression. |

### Notes

If either expression evaluates to [True](#), then *result* is [True](#). If both expressions evaluate to [False](#) then *result* is [False](#).

### Example

This example uses the **Or** operator to perform logical comparisons.

```
Dim a As Boolean
Dim b As Boolean
Dim c As Boolean
Dim d As Boolean
a=True
b=False
d=a Or b //Returns True
d=b Or c //Returns False
```

### See Also

[And](#), [Not](#) operators; [Operator\\_Or](#) function.

---

## OracleDatabase Class

Use with the Oracle plug-in to use an Oracle database as your data source. Oracle 8i and above are supported. The Oracle plug-in must be in your plugins folder. The Oracle OCS client 9i or later must be installed.

### Super Class

[Database](#)

### Properties

| Name  | Type                    | Description                    |
|-------|-------------------------|--------------------------------|
| Debug | <a href="#">Integer</a> | Set to 1 to turn on debugging. |

**Example** The following example opens an existing Oracle database.

```
Dim db as OracleDatabase
'open a database connection
db = New OracleDatabase
db.DatabaseName = "ALEXIS"
db.UserName = "scott"
db.Password = "tiger"
db.debug = 1 'look in the console
If Not db.connect() then
  MsgBox db.ErrorMessage
  Return
End if
```

**See Also** [Database](#), [DatabaseRecord](#), [RecordSet](#) classes.

## OSType Data Type

A 4-byte integer known as a “four char code,” typically used only on Macintosh or when declaring into QuickTime. If you pass a REALbasic string of four characters via this data type, it is automatically converted into a four char code [Integer](#).

This data type can be used only with the [Declare](#) statement and cannot be used to declare REALbasic variables, properties, and methods.

**See Also** [Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

## OutOfBoundsException Error

Occurs when code tries to access an array element that doesn’t exist.

**Super Class** [RuntimeException](#) class

**Notes** An **OutOfBoundsException** error occurs when your code uses an array index that is out of range. This occurs, for example, if you forget that arrays in REALbasic are zero-based and you make an error when you try to access the last element.

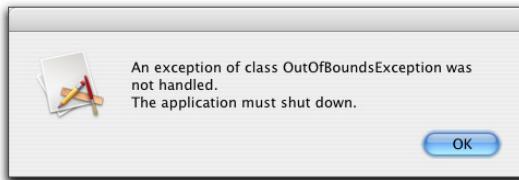
**Examples** This example is supposed to display the fonts installed on the user’s computer in a [ListBox](#). It generates an **OutOfBoundsException** because the loop should be from 0 to

## OutOfBoundsException Error

nFonts-1 rather than from 1 to nFonts. When the loop gets to nFonts, the value of i is out of range and the error occurs.

```
Dim i, nFonts as Integer
nFonts=FontCount
For i=1 to nFonts //don't do this!
    ListBox1.addrow(Font(i))
Next
```

If you run this code in a built application, you will see a generic error message when i reaches the value of nFonts:



The application quits when the user accepts this dialog.

You can handle the exception by adding the following code:

```
Dim i, nFonts as Integer
nFonts=FontCount
For i=1 to nFonts //don't do this!
    ListBox1.addrow(Font(i))
Next
Exception err
If err IsA OutOfBoundsException then
    MsgBox "The value: "+Str(i)+" is out of range!"
end if
```

When this code runs in a built application, the user sees the following dialog box.



The application does not quit when the user accepts the dialog box.

### See Also

[RuntimeException](#) class; [Function](#), [Raise](#), [Sub](#) statements; [Nil](#) keyword, [Exception](#), [Try](#) blocks.

# OutOfMemoryException Error

Raised in certain cases when you run out of memory.

**SuperClass** [RuntimeException](#) class.

**Notes** The **OutOfMemoryException** error may be raised when an operation cannot be completed due to insufficient memory. It is difficult to say when this may occur, especially since all operating systems supported by REALbasic except Mac OS “classic” use dynamic memory allocation. One possibility is when you are attempting to load a large graphics file into memory.

**Example** This example uses the [Catch](#) statement in a window’s Open event handler to handle out of memory exceptions when trying to draw an imported gif image. The variable myPicture is a global property of type [Picture](#). The picture “Logo” has been added to the Project Editor.

```
Sub Open
Try
myPicture=New Picture(Logo.width,Logo.height,32)
myPicture.Graphics.DrawPicture Logo,0,0

Catch err as OutOfMemoryException
MsgBox "Insufficient memory to draw the picture!"
Quit
```

**See Also** [RuntimeException](#) class; [Function](#), [Raise](#) statements; [Nil](#) keyword; [Exception](#), [Try](#) blocks.

## Oval Control

Draws an oval.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name        | Type                    | Description                               |
|-------------|-------------------------|---|
| BorderColor | <a href="#">Color</a>   | The color of the oval’s border.           |
| BorderWidth | <a href="#">Integer</a> | The width of the oval’s border in pixels. |

## OvalShape Class

---

| Name      | Type                  | Description                     |
|-----------|-----------------------|---------------------------------|
| FillColor | <a href="#">Color</a> | The interior color of the oval. |

### Events

| Name      | Parameters   | Description  |
|-----------|--|--|
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Oval region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown.  |
| MouseDrag | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Oval and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the Oval region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.                                  |

**See Also** [RectControl](#) class.

---

## OvalShape Class

Used for drawing (two-dimensional) ellipses in a vector graphics environment.

**Super Class** [RectShape](#)

### Properties

| Name     | Type                    | Description  |
|----------|-------------------------|--|
| Segments | <a href="#">Integer</a> | The number of polygon sides to use when approximation is needed. Zero means it's up to the renderer. |

**Example** Place the following method in the MouseDown event handler of a [window](#). It will draw an oval in the window when the user presses the mouse button.

```
Dim o as New OvalShape
o.width=60
o.height=30
o.Fillcolor=RGB(127,127,255)

Graphics.DrawObject o,x,y
```

**See Also** [ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Group2D](#), [Graphics](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

# PagePanel Control

Similar to the [TabPanel](#) control, but does not include tabs (or any other visual indicator) for navigating to different pages. The [TabPanel](#) is subclassed from the **PagePanel**.

**SuperClass** [RectControl](#)

## Properties

| Name              | Type                    | Description   |
|-------------------|-------------------------|---|
| <b>PanelCount</b> | <a href="#">Integer</a> | Gets the number of panels in the PagePanel.   |
| <b>Value</b>      | <a href="#">Integer</a> | The number of the currently selected page. The first page is numbered zero. To change the page that is displayed, set this value. This is also how you change the displayed panel for the <a href="#">TabPanel</a> control, which is subclassed from <b>PagePanel</b> . Use the <a href="#">Control's</a> PanelIndex property to get or set any <a href="#">RectControl</a> 's panel in either a <b>PagePanel</b> or <a href="#">TabPanel</a> . |

## Events

| Name          | Parameters | Description                     |
|---------------|------------|---------------------------------|
| <b>Change</b> |            | The frontmost page has changed. |

## Methods

| Name          | Parameters                       | Description  |
|---------------|----------------------------------|--|
| <b>Append</b> |                                  | Appends a new page after the last page.  |
| <b>Insert</b> | Index as <a href="#">Integer</a> | Inserts a new page at the specified position. The first page is numbered zero.   |
| <b>Remove</b> | Index as <a href="#">Integer</a> | Removes the page specified by <i>Index</i> . The first page is numbered zero. It will also remove all of the controls on the page. |

## Notes

The **PagePanel** control allows you to add controls to pages, just as you add controls to tab panels in a [TabPanel](#). The difference is that the **PagePanel** control itself is not visible in your built application—only the controls on the currently selected page are visible. You change the displayed page by changing the value of the Value property. PagePanels cannot be embedded inside PagePanels. The following line of code moves a ListBox from its current panel to the second panel of a **PagePanel**.

```
ListBox1.PanelIndex=1
```

You get or set the page on which a control is located via the PanelIndex property of the [Control](#) class. The first page is numbered zero.

In the IDE, you can add, insert, delete, reorder and navigate from page to page, just like a [TabPanel](#). When building an interface using the **PagePanel**, you can take advantage of the Control Hierarchy, discussed in the section on the [RectControl](#) object.

**See Also** [TabPanel](#) control; [RectControl](#) class.

## Paragraph Class

A paragraph of [StyledText](#).

**Super Class** [Object](#)

### Properties

| Name      | Type                    | Description   |
|-----------|-------------------------|---|
| Alignment | <a href="#">Integer</a> | Gets the text alignment of the paragraph. You can use either the integer values or the class constants, AlignDefault, AlignLeft, AlignCenter, or AlignRight, to set or get the alignment:<br>AlignDefault (0): Default alignment<br>AlignLeft (1): Left aligned<br>AlignCenter (2): Centered<br>AlignRight (3): Right aligned<br>To set the alignment of a paragraph, call the ParagraphAlignment method of the <a href="#">StyledText</a> class. You can use <b>Paragraph</b> or the <a href="#">EditField</a> alignment constants to set a paragraph alignment using the ParagraphAlignment method. |
| EndPos    | <a href="#">Integer</a> | End position of the paragraph.  |
| Length    | <a href="#">Integer</a> | Length (characters) of the paragraph.   |
| StartPos  | <a href="#">Integer</a> | Starting position of the paragraph.   |

### Notes

Use the **Paragraph** class to refer to a paragraph of text within a block of [StyledText](#). A paragraph is defined as all the text between two paragraph separators. A separator can be the [EndOfLine](#) function or the correct separator character for the platform on which the application is running. Completely empty paragraphs (e.g., blank lines) do not count as paragraphs.

You use the ParagraphCount method of the [StyledText](#) class to get the number of paragraphs in the [StyledText](#) object and the Paragraph method of the [StyledText](#) class to refer to a particular paragraph.

The **Paragraph** has one style attribute, Alignment. It gets the current alignment for the paragraph. The alignment is set by calling the ParagraphAlignment method of the [StyledText](#) class.

**Example**

The following example sets a paragraph's alignment to Centered. It then gets the starting position and length of the paragraph via the **Paragraph** class and applies several style attributes to the whole paragraph. The [EditField](#) that displays the styled text must have the MultiLine and Styled properties set to [True](#).

```
Dim Text as String
Dim st,ln as Integer

//define four paragraphs in Text
Text = "This is the text that we are going to save " _ 
    +"into our file from the EditField." +EndofLine _ 
    +"Isn't that interesting?" +EndofLine _ 
    +"Man, I sure do love using RB to take care of my projects." +EndofLine _ 
    +"That's because the RS Engineering staff is just so awesome!" 
EditField1.StyledText.Text = text //four paragraphs in Text

//center the second paragraph
EditField1.StyledText.ParagraphAlignment(1)=Paragraph.AlignCenter

//set the second paragraph in Helvetica, 18 pt bold, red
//first get its character positions...
st=EditField1.StyledText.Paragraph(1).StartPos-1
ln=EditField1.StyledText.Paragraph(1).Length

//next apply attributes...
EditField1.StyledText.Bold(st,ln)=True
EditField1.StyledText.Font(st,ln)="Helvetica"
EditField1.StyledText.Size(st,ln)=18
EditField1.StyledText.TextColor(st,ln)=&cFF0000 //red
```

**See Also**

[EditField](#), [Range](#), [StyledText](#), [StyleRun](#) classes.

---

## ParamArray Keyword

Used in a [Sub](#) or [Function](#) statement to indicate that an arbitrary number of parameters of the specified data type can be passed.

**Syntax**

**ParamArray [parameterName As DataType]**

| Part          | Description   |
|---------------|---|
| parameterName | Name of parameter for which an indefinite number of arguments will be passed. |
| DataType      | Data type of the parameter. It can be any valid data type.                    |

## ParamArray Keyword

---

### Notes

The **ParamArray** keyword enables you to pass an indefinite number of values of a specific data type without formally using an array. A call to a method or function that has been declared using **ParamArray** uses a comma-delimited list of values rather than an array.

Normally you would pass a parameter to a method defined using **ParamArray** but the parameter is optional.

### Example

The following function adds a list of numbers that are passed via **ParamArray**.

```
Function AddNumbers(ParamArray Nums as Integer) As Integer
Dim i,Total as Integer
For Each i in Nums
    total=total+i
Next
Return Total
```

You can call this function with the following code:

```
Dim n as Integer
n=AddNumbers(5,10,20)
```

Any number of integers can be passed to AddNumbers.

This approach is equivalent to the use of an array. Consider the following function:

```
Function AddArray(Nums() as Integer) As Integer
Dim i, total as Integer
For Each i in Nums
    total=total+i
Next
Return total
```

You can initialize the array and call the function in the following way:

```
Dim myArray(-1) as Integer //declare array without specifying size
Dim n as Integer
myArray=Array(5,10,20) //create 3 elements and assign values
n=AddArray(myArray)
```

### See Also

[Array](#) function; [Dim](#), [Function](#), [Sub](#) statements.

## Parameter and Default Value Must be of the Same Type Error

You used a value of an incorrect data type as a default value. When you create a method in the Add Method declaration area, you can optionally set a default value for each of the parameters. Of course, the data type of the default value must match the data type of the parameter.

**Example** Consider the following parameter declaration:

```
s as String = 15
```

Since s is declared as a [String](#), its default value must be a [String](#). 15 isn't.

---

## Parameters are not Compatible with this Function Error

You passed parameters of an incorrect data type to a function.

**Example** The [RGB](#) function expects integer values:

```
Dim c as Color
c=RGB ("red","green","blue")
```

---

## ParseDate Function

Parses a date passed as a [string](#) and creates a [Date](#) object.

### Syntax

**result=ParseDate(Text, ByRef ParsedDate)**

| Part       | Type                    | Description   |
|------------|-------------------------|---|
| result     | <a href="#">Boolean</a> | Returns <a href="#">True</a> if Text was successfully parsed.                     |
| Text       | <a href="#">String</a>  | The <a href="#">date</a> string to be parsed.                                     |
| ParsedDate | <a href="#">Date</a>    | After successful call, contains the parsed date as a <a href="#">Date</a> object. |

### Notes

The Text parameter must be a valid [string](#) representation of a [date](#). You can use the slash, the dash, the period, or the space to separate the Month, Day, and Year. For example, the following are valid [string](#) representations: 12/14/1341, 12-14-1341, 12.14.1341, 14Dec1341, 14.Dec.1341, and 14 Dec. 1341.

## Permissions Class

---

**ParseDate** honors the Date Format settings of the user's computer. For example, the string "12/11/2004" will be parsed as the eleventh of December in the US. If the computer is configured for the British date format, it will be parsed as the twelfth of November.

**ParseDate** will parse dates based on the user's locale even if the user's locale is a Unicode-only locale.

If you use only the last two digits of the year, the current century is assumed.

**ParseDate** does not parse the Time value of a [date](#), if present.

### Example

The following code displays the parsed [date](#) in a message box in the Abbreviated Date format. You can use any [Date](#) property to return corresponding values, e.g., the Day, Month, and Year properties hold the day, month, and year of the passed [date](#).

```
Dim theDate as New Date
Dim theTrueBool as Boolean
thetruebool=ParseDate("14Dec1341",theDate)
If theTrueBool then
  MsgBox thedate.AbbreviatedDate
else
  MsgBox "Invalid Date format!"
end if
```

### See Also

[Date](#) class, [Database](#) class examples.

---

## Permissions Class

Enables you to test, set, and clear the permissions of [FolderItems](#) in operating systems that support this concept (Mac OS X and Linux). You can get and set the permissions of a [FolderItem](#) as an octal [Integer](#) via the Permissions property of the [FolderItem](#) class.

**Super Class** [Object](#)

**Constructor** You must pass the [Integer](#) permissions value of the [FolderItem](#) in the Permissions constructor. For example:

```
Dim perms as New Permissions(FolderItem1.Permissions)
```

### Properties

| Name   | Type                    | Description   |
|--------|-------------------------|---|
| GIDBit | <a href="#">Boolean</a> | Set Group ID bit. If <a href="#">True</a> , the current user has the permissions of the owning Group without altering the Read/Write/Execute permissions for the <a href="#">FolderItem</a> . |

| Name         | Type    | Description  |
|--------------|---------|--|
| GroupExecute | Boolean | If <a href="#">True</a> , a member of the Group that owns the <a href="#">FolderItem</a> can execute the <a href="#">FolderItem</a> .  |
| GroupRead    | Boolean | If <a href="#">True</a> , a member of the Group that owns the <a href="#">FolderItem</a> can read the <a href="#">FolderItem</a> .   |
| GroupWrite   | Boolean | If <a href="#">True</a> , a member of the Group that owns the <a href="#">FolderItem</a> can write to the <a href="#">FolderItem</a> .   |
| OtherExecute | Boolean | If <a href="#">True</a> , a user outside the Group that owns the <a href="#">FolderItem</a> can execute the <a href="#">FolderItem</a> .   |
| OtherRead    | Boolean | If <a href="#">True</a> , a user outside the Group that owns the <a href="#">FolderItem</a> can read the <a href="#">FolderItem</a> .  |
| OtherWrite   | Boolean | If <a href="#">True</a> , a user outside the Group that owns the <a href="#">FolderItem</a> can write to the <a href="#">FolderItem</a> .  |
| OwnerExecute | Boolean | If <a href="#">True</a> , the owner of the <a href="#">FolderItem</a> can execute the <a href="#">FolderItem</a> .   |
| OwnerRead    | Boolean | If <a href="#">True</a> , the owner of the <a href="#">FolderItem</a> can read the <a href="#">FolderItem</a> .  |
| OwnerWrite   | Boolean | If <a href="#">True</a> , the owner of the <a href="#">FolderItem</a> can write to the <a href="#">FolderItem</a> .  |
| StickyBit    | Boolean | If <a href="#">True</a> , the operating system will not allow someone to remove or rename the file or files in the directory that he does not own, even he has Read/Write permissions. This can be used for directories in which different users need to be able to create files but should not touch others' files. If the StickyBit is set, a 1 appears to the left of the Owner's digit in the Octal representation of the <a href="#">FolderItem</a> 's permissions. |
| UIDBit       | Boolean | Set User ID bit. If <a href="#">True</a> , the current user has the permissions of the owner without altering the Read/Write/Execute permissions for the <a href="#">FolderItem</a> .  |

## Notes

On Unix-based operating systems, permissions can be represented as a three-digit numeric code, in which each digit ranges from 0 to 7 (a.k.a, octal number). The digits correspond to the permissions of the [FolderItem](#) Owner, the owning Group, and Other users not in the owning Group, in that order, from left to right.

The code for each digit is computed using the following values:

| Value | Permission Type     | Description  |
|-------|---------------------|--|
| 4     | Read permissions    | User or group can open and read the <a href="#">FolderItem</a> .     |
| 2     | Write permissions   | User or group can open and write to the <a href="#">FolderItem</a> . |
| 1     | Execute permissions | User or group can execute the file or read the directory.            |

For directories, Execute permissions means that the user or group can list the contents of the directory and examine the files that the user has permission to read.

## Picture Class

---

For each digit, the permissions are expressed by adding up the values. Each digit can take on eight possible values:

| Value | Description                          |
|-------|--------------------------------------|
| 0     | No permissions                       |
| 1     | Execute permissions                  |
| 2     | Write permissions                    |
| 3     | Write and Execute permissions        |
| 4     | Read permissions                     |
| 5     | Read and Execute permissions         |
| 6     | Read and Write permissions           |
| 7     | Read, Write, and Execute permissions |

For example, the code “764” is interpreted as follows:

| Value | User Type | Description                          |
|-------|-----------|--------------------------------------|
| 7     | Owner     | Read, Write, and Execute permissions |
| 6     | Group     | Read and Write permissions           |
| 4     | Others    | Read permissions                     |

**See Also** [FolderItem](#) class.

---

## Picture Class

A **Picture** object contains a picture.

**Super Class** [Object](#)

### Properties

| Name         | Type                     | Description   |
|--------------|--------------------------|---|
| Depth        | <a href="#">Integer</a>  | The depth (bytes) of the picture. A depth of zero indicates that the picture has no bitmap at all, but is for vector graphics.    |
| ImageCount   | <a href="#">Integer</a>  | Number of indexed images in the picture (QuickTime 4 and above only).   |
| IndexedImage | <a href="#">Picture</a>  | Returns $i^{\text{th}}$ image in the picture (QuickTime 4 and above only). Parameter is <i>Index</i> as <a href="#">Integer</a> . |
| Graphics     | <a href="#">Graphics</a> | The actual picture.   |
| Height       | <a href="#">Integer</a>  | The height (in pixels) of the picture.  |

| Name                 | Type                       | Description  |
|----------------------|----------------------------|--|
| HorizontalResolution | <a href="#">Integer</a>    | Horizontal resolution of the picture. Established when you use the <a href="#">OpenAsPicture</a> method of the <a href="#">FolderItem</a> class and written out when using the <a href="#">SaveAsPicture</a> method (of the <a href="#">FolderItem</a> class). When creating a picture, the default resolution is 72 (pixels). |
| Mask                 | <a href="#">Picture</a>    | Allows you to access a mask that controls the transparency of the picture.   |
| Objects              | <a href="#">Group2D</a>    | A set of vector graphics associated with the picture (optional).   |
| RGBSurface           | <a href="#">RGBSurface</a> | Provides pixel-level access to the picture's bitmap. Used only for pictures built with <a href="#">NewPicture</a> with a depth of 16 or 32, otherwise returns <a href="#">Nil</a> . Returns an <a href="#">RGBSurface</a> object.  |
| Transparent          | <a href="#">Integer</a>    | Controls the transparency of the white 'layer' of the picture.<br>0 - Not transparent.<br>1 - White is transparent.  |
| VerticalResolution   | <a href="#">Integer</a>    | Vertical resolution of the picture. Established when you use the <a href="#">OpenAsPicture</a> method of the <a href="#">FolderItem</a> class and written out when you use the <a href="#">SaveAsPicture</a> method (of the <a href="#">FolderItem</a> class). When creating a picture, the default resolution is 72 (pixels). |
| Width                | <a href="#">Integer</a>    | The width (in pixels) of the picture.  |

## Notes

A **Picture** object is created by adding a picture to the Project Editor, by calling the [OpenAsPicture](#) or [OpenAsVectorPicture](#) methods of a [FolderItem](#) object, by calling the [NewPicture](#) function, or by using the [New](#) operator. When you use the [New](#) operator, you must pass the same parameters as when using [NewPicture](#): Width as [Integer](#), Height as [Integer](#), and Depth as [Integer](#). Width and Height are in pixels and the values of Depth may be 0, 1, 2, 4, 8, 16, or 32. If you use the [New](#) operator but run out of memory, an [OutOfMemoryException](#) error will be raised. If you specify a depth of zero REALbasic creates a picture with no pixel map at all, but with a preinitialized Objects property. Use this option for creating vector graphics pictures via the set of [Object2D](#) subclasses.

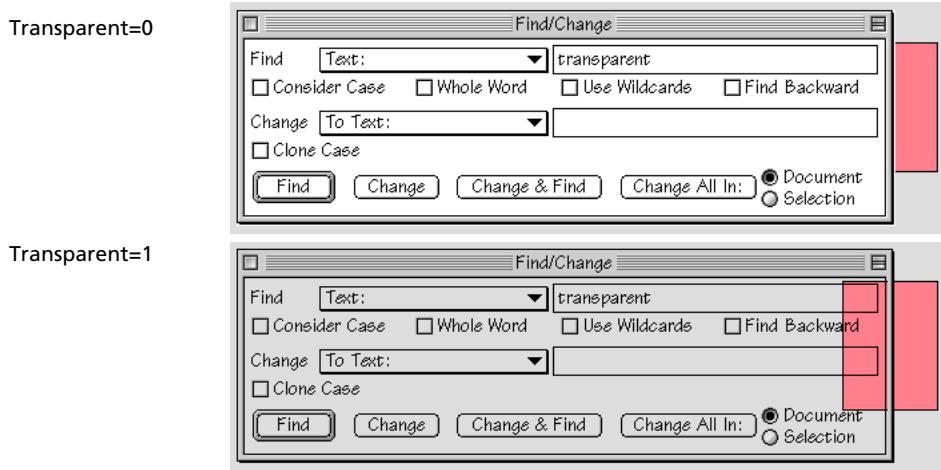
**Picture** objects have no methods or events.

The Transparent property works only if making the color white transparent produces the desired effect. In the example shown below, the picture is the screen shot of the

## PixmapShape Class

---

FrameMaker Find dialog box. Making the white ‘layer’ transparent makes the picture object transparent.



In many cases, you need to work with the Mask property. For example, if you have a picture, myPicture, and a custom mask, myMask, stored in the Project Editor, the code:

```
myPicture.Mask.Graphics.DrawPicture myMask,0,0
```

will assign the custom mask to the Mask property of myPicture.

**See Also** [NewPicture](#) function; [RGBSurface](#) object; [Graphics](#) class.

---

## PixmapShape Class

Used to place a picture in a vector graphics environment.

**Super Class** [RectShape](#)

### Properties

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| Image        | <a href="#">Picture</a> | The source image to use.   |
| SourceHeight | <a href="#">Integer</a> | Height (pixels) of source image, from <i>SourceTop</i> , <i>SourceLeft</i> . Used to define a subregion of image to use. |
| SourceLeft   | <a href="#">Integer</a> | Number of pixels from left edge of source image. Used to define subregion of image to draw.                              |
| SourceTop    | <a href="#">Integer</a> | Number of pixels from top edge of source image. Used to define subregion of image to draw.                               |

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| SourceWidth | <a href="#">Integer</a> | Width (pixels) from SourceLeft of image to draw. Used to define a subregion of image to use. |

## Constructor

| Name        | Parameters                       | Description                                |
|-------------|----------------------------------|--|
| PixmapShape | Image as <a href="#">Picture</a> | Initializes object to entire source image. |

## Example

The following method in the Action event of a [PushButton](#) draws a red oval in the parent window.

```
Dim p as Picture
Dim px as PixmapShape

p=New Picture(240,200,32)
p.graphics.foreColor = &cFF0000
p.graphics.fillOval 0,0,240,200

px=New PixmapShape(p)
px.rotation = 45/57.2958 //45 Degrees in radians

Graphics.drawObject px,150,150
```

## See Also

[ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Group2D](#), [Graphics](#), [Object2D](#), [OvalShape](#), [Picture](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

# Placard Control

Used to add a placard to a window. A placard is a control that you can use as an information display or as background fill for a control area. **Placards** have three states: normal, pressed, and disabled.

Perhaps the most familiar use of the **Placard** control is as a small information panel, often placed at the bottom of a window to the left of the horizontal scroll bar.

## Super Class

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Enabled | <a href="#">Boolean</a> | If <a href="#">True</a> , draws a shaded border around the object; if <a href="#">False</a> , the border is a line drawing and the placard is disabled.   |
| Value   | <a href="#">Boolean</a> | If Enabled is <a href="#">True</a> , Value shades the interior of the placard (Pressed state). If Enabled is <a href="#">True</a> and Value is <a href="#">False</a> , the interior is unshaded (Normal state). |

### Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| MouseDown | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> | The mouse button has been pressed inside the window at the x,y local coordinates passed.  |
| MouseUp   | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> | The mouse button has been released inside the window at the x,y local coordinates passed.   |
| MouseDrag | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> | The mouse button was pressed inside the window and moved (dragged) at the location local to the window passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |

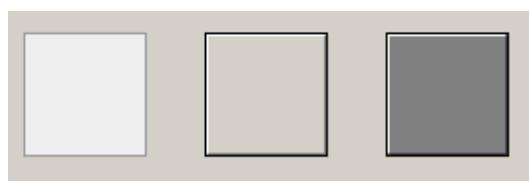
### Note

On Mac OS X, you can solve some display problems with Placards by turning on the parent window's Composite property. For an example, see the Notes on the [Window](#) class.

### Examples

The following illustrations show a **Placard** control in its various states:

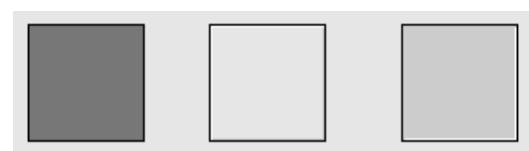
Windows



Macintosh



Linux



Disabled

Value=False

Value=True

**See Also** [RectControl](#) class; [Canvas](#) control.

## Pop Method

Removes the last element from an array and returns its value.

**Syntax**

**result=array.Pop**

| Part   | Description   |
|--------|---|
| result | The element of <i>array</i> that was removed.                 |
| array  | Required. The array from which the last value will be popped. |

**Notes**

The [Append](#) and **Pop** methods can be used together to treat an array as a stack. The [Append](#) method pushes a new element onto the array/stack and the **Pop** method removes the last element from the array/stack and returns its value.

**Example**

This is a directory-crawling routine that performs a depth-first traversal without using recursion:

```
Dim itemsToInspect(0) As FolderItem
Dim item As FolderItem
Dim ctr As Integer
itemsToInspect(0) = directory
While Ubound(itemsToInspect) >= 0
    item = itemsToInspect.Pop
    If item.Directory Then
        For ctr = 1 To item.Count
            itemsToInspect.Append item.Item(i)
        Next
    End If
Wend
```

**See Also**

[Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

## POP3SecureSocket Class

Used to retrieve messages on a POP3 mail server using SSL or TLS encryption.

**Super Class** [SSLSocket](#)

## Properties

| Name            | Type                    | Description   |
|-----------------|-------------------------|---|
| Username        | <a href="#">String</a>  | The username to use for authentication when connecting to the mail server.              |
| Password        | <a href="#">String</a>  | The password to use for security when connecting to the mail server.                    |
| EncryptPassword | <a href="#">Boolean</a> | If <a href="#">True</a> , the password is encrypted when being sent to the mail server. |

## Methods

| Name                  | Parameters  | Description  |
|-----------------------|---|--|
| CheckServerConnection |   | Sends a "NOOP" command to the mail server. This is simply a command that asks the server to reply. This can be useful to check that the mail server is still responding and also tells the mail server that you are still connected if there has been no activity for a long period of time. |
| Connect               |   | Connects to the mail server and logs in with the Username and Password.  |
| CountMessages         |   | Asks the server for the number of messages in the mailbox.   |
| DeleteMessage         | Index as <a href="#">Integer</a>  | Tells the mail server to delete the specified message.   |
| DisconnectFromServer  |   | Disconnects from the mail server. This sends a "QUIT" command to the mail server and waits for it to close the connection.   |
| ListMessages          | [Index as <a href="#">Integer</a> ]                                     | Requests a message listing. This list consists of the message index and the size of the message. If no index is passed, it gets the entire list from the server. If a specific index is passed, it will return just the <i>index</i> message and size of the message.                        |
| RetrieveLines         | Index as <a href="#">Integer</a> , LineCount as <a href="#">Integer</a> | Returns the specified number of lines of a message. The mail server will return the first <i>LineCount</i> of lines that exist in the message you are requesting via the <i>Index</i> parameter. If <i>LineCount</i> is zero, then the mail server returns only the headers for the message. |

| Name              | Parameters                        | Description  |
|-------------------|-----------------------------------|--|
| RetrieveMessage   | Index as <a href="#">Integer</a>  | Reads the entire message specified by <i>Index</i> .   |
| RollbackServer    |                                   | Resets the mail server to the state that it was when you logged in. Can be used to Undo deletions that occur by accident. The changes aren't committed until the connection is closed. RollbackServer will roll back changes that have not yet been committed. |
| SendServerCommand | Command as <a href="#">String</a> | Sends the command specified by <i>Command</i> to the mail server. This is useful when you need to send a command that REALbasic doesn't yet support.   |

## Events

| Name                  | Parameters   | Description   |
|-----------------------|--|---|
| ConnectionEstablished |  | Occurs when a connection has been established.  |
| Disconnected          |  | Occurs when the connection has been terminated.   |
| ListReceived          | List as <a href="#">String</a>   | Executes when the ListMessages method is called. <i>List</i> contains the message listing.  |
| LoginSuccessful       |  | Executes when the login process initiated by calling the Connect method is complete.  |
| MessageCount          | Count as <a href="#">Integer</a>   | Executes when the mail server replies to a CountMessages call and contains the number of messages in the mailbox.   |
| MessageDeleted        | Index as <a href="#">Integer</a>   | Executes when the mail server replies to a DeleteMessage call and contains the index number of the deleted message.   |
| MessageReceived       | Index as <a href="#">Integer</a> ,<br>Message as<br><a href="#">emailMessage</a> | Executes when a message has been received from the mail server, in response to a call to RetrieveMessage. <i>Index</i> contains the index number of the retrieved message and the message contents is in <i>Message</i> . |

## POP3SecureSocket Class

| Name               | Parameters   | Description   |
|--------------------|--|---|
| RollbackSuccessful |  | Executes in response to a call to RollbackServer and indicates that the state of the mailbox has been reset.  |
| ServerAvailable    |  | Executes when the mail server has replied to a call to CheckServerConnection and indicates that the mail server has replied to the call.  |
| ServerError        | ErrorCode as <a href="#">Integer</a> ,<br>ErrorMessage as <a href="#">String</a> ,<br>MessageID as <a href="#">Integer</a> | Executes when a protocol-related error occurs. The error codes are as follows:<br>0 - Unknown Error Message<br>1 - Incorrect Password<br>2 - IncorrectUsername<br>3 - Delete Message Failed<br>4 - List Messages Failed<br>5 - Retrieve Lines Failed<br>6 - Retrieve Message Failed |
| TopLinesReceived   | Index as <a href="#">Integer</a> ,<br>Data as <a href="#">EmailMessage</a>   | Executes in response to a call to RetrieveLines. <i>Index</i> contains the index number of the partial message being retrieved and <i>Data</i> contains the requested lines of the message.   |

### Notes

The **POP3SecureSocket** class is identical to the [POP3Socket](#) class, except that it is derived from the [SSLocket](#) class instead of the [TCPSocket](#) class. This enables you to send secure email by setting the Secure property of the [SSLocket](#) class.

If you use a constructor of a subclass of **POP3SecureSocket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

### Example

The following code in the MessageReceived event handler places the body of an email message in an [EditField](#).

```
Sub MessageReceived(ID as Integer, email as emailMessage)
  Dim s as String
  s=email.bodyHTML
  If s=" " then
    s=email.bodyPlainText
  end
  EditField1.text=s
```

### See Also

[EmailMessage](#), [HTTPSSocket](#), [POP3Socket](#), [SMTPSecureSocket](#), [SMTPSocket](#), [SSLocket](#), [SocketCore](#), [TCPSocket](#) classes.

# POP3Socket Class

Used to retrieve and manage messages on a POP3 mail server.

**Super Class** [TCPSocket](#)

## Properties

| Name            | Type                    | Description   |
|-----------------|-------------------------|---|
| Username        | <a href="#">String</a>  | The username to use for authentication when connecting to the mail server.              |
| Password        | <a href="#">String</a>  | The password to use for security when connecting to the mail server.                    |
| EncryptPassword | <a href="#">Boolean</a> | If <a href="#">True</a> , the password is encrypted when being sent to the mail server. |

## Methods

| Name                  | Parameters                         | Description  |
|-----------------------|------------------------------------|--|
| CheckServerConnection |                                    | Sends a "NOOP" command to the mail server. This is simply a command that asks the server to reply. This can be useful to check that the mail server is still responding and also tells the mail server that you are still connected if there has been no activity for a long period of time. |
| Connect               |                                    | Connects to the mail server and logs in with the Username and Password.  |
| CountMessages         |                                    | Asks the server for the number of messages in the mailbox. It triggers the MessageCount event, from which you can get the total.   |
| DeleteMessage         | <a href="#">Index as Integer</a>   | Tells the mail server to delete the specified message.   |
| DisconnectFromServer  |                                    | Disconnects from the mail server. This sends a "QUIT" command to the mail server and waits for it to close the connection.   |
| ListMessages          | <a href="#">[Index as Integer]</a> | Requests a message listing. This list consists of the message index and the size of the message. If no index is passed, it gets the entire list from the server. If a specific index is passed, it will return just the <i>index</i> message and size of the message.                        |

| Name              | Parameters  | Description  |
|-------------------|---|--|
| RetrieveLines     | Index as <a href="#">Integer</a> , LineCount as <a href="#">Integer</a> | Returns the specified number of lines of a message. The mail server will return the first <i>LineCount</i> of lines that exist in the message you are requesting via the <i>Index</i> parameter. If <i>LineCount</i> is zero, then the mail server returns only the headers for the message. |
| RetrieveMessage   | Index as <a href="#">Integer</a>  | Reads the entire message specified by <i>Index</i> .   |
| RollbackServer    |   | Resets the mail server to the state that it was when you logged in. Can be used to Undo deletions that occur by accident. The changes aren't committed until the connection is closed. RollbackServer will roll back changes that have not yet been committed.                               |
| SendServerCommand | Command as <a href="#">String</a>                                       | Sends the command specified by <i>Command</i> to the mail server. This is useful when you need to send a command that REALbasic doesn't yet support.   |

## Events

| Name                  | Parameters                       | Description   |
|-----------------------|----------------------------------|---|
| ConnectionEstablished |                                  | Occurs when a connection has been established.  |
| Disconnected          |                                  | Occurs when the connection with the server has been lost.   |
| ListReceived          | List as <a href="#">String</a>   | Executes when the ListMessages method is called. <i>List</i> contains the message listing.                          |
| LoginSuccessful       |                                  | Executes when the login process initiated by calling the Connect method is complete.                                |
| MessageCount          | Count as <a href="#">Integer</a> | Executes when the mail server replies to a CountMessages call and contains the number of messages in the mailbox.   |
| MessageDeleted        | Index as <a href="#">Integer</a> | Executes when the mail server replies to a DeleteMessage call and contains the index number of the deleted message. |

| Name               | Parameters   | Description   |
|--------------------|--|---|
| MessageReceived    | Index as <a href="#">Integer</a> ,<br>Message as <a href="#">EmailMessage</a>  | Executes when a message has been received from the mail server, in response to a call to RetrieveMessage. <i>Index</i> contains the index number of the retrieved message and the message contents is in <i>Message</i> .   |
| RollbackSuccessful |  | Executes in response to a call to RollbackServer and indicates that the state of the mailbox has been reset.  |
| ServerAvailable    |  | Executes when the mail server has replied to a call to CheckServerConnection and indicates that the mail server has replied to the call.  |
| ServerCommandReply | Command as <a href="#">String</a> ,<br>Data as <a href="#">String</a>  | Executes in response to a call to SendServerCommand and contains the mail server's response to the command passed.  |
| ServerError        | ErrorCode as <a href="#">Integer</a> ,<br>ErrorMessage as <a href="#">String</a> ,<br>MessageID as <a href="#">Integer</a> | Executes when a protocol-related error occurs. The error codes are as follows:<br>0 - Unknown Error Message<br>1 - Incorrect Password<br>2 - IncorrectUsername<br>3 - Delete Message Failed<br>4 - List Messages Failed<br>5 - Retrieve Lines Failed<br>6 - Retrieve Message Failed |
| TopLinesReceived   | Index as <a href="#">Integer</a> ,<br>Data as <a href="#">EmailMessage</a>   | Executes in response to a call to RetrieveLines. <i>Index</i> contains the index number of the partial message being retrieved and <i>Data</i> contains the requested lines of the message.   |

**Notes**

If you use a constructor in a subclass of **POP3Socket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

**Examples**

The following example in the Action event of a [PushButton](#) establishes a connection to a POP3 server. The **POP3Socket** control, POP3Socket1, has been added to the

window. The method gets the name of the server, username, and password from the contents of the [EditFields](#), ServerFld, UserNameFld, and PasswordFld.

```
If Me.caption = "Connect" then
//get POP3 sever address and port to use
POP3socket1.Address = ServerFld.text
POP3socket1.Port = 110

//get account info
POP3Socket1.Username = UsernameFld.text
POP3Socket1.Password = passwordFld.text

//establish connection
POP3Socket1.connect
Me.Caption = "Disconnect" //change button text while connected
else
POP3Socket1.DisconnectFromServer
Me.caption = "Connect"
End if
```

The following example in the MessageReceived event of the POP3Socket control, displays the message in the multiline EditField, BodyFld.

```
Sub MessageReceived (ID as Integer, Email as EmailMessage)
Dim s as String

// display the message
s = Email.bodyHTML
if s = "" then
s = Email.bodyPlainText
end if
BodyFld.text = ReplaceAll(s,Chr(13)+Chr(10),Chr(13))
```

The following example in the TopLinesReceived event populates a [ListBox](#) with summary information on the email that was received. After populating the [ListBox](#), it checks against the window property MessageTotal, which contains the result from a call

to CountMessages, to see if there are any more message on the server. MessageTotal is assigned the value of the Count parameter in the MessageCount event.

```
Sub TopLinesReceived (ID as Integer, Email as EmailMessage)
// headers received. populate the Listbox
ListBox1.AddRow Email.Subject()
ListBox1.Cell(ListBox1.lastIndex,1) = Email.FromAddress()
ListBox1.Cell(ListBox1.lastIndex,2) = Str(ID)

if ID < MessageTotal then // there are still messages left
  Me.RetrieveLines id+1,0/ / get the next message headers
End if
```

**See Also** [EmailMessage](#), [HTTPSocket](#), [SMTPSocket](#), [SocketCore](#), [TCPSocket](#) classes.

## PopupArrow Control

Used to add a popup arrow to a window.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name   | Type                    | Description   |
|--------|-------------------------|---|
| Facing | <a href="#">Integer</a> | Controls direction and size of the PopupArrow<br>0—East<br>1—West<br>2—North<br>3—South<br>4—Small East<br>5—Small West<br>6—Small North<br>7—Small South |

**Example** The following line of code changes the size and direction of a **PopupArrow**:

```
popuparrow1.facing=3
```

**See Also** [RectControl](#) class.

# PopupMenu Control

A control that presents a list of items. The user can select one item. The [ComboBox](#) control is similar except that the user also has the option of typing an item instead of choosing one from the list of items.

## Super Class [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name                | Type                            | Description   |
|---------------------|---------------------------------|---|
| <b>Bold</b>         | <a href="#">Boolean</a>         | Applies the bold style to all items in the list.  |
| <b>DataField</b>    | <a href="#">String</a>          | Relevant only if the PopupMenu is used in conjunction with a <a href="#">DataControl</a> to display the contents of a field in a database table. The name of a Field in the table referenced by the <a href="#">DataControl</a> .   |
| <b>DataSource</b>   | <a href="#">DataControl</a>     | The <a href="#">DataControl</a> that references a table in a database whose fields are displayed. Use the DataField property to link a field to be displayed in a PopupMenu. In the IDE, a popup menu of field names will appear. When setting in code, it takes a <a href="#">String</a> . Assign it to the Name property of a <a href="#">DataControl</a> . Note: Database fields can also be displayed via a <a href="#">DataControl</a> using <a href="#">ComboBoxes</a> , <a href="#">EditFields</a> , <a href="#">ListBoxes</a> , <a href="#">Checkboxes</a> , and <a href="#">StaticText</a> controls. |
| <b>InitialValue</b> | <a href="#">String</a>          | A list of the default items separated by returns. This property is unreadable at runtime because the list is removed from memory once the control is created. You can set a default value for the control by populating the pop-up with <i>InitialValue</i> and setting the default value with <i>ListIndex</i> .   |
| <b>Italic</b>       | <a href="#">Boolean</a>         | Applies the italic style to all items in the list.  |
| <b>List</b>         | Array as <a href="#">String</a> | The list of items. The first item is numbered zero.   |
| <b>ListCount</b>    | <a href="#">Integer</a>         | The number of items in the list.  |
| <b>ListIndex</b>    | <a href="#">Integer</a>         | Used to get or set the selected item number. The first item is numbered zero. A value of -1 means no selection. May be set using the Properties pane to set the default value of PopupMenu control. If you use the InitialValue property to populate the PopupMenu control and set a value of ListIndex that corresponds to a default item, that item will be displayed in the IDE.   |
| <b>Text</b>         | <a href="#">String</a>          | The text of the currently selected item.  |
| <b>TextFont</b>     | <a href="#">String</a>          | Name of the font used to display the text.  |

| Name      | Type                    | Description   |
|-----------|-------------------------|---|
| TextSize  | <a href="#">Integer</a> | Size of the font used to display the text.            |
| Underline | <a href="#">Boolean</a> | Applies the underline style to all items in the list. |

## Events

| Name      |  | Description  |
|-----------|--|--|
| Change    |  | The selected item has changed.   |
| GotFocus  |  | (Windows and Linux) The control has received the focus. If the user tabbed to select it, the current item is highlighted. On Linux, the control gets an insertion point when it has the focus, but the user cannot edit items. |
| LostFocus |  | (Windows and Linux) The control has lost the focus.  |
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the control's region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown.   |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the control's region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |

## Methods

| Name          | Parameters   | Description  |
|---------------|--|--|
| AddRow        | Item as <a href="#">String</a>   | Appends Item in a new row to the end of the list.  |
| AddSeparator  |  | Adds a separator line to the popup menu.   |
| DeleteAllRows |  | Deletes all rows in the list.  |
| InsertRow     | RowNumber as <a href="#">Integer</a> ,<br>Item as <a href="#">String</a> | Creates a new row at RowNumber (moving the existing row down).   |
| RemoveRow     | RowNumber as <a href="#">Integer</a>                                     | Deletes the specified row.   |
| RowTag        | Row as <a href="#">Integer</a>   | A 'hidden' identifier associated with the item identified by Row. See the example in Chapter 10 of the User's Guide of a database created using object bindings for an example of how RowTag is used to store the ID associated with each PopupMenu item. Returns a <a href="#">Variant</a> . If you want to compare the value of RowTag to another value, you must first convert the value of RowTag to that data type. |

- Examples** This code in the Open event handler populates a popup menu and sets the initial value to the current month:

```
Dim s as String
Dim i,last as Integer
Dim d as New Date
s="January,February,March,April,May,June,July,_
+ "August,September,October,November,December"
last=CountFields(s, ", ")
For i=1 to last
    me.addRow NthField(s, ", ", i)
Next
me.ListIndex=d.Month-1
```

Examine the value of ListIndex in the Change event handler to determine which item was selected.

This example adds an item to PopupMenu1.

```
PopupMenu1.addrow "October"
```

This example populates the RowTag identifier with a sequence number:

```
Dim i,nItems as Integer
nItems=Me.ListCount-1
For i=0 to nItems
    Me.rowTag(i)=i
Next
```

Since RowTag is a [Variant](#), you must first convert it to an [Integer](#) if you want to compare it to another integer. Do this with a simple assignment statement such as:

```
Dim RecID as Integer
RecID=PopupMenu1.Rowtag(1)
```

Then compare RecID to another [Integer](#).

This example opens a new window when an item is chosen.

```
Sub Change()
    Dim w As ListEditorWindow
    If PopupMenu1.text="Edit List..." Then
        w=New ListEditorWindow
    End If
```

Changing the selected item in a PopupMenu:

```
PopupMenu1.listIndex=3
```

Displaying the RowTag of the selected menu item:

```
EditField1.text=popupmenu1.rowtag(popupMenu1.listindex)
```

**See Also** [BevelButton](#), [ContextMenu](#), [ComboBox](#), [EditField](#) controls; [RectControl](#) class.

## PostgreSQLDatabase Class

Used to open a PostgreSQL database.

**Super Class** [Database](#)

### Properties

| Name | Type                    | Description   |
|------|-------------------------|---|
| Port | <a href="#">Integer</a> | The port of the PostgreSQL database to connect to. PostgreSQL's default port is 5432. |

### Notes

The **PostgreSQLDatabase** class requires the PostgreSQL Database plug-in. Place this plug-in in the REALbasic Plugins folder. The REALbasic CD includes all database plug-ins and you can always find the most current versions on REAL Software's web site, <http://www.realsoftware.com>.

A field specified as type Float is actually implemented by PostgreSQL as a double (8 bytes).

The IsPrimary column of the FieldSchema for a PostgreSQL database always returns [False](#).

PostgreSQL dates: field types Date, Time, andTimeStamp are now properly supported; use the [DateColumn](#) and [DateValue](#) methods for a date representation of this data, or use [StringValue](#) to get a human-readable date and/or time.

## Pow Function

---

**Example** This example opens an existing PostgreSQL database.

```
Dim db as PostgreSQLDatabase
db=New PostgreSQLDatabase
db.host="192.168.1.172"
db.port=5432
db.DatabaseName="myDatabase"
db.Username="Charlie"
db.Password="mashie"
If db.connect then
    //proceed with database operations
else
    MsgBox "The connection failed."
end if
```

**See Also** [Database](#), [DatabaseRecord](#), [RecordSet](#) classes.

---

## Pow Function

Returns the value specified raised to the power specified.

**Syntax** *result=Pow (value, power)*

| Part   | Type                   | Description                            |
|--------|------------------------|--|
| result | <a href="#">Double</a> | Value raised to power.                 |
| value  | <a href="#">Double</a> | The value you want to raised to power. |
| power  | <a href="#">Double</a> | The power value is raised to.          |

**Examples** This example uses the **Pow** function to return four raised to the power of seven.

```
Dim d As Double
d=Pow(4,7) //returns 16384
```

**See Also** [^ operator](#).

---

## PowerPointApplication Class

Used to automate Microsoft PowerPoint.

**Super Class** [OLEObject](#)

**Notes**

The language that you use to automate Microsoft Office applications is documented by Microsoft and numerous third-party books on Visual Basic for Applications (VBA). Microsoft Office applications provide online help for VBA. To access the online help, choose Macros from the Tools Menu of your MS Office application, and then choose Visual Basic Editor from the Macros submenu. When the Visual Basic editor appears, choose Microsoft Visual Basic Help from the Help menu. The help is contextual in the sense that it provides information on automating the Office application from which you launched the Visual Basic editor.

If VBA Help does not appear, you will need to install the help files. Windows Office 2003 prompts you to install the VBA help files when you first request VBA help. You don't need the master CD. On Macintosh, Office v.X does not install the VBA help files as part of the full install. Quit out of Office and locate your master CD. Open the "Value Pack" folder and double-click the Value Pack installer. In the Value Pack installer dialog, scroll down to the Programmability topic, select it, and click Continue. The installer will then add the VBA help files and examples to your Office installation. When the install finishes, the VBA help files will be available to the Visual Basic editor within all your Office X applications.

Microsoft has additional information on VBA at <http://msdn.microsoft.com/vbasic/> and have published their own language references on VBA. One of several third-party books on VBA is "VB & VBA in a Nutshell: The Language" by Paul Lomax (ISBN: 1-56592-358-8).

**Example**

This example creates a PowerPoint document with a title page and a user-selected image. The code is in the Action event handler of a Pushbutton. When the user clicks

## PowerPointApplication Class

the button he is prompted to select a picture. The picture is then used as a graphic on the title page of the presentation.

```
Dim ppApp as PowerPointApplication
Dim ppPres as PowerPointPresentation
Dim ppSlide1 as PowerPointSlide
Dim cTop, cWidth, cHeight, cLeft as Single
Dim f as FolderItem

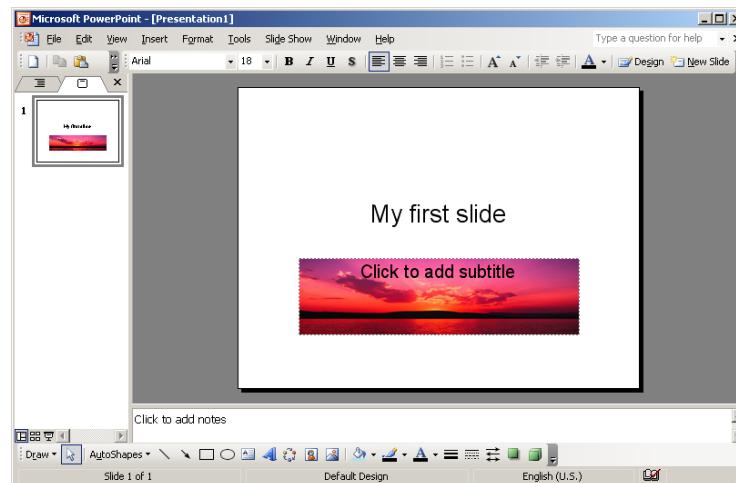
f = GetOpenFolderItem("????")

if f = Nil then
    MsgBox "Please select a picture"
else
    ppApp = New PowerPointApplication
    ppApp.Visible = 1
    ppPres = ppApp.Presentations.Add
    ppSlide1 = ppPres.Slides.Add(1, 1)
    ppSlide1.Shapes.Item(1).TextFrame.TextRange.Text = "My first slide"
    ppSlide1.Shapes.Item(2).TextFrame.TextRange.Text = ""
    ppSlide1.Shapes.Item(2).Fill.UserPicture f.AbsolutePath

    ppApp.Activate
end if

Exception err as OLEException
Msgbox err.message
```

A sample title page looks like this.



### See Also

[ExcelApplication](#), [Office](#), [OLEException](#), [OLEObject](#), [WordApplication](#) classes.

# PreferencesFolder Function

Used to access the Preferences folder.

## Syntax

**result=PreferencesFolder**

| Part   | Type                       | Description  |
|--------|----------------------------|--|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the Preferences Folder or Application Data directory. |

## Notes

Use the **PreferencesFolder** function to access the user's Preferences folder. If your application has user-defined preferences, it is a good idea to store the preference values in a file and place it in the Preferences folder. This allows the user to reinstall your application if necessary and retain their current preferences.

The **PreferencesFolder** function provides a way to access the Preferences folder that will work under different language systems, and will continue to work if the operating system is reorganized.

The [SpecialFolder](#) module enables you to access many other special folders that are managed by the OS.

## Windows

On Windows, PreferencesFolder returns a reference to the c:\Documents and Settings\username\Application Data directory, if available. On Windows 98 it accesses the Windows directory.

## Macintosh

On Mac OS X, **PreferencesFolder** returns a reference to the /Users/username/Library/Preferences folder. On Macintosh "classic", it returns a reference to the Preferences folder in the System folder.

## Linux

On Linux, **PreferencesFolder** returns a reference to the current user's Home directory, /Home/username.

## Example

This example displays the absolute path to the Preferences folder or Application Data directory, if it exists on the user's machine.

```
Dim f As FolderItem
f=PreferencesFolder
if f <> Nil Then
  MsgBox f.AbsolutePath
else
  MsgBox "The folderItem does not exist."
end if
```

## See Also

[ApplicationSupportFolder](#), [DesktopFolder](#), [FontsFolder](#), [StartupItemsFolder](#), [SystemFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

## PrefsMenuItem Class

**PrefsMenuItem** is designed to handle the Preferences menu item for Mac OS X applications. A menu item derived from the **PrefsMenuItem** class is automatically moved to the application menu under Mac OS X (or the REALbasic menu when running within the IDE). Under other operating systems, such a menu item stays where you put it in the Menu Editor.

**Super Class** [MenuItem](#)

Since this class is based on [MenuItem](#), please refer to this class for information on its Properties, Methods, and Events.

### Notes

This class is for creating the Preferences menu item for applications that will run on Mac OS X. Create the menu item under the menu that you want to use for the Windows, Linux, and Mac OS “classic” builds and then use the Properties pane to change its super class to **PrefsMenuItem**. Otherwise, set up the menu item normally. In your Mac OS X build, it will magically move to the built application’s menu.

According to Apple user interface guidelines, ⌘-, (comma) is the recommended keyboard shortcut for an application’s Preferences menu item.

**See Also** [AppleMenuItem](#), [MenuItem](#), [QuitMenuItem](#) classes.

---

## Print Method

Used to print to the terminal in [ConsoleApplications](#).

### Syntax

**Print (data)**

| Part | Type                   | Description  |
|------|------------------------|--|
| data | <a href="#">String</a> | The text to be printed. <b>Print</b> appends the Newline character to the end of <i>data</i> . |

### Notes

The **Print** method performs the same operation as a call to the **WriteLine** method of the [StandardOutputStream](#) class.

### See Also

[ConsoleApplication](#), [StandardInputStream](#), [StandardOutputStream](#) classes; [TargetHasGUI](#) constant.

# PrinterSetup Class

Used to get and set the page setup settings.

**Super Class** [Object](#)

## Properties

| Name                           | Type                    | Description   |
|--------------------------------|-------------------------|---|
| <b>Height</b>                  | <a href="#">Integer</a> | The height of the printable area on the page in pixels. The height is the PageHeight minus the margins.   |
| <b>HorizontalResolution</b>    | <a href="#">Integer</a> | The horizontal resolution in dots per inch of the output device (usually a printer).  |
| <b>Left</b>                    | <a href="#">Integer</a> | The left origin of the printable area. This should always be zero.  |
| <b>MaxHorizontalResolution</b> | <a href="#">Integer</a> | Set to the maximum horizontal resolution at which you wish to print. The default value is 72. You can change it to the maximum printer resolution you're prepared to handle or -1 for the highest possible resolution of the output device. After calling PageSetupDialog, OpenPrinter, or OpenPrinterDialog, the HorizontalResolution property will automatically be set to the highest resolution supported by the printer driver, within your specified constraints. Printing will then occur at that resolution. Will not affect existing code. |
| <b>MaxVerticalResolution</b>   | <a href="#">Integer</a> | Set to the maximum vertical resolution at which you wish to print. The default value is 72. You can change it to the maximum printer resolution you're prepared to handle or -1 for the highest possible resolution of the output device. After calling PageSetupDialog, OpenPrinter, or OpenPrinterDialog, the VerticalResolution property will automatically be set to the highest resolution supported by the printer driver, within your specified constraints. Printing will then occur at that resolution. Will not affect existing code.     |
| <b>PageHeight</b>              | <a href="#">Integer</a> | The height of the page in pixels.   |
| <b>PageLeft</b>                | <a href="#">Integer</a> | The offset distance (in pixels) from the left margin of the printable area to the left edge of the physical page, i.e., the left margin.  |
| <b>PageTop</b>                 | <a href="#">Integer</a> | The offset distance (in pixels) from the top margin of the printable area to the top edge of the physical page, i.e., the top margin.   |
| <b>PageWidth</b>               | <a href="#">Integer</a> | The width of the page in pixels.  |

| Name               | Type                    | Description  |
|--------------------|-------------------------|--|
| SetupString        | <a href="#">String</a>  | A value that represents all of the other properties. When the user clicks OK in the Page Setup dialog box, this value will be populated. You can then store this value to store the users Page Setup settings. Assigning one of these stored values to this property will then update all of the other properties restoring the page setup settings. |
| Top                | <a href="#">Integer</a> | The top origin of the printable area. This should always be zero.  |
| VerticalResolution | <a href="#">Integer</a> | The vertical resolution in dots per inch of the output device (usually a printer).   |
| Width              | <a href="#">Integer</a> | The width of the printable area on the page in pixels. The width is the PageWidth minus the margins.   |

## Methods

| Name            | Parameters                         | Description   |
|-----------------|------------------------------------|---|
| PageSetupDialog | [window] as <a href="#">Window</a> | Displays the standard Page Setup dialog box. If the SetupString property has been populated before this method is called, the Page Setup dialog box will reflect the settings stored in the SetupString property. After the user clicks the OK button to close the Page Setup dialog box, all of the PrinterSetup properties will be updated to reflect the settings the user chose.<br>Takes an optional parameter, <i>window</i> . If passed, <i>window</i> is a sheet window that you wish to use as the Page Setup dialog box instead of the built in dialog. Returns a <a href="#">Boolean</a> . This function returns <a href="#">True</a> if the User clicks OK and <a href="#">False</a> if the user clicks Cancel. |

## Notes

Passing a PrinterSetup object to the [OpenPrinter](#) or [OpenPrinterDialog](#) functions will cause the printer to utilize those PrinterSetup object's properties when printing. For example, if the user chose 200% for the scale in the Page Setup dialog box, the printer would automatically print at 200%.

The Page Setup dialog is not supported on Linux builds. Calling this function will return [False](#) with no dialog presented to the user.

## MaxHorizontalResolution and MaxVerticalResolution

These properties enable you to print at higher resolutions than 72 dpi. In general, you will need to scale the material being printed (and perhaps the size of the controls) to get the desired results. For example, if you are printing styled text in an [EditField](#) using DrawBlock, you will need to make commensurate changes to the font size(s) to get

WYSIWYG output. Doubling the resolution will require that you double the font size; otherwise you will get text that is half the size of the screen font.

- Examples** This example displays the Page Setup dialog box then stores the settings the user chose in a variable:

```
Dim settings as String
Dim PageSetup as PrinterSetup
PageSetup=New PrinterSetup
If PageSetup.PageSetupDialog Then
    settings=PageSetup.SetupString
End If
```

This example restores the page setup settings stored in a variable called “settings” and then displays the Page Setup dialog box with those settings:

```
Dim PageSetup as PrinterSetup
PageSetup=New PrinterSetup
PageSetup.SetupString=settings
If PageSetup.PageSetupDialog Then
    settings=PageSetup.SetupString
End If
```

This example displays the Page Setup dialog box and then passes the settings the user chose to the [OpenPrinterDialog](#) function. It then prints a sample string:

```
Dim g as Graphics
Dim p as PrinterSetup
p=New PrinterSetup
If p.PageSetupDialog then
    g=OpenPrinterDialog(p)
    If g<> Nil then
        g.DrawString "Hello World", 50,50
    End if
End if
```

This example displays the Page Setup box and then displays the page size, printable area, and margins in StaticText controls. Results, of course, depend on the page size that the user selects. Since PageLeft and PageTop are the horizontal and vertical

margins as measured from the printable area rather than the edge of the page, they are negative.

```
Dim settings as String
Dim p as PrinterSetup
p=New PrinterSetup
If p.PageSetupDialog Then
    settings=p.SetupString
End If
staticText1.text="PageLeft="+Str(p.pageLeft)
staticText2.text="PageTop="+Str(p.pagetop)
staticText3.text="PageHeight="+Str(p.pageheight)
staticText4.text="PageWidth="+Str(p.pagewidth)
staticText5.text="Height="+Str(p.Height)
staticText6.text="Width="+Str(p.width)
staticText7.text="Computed height="+Str(p.height-2*p.pagetop)
staticText8.text="Computed width="+Str(p.width-2*p.pageleft)
```

**See Also** [Graphics](#), [StyledTextPrinter](#) classes; [OpenPrinter](#), [OpenPrinterDialog](#) functions.

---

## ProgressBar Control

Displays a progress indicator.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name    | Type                    | Description  |
|---------|-------------------------|--|
| Maximum | <a href="#">Integer</a> | The maximum value of the progress bar. The maximum value of <i>Maximum</i> is 65536. |
| Value   | <a href="#">Integer</a> | The current value of the progress bar.   |

### Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| MouseDown | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the ProgressBar at the location passed in to x,y. Returns a <a href="#">Boolean</a> . <a href="#">Return True</a> if you are going to handle the MouseDown. |

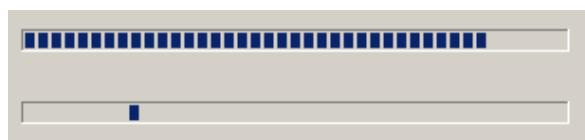
| Name      | Parameters                   | Description   |
|-----------|------------------------------|---|
| MouseDown | x as Integer<br>y as Integer | The mouse button was pressed inside the ProgressBar and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |
| MouseUp   | x as Integer<br>y as Integer | The mouse button was released inside the ProgressBar at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |

**Notes**

When you want to show progress of an indeterminate length, set the Maximum property to zero to display an indeterminate progress bar rather than a progress indicator. On Macintosh, the indeterminate progress indicator looks like a barber pole. On Windows and Linux, it appears as a single bar that moves back and forth.

The illustration below shows a progress indicator and an indeterminate progress bar as they appear on Windows, Macintosh, and Linux.

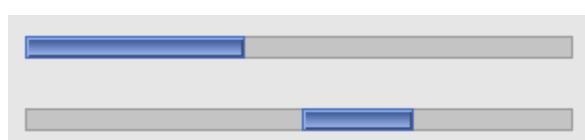
Windows



Macintosh



Linux

**Example**

This line of code sets the maximum value of the progress bar:

```
ProgressBar1.maximum=350
```

**See Also**

[RectControl](#) class.

## ProgressWheel Control

Adds a rotating progress wheel to a window.

**Super Class** [RectControl](#)

## PString Data Type

---

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

**Notes** Use the Visible property of the control to show or hide it. When it is visible, it is animated automatically.

**See Also** [RectControl](#) class.

---

## PString Data Type

A “Pascal” string, typically used only with Mac OS “classic.” The first byte is the length of the string; as a result, it is limited to 255 characters. You can pass a regular REALbasic [String](#) to it and it will be converted to a PString automatically.

PString can be used only with a [Declare](#) statement and it cannot be used to declare REALbasic variables, properties, and methods.

**See Also** [Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

---

## Ptr Data Type

A 4-byte pointer to a chunk of memory. You can pass a [MemoryBlock](#) object via this data type and it will be treated as a pointer to the memory contained within the [MemoryBlock](#).

This data type can be used only with the [Declare](#) statement and it cannot be used to declare REALbasic variables, properties, and methods.

**See Also** [Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

---

## PushButton Control

The standard push button.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name      | Type                    | Description   |
|-----------|-------------------------|---|
| Bold      | <a href="#">Boolean</a> | Applies the bold style to the button caption.   |
| Cancel    | <a href="#">Boolean</a> | If <a href="#">True</a> , the Escape key and Command-Period key sequence are mapped to the button.                                    |
| Caption   | <a href="#">String</a>  | The button's text.  |
| Default   | <a href="#">Boolean</a> | If <a href="#">True</a> , the default button indicator is added to the button and the Return and Enter keys are mapped to the button. |
| Italic    | <a href="#">Boolean</a> | Applies the italic style to the button caption.   |
| TextFont  | <a href="#">String</a>  | Name of the font used to display the button caption.  |
| TextSize  | <a href="#">Integer</a> | Size of the font used to display the button caption.  |
| Underline | <a href="#">Boolean</a> | Applies the underline style to the button caption.  |

## Methods

| Name     | Parameters | Description  |
|----------|------------|--|
| Push     |            | Causes the button to be visually pushed (clicked), as well as triggering the Action event. Call Push when the user has pressed the keyboard equivalent for the button to give the appropriate visual feedback. |
| SetFocus |            | (Windows and Linux) Sets the focus to the PushButton.  |

## Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| Action    |  | The button has been clicked. A right-click does not trigger the Action event.   |
| GotFocus  |  | (Windows and Linux) The button has received the focus and has a selection marquee around the text of the Pushbutton.  |
| LostFocus |  | (Windows and Linux) The button has lost the focus.  |
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the control's region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown. The Action event will not execute and the state of the object will not change. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the control's region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. The return value is ignored.                               |

## Note

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. This is done to make applications on all platforms behave consistently.

## QuickTime Object

---

The **PushButton's** Caption property can show a caption that uses an encoding that doesn't match the application's region code (or while running within the IDE, the IDE's region code). For example, an English application can set a Japanese caption, as long as it first sets the button's TextFont to Osaka.

**See Also** [BevelButton](#) control; [RectControl](#) class.

---

## QuickTime Object

Provides access to certain properties of the QuickTime software installed on the user's system. Accessible via the [System](#) Object.

### Properties

| Name      | Type                    | Description  |
|-----------|-------------------------|--|
| Installed | <a href="#">Boolean</a> | Returns <a href="#">True</a> if QuickTime is installed on the user's computer. |

**Example** This example determines whether QuickTime is installed.

```
If System.QuickTime.Installed then  
    Beep  
    MsgBox "QuickTime is installed!"  
End if
```

**See Also** [System](#) object.

---

## QT3DAudio Class

A storage container for 3D Audio settings. The properties of this class enable you to store such settings.

**Super Class** [Object](#)

### Properties

| Name            | Type                    | Description  |
|-----------------|-------------------------|--|
| ConeAngleCos    | <a href="#">Double</a>  | Cosine (angle/2) of the attenuation cone, in degrees. The range is from zero to 360. |
| ConeAttenuation | <a href="#">Double</a>  | Amount of attenuation (in decibels) outside the cone.                                |
| CPULoad         | <a href="#">Boolean</a> | CPU load versus quality: zero is best quality.                                       |

| Name                     | Type                    | Description  |
|--------------------------|-------------------------|--|
| Humidity                 | <a href="#">Double</a>  | The humidity (per cent) when the medium is air. The range is from zero to 100.                   |
| LocationAzimuth          | <a href="#">Double</a>  | The range is from -1.57 to 1.57.   |
| LocationDistance         | <a href="#">Double</a>  | The units are meters and range is from 1.0 to 10.0.  |
| LocationElevation        | <a href="#">Double</a>  | The units are meters and range is from -3.14 to 3.14.  |
| LocationListenerVelocity | <a href="#">Double</a>  | The units are meters/second. The minimum value is zero.  |
| LocationProjectionAngle  | <a href="#">Double</a>  | Units are degrees and range is from zero to 180.   |
| LocationSourceVelocity   | <a href="#">Double</a>  | The units are meters/second. The minimum value is zero.  |
| Medium                   | <a href="#">Integer</a> | Medium for sound propagation:<br>0—Air,<br>1—Water.  |
| ReferenceDistance        | <a href="#">Double</a>  | Nominal distance for recording. The units are meters and minimum distance is zero.               |
| ReverbAttenuation        | <a href="#">Double</a>  | Reverberation model: mix level. The units are decibels.  |
| RoomReflectivity         | <a href="#">Double</a>  | Reverberation model: Bounce attenuation. The units are decibels.                                 |
| RoomSize                 | <a href="#">Double</a>  | Reverberation model: Distance between walls. The units are meters and the minimum value is zero. |
| SourceMode               | <a href="#">Integer</a> | Type of filtering to apply.<br>0—Unfiltered,<br>1—Localized<br>2—Ambient,<br>3—Binaural.         |

**Notes**

The **QT3DAudio** class is for adding audio spatialization (special effects) to audio tracks. Add the **QT3DAudio** object to a **QTSoundTrack** with **QTSoundTrack.Set3D** method.

The properties of this class enable you to set QuickTime parameters that are documented at the following URL:

<http://developer.apple.com/techpubs/quicktime/qtdevdocs/APIREF/SOURCESIV/ssplocalizationdata.htm>

**See Also**

[EditableMovie](#), [QTSoundTrack](#) classes.

---

## QTEffect Class

A QuickTime effect, generated by [GetQTSMPTEEffect](#) or [GETQTCrossFadeEffect](#) using a [QTEffectSequence](#) instance. See the example for the [QTEffectSequence](#) class.

The **QTEffect** class has no properties, methods, or events.

# QTEffectSequence Class

Builds a QuickTime movie from two pictures and a QuickTime effect.

Use **QTEffectSequence** as a constructor to create a **QTEffectSequence** object.

**Super Class** [Object](#)

**Syntax** **result = New QTEffectSequence(effect,image1,image2,frames)**

| Part   | Type                     | Description  |
|--------|--------------------------|--|
| result | <b>QTEffectSequence</b>  | QTEffectSequence object, created from the values of the passed parameters. |
| effect | <a href="#">QTEffect</a> | QT effect used in creating the sequence.                                   |
| image1 | <a href="#">Picture</a>  | Initial image in sequence.   |
| image2 | <a href="#">Picture</a>  | Final image in sequence.   |
| frames | <a href="#">Integer</a>  | Number of frames in the sequence.  |

## Properties

| Property | Type                    | Description  |
|----------|-------------------------|--|
| Image    | <a href="#">Picture</a> | Image associated with a particular frame in the QT sequence. |
| Frame    | <a href="#">Integer</a> | Frame of QT sequence.  |

**Notes** Use [GetQTSMPTEEffect](#) or [GETQTCrossFadeEffect](#) to obtain the value of *effect*.

**Example** This example creates a QuickTime movie from two pictures, p1 and p2, using the cross-fade effect.

```
Dim sequence as QTEffectSequence
Dim theEffect as QTEffect
theEffect=GetQTCrossFadeEffect
sequence=New QTEffectSequence (theEffect,p1,p2,96)
```

This example creates a QuickTime movie from two pictures that are displayed in [ImageWells](#), ImageWell1 and ImageWell2. The movie demonstrates either a cross-fade effect (as in the first example) or a user-selected SMPTE effect. All of the SMPTE effects are listed in a ListBox named EffectsList. To select an SMPTE effect, the user selects RadioButton2 and clicks on the desired SMPTE effect displayed in EffectsList.

This code is in the Action event of a [PushButton](#). The user clicks it after choosing either the cross-fade effect or a particular SMPTE effect.

```

Dim sequence as QTEffectSequence
Dim i,effectID as Integer
Dim theEffect as QTEffect
Dim track as QTVideoTrack
Dim m as EditableMovie
Dim f as FolderItem
f=GetFolderItem("Movie")
m=f.CreateMovie
if Radiobutton1.value then
    theEffect=GetOTCrossFadeEffect
else
    theEffect=GetOTSMPTEEffect(Val(EffectsList.Cell(EffectsList.ListIndex,1)))
End if
sequence=New QTEffectSequence(theEffect,ImageWell1.Image,
    ImageWell2.Image,96)
track=m.NewVideoTrack(ImageWell1.Image.width, ImageWell1.Image.height, 32)

For i=1 to 96
    sequence.frame=i
    track.appendpicture sequence.image
    progressBar1.value=i
Next
f.launch

```

**See Also**

[EditableMovie](#), [QTEffect](#), [QTEffectSequence](#), [QTTrack](#) classes;  
[GetOTCrossFadeEffect](#), [GetOTSMPTEEffect](#) functions; [MoviePlayer](#) control.

## QTGraphicsExporter Class

Used to export a graphic.

**Super Class** [Object](#)

### Properties

| Name               | Type                    | Description   |
|--------------------|-------------------------|---|
| CompressionQuality | <a href="#">Integer</a> | 0 = Minimum quality<br>256 = Low quality<br>512 = Normal quality<br>768 = High quality<br>1023 = Maximum quality<br>1024 = Lossless |

| Name                       | Type                    | Description   |
|----------------------------|-------------------------|---|
| <b>DefaultExtension</b>    | <a href="#">String</a>  | Default file extension.   |
| DesiredTargetDataSize      | <a href="#">Integer</a> | Target data size in bytes.  |
| <b>HasSettingsDialog</b>   | <a href="#">Boolean</a> | <a href="#">True</a> if the exporter object has a settings dialog box that can be displayed by RequestSettings. |
| OutputFileCreator          | <a href="#">String</a>  | Macintosh File Creator.   |
| OutputFileType             | <a href="#">String</a>  | Macintosh File Type   |
| <b>RequestSettings</b>     | <a href="#">Boolean</a> | Presents a dialog box. See Notes, below.  |
| <b>SettingsDescription</b> | <a href="#">String</a>  | Description of settings.  |

## Methods

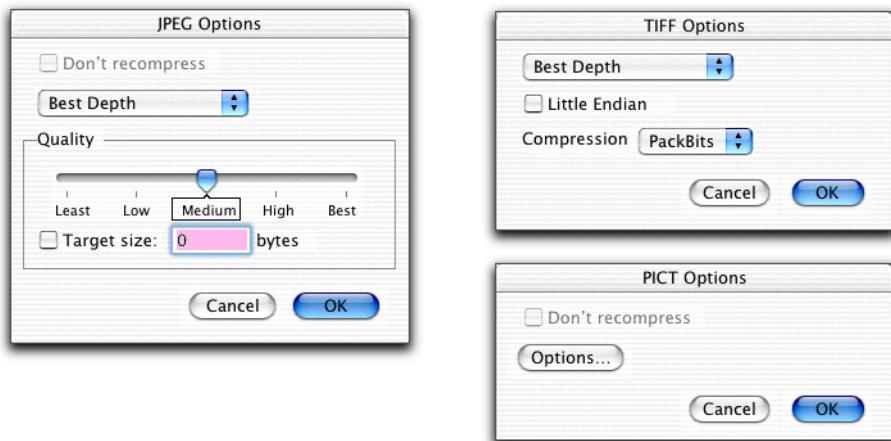
| Name        | Parameters  | Description  |
|-------------|---|--|
| SavePicture | Location as <a href="#">FolderItem</a> , Image as <a href="#">Picture</a> | Saves the image. Returns a <a href="#">Boolean</a> . |

## Notes

The Request Settings dialog box has different options for different file types.

The following are some example RequestSettings dialog boxes:

### Example RequestSettings dialog boxes.



**Example**

The following example exports a graphic, “lois”, that has been added to the Project Editor. You can either specify the exporter’s properties (as shown) or display the Request Settings dialog box.

```
Dim f as FolderItem
Dim exporter as QTGraphicsExporter
Dim mybool as Boolean
f=GetFolderItem(" ")
exporter=GetOTGraphicsExporter("JPEG")
mybool=exporter.RequestSettings
exporter.DesiredTargetDatasize=36
exporter.CompressionQuality=512
exporter.OutputFileType="JPEG"
exporter.OutputFileCreator="ogle"
mybool=exporter.SavePicture(f,lois)
```

**See Also**

[GetQTGraphicsExporter](#) function.

## QTSoundTrack Class

Enables you to add sound to [EditableMovies](#).

**Super Class** [Object](#)

**Properties**

| Name       | Type                    | Description   |
|------------|-------------------------|---|
| Balance    | <a href="#">Integer</a> | Gets or sets the left-right balance of the <b>QTSoundTrack</b> . The range is $\pm 128$ .   |
| Bass       | <a href="#">Integer</a> | Gets or sets the bass level of the <b>QTSoundTrack</b> . The range is $\pm 255$ .   |
| Compressor | <a href="#">String</a>  | Gets the compressor used by the <b>QTSoundTrack</b> . The returned value is a Mac OS API OSType. For example, “raw” or “MAC3”. Use this property to compare QTSoundTracks to avoid combining different compression codecs on the same QTSoundTrack when inserting or appending. |
| SampleRate | <a href="#">Integer</a> | Use to get the sample rate of the <b>QTSoundTrack</b> . Use this property to compare QTSoundTracks to avoid combining different sample rates on the same QTSoundTrack when inserting or appending.  |
| SampleSize | <a href="#">Integer</a> | Use to get the sample size (bit depth) of the <b>QTSoundTrack</b> . Use this property to compare QTSoundTracks to avoid combining different sample sizes on the same QTSoundTrack when inserting or appending.  |

| Name   | Type                    | Description  |
|--------|-------------------------|--|
| Stereo | <a href="#">Boolean</a> | Use to determine whether the <b>QTSoundTrack</b> is in stereo. <a href="#">True</a> indicates Stereo.<br>Use this property to compare QTSoundTracks. |
| Treble | <a href="#">Integer</a> | Use to get or set the treble level of the <b>QTSoundTrack</b> . The range is ± 255.  |
| Volume | <a href="#">Integer</a> | Use to get or set the volume level of the <b>QTSoundTrack</b> . The range is 0 to 512.   |

## Methods

| Name                    | Parameters  | Description   |
|-------------------------|---|---|
| AppendSoundTrackSegment | SourceTrack As <b>QTSoundTrack</b> ,<br>SourcePosition As <a href="#">Double</a> ,<br>SourceDuration As <a href="#">Double</a> ,<br>CopyMedia As <a href="#">Boolean</a> ,<br>ShowProgress As <a href="#">Boolean</a> | Adds a segment from another <b>QTSoundTrack</b> to the end of the calling <b>QTSoundTrack</b> .<br><i>SourcePosition</i> (seconds) is the beginning of the segment in <i>SourceTrack</i> to be appended to the calling <b>QTSoundTrack</b> .<br><i>SourceDuration</i> (seconds) is the length of the segment in <i>SourceTrack</i> to be appended to the calling <b>QTSoundTrack</b> .<br>If <i>CopyMedia</i> is <a href="#">True</a> , the media in <i>SourceTrack</i> will be copied to the destination track; otherwise a reference to <i>SourceTrack</i> will be used.<br>If <i>ShowProgress</i> is <a href="#">True</a> and the operation is lengthy, QuickTime will display a modal dialog box with a progress indicator as the operation proceeds. |
| Get3D                   |   | Obtains a reference to the <b>QT3DAudio</b> object in the movie. Returns the <b>QT3DAudio</b> object.   |

| Name                         | Parameters  | Description   |
|------------------------------|---|---|
| InsertEmptySoundTrackSegment | ThePosition As <a href="#">Double</a> , TheDuration As <a href="#">Double</a>   | Inserts blank space in a <b>QTSoundTrack</b> that already has media on it. The parameters <i>ThePosition</i> and <i>TheDuration</i> specify in seconds the starting point and duration of the blank segment to be inserted. This method has no effect on a <b>QTSoundTrack</b> that has no media on it.   |
| InsertSoundTrackSegment      | SourceTrack As <b>QTSoundTrack</b> , SourcePosition As <a href="#">Double</a> , SourceDuration As <a href="#">Double</a> , destinationPosition As <a href="#">Double</a> , copyMedia As <a href="#">Boolean</a> , ShowProgress As <a href="#">Boolean</a> | Adds a segment from another <b>QTSoundTrack</b> to the calling <b>QTSoundTrack</b> . <i>SourcePosition</i> is the starting point (seconds) in <i>SourceTrack</i> of the segment to be inserted; <i>SourceDuration</i> is the length (seconds) of the segment; <i>DestinationPosition</i> is the starting point (seconds) in the calling <b>QTSoundTrack</b> where <i>SourceTrack</i> will be inserted. If <i>CopyMedia</i> is <a href="#">True</a> , the media in <i>SourceTrack</i> will be copied to the destination track; otherwise a reference to <i>SourceTrack</i> will be used. If <i>ShowProgress</i> is <a href="#">True</a> and the operation is lengthy, QuickTime will display a modal dialog box with a progress indicator as the operation proceeds. |
| Set3D                        | 3DAudio as <a href="#">OT3DAudio</a>  | Sets audio localization of a <b>QTsoundtrack</b> .  |

**See Also**

[EditableMovie](#), [QTTrack](#), [QTVideoTrack](#) classes; [MoviePlayer](#) control.

# QTTrack Class

A QuickTime track, created from an [EditableMovie](#).

Super Class [Object](#)

## Properties

| Name                | Type                    | Description   |
|---------------------|-------------------------|---|
| <b>Duration</b>     | <a href="#">Double</a>  | Duration of track in seconds.   |
| Enabled             | <a href="#">Boolean</a> | <a href="#">True</a> if Enabled. A disabled track will not be displayed (if video) or heard (if audio). |
| MediaType           | <a href="#">String</a>  | Four-character media type (soun, vide).   |
| <b>TimeDuration</b> | <a href="#">Integer</a> | Duration of the movie, in TimeScale units.  |
| TimeScale           | <a href="#">Integer</a> | Defines the timescale of the track, in 1/TimeScale units per second.                                    |
| <b>TrackID</b>      | <a href="#">Integer</a> | ID of track.  |

## Methods

| Name                      | Parameters  | Description  |
|---------------------------|---|--|
| <b>DataSize</b>           | StartTime as <a href="#">Integer</a><br>DurationTime as <a href="#">Integer</a> | Returns <a href="#">Integer</a> , number of bytes in segment specified by StartTime and DurationTime.  |
| <b>Delete</b>             |   | Deletes the QTTrack  |
| <b>DeleteTrackSegment</b> | StartTime as <a href="#">Double</a> , Duration as <a href="#">Double</a>        | Deletes a segment from a QTTrack. <i>StartTime</i> is the beginning of the segment to be deleted (seconds) and <i>Duration</i> is the length (seconds) of the segment. |
| <b>FindFirstSample</b>    |   | Finds first sample (frame in the case of a video track).   |
| <b>FindNextSample</b>     |   | Finds next sample (frame for video track).   |
| <b>FindFirstKeyFrame</b>  |   | Finds first key frame (base frame from which differences are calculated)   |
| <b>FindNextKeyFrame</b>   |   | Finds next key frame.  |

## Notes

The TimeScale parameter defines the number of time units that pass each second. For example, a TimeScale value of 60 sets the timescale in sixtieths of a second. The TimeDuration parameter is expressed in TimeScale units.

## See Also

[EditableMovie](#), [QTVideoTrack](#) classes; [MoviePlayer](#) control.

# QTUserData Class

Used to add and retrieve QuickTime user data, such as copyright notice, credits, and miscellaneous information.

**Super Class** [Object](#)

## Methods

| Name            | Parameters   | Description  |
|-----------------|--|--|
| AddUserData     | udType as <a href="#">String</a><br>Value as <a href="#">String</a>  | Adds a user data item of type udType to that user data list.   |
| GetUserData     | udType as <a href="#">String</a><br>Index as <a href="#">Integer</a><br><a href="#">ByRef</a> Value as <a href="#">String</a>                                      | Gets user data of type udType at position given by Index. After successful call, the user data is stored in Value. Returns a <a href="#">Boolean</a> . |
| GetUserDataText | udType as <a href="#">String</a><br>Index as <a href="#">Integer</a><br><a href="#">ByRef</a> Value as <a href="#">String</a><br>Region as <a href="#">Integer</a> | Gets user data of type udType for Region. After successful call, the user data is stored in Value. Returns a <a href="#">Boolean</a> .                 |
| SetUserDataText | udType as <a href="#">String</a><br>Index as <a href="#">Integer</a><br>Value as <a href="#">String</a><br>Region as <a href="#">Integer</a>                       | Adds user data item in position given by Index of type udType for Region.  |
| UserDataCount   | udType as <a href="#">String</a>   | Number of items in udType. Returns an <a href="#">Integer</a>  |

## Notes

A QuickTime movie can contain user data lists, denoted by the udType parameter. UserData contains additional information about the movie, including copyright information, etc. The QTUserData class is used to manage the user data lists. UserData is a property of [EditableMovie](#).

The udType parameter is a four character code for the type of user data. Apple's currently-defined user data types are shown in the following table:

| udType       | Description                   |
|--------------|-------------------------------|
| @nam         | Movie's name.                 |
| @cpy         | Copyright statement.          |
| @day         | Date the movie was created.   |
| @dir         | Name of the movie's director. |
| @ed1 to @ed9 | Edit dates and descriptions.  |
| @fmt         | Indication of movie format.   |
| @inf         | Information about the movie.  |

| udType | Description  |
|--------|--|
| ©prd   | Name of movie's producer.                            |
| ©prf   | Names of performers.                                 |
| ©req   | Special hardware and software requirements.          |
| ©src   | Credits for those who provided movie source content. |
| ©wrt   | Name of movie's writer.                              |

Apple has reserved all lowercase data type values; however, you can create user data type values using uppercase letters. Apple recommends using type values beginning with the © character to specify user data items that store text data.

It is possible to create more than one user data item in a user data list. Therefore, each user data item is referenced by the `Index` parameter, where the first item is numbered 1. Please see the example for multiple items within a data type.

You may also provide alternate text for a given user data item, for example, to support multiple languages. The alternate versions are indicated by the value of the `Region` parameter. It represents a region code. The example shows English and French titles for a movie using the `Region` parameter.

## Examples

The following code adds several performers and a director to the movie and then displays the names of the performers in a [ListBox](#) and the director in an [EditField](#). It also adds English and French titles to the movie.

```
Dim mybool as Boolean
Dim myval as String
Dim i, myint as Integer
Dim f as FolderItem
Dim m as EditableMovie
f=GetFolderItem("FrmAmngTheDead.Mov")
If f <> Nil then
    m=f.EditableMovie
    //Performers
    m.UserData.AddUserData("@prf","Kim Novak")
    m.UserData.AddUserData("@prf","James Stewart")
    m.UserData.AddUserData("@prf","Barbara Bel Geddes")
    m.UserData.AddUserData("@prf","Tom Helmore")
    //number of performers
    MyInt=m.UserData.UserDataCount("@prf")
    //the director...
    m.UserData.AddUserData("@dir","Alfred Hitchcock")
    //English and French titles
    m.UserData.SetUserDataText("@nam",1,"Vertigo",1)
    m.UserData.SetUserDataText("@nam",1,"D'Entre les Morts",2)
    //display Director's name
    mybool=m.UserData.GetUserData("@dir",1,myval)
    EditField1.text=myval
    //display cast members in ListBox
    For i=1 to MyInt
        mybool=m.UserData.GetUserData("@prf",i,myval)
        ListBox1.AddRow(myval)
    Next
    end if
```

## See Also

[EditableMovie](#) class; [MoviePlayer](#) control.

# QTVideoTrack Class

A QuickTime video track.

Super Class [QTTrack](#)

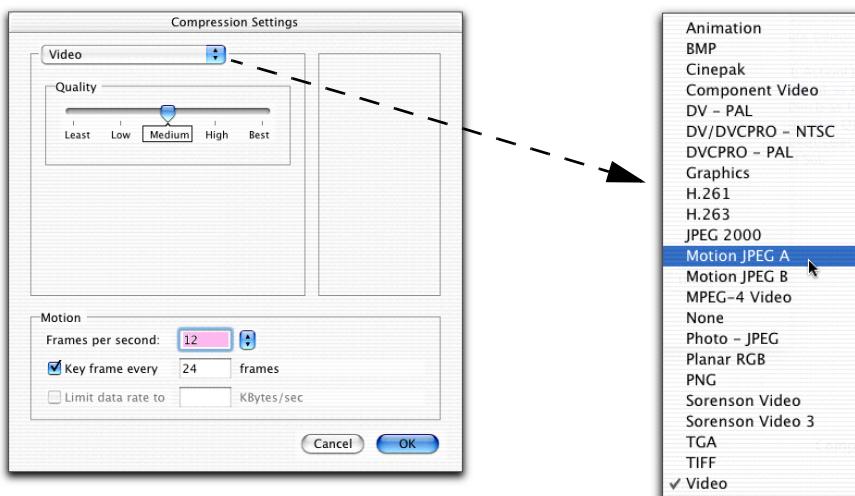
## Methods

| Name                    | Parameters   | Description  |
|-------------------------|--|--|
| AppendPicture           | Pic as <a href="#">Picture</a>   | Picture to be appended to video track. Used to build a movie frame-by-frame.   |
| AppendVideoTrackSegment | SourceTrack As <a href="#">QTVideoTrack</a> ,<br>SourcePosition As <a href="#">Double</a> ,<br>SourceDuration As <a href="#">Double</a> ,<br>CopyMedia As <a href="#">Boolean</a> ,<br>ShowProgress As <a href="#">Boolean</a> | Appends a segment from <i>SourceTrack</i> to the calling <b>QTVideoTrack</b> . <i>SourcePosition</i> is the starting point or the segment (seconds) in <i>SourceTrack</i> and <i>SourceDuration</i> is the length (seconds) of the segment. If <i>CopyMedia</i> is <a href="#">True</a> , the media in <i>SourceTrack</i> will be copied to the destination track; otherwise a reference to <i>SourceTrack</i> will be used. If <i>ShowProgress</i> is <a href="#">True</a> and the operation is lengthy, QuickTime will display a modal dialog box with a progress indicator as the operation proceeds. |

| Name                      | Parameters   | Description   |
|---------------------------|--|---|
| InsertVideoTrackSegment   | SourceTrack As <b>QTVideoTrack</b> ,<br>SourcePosition As <b>Double</b> ,<br>SourceDuration As <b>Double</b> ,<br>DestinationPosition As <b>Double</b> , CopyMedia As <b>Boolean</b> ,<br>ShowProgress As <b>Boolean</b> | Inserts a segment from <i>SourceTrack</i> to the calling <i>QTVideoTrack</i> at <i>DestinationPosition</i> .<br><i>SourcePosition</i> is the starting point or the segment (seconds) in <i>SourceTrack</i> and <i>SourceDuration</i> is the length (seconds) of the segment.<br><i>DestinationPosition</i> is the position in the calling <i>QTVideoTrack</i> at which the segment will be inserted.<br>If <i>CopyMedia</i> is <b>True</b> , the media in <i>SourceTrack</i> will be copied to the destination track; otherwise a reference to <i>SourceTrack</i> will be used.<br>If <i>ShowProgress</i> is <b>True</b> and the operation is lengthy, QuickTime will display a modal dialog box with a progress indicator as the operation proceeds. |
| SelectCompressionSettings |  | Brings up the standard QT compression settings dialog box.<br>Returns a <b>Boolean</b> that is <b>True</b> if the user clicked OK.  |

**Notes**

The SelectCompressionSettings method displays the following dialog box.



Some video formats add an Options button below the Quality slider control.

- Example** The following example creates a QuickTime movie from two images and a user-selected video effect.

```
Dim sequence as QTEffectSequence
Dim i,effectID as Integer
Dim theEffect as QTEffect
Dim track as QTVideoTrack
Dim m as EditableMovie
Dim f as FolderItem
Dim mybool as Boolean
f=GetFolderItem("Movie")
m=f.CreateMovie //create null movie
If radiobutton1.value then //get user's chosen effect
  theEffect=GetOTCrossFadeEffect
else
  theEffect=GetOTSMPTEEffect\(Val\)(EffectsList.cell(EffectsList.ListIndex,1))
end if
sequence=New QTEffectSequence(theEffect,ImageWell1.Image,_
ImageWell2.Image,96)
track=m.NewVideoTrack(ImageWell1.Image.width,_
ImageWell1.Image.height, 32)
mybool=track.SelectCompressionSettings
For i=1 to 96
  sequence.frame=i
  track.appendpicture sequence.image
  progressBar1.value=i
Next
f.launch //open movieplayer
```

- See Also** [EditableMovie](#), [FolderItem](#), [QTEffect](#), [QTEffectSequence](#), [QTTrack](#) classes;  
[GetCrossFadeEffect](#), [GetOTSMPTEEEffect](#) functions.

---

## Quaternion Class

A 3D object for representing orientation or rotation within an [RB3DSpace](#) world.

### Properties

| Name | Type                   | Description                        |
|------|------------------------|------------------------------------|
| W    | <a href="#">Double</a> | The W component of the Quaternion. |
| X    | <a href="#">Double</a> | The X component of the Quaternion. |

| Name | Type                   | Description                        |
|------|------------------------|------------------------------------|
| Y    | <a href="#">Double</a> | The Y component of the Quaternion. |
| Z    | <a href="#">Double</a> | The Z component of the Quaternion. |

## Methods

| Name               | Parameters  | Description   |
|--------------------|---|---|
| Inverse            |   | Returns the opposite of the Quaternion.<br>Returns a Quaternion.  |
| Invert             |   | Reverses the Quaternion to represent the opposite direction.  |
| MultiplyBy         | quat as <a href="#">Quaternion</a>  | Multiplies this Quaternion by another Quaternion, updating this Quaternion. Same as the Time method, except that it updates this Quaternion rather than returning the result as a Quaternion. |
| Normalize          |   | Scales Quaternion to a standard size.   |
| SetBetween         | quat1 as <a href="#">Quaternion</a> , quat2 as <a href="#">Quaternion</a> , position as <a href="#">Double</a>                          | Sets this Quaternion to a combination of <i>quat1</i> and <i>quat2</i> .  |
| SetRotateAboutAxis | axisX as <a href="#">Double</a> , axisY as <a href="#">Double</a> , axisZ as <a href="#">Double</a> , radians as <a href="#">Double</a> | x, y, and z coordinates and the object's angle (radians) Used to rotate an object about the axis at the specified angle.  |
| Times              | quat as <a href="#">Quaternion</a>  | Multiplies <i>quat</i> by another Quaternion and returns the product. Returns a Quaternion.   |
| Transform          | vector as <a href="#">Vector3D</a>  | Returns the result of rotating <i>vector</i> by this Quaternion. Returns a <a href="#">Vector3D</a> .   |

## Notes

A **Quaternion** is a compact, efficient way to represent orientation or rotation in three-dimensional space. They do the same job for orientation and rotation that vectors do for position and movement. Any possible rotation can be represented as an angle turned around a certain axis. A **Quaternion** is essentially an axis plus an angle, although the internal representation is slightly different from this for mathematical reasons.

Just as a vector can be used to represent either position or displacement, a **Quaternion** can be used to represent either orientation or rotation. To represent absolute orientation, we simply consider the quaternion as a rotation relative to a “null” or standard orientation.

The neat thing about quaternions is that they can be combined in a very straightforward manner. When you multiply quaternion A by quaternion B, the result is a quaternion which we'll call C. Rotation C puts the object in exactly the same

orientation as you'd get by doing rotation A followed by rotation B. Multiplying quaternions is exactly the same as composing rotations.

To make a quaternion represent a particular rotation, use the SetRotateAboutAxis method, giving it the axis and the angle (in radians) you want to rotate. For example, to make a quaternion that rotates 90 degrees around the X axis, you'd use:

```
q.SetRotateAboutAxis 1,0,0, Pi/2
```

To rotate a vector, use the Transform method of a **Quaternion**. Give it a vector, and the result will be a new vector, rotated about the origin (0,0,0).

To interpolate between two Quaternions: this is used when you think of the quaternions as representing orientation. You have q1 representing the object in one orientation and q2 representing it in another. You want to animate the object smoothly rotating between q1 and q2. To do this, set the object's orientation to something in between q1 and q2 by using the SetBetween method. The final parameter is the position between these two, with 0.0 meaning q1 and 1.0 meaning q2. 0.5 would be a position exactly halfway in between.

One common application is to update an object's orientation by some rotation. For example, suppose [Object3D](#) didn't have a built-in Yaw method. You could do the same thing yourself as follows:

```
Dim yawQuat as Quaternion  
yawQuat = New Quaternion  
yawQuat.SetRotateAboutAxis 0,1,0, yawAngle // "yaw" -rotate about Y axis  
obj.orientation = yawQuat.Times(obj.orientation)
```

Note that when multiplying quaternions, order does matter. A.Times(B) is not the same as B.Times(A); this is just the nature of rotations in 3D space.

Finally, note that quaternions also have a MultiplyBy function;  $Q1 = Q1.Times(Q2)$  can be written more efficiently as  $Q1.MultiplyBy Q2$ . But that doesn't help in the example above, because we need  $Q2 = Q1.Times(Q2)$  in order to get the correct ordering. We want to start with the current orientation of the object, so it must appear on the right.

|                      |  |
|----------------------|--|
| <b>Interpolation</b> | Use the SetBetween method, giving it the two quaternions you want to interpolate between, and the interpolation factor: 0.0 to exactly match the first quaternion, 1.0 to exactly match the second, and something else to assume some position in between: |
|----------------------|--|

```
Q.SetBetween Q1, Q2, 0.25 // set Q to 25% of Q1 plus 75% of Q2
```

This is an amazingly useful technique. It lets you produce smooth animation between the orientation that you have and an orientation you want to have. For example, if you

have two “keyframe” (pre-defined or pre-computed) positions and orientations the object is supposed to interpolate between, in 11 steps, you could do it this way:

```
For i = 0 to 10
    obj.position.x = position1.x * i/10 + position2.x * (1 - i/10)
    obj.position.y = position1.y * i/10 + position2.y * (1 - i/10)
    obj.position.z = position1.z * i/10 + position2.z * (1 - i/10)
    obj.orientation.SetBetween orientation1, orientation2, i/10
Rb3DSpace1.Update
next
```

**See Also**

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

## Quit Method

Quits the application.

**Syntax**

**Quit([status])**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| status | <a href="#">Integer</a> | For console applications only, optional parameter that returns a status code. When you call Quit, it will terminate your application and returns the status code in <i>status</i> . |

**Notes**

Calling the **Quit** method will call the [Application](#) class's CancelClose event. To cancel the close, return [True](#) from this event. If it does not return [True](#), each window's CancelClose event handler will execute. CancelClose and Close are called as a pair for each open window.

If a CancelClose event handler returns [False](#) (the default action) then the window's Close event handler will be executed. If any window's CancelClose event handler returns [True](#), REALbasic will stop sending CancelClose or Close events and the application will not quit.

**Example**

This example quits the application.

```
Quit
```

**See Also**

[Application](#), [QuitMenuItem](#) classes.

## QuitMenuItem Class

The Quit menu item is derived from the **QuitMenuItem** class.

### Super Class [MenuItem](#)

Since this class is based on [MenuItem](#), please see that class for information on its Properties, Methods, and Events.

### Notes

The Quit menu item that is added by default to all applications is an instance of the **QuitMenuItem** class. It differs from menu items derived from the [MenuItem](#) class in that it is enabled by default and automatically calls the [Quit](#) method to quit the application. Thus, there is no need to enable the menu item via code or write its handler.

### Examples

You normally have no need to create additional instances of the **QuitMenuItem** class, but you can use its properties to modify the default appearance or behaviour of the Quit menu item.

`FileQuit.BalloonHelp="Exits the application"`

### See Also

[EnableMenuItems](#) function; [New](#) operator; [AppleMenuItem](#), [MenuItem](#), [PrefsMenuItem](#) classes.

---

## RadioButton Control

The standard radio button control. If you enclose a group of radio button controls within a parent control (or just the parent window), REALbasic will automatically deselect the remaining radio buttons when the user clicks a radio button in the group.

### Super Class [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name     | Type                    | Description  |
|----------|-------------------------|--|
| Bold     | <a href="#">Boolean</a> | Applies the bold style to the button caption.        |
| Caption  | <a href="#">String</a>  | The button's text.                                   |
| Italic   | <a href="#">Boolean</a> | Applies the italic style to the button caption.      |
| TextFont | <a href="#">String</a>  | Name of the font used to display the button caption. |

| Name      | Type                    | Description   |
|-----------|-------------------------|---|
| TextSize  | <a href="#">Integer</a> | Size of the font used to display the button caption.    |
| Underline | <a href="#">Boolean</a> | Applies the underline style to the button caption.      |
| Value     | <a href="#">Boolean</a> | Value of the radio button when its owning window opens. |

## Events

| Name      |  | Description   |
|-----------|--|---|
| Action    |  | The RadioButton has been clicked. A right+click does not trigger the Action event.  |
| GotFocus  |  | (Windows and Linux only) The radio button has received the focus and has a selection marquee around its caption. On Linux, a RadioButton is not included in the Tab order but gets the focus when selected with the mouse or the arrow keys.    |
| LostFocus |  | (Windows and Linux) The radio button has lost the focus and no longer has a selection marquee.  |
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the control's region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown. The Action event will not execute and the state of the object will not change. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the control's region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. The return value is ignored.                               |

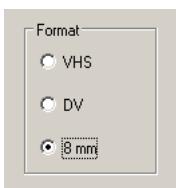
## Notes

If you have more than one group of radio buttons in a window, each group should have its own parent control. Radiobuttons within a group will be enabled and disabled automatically as the user clicks on a radiobutton within a group. You can use controls such as [GroupBoxes](#), [TabPanels](#), [PagePanels](#), or even [Canvases](#) as parent controls.

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. This is done to make applications on all platforms behave consistently.

## Example

The following interface uses a [GroupBox](#) and three RadioButtons:



The three RadioButtons are set up as a control array. That is, each RadioButton is named "RadioButton1" and their Index properties are set to 0, 1, and 2, respectively.

## Raise Statement

---

RadioButton1's Action event is passed the parameter, *Index* as [Integer](#). The parameter *Index* contains the value of Index for the RadioButton that was clicked. Use a [Select](#) Case statement to determine which button was clicked:

```
Sub Action (Index as Integer)
Select Case index
Case 0
    StaticText1.text="VHS"
Case 1
    StaticText1.text="DV"
Case 2
    StaticText1.Text="8 mm"
End Select
```

**See Also** [GroupBox](#), [PagePanel](#), [TabPanel](#) control; [RectControl](#) class.

---

## Raise Statement

The Raise statement is used to raise an exception to the next level in the calling chain.

**Syntax** **Raise expression**

Expression must evaluate to an object that is a subclass of [RuntimeException](#) — and will be the object that is passed to any exception handlers.

**Example** You might want to use **Raise** to handle certain types of exceptions differently than others. For example, if you want to handle all exceptions except out of bounds exceptions, you can raise only those and catch the others. For example

```
Exception err as OutofBoundsException
Raise Err
Exception err
MsgBox "An exception occurred here. Never mind."
```

**See Also** [Function](#) statement; [RuntimeException](#) class; [Exception](#), [Try](#) blocks.

---

## RaiseEvent Statement

Calls the event that is passed to it.

**Syntax****RaiseEvent** *EventName* [(*parameters*)]

| Part       | Type | Description                                     |
|------------|------|---|
| EventName  |      | The name of the event that you want to trigger. |
| Parameters |      | Parameters to <i>EventName</i> , if any.        |

**Notes**

*EventName* is the name of the event that you want to call. If the event requires parameters, then they are passed as the optional parameter to **RaiseEvent**. **RaiseEvent** is useful when you have an Event Definition that has the same name as a method and the ordinary way to call an event definition is ambiguous.

## Random Class

Use to generate random numbers.

**Super Class** [Object](#)

**Properties**

| Name | Type                    | Description  |
|------|-------------------------|--|
| Seed | <a href="#">Integer</a> | Gets or sets the seed used by the random number generator. |

**Methods**

| Name     | Parameters   | Description   |
|----------|--|---|
| Gaussian |  | Returns a random number drawn from a Gaussian (a.k.a. Normal) distribution rather than from a uniform distribution. The Gaussian distribution has a mean of zero and a standard deviation of 1. |
| InRange  | minR as <a href="#">Integer</a><br>maxR as <a href="#">Integer</a> | Returns a random number as an <a href="#">Integer</a> in the range from <i>minR</i> to <i>maxR</i> .  |
| LessThan | Range as <a href="#">Integer</a>                                   | Returns a random number as an <a href="#">Integer</a> in the range greater than or equal to 0 and less than <i>Range</i> -1   |
| Number   |  | Returns a random number as a <a href="#">Double</a> in the range greater than or equal to 0 and less than 1.  |

**Notes**

The **Random** class supersedes the [Rnd](#) function. It performs the same functions and more.

## Range Class

---

You should create only one **Random** object per application. Creating an new **Random** object each time you need another random number is inefficient and can lead to some hard-to-find bugs.

**Example** The following example generates a random integer in the range from 0 to 1,000.

```
Dim r as New Random  
Statictext1.Text=Str(r.InRange(0,1000))
```

**See Also** [Rnd](#) function.

## Range Class

Used to get the starting position, length, and end position of a [StyleRun](#).

**Super Class** [Object](#)

### Properties

| Name     | Type                    | Description   |
|----------|-------------------------|---|
| StartPos | <a href="#">Integer</a> | The starting position of the <a href="#">StyleRun</a> . The first position is zero. |
| EndPos   | <a href="#">Integer</a> | The ending position of the <a href="#">StyleRun</a> .                               |
| Length   | <a href="#">Integer</a> | The length of the <a href="#">StyleRun</a> .  |

**Notes** A [StyleRun](#) is a series of characters with the same style attributes in a [StyledText](#) object. A block of [StyledText](#) can be thought of as a group of [StyleRuns](#) that are appended to each other. Use the **Range** class to access the position of a [StyleRun](#) within the styled text.

**Example** The following example loops through the [StyleRuns](#) in a block of [StyledText](#) that is displayed in an [EditField](#). It uses the StartPos and Length properties of the **Range** class to get the position of each [StyleRun](#) and displays it and the text of each [StyleRun](#) in a [ListBox](#).

```
Dim count, i as Integer  
count=EditField1.StyledText.StyleRunCount //get the number of StyleRuns  
For i=0 to count-1 //loop through them  
    ListBox1.AddRow Str(EditField1.StyledText.StyleRunRange(i).StartPos)  
    ListBox1.Cell(i,1)=Str(EditField1.StyledText.StyleRunRange(i).Length)  
    ListBox1.Cell(i,2)=EditField1.StyledText.StyleRun(i).Text  
Next
```

**See Also** [EditField](#), [Paragraph](#), [StyledText](#), [StyleRun](#) classes; [EditField](#) control.

## RB3DSpace Control

Used to view and manipulate objects in 3D space.

**Super Class** [RectControl](#)

### Properties

| Name           | Type                      | Description  |
|----------------|---------------------------|--|
| AmbientColor   | <a href="#">Color</a>     | The color of the built-in ambient light.   |
| AmbientLight   | <a href="#">Integer</a>   | Brightness of ambient light. Range is from 0 to 100.   |
| Background     | <a href="#">Object3D</a>  | Background object or grouped object ( <a href="#">Group3D</a> ).   |
| Camera         | <a href="#">Element3D</a> | Camera position and orientation. To move the camera, change the values of its Position.X, Position.Y, and/or Position.Z properties, e.g., RB3DSpace1.Camera.Position.Z=100. To rotate the camera, use the Yaw, Pitch, and Roll methods of the <a href="#">Element3D</a> class, which take a <a href="#">Vector3D</a> or the Orientation property, which takes a <a href="#">Quaternion</a> . |
| DebugCube      | <a href="#">Boolean</a>   | If <a href="#">True</a> , draws an x, y, z set of axes in the background of the RB3DSpace to help orient you. The debug cube appears only in the IDE.  |
| FieldOfView    | <a href="#">Integer</a>   | Angle of view (degrees) that's shown in the area of the RB3DSpace. A typical field of view for a game is about 40-50 degrees. You can provide a "zoom" effect by temporarily reducing the FieldOfView or a wide-angle effect by increasing its value. The default is 50 degrees.   |
| FloodColor     | <a href="#">Color</a>     | The color of the light from the built-in floodlight. The default color is white.   |
| FloodDirection | <a href="#">Vector3D</a>  | Specifies the direction of the built-in floodlight.  |
| FloodLight     | <a href="#">Integer</a>   | Brightness of the floodlight. Range is from 0 to 100.  |
| FogStart       | <a href="#">Integer</a>   | If FogVisible is <a href="#">True</a> , Fogstart indicates how far away from the camera the fog should begin. The fog gets thicker from that point (or Hither), whichever is farther away, until the Yon distance, where it is fully opaque. The default is zero. The fog color is the same as the SkyColor.   |
| FogVisible     | <a href="#">Boolean</a>   | If <a href="#">True</a> , fog is drawn according to the FogStart parameter. The default is <a href="#">False</a> .   |
| Hither         | <a href="#">Integer</a>   | Near clipping plane. Nothing closer than the Hither value gets drawn. The units are arbitrary, but are the same for Yon and FogStart.  |

| Name       | Type                    | Description  |
|------------|-------------------------|--|
| LightCount | <a href="#">Integer</a> | The number of lights in the scene.   |
| Objects    | <a href="#">Group3D</a> | Objects to be rendered.  |
| SkyColor   | <a href="#">Color</a>   | The background color.  |
| WireFrame  | <a href="#">Boolean</a> | If <a href="#">True</a> , render the objects as wireframes.  |
| ViewHandle | <a href="#">Integer</a> | Allows you to access the TQ3ViewObject associated with the control, for use with Declares to Quesa/QD3D. The object returned is not locked, so you should not dispose of it nor attempt to use it after the control is closed. |
| Yon        | <a href="#">Integer</a> | Far clipping plane. Nothing farther away from the Yon value is drawn. The units are arbitrary, but are the same as Hither and FogStart.  |

## Methods

| Name        | Parameters   | Description  |
|-------------|--|--|
| AddLight    | Light as <a href="#">Light3D</a>   | Adds a light to the scene.   |
| FindObject  | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a>   | Finds object at x, y (pixels) within the RB3DSpace. The coordinates. Returns <a href="#">Nil</a> if the point contains no object. Returns the object drawn at that point as an <a href="#">Object3D</a> .                            |
| FindPoint   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>   | Finds point at x, y (pixels) within the RB3DSpace. Returns <a href="#">Nil</a> if the x, y, contains no point. Returns returns the 3D world coordinates of the part of the scene drawn at that point as a <a href="#">Vector3D</a> . |
| GetPicture  | [Width as <a href="#">Integer</a> ]<br>[,Height as <a href="#">Integer</a> ]<br>[,Depth as <a href="#">Integer</a> ] | Renders the current scene into a newly created <a href="#">Picture</a> and returns it. If the parameters are omitted the control's current width and height and a depth of 32 are used.  |
| Light       | Index as <a href="#">Integer</a>   | Gets or sets a light. Returns a <a href="#">Light3D</a> .  |
| RemoveLight | Index as <a href="#">Integer</a><br>OR<br>Light as <a href="#">Light3D</a>   | Removes the specified light from the scene. You can indicate the light to be removed either by its index or by reference to it.  |

| Name   | Parameters | Description  |
|--------|------------|--|
| Update |            | Redraws the 3D space without erasing or triggering a refresh. To provide a modeless interface for the user (permitting the use of the mouse and the keyboard), call Update from the Action event of a <a href="#">Timer</a> object. Set the Timer's Mode property to 2 (Multiple calls). |

## Notes

- Requirements** The 3D classes in REALbasic require either Quesa or QuickDraw 3D. The latter is a library by Apple Computer and part of QuickTime, while Quesa is an open-source project that provides very similar functionality. The Quesa libraries are available at <http://www.quesa.org>. Which one you use may depend on the operating system you're running.
- Windows** On Windows, copy the Quesa.dll file to your Windows System directory. Under Windows, you have a choice. QuickDraw 3D is part of QuickTime for Windows, so you can use that or Quesa (and OpenGL). The **Rb3DSpace** control is written assuming Quesa, and since Windows locates libraries based on their file names, to use QuickDraw 3D you'll need to rename QD3D.DLL to QUESA.DLL. (This is not recommended; Quesa gives better performance under Windows.)
- Macintosh** On Macintosh, copy the file "Quesa" to your /Library/CFMSupport folder. Under Mac OS "classic," QuickDraw 3D is installed by default, so you don't need to install any additional libraries. You can instead install Quesa Classic if you prefer.

- Linux** Linux requires OpenGL to be installed. If the required 3D software is not installed, you will get a [NilObjectException](#) when the compiler encounters an **Rb3DSpace** object. If the required libraries are not installed if the Objects property of any **Rb3DSpace** control will be [Nil](#). If the proper libraries are available, this property is automatically initialized to a [Group3D](#) (i.e., it's non-nil). A check such as the following (in the Open event of the **Rb3DSpace** control) is a good idea:

```
If Me.objects = Nil then
  MsgBox "You need to install QuickDraw 3D or Quesa!"
  Return
End if
```

Note that this works only under Mac OS. Under Windows, if the QUESA.DLL library can't be found, your program will not even launch. This is just a side-effect of the way shared libraries work under Windows.

|                   |   |
|-------------------|---|
| <b>Background</b> | <p>The <b>RB3DSpace</b> control is where 3D renderings are done. The axes of the 3D space are displayed when you set the DebugCube property to <a href="#">True</a>. The Z axis is the “depth” dimension and the “size” of this axis is controlled by the Hither and Yon properties. The Hither and Yon properties define the near and far clipping planes (the Z axis); you want to keep Hither and Yon as close together as possible in order to make best use of the Z-buffer of your hardware. FieldOfView is in degrees, and specifies how wide is the angle that the camera sees; make this smaller to zoom in, or larger to zoom out.</p> <p>Note that the camera is defined as an object; it doesn’t need any geometry, but this lets you move the camera exactly as you would move any other object. The background is defined as an <a href="#">Object3D</a>, but you could assign a <a href="#">Group3D</a> to it instead if you want.</p> <p>All other objects in the scene are stored in the Objects property. It is a <a href="#">Group3D</a> object so that you can easily add things to it (or iterate over objects in the scene, etc.). If you need to find the object or point in 3D space which corresponds to a given pixel position within the view, use FindObject and FindPoint. These will return <a href="#">Nil</a> if the point clicked contains no object.</p> <p>The <a href="#">Element3D</a> class represents anything that can go into a <a href="#">Group3D</a> and, therefore, can be displayed by the Objects property of the <b>RB3DSpace</b>. All such objects have a position, orientation, and scale. Note that the <b>Rb3DSpace</b>’s Camera is an <a href="#">Element3D</a>, rather than an <a href="#">Object3D</a>.</p> <p>The <a href="#">Trimesh</a> class allows you to create and manipulate triangular meshes on-the-fly. It works with four “helper” classes that get and set trimesh data: <a href="#">TriangleList</a>, <a href="#">UVList</a>, <a href="#">VectorList</a>, and <a href="#">ColorList</a>. Also, the <a href="#">Material</a> class lets you create and manipulate colors and textures on-the-fly.</p> <p><b>Specifying a Location in 3D Space</b></p> <p>Any location in a 3D space can be specified the three coordinates: X, Y, and Z. If you imagine the world as one infinite, flat plane, you would pick some arbitrary point as the “origin.” Then you can specify any location by a distance East/West from the origin, a distance North/South, and a height/depth.</p> <p>The arbitrary center of the virtual universe is called the “origin.” By convention, the three coordinates in 3D graphics are called X, Y, and Z and the origin is the point 0,0,0. The X-axis is the line through the origin formed by assuming Y=0 and Z=0; similarly, the XY plane is the infinite plane through the origin where Z=0.</p> <p>The standard interpretation of the axes is that the XZ plane is the ground (or parallel to the ground), and the Y coordinate is height. In REALbasic, the three axes in REALbasic are arranged such that if you’re looking in the -Z direction, with +Y pointing up, then +X is to the right. (This is the default orientation of the camera, too.) So, if you’re looking in this direction, than an increase in the X coordinate of an object’s position will cause it to move to the right from your point of view. An increase in Z will cause it to move closer to you, and a decrease in Z will cause it to move further away.</p> |
|-------------------|---|

### The FieldOfView Property

FieldOfView is the angle, in degrees, that is shown in the area of the **RB3DSpace** control. A normal human has a field of view of about 180 degrees, but the monitor itself is only on the order of 10 degrees across for a typical viewing distance. If you try to compress a large field of view down to a display only 10 degrees across, you'll cause a "fisheye" effect which looks very unnatural. On the other hand, if you use only a 10 degree field of view in a typical 3D environment, users will feel like they've got tunnel vision and will constantly be panning the camera around. The FieldOfView you choose is a balance between these two conflicting constraints: You want to see a useful amount of the scene, but you want it drawn as realistically as possible.

A typical FieldOfView value for many 3D games is about 40 to 50 degrees. However, you can provide a "zoom" (telescopic vision) option simply by reducing the FieldOfView temporarily — a smaller FieldOfView means that you're displaying a smaller part of the world in the same area on screen, hence everything in that part looks bigger.

Finally, the telescopic quality of a small FieldOfView also allows you to reduce (and nearly eliminate) perspective when you need to. Put the camera very far away from the scene and use a small FieldOfView; this is equivalent to looking at the ground from orbit, and under those conditions, there is almost no foreshortening.

### The Hither and Yon Properties

These are standard 3D graphics jargon for the "near-far" (Z) axis. Nothing closer to the camera than the Hither distance gets drawn; and if an object happens to be halfway straddling Hither, it will simply be cut in half. Everything closer than Hither is clipped out of the scene (thus, the "near clipping plane"). Similarly, nothing farther from the camera than Yon is drawn.

If there were nothing more to it, you would simply set Hither to zero and Yon to some very large number so that things aren't clipped. But in reality, that's a very bad idea. The reason is that video cards have a limited amount of resolution in the "depth buffer" that's used to keep track of what's in front of what. The ratio Yon/Hither must be evenly divided by the number of bits in the card's depth buffer. If that ratio is very large, then the depth resolution will be inadequate and you'll see all sorts of visual artifacts — objects mixed together in odd ways, further-away objects being drawn in front of closer ones, etc. So it's very important to the quality of your rendering that you keep the Yon/Hither ratio as small as possible. Increase Hither and decrease Yon as much as you can in your application.

### Orienting an object

There are an infinite number of orientations that all point towards a certain point of interest (or "POI"). Prove this to yourself by taking (or imagining) an actual camera with a view-finder. Look through the view-finder at some object, say a tree down the street. Now, without losing sight of the tree, rotate the camera around the axis between it and the POI. There are an infinite number of different orientations you could stop the camera in, even though they all look at the tree.

The first thing one must do is define some additional constraint on the orientation. A common one is to keep the object as “upright” as possible. There are still many ways to interpret this, but here’s one solution:

```
Sub PointObjectAt(obj As Object3D, POI As Vector3D)
    // Make object 'obj' point towards the point of interest 'POI',
    // by first doing a yaw (from the default orientation), and
    // then a pitch. The object's previous orientation is not
    // involved.

    Dim yawAngle As Double
    Dim pitchAngle As Double
    Dim v1, v2 As Vector3D

    // Find the yaw angle needed to get in the right general direction
    // (i.e., ignoring Y, get it facing the right XZ direction).
    // The Atan2 function provides a very easy way to find this.
    yawAngle = Atan2(POI.x-obj.position.x, POI.z-obj.position.z)

    // Set the object's orientation to a yaw by that amount.
    obj.orientation.SetRotateAboutAxis 0,1,0, yawAngle

    // Now, find the pitch angle needed to account for Y.
    // We use a dot product to get the angle between a vector in
    // the direction the object's facing now, and a vector that
    // points towards the POI.
    v1 = New Vector3D
    v1.z = 1.0
    v1 = obj.orientation.Transform(v1)
    v2 = New Vector3D
    v2.x = POI.x - obj.position.x
    v2.y = POI.y - obj.position.y
    v2.z = POI.z - obj.position.z
    v2.Normalize
    pitchAngle = Acos( Min(Max(v1.Dot(v2),-1.0),1.0) )
    If v2.y > 0 then
        pitchAngle = -pitchAngle
    end if

    // Apply that pitch.
    obj.Pitch pitchAngle
```

#### Orbiting an Object around a POI

This is another task that can be interpreted in many ways. Start by figuring out exactly what want to do. If the object in “orbit” is a camera that you want to move around a Point Of Interest (POI), there are several reasonable options:

- Move the camera on the surface of an imaginary, Y-axis-aligned cylinder which surrounds the object. Left-right movements will move the camera laterally around the

cylinder, and up-down movements will move the camera up and down the side of the cylinder. Note that this means that the camera gets further away from the POI when you move up or down. But it also means that the left-right/up-down directions are always very well defined and obvious.

- Move the camera on the surface of an imaginary sphere around the object. The camera object will have a certain direction it is facing (towards the center), and left-right/up-down will move the camera on the surface of the sphere relative to that direction. This keeps a constant distance from the POI, but the controls can get confusing since the moving object can easily end up upside down or sideways.
- Move the camera on the surface of an imaginary sphere around the object. In this case, let left-right control the longitudinal position of the camera, and up-down control the latitude. This keeps the camera a constant distance from the POI, and is fairly easy to control since (like the cylindrical case) the left-right/up-down directions are always well defined.

In all these cases, the position of the object can be described by two parameters. Left-right controls increase or decrease one of the parameters, and up-down controls adjust the other. The only difference among them is how these parameters are converted into 3D coordinates (and, therefore, what the parameters mean). In all cases, you could add a third parameter, radius, to change the size of the orbit.

Here is the code for the first option:

```
Function OrbitCylinder(POI As Vector3D, radius As Double, angle As Double,  
altitude As Double) As Vector3D  
    // Compute a position on a virtual Y-aligned cylinder, centered on POI,  
    // with the given radius, angle, and altitude.  
Dim out As Vector3D  
out = New Vector3D  
out.x = POI.x + radius * Cos(angle)  
out.z = POI.z + radius * Sin(angle)  
out.y = POI.y + altitude  
Return out
```

## Example

This example shows how to load a 3DMF file into an **RB3DSpace**. This code is placed in the Open event. Note that “3DMF” should be declared in via the [FileType](#) class or with the File Type Sets Editor, with Type code “3DMF”. In this example, the File Type

Set called “FileTypes1” contains the required file types. The All method does automatic text conversion and returns all the file types defined in this set

```
Dim f As FolderItem
Dim obj As Object3D

//allow only 3DMF files types to be shown
f = GetOpenFolderItem(FileTypes1.All)
If f <> Nil then

    // create an object
    obj = New Object3D

    // give the object a shape specified by the 3DMF
    obj.AddShapeFromFile f

    If obj.Shapecount < 0 then
        MsgBox "No valid 3DMF data found in "+.DisplayName+".
        Return
    End if

    // add the object to this Rb3DSpace
    Me.objects.Append obj
    end if

    // move the camera away from the origin so that we can see the 3DMF
    Dim v as Vector3D
    v=New Vector3D

    v.X=0
    v.Y=2
    v.Z=10
    Me.Camera.Position= v
```

The last line moves the camera away from the object so that the 3DMF is visible. By default, the camera is at location 0,0,0 (a.k.a., the “origin” in the 3D space). By default, objects are also loaded at the same place. This means that the camera is inside your object, which is usually not where you want it to be. Fix this by moving the camera away from the origin. How far you need to move depends on how big your object is.

The Position property is a [Vector3D](#) that specifies the X, Y, and Z coordinates of the camera. You can pass a [Vector3D](#) as in this example or with lines like:

```
Me.Camera.Position.Z=10
```

### See Also

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes.

# RBScript Control

Used to execute REALbasic code within a running (compiled) application.

**Super Class** [Object](#).

## Properties

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| Context      | <a href="#">Object</a>  | Object (e.g., window or class) that is made available to the RBScript. For example, you can create a custom class and assign it to the Context property. The script would then have access to the methods and properties of this class. The methods become global methods and the properties become global properties.                       |
| EncodingFont | <a href="#">String</a>  | Useful only for double-byte character systems. This font is used to determine what script system is used to interpret strings in functions such as <a href="#">Len</a> and <a href="#">Mid</a> . For example, to force interpretation of strings as Japanese script, set this property to "Osaka". If left blank, the system script is used. |
| Source       | <a href="#">String</a>  | The source code the compiler will run.   |
| State        | <a href="#">Integer</a> | State of the script compiler:<br>0=Ready<br>1=Running<br>2=Complete<br>3=Aborted.  |

## Methods

| Name         | Parameters                          | Description  |
|--------------|-------------------------------------|--|
| PartialRun   | millisec as <a href="#">Integer</a> | Runs the code <i>Source</i> until it is done or <i>millisec</i> has elapsed. |
| Precompile() |                                     | Runs the parser immediately instead of waiting until the next call to Run.   |
| Reset()      |                                     | Rewinds the compiler to zero.  |
| Run()        |                                     | Runs the code <i>Source</i> until it is done.                                |

## Events

| Name          | Parameters  | Description   |
|---------------|---|---|
| CompilerError | line as <a href="#">Integer</a> ,<br>errorNumber as <a href="#">Integer</a> ,<br>errorMsg as <a href="#">String</a> | A compiler error has occurred. This could be caused by a syntax error in the code. See the error numbers in Notes, below. |

| Name         | Parameters  | Description  |
|--------------|---|--|
| Input        | prompt as <a href="#">String</a>  | The script requests input from the end user. Returns a <a href="#">String</a> . Used in IDE scripts to get input from the user.                        |
| Print        | msg as <a href="#">String</a>   | The compiler is returning the results of the script in <i>msg</i> . Used in IDE scripts to display the passed string via a <a href="#">MsgBox</a> box. |
| RuntimeError | line as <a href="#">Integer</a> ,<br>error as<br><a href="#">RuntimeException</a> | A runtime error has occurred.  |

## IDE Scripting

The REALbasic IDE has its own RBScript editor that you can use to automate many aspects of the development process. Choose File ▶ IDE Scripts ▶ New IDE Script to access this code editor. You can write scripts manually or turn on the “record” feature to record your actions within the IDE. Once you are in record mode, click Record again to stop recording. Every menu item and toolbar button has a command name that can be called from an IDE script via the DoCommand. If you have saved scripts, they are added to the IDE Scripts submenu. You will be given a chance to save your script when you close the IDE Script editor.

### Recording

When you click the Record button in the IDE Script editor, the editor goes into “recording” mode and the button stays down as long as it’s in this mode. While recording, any scriptable action you do in the IDE is added to the editor as RBScript code. You then return to the Script editor, stop the recording by clicking the Record button again, and edit the script -- deleting unimportant commands, changing constants to variables as required, and so on.

Use the record feature to get the names of commands to pass to the DoCommand method.

Script recording is a powerful feature for automating repetitive tasks; it does most of the work for you, freeing you from having to know the syntax for most actions.

Your scripts have access to all the standard functions in RBScript, including string manipulation, math, and so on. The RBScript Print command displays a message box, and the Input command gets input via a simple modal dialog. As in any RBScript, you can also write methods, classes, or modules within your script if needed to help organize your code (though many scripts will be so simple as to not need them).

### IDE Scripting Commands

In addition to the standard RBScript functions, there are additional functions and properties that are defined only for IDE scripts, which manipulate projects in the IDE in various ways. These functions are detailed in the table that follows. Most of them operate on the frontmost Project Window, and on the current project item or method within that window. However, there are ways to change the current window or location for cases where your script needs to act upon a particular item.

Most of the things your script can do fall into one of the following categories and use the indicated RBScript commands:

- Changing the context (SelectWindow, Location)
- Changing project item properties (PropertyValue)
- Changing build settings (BuildLanguage, BuildLinux, etc.)
- Executing commands, as if chosen from a toolbar or menu (DoCommand)
- Interacting with the user (Print, Input, Beep, Speak)
- Inspecting or changing source code (Text, SelText, SelStart, SelLength)

There are also a few utility functions that don't fit neatly into the above categories (e.g., Clipboard).

Here is the list of IDE scripting commands.

| Name            | Type                    | Description   |
|-----------------|-------------------------|---|
| Beep            |                         | Plays the system beep.  |
| BuildLanguage   | <a href="#">String</a>  | Gets or sets the current build language.  |
| BuildLinux      | <a href="#">Boolean</a> | Gets or sets whether to build for the Linux platform.   |
| BuildMacClassic | <a href="#">Boolean</a> | Gets or sets whether to build for the Mac OS classic platform.  |
| BuildMacMachO   | <a href="#">Boolean</a> | Gets or sets whether to build a Mach-O Macintosh application.   |
| BuildMacPEF     | <a href="#">Boolean</a> | Gets or sets whether to build a PEF Macintosh application.  |
| BuildRegion     | <a href="#">String</a>  | Gets or sets the region code to be used in the built application.   |
| BuildWin32      | <a href="#">Boolean</a> | Gets or sets whether to build for the Windows platform.   |
| Clipboard       | <a href="#">String</a>  | Gets or sets the contents of the copy/paste buffer, aka "clipboard."  |
| DoCommand       |                         | Parameter is cmdName as <a href="#">String</a> . Executes the passed command, as if the user had pressed the corresponding toolbar button or chosen the corresponding menu item. Every such command in the IDE has a unique name which is used as the argument for DoCommand. To find the name of a particular command, use the script recording feature. |

| Name           | Type                   | Description  |
|----------------|------------------------|--|
| DoShellCommand | <a href="#">String</a> | Parameters are cmd as <a href="#">String</a> and timeout as <a href="#">Integer</a> . The default value for timeout is 3000. Executes a shell command. The shell environment is set up with the following variables:<br>IDE_PATH to point to the directory containing the IDE; PROJECT_PATH to point to the directory containing the project in the frontmost window; and PROJECT_FILE to point to the actual project file. The command is executed in synchronous (blocking) mode, and returns any output as the result.                      |
| EndOfLine      | <a href="#">String</a> | Returns the default line ending for the platform on which the application is running. This is also the line separator in source code text, i.e., that returned by Text and SelText.  |
| Location       | <a href="#">String</a> | Gets or sets the current location of the project, as shown in the Location field in the Main Toolbar. A script can use this both to see its current location and to navigate to another part of the project.   |
| OpenFile       |                        | Parameter is path as <a href="#">String</a> . Attempts to open a REALbasic project file specified by the passed path. This can be either a native or shell path.   |
| ProjectItem    | <a href="#">String</a> | Returns the name of the current project item, that is, the project item that is currently being edited or is selected in the IDE tab bar.  |
| PropertyValue  |                        | Parameter is propName as <a href="#">String</a> . Gets or sets the value of a project name property. The propName argument may be just the property name, in which case the property is found on the current project item; or it may be the name of a project item plus the property name, separated by a dot. You may also use the special keyword "App" to refer to the blessed Application subclass, even if that is not actually named App. The following are all valid property references: "Width", "Window1.Width", "App.MajorVersion". |
| RunScript      |                        | Parameter is scriptName as <a href="#">String</a> . Runs the passed script, found in the Scripts folder, either next to the IDE or next to the frontmost project file.   |
| SelectWindow   |                        | Parameter is windowTitle as <a href="#">String</a> . Brings the passed window to the front. If there is more than one window named windowTitle, then the one closest to the front is the one brought to the front.   |
| SelectWindow   |                        | Parameter is Index as <a href="#">Integer</a> . Brings a window to the front identified by its current index in the Window list (see the WindowCount and WindowTitle functions).   |

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| SelLength    | <a href="#">Integer</a> | Gets or sets the length in characters of the current selection in the current Code Editor, assuming that there is a selection.   |
| SelStart     | <a href="#">Integer</a> | Gets or sets the offset of the selection or insertion point in the current Code Editor.  |
| SelText      | <a href="#">String</a>  | Gets or sets the selected text in the current Code Editor. Note that after assigning to SelText, the selection will be empty and positioned right after the inserted text. (This is the same behavior as EditField.SelText.)   |
| Speak        |                         | Parameters are text as <a href="#">String</a> and Interrupt as <a href="#">Boolean = False</a> . Speaks the passed text via the system's speech synthesis engine. If interrupt is set to <a href="#">True</a> , then it interrupts any previous speech. Otherwise, it waits for the previous speech to finish. This is the same behavior as the built-in Speak command.  |
| Sublocations |                         | Parameter is baseLocation as <a href="#">String</a> . Returns all the locations within the given base location as a space-delimited string. For example, if baseLocation is "App", then this might return "App.Activate App.CancelClose App.Close" (and so on). The set of locations returned should be the same ones suggested by autocompletion in the Location field within the IDE. If baseLocation is an empty string, then this returns the set of project items in the frontmost project. |
| Text         | <a href="#">String</a>  | Gets or sets the entire text of the current Code Editor.   |
| WindowCount  | <a href="#">Integer</a> | Returns the number of windows in the IDE.  |
| WindowTitle  | <a href="#">String</a>  | Parameter is index as <a href="#">Integer</a> . Returns the title of a window indicated by the passed Index in the window list. An index of zero is the frontmost window.  |

## Examples

The following script issues the traditional greeting via a MsgBox:

```
Print "Hello world!"
```

This script sets the ShortVersion to “1.0J”, sets the build language to Japanese, and then builds a Mac version of the application:

```
PropertyValue("App.ShortVersion") = "1.0J"
BuildLanguage="Japanese"
BuildWin32=True
BuildMacMachO=True
DoCommand "BuildApp"
```

Here is a script that automates a build for Windows and Linux. You may need to use the appropriate SelectWindow commands to display the correct window for the commands.

```
PropertyValue("App.StageCode")="1"
DoCommand "BuildSettings"
PropertyValue("App.Windows.AppName")="TextEdit.exe"
BuildWin32=True
BuildLinux=True
BuildMacPEF=False
BuildMacMachO=False
BuildMacClassic=False
BuildRegion="United States"
BuildLanguage="English"
DoCommand "BuildApp"
```

### Notes

The **RBScript** language is an implementation of the REALbasic language that allows end users to write and execute REALbasic code within a compiled application. Scripts are compiled into machine language, rather than being interpreted.

Since **RBScript** is a class, you use it by creating an instance of this class either via code or by adding an **RBScript** control to a window. The easiest way to use **RBScript** is to assign the code to the Source property of the **RBScript** object and call the Run method.

To provide information to an **RBScript** while it's running, use the Input function. This function calls the Input event of the **RBScript** object where you can return the information you wish returned by the Input function in your **RBScript** code. In the following example, the results of the Input function are assigned to a variable:

```
Dim Years, Days as Integer
Years = Val(Input(" "))
Days = Years * 365
```

The Input function takes a **String** that can be used to provide a prompt in case you are going to provide a dialog box in which the user enters the information. Since the Input function returns a **String** and we want to store the value as an integer, the **Val** function is used to convert the string to an integer. In this case, if the number of years is going to be entered into an **EditField** called Editfield1, then the Input event of the **RBScript** object would look like this:

```
Function Input(prompt as String) as String
Return Editfield1.text
```

When the Run method of the **RBScript** object is called, the code will be compiled and then executed. Since the Input function is called, the Input event of the **RBScript** object is executed and the contents of the Text property of the **EditField** is returned and assigned to the Years variable. Then the Days value is calculated.

## Getting Information From RBScript

You obtain data from the **RBScript** using the Print method. This method takes a [String](#) and passes it to the Print event of the **RBScript** object. Here is the example modified to use the Print function:

```
Dim Years, Days as Integer
Years = Val(Input(" "))
Days = Years * 365
Print Str(days)
```

You access the value passed to the Print method through the Print event. For example, if you want to assign the value to the Text property of a [StaticText](#) object, the code for the Print event of the **RBScript** object would look like this:

```
Sub Print(msg as String)
    StaticText1.text = msg
```

## Handling Errors in Your Code

If an error occurs while **RBScript** is compiling your code, the CompilerError event of the **RBScript** object will be called and will be passed appropriate error information so you can then decide how to handle the error. If the error occurs while the code is running, the RuntimeError event of the **RBScript** object is called and is passed appropriate error information. You can then decide how to respond to the error.

## Variables and Constants

All numeric and alphanumeric operators are supported:

[+, -](#), [\\*](#), [/](#), [\](#), [Mod](#), [<](#), [=](#), [>](#), [<=](#), [>=](#), [<>](#).

Logical operators: [And](#), [Not](#), [Or](#).

All forms of comments are supported: [!](#), [//](#), and [REM](#).

## Data Types

**RBScript** supports the following data types:

- [Integer](#)
- [Single](#)
- [Double](#)
- [Boolean](#)
- [String](#)
- [Color](#)
- [Object](#)
- [Variant](#)

Arrays can use any of these types.

## Control Structures

All the control structures specified in this *Language Reference* are supported. This includes:

- [Function...](#)
- [Sub...](#)
- [For...](#) Next
- [Do...](#) Loop
- [If...](#) Then... End If
- [Select Case...](#) End Select
- [While...](#) Wend
- [Return](#) statement

**Classes** You can create a class using the Class...End Class structure. A new class can have properties, methods, and events.

```
Class NewClass
    Dim i as Integer //Property definitions..
    Dim c as Color
    Sub myMethod (x as Integer, y as Integer)
        //Method definition goes here
    End Sub
    //Class definition
End Class
```

A class can be subclassed from another class using the “Inherits” declaration.

**Modules** You can create modules using the Module structure. For example:

```
Module Foo
    Dim bar As Integer
    Sub Baz ()
        bar = 42
    End Sub
End Module
```

Modules are similar to REALbasic modules. The differences are that properties are always protected and constants are not allowed. You can get the effect of a public property or constant by simply putting the [Dim](#) or [Const](#) statement outside of the module.

**TypeCasting** Typecasting is supported. Use the desired type’s name as a function whose only argument is the object reference you want to cast. If it fails, it will trigger an [IllegalCastException](#) error.

**“Super” keyword** The new “Super” keyword lets you call overridden methods without having to specify which class the method came from. This works for constructors as well. You can call overridden superclass methods using REALbasic’s `classname.methodname()` syntax.

**Inheritance** Class inheritance is implemented with the “Inherits” declaration.

```
Class MyNewSubClass
Inherits NewClass
//continue coding here....
End Class
```

**Events** Classes can declare new events using a New Event declaration.

```
Class myClass
>New Event myEvent (myParameter as DataType) as DataType
End Class
```

Methods can handle such events using the Handles Event declaration. For example:

```
Class myClass
>New Event myEvent (myParameter as DataType) as DataType
End Class
Class mySubClass
Inherits myClass
Function myEvent (myParameter as DataType) Handles Event
//continue with function here
End Function
End Class
```

**Interfaces** RBScript now supports interfaces, declared by the Interface...End Interface structure. Interface methods are declared with a [Sub](#) or [Function](#) line just as they would be declared inside a class, but they have no contents and need no End Sub or End Function line:

```
Interface MovableItem
Sub Move( x As Integer, y As Integer)
Function LocationX() As Integer
Function LocationY() As Integer
End Interface
```

*Note* Function declarations with no parameters must have empty parentheses, as shown above.

A class may declare that it implements an interface with the Implements keyword:

```
Class Box
Implements MovableItem
End Class
```

Classes can implement any number of interfaces, but they must provide all of the methods listed in the interface. Interfaces can be used in all the situations as in REALbasic.

## Input and Output

| Function                             | Comments   |
|--------------------------------------|--|
| <code>Input(Prompt as String)</code> | Retrieves input from the user. This function triggers the Input event.             |
| <code>Print(String)</code>           | Sends output to the implied output device. This function triggers the Print event. |

## Functions

Standard library functions are supported as follows. These functions work as specified in this *Language Reference* unless comments indicate otherwise. String functions can be called using the syntax of methods, just as in REALbasic.

| Function   | Comment             |
|--|---------------------|
| <code>Abs(Double) As Double</code>                   |                     |
| <code>Acos(Double) As Double</code>                  |                     |
| <code>Asc(String) As Integer</code>                  |                     |
| <code>AscB(String) As Integer</code>                 |                     |
| <code>Asin(Double) As Double</code>                  |                     |
| <code>Atan(Double) As Double</code>                  |                     |
| <code>Atan2(Double, Double) As Double</code>         |                     |
| <code>BitwiseAnd(Integer, Integer) As Integer</code> |                     |
| <code>BitwiseOr(Integer, Integer) As Integer</code>  |                     |
| <code>BitwiseXor(Integer, Integer) As Integer</code> |                     |
| <code>ByRef keyword</code>                           |                     |
| <code>ByVal keyword</code>                           |                     |
| <code>CDbl(String) As Double</code>                  | Identical to Val(). |
| <code>Ceil(Double) As Double</code>                  |                     |
| <code>Chr(Double) As String</code>                   |                     |
| <code>ChrB(Double) As String</code>                  |                     |
| <code>CMY</code>                                     |                     |
| <code>Color</code>                                   |                     |
| <code>Const name = value</code>                      |                     |
| <code>Cos(Double) As Double</code>                   |                     |
| <code>CountFields(String, String) As Integer</code>  |                     |
| <code>CStr(Double) as String</code>                  |                     |

| Function   | Comment   |
|--|---|
| <a href="#">Dim</a>  | Allows you to declare multiple variables with different types rather than only multiple variables of the same type.<br>Empty arrays can be created by specifying -1 elements. |
| <a href="#">Do...Loop</a>                                  |   |
| <a href="#">Exit</a>                                       |   |
| <a href="#">Exp(Double) As Double</a>                      |   |
| <a href="#">False</a>                                      |   |
| <a href="#">Floor(Double) As Double</a>                    |   |
| <a href="#">For...Next</a>                                 | Allows floating point loop counters and step values.  |
| <a href="#">Format(Double, String) As String</a>           |   |
| <a href="#">Function</a>                                   | Functions can be called before they are defined.  |
| <a href="#">Goto keyword</a>                               |   |
| <a href="#">Hex(Integer) As String</a>                     |   |
| <a href="#">HSV</a>  |   |
| <a href="#">If...Then...Else</a>                           |   |
| <a href="#">InStr(Integer, String, String) As Integer</a>  |   |
| <a href="#">InStrB(Integer, String, String) As Integer</a> |   |
| <a href="#">IsA operator</a>                               | Can be used with <a href="#">Object</a> ; it will return <a href="#">True</a> for any non-null object.  |
| <a href="#">Left(String, Integer) As String</a>            |   |
| <a href="#">LeftB(String, Integer) As String</a>           |   |
| <a href="#">Len(String) As Integer</a>                     |   |
| <a href="#">LenB(String) As Integer</a>                    |   |
| <a href="#">Log(Double) As Double</a>                      |   |
| <a href="#">Lowercase(String) As String</a>                |   |
| <a href="#">LTrim(String) As String</a>                    |   |
| <a href="#">Max(Double, Double) As Double</a>              |   |
| <a href="#">Me</a>   |   |
| <a href="#">Microseconds As Double</a>                     |   |
| <a href="#">Mid(String, Integer, Integer) As String</a>    |   |
| <a href="#">MidB(String, Integer, Integer) As String</a>   |   |
| <a href="#">Min(Double, Double) As Double</a>              |   |
| <a href="#">Nil</a>  |   |

| Function  | Comment   |
|---|---|
| <a href="#">NthField(String, String, Integer) As String</a>   |   |
| <a href="#">Oct(Integer) As String</a>                        |   |
| <a href="#">Pow(Double, Double) As Double</a>                 |   |
| <a href="#">Redim</a>   |   |
| <a href="#">Rem</a>   |   |
| <a href="#">Replace(String, String, String) As String</a>     |   |
| <a href="#">ReplaceB(String, String, String) As String</a>    |   |
| <a href="#">ReplaceAll(String, String, String) As String</a>  |   |
| <a href="#">ReplaceAllB(String, String, String) As String</a> |   |
| <a href="#">RGB</a>   |   |
| <a href="#">Right(String, Integer) As String</a>              |   |
| <a href="#">RightB(String, Integer) As String</a>             |   |
| <a href="#">Rnd As Double</a>                                 |   |
| <a href="#">Round(Double) As Double</a>                       |   |
| <a href="#">RTrim(String) As String</a>                       |   |
| <a href="#">Select Case</a>                                   |   |
| <a href="#">Self</a>  |   |
| <a href="#">Sin(Double) As Double</a>                         |   |
| <a href="#">Sqrt(Double) As Double</a>                        |   |
| <a href="#">Str(Double) As String</a>                         |   |
| <a href="#">StrComp(String, String, Integer) As Integer</a>   |   |
| <a href="#">Sub</a>   |   |
| <a href="#">Tan(Double) As Double</a>                         |   |
| <a href="#">Ticks as Integer</a>                              |   |
| <a href="#">Titlecase(String) As String</a>                   |   |
| <a href="#">Trim(String) As String</a>                        |   |
| <a href="#">True</a>  |   |
| <a href="#">Ubound(array) As Integer</a>                      | Supports second parameter for multi-dimensional arrays. |
| <a href="#">Uppercase(String) As String</a>                   |   |
| <a href="#">Val(String) As Double</a>                         | Scientific format not recognized.                       |
| <a href="#">While...Wend</a>                                  |   |

Compiler error numbers returned in *errorNumber* are shown below:

| Error Number | Description  |
|--------------|--|
| 1            | Syntax does not make sense.                                    |
| 2            | Type mismatch.   |
| 3            | Select Case does not support that type of expression.          |
| 4            | The compiler is not implemented (obsolete).                    |
| 5            | The parser's internal stack has overflowed.                    |
| 6            | Too many parameters for this function.                         |
| 7            | Not enough parameters for this function call.                  |
| 8            | Wrong number of parameters for this function call.             |
| 9            | Parameters are incompatible with this function.                |
| 10           | Assignment of an incompatible data type.                       |
| 11           | Undefined identifier.  |
| 12           | Undefined operator.  |
| 13           | Logic operations require <a href="#">Boolean</a> operands.     |
| 14           | Array bounds must be integers.                                 |
| 15           | Can't call a non-function.                                     |
| 16           | Can't get an element from something that isn't an array.       |
| 17           | Not enough subscripts for this array's dimensions.             |
| 18           | Too many subscripts for this array's dimensions.               |
| 19           | Can't assign an entire array.                                  |
| 20           | Can't use an entire array in an expression.                    |
| 21           | Can't pass an expression as a <a href="#">ByRef</a> parameter. |
| 22           | Duplicate identifier.  |
| 23           | The backend code generator failed.                             |
| 24           | Ambiguous call to overloaded method.                           |
| 25           | Multiple inheritance is not allowed.                           |
| 26           | Cannot create an instance of an interface.                     |
| 27           | Cannot implement a class as though it were an interface.       |
| 28           | Cannot inherit from something that is not a class.             |
| 29           | This class does not fully implement the specified interface.   |
| 30           | Event handlers cannot live outside of a class.                 |
| 31           | It is not legal to ignore the result of a function call.       |
| 32           | Can't use "Self" keyword outside of a class.                   |
| 33           | Can't use "Me" keyword outside of a class.                     |
| 34           | Can't return a value from a Sub.                               |
| 35           | An exception object required here.                             |
| 36-39        | Obsolete.  |
| 40           | Destructors can't have parameters.                             |
| 41           | Can't use "Super" keyword outside of a class.                  |
| 42           | Can't use "Super" keyword in a class that has no parent.       |

## RbScriptAlreadyRunningException error

Occurs if the user tries to modify an [RBScript's](#) source code while it is running or you try to change the Context object while the script is running. Don't do this.

**Super Class** [RuntimeException](#).

**See Also** [RBScript](#) class; [Exception](#), [Try](#) blocks.

---

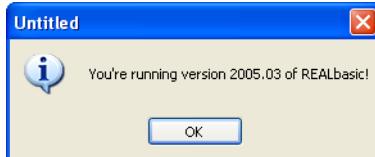
## RBVersion Constant

Reports the major and minor version number of REALbasic.

**Syntax** `result=RBVersion`

| Part   | Type                   | Description   |
|--------|------------------------|---|
| result | <a href="#">Double</a> | Constant indicating major and minor version number. |

**Example** The following line displays the version of REALbasic that is running.



```
MsgBox "You're running version "+Str\(RBVersion\)+" of REALbasic!"
```

**Note**

Version 2006 returns a number of the form 2006.xx, where xx is the Release number.

When you build your application, you can enter version information about your application in the App class's Properties pane. This information is stored in your application's 'vers' resource. For more information, see the chapter on building applications in the *User's Guide*.

You can use **RBVersion** in an [#If](#) statement to determine whether a use is running a particular version of REALbasic. Based on the result, you conditionally compile code that is available only for that version.

**Example**

The following includes code only for applications built using REALbasic version 5 or above.

```
#If RBVersion>5
//include 5.0 code here
#endif
```

**See Also**

[DebugBuild](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetWin32](#) constants; [#If](#) statement.

## RBVersionString Constant

Returns the version of REALbasic as a [string](#).

**Syntax**

**result=RBVersionString**

| Part   | Type                   | Description                           |
|--------|------------------------|---------------------------------------|
| result | <a href="#">String</a> | Version of REALbasic that is running. |

**Example**

The following line displays the version of REALbasic that is running:



```
MsgBox "You're running version "+Rbversionstring+" of REALbasic!"
```

**See Also**

[DebugBuild](#), [RBVersion](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetWin32](#) constants; [#If](#) statement.

## Readable Class Interface

Provides “specs” for reading with the **Readable** class interface.

## REALdatabase Class

---

### Methods

| Name      | Parameters   | Description   |
|-----------|--|---|
| EOF       |  | Returns a <a href="#">Boolean</a> . Returns <a href="#">True</a> when the read reaches the end of the stream.   |
| Read      | Count as<br><a href="#">Integer</a><br>[,enc as<br><a href="#">TextEncoding</a><br>] | Reads <i>Count</i> bytes from the input stream and returns a <a href="#">String</a> . If provided, the optional parameter <i>Enc</i> specifies the text encoding to be defined for the <a href="#">String</a> to be read. |
| ReadError |  | Returns a <a href="#">Boolean</a> . Returns <a href="#">True</a> if an error of any type occurred.  |

### Notes

If you implement this interface in your application, you must implement the methods with the parameters as shown here.

Class interfaces are described in the chapter on “Classes” in the *REALbasic User’s Guide*.

### See Also

[BinaryStream](#), [IPCSocket](#), [Serial](#), [StdIn](#), [TCPSocket](#), [TextInputStream](#) classes.

---

## REALdatabase Class

The **REALdatabase** class provides access to the REALdatabase data source, a.k.a., database engine or database back-end that was used in REALbasic 5.5. This class is obsolete and included for compatibility reasons and for users who need to deploy a database on Mac OS “classic.” The [REALSQLdatabase](#) class is recommended for all other uses.

**Super Class** [Database](#) class.

### Properties

| Name         | Type                       | Description   |
|--------------|----------------------------|---|
| DatabaseFile | <a href="#">FolderItem</a> | The <a href="#">FolderItem</a> for the REALdatabase file. Set this property to a directory to store the database as separate files rather than as a <a href="#">VirtualVolume</a> . |

### Methods

| Name               | Parameters | Description  |
|--------------------|------------|--|
| CreateDataBaseFile |            | Creates a new REALdatabase. It uses the DatabaseFile property as the <a href="#">FolderItem</a> for the database to create. Returns a <a href="#">Boolean</a> — <a href="#">True</a> if the new database was created successfully, and <a href="#">False</a> if otherwise. |

| Name              | Parameters                          | Description   |
|-------------------|-------------------------------------|---|
| GetSchemaData     |                                     | Returns a <a href="#">String</a> containing a representing of the current database schema. GetSchemaData will fail if a transaction is in progress.                                 |
| LastRowID         | tablename as <a href="#">String</a> | Returns as <a href="#">Integer</a> the value of the last rowID added to the specified table.  |
| ReplaceSchemaData | NewSchema as <a href="#">String</a> | Replaces the current database schema with the schema passed. ReplaceSchemaData will fail if a transaction is in progress. Also, the change cannot be rolled back. Use with caution. |

**Notes**

**REALdatabase** tables may be added, dropped, or modified in various ways (columns or indexes added or dropped) without losing or recopying the existing data.

The new **REALdatabase** supports a subset of SQL/92 and SQL/99 (details below), including queries that involve self-joins, aggregate functions, and more. For the set of features the new **REALdatabase** engine supports, its syntax is fully SQL compliant (with a few minor extensions, like the Boolean data type). It also returns standard SQL error codes.

Every new **REALdatabase** table has a special identifier column called “\_rowid” which is a unique integer identifier for that row. The identifier is added automatically and serves as a convenient join field for building relational databases.

A call to SQLSelect returns a dynamic [RecordSet](#); you can move forward, backward, or jump to the beginning or end as much as you like.

The **REALdatabase** engine supports transactions, both for schema changes and for data changes. A transaction is started automatically when you make any change to the database and is ended by calling either the Commit or Rollback methods of the [Database](#) class.

**Result Codes**

The **REALdatabase** engine sets the [Database](#) class’s Error flag after each operation and returns values in the ErrorCode and ErrorMessage properties. When the Error flag is [False](#), the ErrorCode is “0” and the ErrorMessage is “Success”. If the Error flag is [True](#), the following codes and messages are returned in ErrorMessage. The SQL Error code appears in brackets following the text of the message.

| SQL Error Code | Message                             |
|----------------|-------------------------------------|
| 02000          | No records found.                   |
| 01004          | Warning: String truncated on right. |
| 01S09          | Warning: Invalid SQL keyword.       |
| 07000          | Error: Dynamic SQL error.           |
| 42S01          | Error: Table already exists.        |
| 42S02          | Error: Table not found.             |

## REALdatabase Class

---

| SQL Error Code | Message  |
|----------------|--|
| 42S11          | Error: Index already exists.                         |
| 42S12          | Error: Index not found.                              |
| 42S21          | Error: Column already exists.                        |
| 42S22          | Error: Column not found.                             |
| 21S01          | Error: Insert value list does not match column list. |
| 22000          | Error: Data exception.                               |
| 22023          | Error: Invalid parameter.                            |
| 25000          | Error: Invalid transaction state.                    |
| 0A000          | Error: Unsupported SQL feature.                      |
| 72000          | Error: Internal SQL processor error.                 |

### Data Types

The **REALdatabase** engine supports the following data types:

| Type      | Code | Comments  |
|-----------|------|---|
| Integer   | 3    | 32-bit signed integer.  |
| VarChar   | 5    | Text up to $2^{31}$ bytes <sup>a</sup> .  |
| Double    | 7    | 64-bit floating-point number.   |
| Date      | 8    | Day, in YYYY-MM-DD format. The <a href="#">Date</a> class's SQLDate property supports this format.                      |
| Time      | 9    | Time, in HH:MM:SS format.   |
| TimeStamp | 10   | Time stamp, in YYYY-MMM-DD HH:MM:SS format. The <a href="#">Date</a> class's SQLDateTime property supports this format. |
| Boolean   | 12   | Boolean value, TRUE or FALSE <sup>b</sup> .   |
| Binary    | 14   | Binary data up to $2^{31}$ bytes  |
| String    | 18   | Text up to $2^{31}$ bytes <sup>a</sup> .  |

a. Text fields in the REALdatabase store text encoding information as well as the text itself. When you insert records, the encoding of the text fields is stored along with the contents of the fields. When you retrieve the data, the encoding is restored.

b. Boolean is not a SQL data type and TRUE or FALSE are not SQL keywords. These are REALbasic extensions to SQL.

### Creating a REALdatabase

A REALdatabase can be created via the **REALdatabase** class as shown in the first example.

No table can be greater than 2 GB total size. When storing files in a [VirtualVolume](#), the entire database must also be under 2 GB. Row IDs increment automatically and are never reused; so you can't insert more than  $2^{32}$  (about 4 billion) records into any table.

**Examples**

The following example creates a new REALdatabase:

```
Dim db as REALdatabase
Dim f as FolderItem
f=New FolderItem("mydb")
db=New REALdatabase
db.databaseFile=f
If db.CreateDatabaseFile then
    //proceed with database operations...
else
    MsgBox "Database not created"
end if
```

The following example opens an existing REALdatabase.

```
Dim dbFile as FolderItem
Dim db as REALdatabase
db=New REALdatabase
dbFile = GetFolderItem("Pubs")
db.DatabaseFile=dbFile
If db.Connect() then
    //proceed with database operations here..
else
    Beep
    MsgBox "The database couldn't be opened."
end if
```

The following example adds a record to a table.

```
Dim dbFile as FolderItem
Dim db as REALdatabase
db=New REALdatabase
Dim rs As New Recordset
dbFile=New FolderItem("Employees")
db.databaseFile=dbFile
if db.error then
    MsgBox db.errormessage
else
    db.sqlexecute ("Insert into Employees (Name,Job,YearJoined) Values " _ 
        +"('Dr.Strangelove','Adviser',1962)")
If db.error then
    MsgBox db.errormessage
else
    db.Commit
end if
end if
```

**See Also**

[Database](#), [DatabaseRecord](#), [RecordSet](#) classes.

# REALSQLdatabase Class

The **REALSQLdatabase** class provides access to the REALSQLdatabase data source, a.k.a., database engine or database back-end. It is based on SQLite, which is described at <http://www.SQLite.org>.

Both the Standard and Professional versions of REALbasic fully support the REALSQLdatabase database engine. Plug-ins for other database engines are supported only in the Professional version of REALbasic.

Use the **REALSQLdatabase** class to open or create REALSQLdatabases programmatically. A REALSQLdatabase can also be created or opened with the Project ▶ Add ▶ Database submenu and designed with a graphical user interface.

The **REALSQLdatabase** is not supported on Mac OS “classic” builds. Use the [REALdatabase](#) engine that was introduced in version 5.5 of REALbasic.

**Super Class** [Database](#) class.

## Properties

| Name             | Type                       | Description   |
|------------------|----------------------------|---|
| DatabaseFile     | <a href="#">FolderItem</a> | The <a href="#">FolderItem</a> for the REALSQLdatabase file. Set this property to a directory to store the database as separate files rather than as a <a href="#">VirtualVolume</a> .  |
| ShortColumnNames | <a href="#">Boolean</a>    | If <a href="#">True</a> , REALSQLdatabase will use the column names as they appear in table schemas for query results whenever it can. Aliased columns and ambiguous columns will be fully qualified. If <a href="#">False</a> , REALSQLdatabase will return column names exactly as they appeared in the original query. The default value is <a href="#">True</a> . |

## Methods

| Name               | Parameters   | Description   |
|--------------------|--|---|
| AttachDatabase     | file as <a href="#">FolderItem</a><br>databaseName as <a href="#">String</a> | Attaches the REAL SQL Database referred to by <i>file</i> to the database. It gives the newly attached database the name <i>databaseName</i> . When a database has been attached, it is possible to do cross-database queries.  |
| CreateDataBaseFile |  | Creates a new REALSQLdatabase. It uses the DatabaseFile property as the <a href="#">FolderItem</a> for the database to create. If the database already exists, it is deleted and a new one is created. Returns a <a href="#">Boolean</a> — <a href="#">True</a> if the new database was created successfully, and <a href="#">False</a> if otherwise. |

| Name           | Parameters                             | Description  |
|----------------|--|--|
| DetachDatabase | databaseName as <a href="#">String</a> | Detaches the passed database, which was previously attached with AttachDatabase.                   |
| LastRowID      |  | Returns as <a href="#">Integer</a> the value of the last rowID added to any table in the database. |

**Notes**

The **REALSQLdatabase** supports a subset of SQL/92 and SQL/99, including queries that involve self-joins, aggregate functions, and more. For the set of features the new **REALSQLdatabase** engine supports, its syntax is fully SQL compliant. It returns SQLite error codes.

A call to SQLSelect returns a dynamic [RecordSet](#); you can move forward, backward, or jump to the beginning or end as much as you like.

The **REALSQLdatabase** engine supports transactions, both for schema changes and for data changes. A transaction is started automatically when you make any change to the database and is ended by calling either the Commit or Rollback methods of the [Database](#) class.

**Result Codes**

The **REALSQLdatabase** engine sets the [Database](#) class's Error flag after each operation and returns values in the ErrorCode and ErrorMessage properties. When the Error flag is [False](#), the ErrorCode is "0" and the ErrorMessage is "Not an error". If the Error flag is [True](#), the following codes and messages are returned in ErrorMessage

REAL SQL Database returns SQLite error codes as follows.

| Error code | Error Message                        |
|------------|--------------------------------------|
| 0          | Not an error                         |
| 1          | SQL logic error or missing database  |
| 2          | Internal SQLite implementation flaw  |
| 3          | Access permission denied             |
| 4          | Callback requested query abort       |
| 5          | Database is locked                   |
| 6          | Database table is locked             |
| 7          | Out of memory                        |
| 8          | Attempt to write a readonly database |
| 9          | Interrupted                          |
| 10         | Disk I/O error                       |
| 11         | Database disk image is malformed     |
| 12         | Table or record not found            |
| 13         | Database is full                     |
| 14         | Unable to open database file         |
| 15         | Database locking protocol failure    |
| 16         | Table contains no data               |
| 17         | Database schema has changed          |

| Error code | Error Message                          |
|------------|--|
| 18         | Too much data for one table row        |
| 19         | Constraint failed                      |
| 20         | Datatype mismatch                      |
| 21         | Library routine called out of sequence |
| 22         | Kernel lacks large file support        |
| 23         | Authorization denied                   |
| 24         | Auxiliary database format error        |
| 25         | Bind or column index out of range      |
| 26         | File is encrypted or is not a database |

**Primary Keys** All REAL SQL Database tables have an Integer Primary Key column. If you don't explicitly define such a column, one will be created for you. You can refer to the INTEGER PRIMARY KEY column using the "rowid" keyword. But, if you don't explicitly define your own INTEGER PRIMARY KEY column, you won't get the 'rowid' column unless you ask for it in queries, such as in the statement:

```
SELECT rowid,* FROM tableName
```

**Data Types** The following table contains information about the data types used by REAL SQL Database.

| FieldType | Description   |
|-----------|---|
| Binary    | Stores code, images, and hexadecimal data. Consult the documentation of your data source for information on the maximum size of a Binary field.   |
| Blob      | Stores a binary object. The REAL SQL Database supports blobs of up to any size. Furthermore, a blob can be stored in a column of any declared data affinity. If you are using another data source, check the documentation of your data source. Blob data can be inserted into a REAL SQL Database using the BlobColumn method of the <a href="#">DatabaseRecord</a> class. |
| Boolean   | Stores the values of TRUE or FALSE.   |
| Date      | Stores year, month, and day values of a date in the format YYYY-MM-DD. The year value is four digits; the month and day values are two digits.  |
| Double    | Stores double-precision floating-point numbers.   |
| Float     | Stores floating-point numeric values with a precision that you specify, i.e., FLOAT (5).  |
| Integer   | A numeric data type with no fractional part. The maximum number of digits is implementation-specific. The REAL database supports 4-byte integers, which provide a range of ±2,000,000,000. If you are using another data source, check the documentation of your data source.   |

| FieldType | Description  |
|-----------|--|
| SmallInt  | A numeric data type with no fractional part. The maximum number of digits is implementation-specific, but is usually less than or equal to INTEGER. The REAL database supports 2-byte smallints, which allow you to store values in the range of ±32,767. If you are using another data source, check the documentation of your data source.   |
| Time      | Stores hour, minute, and second values of a time in the format HH:MM:SS. The hours and minutes are two digits. The seconds values is also two digits, may include a optional fractional part, e.g., 09:55:25.248. The default length of the fractional part is zero.   |
| TimeStamp | Stores both date and time information in the format YYYY-MM-DD HH:MM:SS. The lengths of the components of a TimeStamp are the same as for Time and Date, except that the default length of the fractional part of the time component is six digits rather than zero. If a TimeStamp values has no fractional component, then its length is 19 digits If it has a fractional component, its length is 20 digits, plus the length of the fractional component. |
| VarChar   | Stores alphabetic data, in which the number of characters vary from record to record, but you don't want to pad the unused characters with blanks.<br>REAL SQL Database converts text to UTF-8 text encoding. SQLite expects text to be in that encoding.  |

## Creating a REALSQLdatabase

You can create a REALSQLdatabase via the IDE by choosing Project ▶ Add ▶ Database ▶ New REAL SQL Database. REALbasic will then present a standard save-file dialog box in which you can name the database and specify the directory in which it will be stored.

When you click OK, a REALSQLdatabase will be added to your project, with the name you entered in the dialog. You can double-click the item to add tables, fields, and indexes. When data have been entered, you can use a table viewer to view the data. For more information, see the chapter on Databases in the *Users Guide*.

## Examples

The following example creates a new REALSQLdatabase:

```
Dim db as REALSQLdatabase
Dim f as FolderItem
f=New FolderItem("mydb")
db=New REALSQLdatabase
db.databaseFile=f
If db.CreateDatabaseFile then
    //proceed with database operations...
else
    MsgBox "Database not created"
end if
```

The following example opens an existing REALSELdatabase.

```
Dim dbFile as FolderItem
Dim db as REALSELdatabase
db=New REALSELdatabase
dbFile = GetFolderItem("Pubs")
db.DatabaseFile=dbFile
If db.Connect() then
    //proceed with database operations here..
else
    Beep
    MsgBox "The database couldn't be opened."
end if
```

The following example adds a record to a table.

```
Dim dbFile as FolderItem
Dim db as REALSELdatabase
db=New REALSELdatabase
Dim rs As New Recordset
dbFile=New FolderItem("Employees")
db.databaseFile=dbFile
if db.error then
    MsgBox db.errormessage
else
    db.sqlexecute ("Insert into Employees (Name,Job,YearJoined) Values " +
        + "('Dr.Strangelove','Advisor',1962)")
If db.error then
    MsgBox db.errormessage
else
    db.Commit
end if
end if
```

**See Also** [Database](#), [DatabaseRecord](#), [RecordSet](#) classes.

---

## RecordSet Class

A **RecordSet** is a group of [Database](#) records. Use the methods and properties of the **RecordSet** class to navigate among this set of records and edit and update individual records.

**Super Class** [Object](#)

## Properties

| Name               | Type                    | Description   |
|--------------------|-------------------------|---|
| <b>BOF</b>         | <a href="#">Boolean</a> | Beginning of the set of records.  |
| <b>EOF</b>         | <a href="#">Boolean</a> | End of the set of records.  |
| <b>FieldCount</b>  | <a href="#">Integer</a> | Number of fields in the <b>RecordSet</b> .  |
| <b>RecordCount</b> | <a href="#">Integer</a> | The number of records in the <b>RecordSet</b> . Currently supported in the REALSQLdatabase, OpenBase, MySQL, PostgreSQL, and 4D plug-ins. For databases that do not support function, RecordCount returns -1. |

## Methods

| Name                | Parameters                       | Description   |
|---------------------|----------------------------------|---|
| <b>Close</b>        |                                  | Closes an open <b>RecordSet</b> . If you quit the application without calling Close, REALbasic does an implicit Close.        |
| <b>DeleteRecord</b> |                                  | Deletes the record in the <b>RecordSet</b> the record pointer is pointing to.   |
| <b>Edit</b>         |                                  | Call prior to performing modifications to the current record.   |
| <b>Field</b>        | Name as <a href="#">String</a>   | Returns the value of the Field in the row the record pointer is pointing to. Returns a <a href="#">DatabaseField</a> .        |
| <b>IdxField</b>     | Index as <a href="#">Integer</a> | 1-based array. Use to refer to i <sup>th</sup> field in the <b>RecordSet</b> . Returns a <a href="#">DatabaseField</a> .      |
| <b>MoveFirst</b>    |                                  | Moves the record pointer to the first record in the <b>RecordSet</b> . Supported only for the REALSQLdatabase data source.    |
| <b>MoveLast</b>     |                                  | Moves the record pointer to the last record in the <b>RecordSet</b> . Supported only for the REALSQLdatabase data source.     |
| <b>MoveNext</b>     |                                  | Moves the record pointer to the next record in the <b>RecordSet</b> . Supported for all data sources.                         |
| <b>MovePrevious</b> |                                  | Moves the record pointer to the previous record in the <b>RecordSet</b> . Supported only for the REALSQLdatabase data source. |
| <b>Update</b>       |                                  | Call to update <b>RecordSet</b> to reflect changes to the record the record pointer is pointing to.                           |

## Notes

If you are using REALbasic as a front-end in a multi-user environment, keep in mind the following. The Edit method will attempt to lock the current record to other users. If a user tries to access a locked record, the [Database](#) class Error property will return [True](#).

The MoveNext method unlocks the current record if you previously locked it with the Edit method. It also calls the Close method of the class and the Close method of the [Database](#) class.

If you are using OpenBase as your data source and use the **RecordSet** class to update a record using Edit and Update you need to include the primary key column of the table. If you don't, OpenBase won't be able to find the record in order to update it.

### Examples

The following example modifies a field and updates the table.

```
Dim db as Database
Dim rs as RecordSet
.
.
rs = db.SQLSelect("select * from employees where ID=01")
rs.Edit
rs.IdxField(2).Value = "VP"
rs.Update
db.Commit
```

The following method displays a **RecordSet** in a [ListBox](#). It uses the Name and StringValue properties of the [DatabaseField](#) class to display the column headers and the data.

```
Sub DisplayRecordSet (rs as RecordSet)
Dim i as Integer
Dim V as String
Listbox1.DeleteAllRows
Listbox1.ColumnCount = rs.FieldCount
Listbox1.AddRow rs.IdxField(1).Name
Listbox1.CellBold(Listbox1.LastIndex, 0) = True
For i = 2 to rs.FieldCount
    Listbox1.Cell(Listbox1.lastIndex, i - 1) = rs.IdxField(i).Name
    Listbox1.CellBold(Listbox1.LastIndex, i - 1) = True
Next
While Not rs.eof
    v = rs.IdxField(1).StringValue //display first column
    Listbox1.AddRow v
    For i = 2 to rs.fieldcount
        Listbox1.Cell(Listbox1.lastIndex, i - 1) = rs.IdxField(i).StringValue
    next
    rs.MoveNext
Wend
```

### See Also

[Database](#), [Database4DServer](#), [DatabaseField](#), [DatabaseRecord](#), [MySQLDatabase](#), [ODBCDatabase](#), [OpenBaseDatabase](#), [OracleDatabase](#), [PostgreSQLDatabase](#), [REALSQLdatabase](#) classes.

# Rectangle Control

Draws a rectangle.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| BorderWidth      | <a href="#">Integer</a> | The width of the rectangle's border in pixels.            |
| BottomRightColor | <a href="#">Color</a>   | The color of the bottom and right edges of the rectangle. |
| FillColor        | <a href="#">Color</a>   | The color of the inside of the rectangle.                 |
| LeftTopColor     | <a href="#">Color</a>   | The color of the left and top edges of the rectangle.     |

## Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Rectangle region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown.  |
| MouseDrag | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Rectangle and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the Rectangle region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.                                  |

## Example

You can give a rectangle a sunken or raised appearance using the global functions [LightBevelColor](#), [DarkBevelColor](#), and [FillColor](#).

```
//Sunken Appearance
Me.BorderWidth=2
Me.TopLeftColor=DarkBevelColor
Me.BottomRightColor=LightBevelColor
Me.FillColor=FillColor

//Raised Appearance
Me.BorderWidth=2
Me.TopLeftColor=LightBevelColor
Me.BottomRightColor=DarkBevelColor
Me.FillColor=FillColor
```

See Also    [RoundRectangle](#) control; [RectControl](#) class.

---

## RectControl Class

**RectControl** is the base class for most other control classes. This means that all controls inherit the properties of the **RectControl**. A **RectControl** cannot be created or modified directly. Instead, you create and modify the **RectControl** subclasses like [PushButtons](#), [EditFields](#), [ListBoxes](#), etc.

Super Class [Control](#)

Because this is a [Control](#), see the [Control](#) class for other properties and events that are common to all [Control](#) objects.

### Properties

| Name                | Type                    | Description   |
|---------------------|-------------------------|---|
| Active              | <a href="#">Boolean</a> | Indicates whether the RectControl is active. Active is <a href="#">False</a> when the RectControl's window is not in the foreground. When a window is deactivated, its controls are automatically deactivated unless AutoDeactivate is set to <a href="#">False</a> . |
| AutoDeactivate      | <a href="#">Boolean</a> | Determines whether the control should be deactivated (on Mac OS) when the parent window is deactivated. The default is <a href="#">True</a> .   |
| Balloon Help        | <a href="#">String</a>  | Balloon Help text for the control in Mac OS "classic". Use the HelpTag property for other platforms.  |
| DisabledBalloonHelp | <a href="#">String</a>  | The text that appears when the user moves the mouse over the control while the control is disabled and BalloonHelp is on (Mac OS "classic")   |
| Enabled             | <a href="#">Boolean</a> | Determines if the control should be enabled when the owning window is opened. A disabled control cannot be clicked and cannot receive the focus.  |
| Height              | <a href="#">Integer</a> | The height (in pixels) of the control.  |
| HelpTag             | <a href="#">String</a>  | Text of help message displayed as a Windows or Linux "tip" or Carbon/Mac OS X help tag. The tip/tag is displayed when the user places the mouse on the control and leaves it there.   |
| Left                | <a href="#">Integer</a> | The left side of the control in local coordinates (relative to the window).   |

| Name        | Type                        | Description   |
|-------------|-----------------------------|---|
| LockBottom  | <a href="#">Boolean</a>     | Determines whether the bottom edge of the control should stay at a set distance from the bottom edge of the parent control, if there is one, or the owning window. See the section "The Control Hierarchy" on page 149 in the <i>User's Guide</i> for more information on parent and child controls.  |
| LockLeft    | <a href="#">Boolean</a>     | Determines whether the left edge of the control should stay at a set distance from the left edge of the parent control, if there is one, or the owning window. See the section "The Control Hierarchy" on page 149 in the <i>User's Guide</i> for more information on parent and child controls. LockLeft has no effect unless LockRight is <a href="#">True</a> .  |
| LockRight   | <a href="#">Boolean</a>     | Determines whether the right edge of the control should stay at a set distance from the right edge of the parent control, if there is one, or the owning window. See the section "The Control Hierarchy" on page 149 in the <i>User's Guide</i> for more information on parent and child controls.  |
| LockTop     | <a href="#">Boolean</a>     | Determines whether the top edge of the control should stay at a set distance from the top edge of the parent control, if there is one, or the owning window. See the section "The Control Hierarchy" on page 149 in the <i>User's Guide</i> for more information on parent and child controls. LockTop has no effect unless LockBottom is <a href="#">True</a> .  |
| MouseCursor | <a href="#">MouseCursor</a> | The cursor to be displayed while the mouse is within the control and both the <a href="#">Application</a> and <a href="#">Window</a> classes's MouseCursor properties are <a href="#">Nil</a> . If the <a href="#">Application</a> class's MouseCursor property is not <a href="#">Nil</a> or the <a href="#">Window</a> 's MouseCursor property is not <a href="#">Nil</a> , then any control's MouseCursor property is ignored. You can use a cursor stored in the <a href="#"> Cursors</a> object. On Macintosh, you can also obtain a <a href="#">MouseCursor</a> from a resource file. |
| Parent      | <a href="#">Control</a>     | Used to get and set the control's parent control or window. Returns <a href="#">Nil</a> if the parent is the window. Assign <a href="#">Nil</a> to make the control's parent the window, even if it is enclosed by another control. The child control's behavior in the IDE (see "Control Hierarchy" in Notes, below) will reflect the parent's state. If the parent control is somehow in another window, an <a href="#">InvalidParentException</a> will occur.  |

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Top     | <a href="#">Integer</a> | The top of the control in local coordinates (relative to the window).   |
| Visible | <a href="#">Boolean</a> | Determines whether the control is visible when its owning window is opened. The default is <a href="#">True</a> : the control is visible. |
| Width   | <a href="#">Integer</a> | The width (in pixels) of the control.   |
| Window  | <a href="#">Window</a>  | The RectControl's parent window.  |

## Events

| Name                 | Parameters   | Description   |
|----------------------|--|---|
| ConstructContextMenu | Base as <a href="#">MenuItem</a> ,<br>x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | Fires whenever it is appropriate to display a ContextualMenu for the control. This is the recommended way to handle contextual menus because this event figures out whether the user has requested the contextual menu, regardless of how he did it. Depending on platform, it might be in the MouseUp or MouseDown event and it might be a right+click or by pressing the contextual menu key on the keyboard, for example.<br>Returns a <a href="#">Boolean</a> . Base is analogous to the menu bar for the contextual menu. Any items you add to Base will be shown as menu items. If you return <a href="#">False</a> , the event is passed up the parent hierarchy. If you return <a href="#">True</a> , the contextual menu is displayed. The parameters x and y are the mouse locations. If the event was fired because of a non-mouse event, then x and y are both set to -1. See the example of a contextual menu in the examples. |
| ContextMenuAction    | HitItem as <a href="#">MenuItem</a>  | Fires when a contextual menuitem HitItem was selected but the Action event and the MenuHandler for the menuitem did not handle the menu selection. This event gives you a chance to handle the menu selection by inspecting the menuitem's Text or Tag properties to see which item was selected. Use this in conjunction with ConstructContextMenu if you have not specified the Action event or the Menu Handler for the items on the contextual menu.  |

| Name       | Parameters   | Description   |
|------------|--|---|
| DragEnter  | obj as <a href="#">DragItem</a>  | Fired when the <a href="#">DragItem</a> enters the <b>RectControl</b> . Returns a <a href="#">Boolean</a> . Return <a href="#">True</a> from this event to prevent the drop from occurring.   |
| DragExit   | obj as <a href="#">DragItem</a>  | Fired when the <a href="#">DragItem</a> exits the <b>RectControl</b> .  |
| DragOver   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> ,<br>obj as <a href="#">DragItem</a>  | Fired when the <a href="#">DragItem</a> is over the <b>RectControl</b> . The coordinates x and y are relative to the <b>RectControl</b> . Returns a <a href="#">Boolean</a> . Return <a href="#">True</a> from this event to prevent the drop from occurring.   |
| DropObject | Obj as <a href="#">DragItem</a>  | The item represented by <i>Obj</i> has been dropped on the control.   |
| MouseEnter |  | The mouse has entered the area of the control.  |
| MouseExit  |  | The mouse has left the area of the control.   |
| MouseMove  | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a>                                       | The mouse has moved within the control to the coordinates passed. The coordinates are local to the control, not to the window.  |
| MouseWheel | X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ,<br>Delta as <a href="#">Integer</a> | The mouse wheel has been moved. The parameters <i>X</i> and <i>Y</i> are the mouse coordinates relative to the control that has received the event. The parameter <i>Delta</i> is the number of scroll lines the wheel has been moved, as defined by the operating system. This value is positive when the user scrolls down and negative when scrolling up. Returns a <a href="#">Boolean</a> . Return True to prevent the event from propagating further. |

## Methods

| Name              | Parameters                         | Description   |
|-------------------|------------------------------------|---|
| AcceptFileDrop    | FileType as <a href="#">String</a> | Permits documents of type FileType to be dropped on the control. <i>FileType</i> must be a file type that you defined in via the <a href="#">FileType</a> class or the File Type Sets Editor. |
| AcceptRawDataDrop | type as <a href="#">String</a>     | Permits data (of the Type specified) to be dropped on the control.  |
| AcceptPictureDrop |                                    | Permits pictures to be dropped on the control.  |
| AcceptTextDrop    |                                    | Permits text to be dropped on the control.  |

| Name        | Parameters  | Description  |
|-------------|---|--|
| NewDragItem | Left As <a href="#">Integer</a> ,<br>Top As <a href="#">Integer</a> ,<br>Width As<br><a href="#">Integer</a> , Height<br>As <a href="#">Integer</a>   | Defines the drag rectangle (based on the coordinates passed) that will appear when the user drags from the control. Returns a new <a href="#">DragItem</a> object. <i>Left</i> and <i>Top</i> are the coordinates of the left-top corner of the drag rectangle and <i>Width</i> and <i>Height</i> are the dimensions of the drag rectangle. See the <a href="#">NewDragItem</a> global function, which provides equivalent functionality, as well as support for dragging to a window. |
| Refresh     | [EraseBackground as <a href="#">Boolean</a> ]   | Repaints the entire contents of the control. If the optional parameter <i>EraseBackground</i> is <a href="#">True</a> , the background will be erased prior to redrawing the control. The default is <a href="#">True</a> .  |
| RefreshRect | X As <a href="#">Integer</a> ,<br>Y As <a href="#">Integer</a> ,<br>Width As<br><a href="#">Integer</a> , Height<br>As <a href="#">Integer</a><br>[,EraseBackground<br>as <a href="#">Boolean</a> ] | Repaints only the region of the control specified. If the optional parameter <i>EraseBackground</i> is <a href="#">True</a> , the background will be erased prior to redrawing the area. The default is <a href="#">True</a> .   |
| SetFocus    |   | If applicable, sets the focus to the control. KeyDown events are directed to the control. If the control cannot get the focus on the platform on which the application is running, SetFocus does nothing. The SetFocus method of the <a href="#">Window</a> class or the <a href="#">ClearFocus</a> method can be used to remove the focus from the control that currently has the focus, leaving no control with the focus.   |

## Notes

### The Control Hierarchy

In some cases you build portions of your interface by placing some controls within another control. For example, you use the [GroupBox](#) control to organize other controls, usually [RadioButtons](#) or possibly [Checkboxes](#). The [TabPanel](#) and [PagePanel](#) controls also designed to enclose other controls. A good example is the Find and Replace dialog box in the REALbasic *Tutorial*. In this case, [StaticText](#), [EditField](#), and [PushButton](#) controls are all enclosed by the [TabPanel](#).

The control that encloses the others is known as the *parent* control and the controls that are entirely within its borders are the *child* controls. This works automatically if you create the controls in the order of: parent-child. That is, create the control that encloses the others first, then create the child controls or duplicate existing child controls and keep them inside the parent control.

You can use the Parent property of the [RectControl](#) class to either get or set the parent of an existing control. If you create a child control before its parent, you can set its parent using its Parent property.

If a control is not completely enclosed by another control then it doesn't automatically become a child; it simply overlaps the other control.

If you move a child outside its parent, it is no longer a child of that control. If you move it completely inside another control, it becomes the child of that control.

When you copy a parent control, you copy all its child controls as well.

You can create more than one level of nesting. For example, you can place a [GroupBox](#) within a [TabPanel](#) and then place several [RadioButtons](#) within the [GroupBox](#). In this case, the [GroupBox](#) is the parent of the [RadioButtons](#) and the [TabPanel](#) is the parent of the [GroupBox](#). To make this work automatically, you must create them in the order that respects the hierarchy: First create the [TabPanel](#), then the [GroupBox](#), and then the [RadioButtons](#).

Obviously, the parent control must be in the same window as its child controls. If you attempt to get the parent of a control whose parent is in another window, you will get an [InvalidOperationException](#) error.

If a control is not enclosed by another control, then its “parent” is its window. But, since the Parent property of a control is a [Control](#), the Parent property will be [Nil](#)—since a [Window](#) is not subclassed from the [Control](#) class. Before trying to access the Parent property of a control, be sure to first test whether it is [Nil](#). If you don't, you will get a [NilObjectException](#) error if the control's “parent” is the parent window.

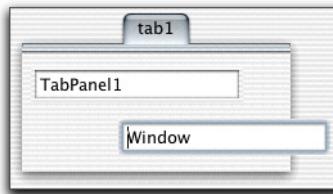
If a control is enclosed by another control, but you don't want it to behave as a child, then set its Parent property to [Nil](#). This breaks the default Parent-Child relationship that is normally maintained by the control hierarchy.

## Control Hierarchy Features

To take advantage of the features of the control hierarchy automatically, create the parent control first and then add the child controls by dragging fully into the interior of the parent control. When you add a child to a parent in this manner, a marquee surrounds the parent. A marquee also surrounds the parent control whenever you select an existing child control. You can disable the marquee by deselecting the “Highlight parent control” preference on the Window Editor screen in Edit ▶ Options (REALbasic ▶ Preferences on Mac OS X). If you leave the feature enabled, you can also select the color of the marquee in that preference screen.

If you duplicate a child control and leave the duplicate within the parent, then it is automatically a child. However, if you move the duplicate outside the parent, it is no longer a child. A control must be fully enclosed by the parent to be considered a child.

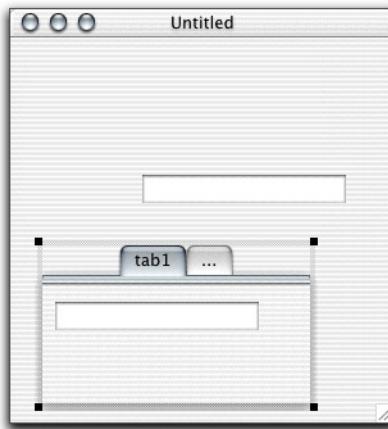
In the following example, both [EditFields](#) report the identity of their parents. The bottom one was duplicated from the top one, but has been moved out of the [TabPanel](#). It is no longer a child.



When you move a child out of the 'scope' of the parent, the marquee surrounding the parent is turned off, indicating that the child is not under parental control anymore.

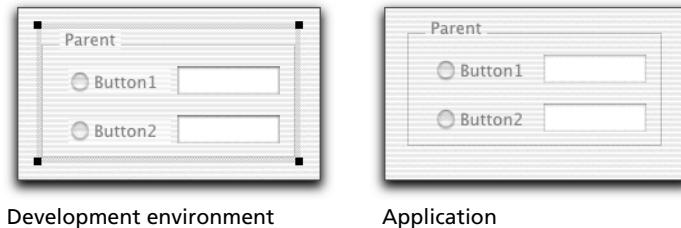
If you move the bottom [EditField](#) back within the [TabPanel](#), it becomes a child of the [TabPanel](#) once again.

- Moving a parent control moves its child controls as well. In the following example, the [TabPanel](#) shown above was moved down. It takes the top [EditField](#) with it, but leaves the bottom one orphaned.



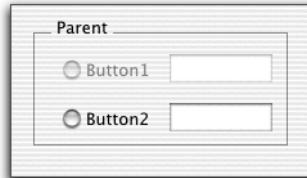
- Deleting a parent control deletes all child controls.
- Hiding a parent hides all child controls, but retains the previous visibility status of all children.
- Showing a parent control shows only the child controls whose visibility is set.
- Disabling a parent control disables all its child controls, but retains the previous visibility status of all children. In the IDE, disabling a container visually disables all the child controls, but does not update the Enabled property.

- Here is a disabled [GroupBox](#) in the IDE in which two [RadioButtons](#) and two [EditFields](#) are children. When the [GroupBox](#) is disabled, all four children are disabled. Since all children are disabled when the parent is disabled, the [EditFields](#) are not enterable in the application.



Enabling a parent control enables only the child controls whose Enabled property is set.

- In the following example, only the bottom row of child controls have the Enabled property set. When the [GroupBox](#) is enabled, the top row of controls remains disabled.



Disabling the [GroupBox](#) disables all child controls, as expected.

## Dragging

There are three possible items a user can drag from a control: text, a picture, and/or a [FolderItem](#). In order for the user to be able to drag, the control must (during the appropriate event handler) create a [DragItem](#) object. The appropriate event handler depends on the type of object. For example, the appropriate event handler for a [Canvas](#) control is the MouseDown event handler since the user must be holding down the mouse button to drag.

The [DragItem](#) object represents the data that the user is dragging. [DragItem](#) objects have properties to hold text, pictures, and [folderItems](#). One or more of these properties must be populated with the values the user wishes to drag. When you create a new [DragItem](#) object using the [NewDragItem](#) method, you can specify the drag rectangle that should appear as the user drags from the control.

[EditFields](#) and [ListBoxes](#) have implicit dragging built-in to them. This means that they will create a new [DragItem](#) and drag rectangle for you. [EditFields](#) automatically populate the Text property of the [DragItem](#) with the text the user is dragging.

[ListBoxes](#) have a DragRow event handler where you can populate the [DragItem](#) with the data to be dragged.

## Dropping

Before a control will accept an item being dropped on it, you must call the AcceptFileDrop, AcceptPictureDrop, and/or AcceptTextDrop methods to indicate the

drop types that will be allowed. This is usually done in the control's Open event handler. When an acceptable item is dropped on the control, the control's DropObject event handler will execute and will be passed the [DragItem](#) that has been dropped on the control. At that point you can use the [DragItem's](#) methods to determine what kind of data is available and access it. If you are only allowing a particular type of drag data (text for example) then there would be no need to test the type of [data](#) that is being dragged. You can simply get the text from the Text property of the [DragItem](#).

### Handling Contextual Menus

The following code in a ConstructContextMenu event builds a simple contextual menu. The parameter **Base** as [MenuItem](#) is passed in as a parameter.

```
Dim m1 as New MenuItem
Dim m2 as New MenuItem

m1.Text="Import"
m2.text="Export"

base.append m1
base.append m2

Return True //display the contextual menu
```

The following [Select](#) statement in the ContextualMenuAction event handler inspects the selected menu item, which is passed in as the **HitItem** as [MenuItem](#) parameter.

```
Select Case hititem.text
case "Import"
MsgBox "You chose Import"
case "Export"
MsgBox "You chose export"
End select

Return True
```

**Examples**

This example shows how the DropObject event handler of an [EditField](#) (named EditField1 in this example) would be coded to handle one or more SimpleText documents or some text being dropped on it:

```
Sub DropObject(Obj as DragItem)
  If Obj.TextAvailable then
    EditField1.Text=Obj.Text
  ElseIf Obj.FolderItemAvailable Then
    Do
      //Load the styled text from the document
      //into EditField1
      Obj.FolderItem.OpenStyledEditField EditField1
    Loop Until Not Obj.NextItem
  End if
```

**The [Window](#) Class's Composite Property**

Under Mac OS X, there is a display problem that affects certain types of **RectControls** that are placed in Metal or Drawer windows or on top of other controls in Document windows. Non-rectangular shaped controls such as [PushButtons](#) or [RadioButtons](#) show their rectangular borders as a white background. To solve this problem, turn on the parent window's Composite property.

**See Also**

[Control](#) class.

## RectShape Class

Draws a (two-dimensional) rectangle in a vector graphics environment.

**SuperClass** [Object2D](#)

**Properties**

| Name   | Type                   | Description                  |
|--------|------------------------|------------------------------|
| Height | <a href="#">Double</a> | The height of the rectangle. |
| Width  | <a href="#">Double</a> | The width of the rectangle.  |

**Methods**

| Name     | Parameters  | Description  |
|----------|---|--|
| Contains | X as <a href="#">Double</a> , Y as <a href="#">Double</a> | Returns a <a href="#">Boolean</a> . Returns <a href="#">True</a> if the rectangle contains the point specified by x,y. |

## Redim Method

---

**Example** The following code in the Paint event of a [window](#) draws a diamond shaped rectangle.

```
Dim r as New RectShape  
r.width=75  
r.height=75  
r.border=100  
r.bordercolor=RGB(0,0,0) //black  
r.fillcolor=RGB(0,127,127) // teal  
r.borderWidth=2.5  
r.rotation=-.78  
g.DrawObject r,100,100
```

**See Also** [ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Group2D](#), [Graphics](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RoundRectShape](#), [StringShape](#) classes.

## Redim Method

Resizes the passed array.

### Syntax

**Redim arrayName(newSize1[,newSizeN])**

| Part      | Type                    | Description   |
|-----------|-------------------------|---|
| arrayName | Variable name           | The name of any previously declared array.  |
| newSize1  | <a href="#">Integer</a> | The new size for the array. In the case of multi-dimensional arrays, the new size for the first dimension of the array. |
| newSizeN  | <a href="#">Integer</a> | The new size for the N <sup>th</sup> dimension of the array.  |

### Notes

The **Redim** method is used to increase or reduce the number of elements in the array specified. Arrays are zero-based (the first element is zero) so you resize the array using a number that is one less than the number of elements you actually want.

### Examples

This example reduces the aNames array to 11 elements.

```
Redim aNames(10)
```

This example adds 10 elements to the aNames array.

```
Redim aNames(UBound(aNames)+10)
```

This example reduces the aPeople array to 11 elements for the first dimension and 6 elements for the second dimension.

```
Redim aPeople(10,5)
```

**See Also**

[Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

## RegistryAccessErrorException Error

Occurs if you try to use the [RegistryItem](#) class without proper access privileges or try to use it under any Macintosh OS or Linux. The Registry items feature is Windows-only.

**Super Class** [RuntimeException](#)

**See Also** [Exception](#), [Try](#) blocks, [RegistryItem](#) class.

## RegistryItem Class

Creates and manages registry items on Windows. Does not apply to Macintosh or Linux.

**Super Class** [Object](#)

### Constructors

| Name         | Parameters   | Description   |
|--------------|--|---|
| RegistryItem | Path as <a href="#">String</a><br>[Create as <a href="#">Boolean</a> ] | Creates the registry item from the path passed to it. Returns a new <a href="#">RegistryItem</a> . The optional parameter Create defaults to <a href="#">True</a> ; if you set it to <a href="#">False</a> , the path will not be created if the path is not found. Instead, a <a href="#">RegistryItemAccessErrorException</a> will occur. |
| RegistryItem | Copy as <a href="#">RegistryItem</a>                                   | Clones the <a href="#">RegistryItem</a> passed to it. Returns a new <a href="#">RegistryItem</a> .  |

### Properties

| Name               | Type                         | Description   |
|--------------------|------------------------------|---|
| <b>FolderCount</b> | <a href="#">Integer</a>      | The number of child folders contained with the <a href="#">RegistryItem</a> .                         |
| <b>KeyCount</b>    | <a href="#">Integer</a>      | The number of keys contained within the <a href="#">RegistryItem</a> .                                |
| <b>Parent</b>      | <a href="#">RegistryItem</a> | The RegistryItem's parent. Note: You cannot get the parent of a hive. Each hive is considered a root. |
| <b>Path</b>        | <a href="#">String</a>       | The full path to the <a href="#">RegistryItem</a> .   |

## Methods

| Name         | Parameters                       | Description   |
|--------------|----------------------------------|---|
| AddFolder    | Name as <a href="#">String</a>   | Adds the passed item as a folder. Returns a <a href="#">RegistryItem</a> .  |
| Child        | Name as <a href="#">String</a>   | Returns the child named as a <a href="#">RegistryItem</a> . Returns a <a href="#">RegistryItem</a> .  |
| DefaultValue |                                  | Returns the default value of the <a href="#">RegistryItem</a> as a <a href="#">Variant</a> .  |
| Delete       | Name as <a href="#">String</a>   | Deletes the passed <a href="#">RegistryItem</a> .   |
| Item         | Index as <a href="#">Integer</a> | Returns as a <a href="#">RegistryItem</a> the <a href="#">RegistryItem</a> corresponding to <i>Index</i> . If <i>Index</i> is greater than KeyCount, then an <a href="#">OutOfBoundsException</a> is raised.  |
| KeyType      | Index as <a href="#">Integer</a> | Returns an <a href="#">Integer</a> . Returns the key type for the key located at the index. Possible values are:<br>Unsupported key = -1,<br>REG_SZ and REG_EXPAND_SZ = 0,<br>REG_DWORD = 1,<br>REG_BINARY = 2,<br>REG_MULTI_SZ = 3                   |
| Name         | Index as <a href="#">Integer</a> | Returns as a <a href="#">String</a> the name of the <a href="#">RegistryItem</a> corresponding to <i>Index</i> . If <i>Index</i> is greater than KeyCount, then an <a href="#">OutOfBoundsException</a> is raised.                                    |
| Value        | Name as <a href="#">String</a>   | Returns the <a href="#">RegistryItem</a> value as a <a href="#">Variant</a> . See notes on the Value method.  |
| Value        | Index as <a href="#">Integer</a> | Returns the <a href="#">RegistryItem</a> value as a <a href="#">Variant</a> . Index refers to a Key contained in the <a href="#">RegistryItem</a> . If <i>Index</i> is greater than KeyCount, then an <a href="#">OutOfBoundsException</a> is raised. |

## Notes

The Value method returns valid information for expandable strings, binary and multi-strings. In the case of expandable strings, REALbasic automatically expands the string for you and returns it as a string.

In the case of multi-strings, REALbasic passes back a [string](#) that is [Chr\(0\)](#)-delimited. Please use the [Split](#) function to get the individual parts, like this:

```
part = Split(regValue, Chr(0))
```

where *part* is a [String](#) array.

In the case of a binary value, REALbasic returns a [MemoryBlock](#). You cannot currently set an expandable string, or a multistring. To set a binary string, please pass in a [MemoryBlock](#).

**See Also** [RegistryAccessErrorException](#) error.

## RegEx Class

Used to do search and replace operations using regular expressions (i.e., perl).

**Super Class** [Object](#)

### Properties

| Name                | Type                         | Description   |
|---------------------|------------------------------|---|
| Options             | <a href="#">RegExOptions</a> | These options are various states which you can set for the Regular Expressions engine. See the <a href="#">RegExOptions</a> class.  |
| ReplacementPattern  | <a href="#">String</a>       | This is the replacement string, which can include references to substrings matched previously, via the standard '\1' or '\$1' notation common in regular expressions. This pattern is used either with the Replace property or passed to the <a href="#">RegExMatch</a> class when Search returns, and subsequently used with Replace if no parameters are specified. |
| SearchPattern       | <a href="#">String</a>       | This is the pattern you are currently searching for.  |
| SearchStartPosition | <a href="#">Integer</a>      | Character position at which you want to start the search if the optional TargetString parameter to Replace is not specified. Keep in mind if you set it, it will only be used if you don't specify a TargetString, since setting a new targetString resets the value.   |

### Methods

| Name    | Parameters  | Description   |
|---------|---|---|
| Replace | [TargetString as <a href="#">String</a> ]<br>[,SearchStartPosition as <a href="#">Integer</a> ] | Finds <i>SearchPattern</i> in <i>Target</i> , Replaces the contents of <i>SearchPattern</i> with <i>ReplacementPattern</i> . Returns the resulting string. Replace can take the optional parameters shown at left. Returns a <a href="#">String</a> . |

| Name   | Parameters  | Description   |
|--------|---|---|
| Search | [ <i>TargetString</i> as <a href="#">String</a> ]<br>[, <i>SearchStartPosition</i> as <a href="#">Integer</a> ] | Finds <i>SearchPattern</i> in <i>TargetString</i> , beginning at <i>SearchStartPosition</i> . If it succeeds, it returns a <a href="#">RegExMatch</a> . Both parameters are optional; if <i>TargetString</i> is omitted, it assumes the previous <i>TargetString</i> , so you will want to pass a <i>TargetString</i> the first time you call Search. If you call Search with a <i>TargetString</i> and omit <i>SearchStartPosition</i> , zero is assumed. If you call Search with no parameters after initially passing a <i>TargetString</i> , it assumes the previous <i>TargetString</i> and will begin the search where it left off in the previous call. This is the easiest way to find the next occurrence of <i>SearchPattern</i> in <i>TargetString</i> .<br>The <a href="#">RegExMatch</a> will remember the <i>ReplacementPattern</i> specified at the time of the search. Returns a <a href="#">RegExMatch</a> . |

**Notes**

This section describes the syntax of regular expressions in REALbasic.

| Pattern   | Description   |
|-----------|---|
| .         | Matches any character except newline.                       |
| [a-z0-9]  | Matches any single character of set.                        |
| [^a-z0-9] | Matches any single character not in set.                    |
| \d        | Matches a digit. Same as [0-9].                             |
| \D        | Matches a non-digit. Same as [^0-9].                        |
| \w        | Matches an alphanumeric (word) character — [a-zA-Z0-9_].    |
| \W        | Matches a non-word character [^a-zA-Z0-9_].                 |
| \s        | Matches a whitespace character (space, tab, newline, etc.). |
| \S        | Matches a non-whitespace character.                         |
| \n        | Matches a newline (line feed).                              |
| \r        | Matches a return.   |
| \t        | Matches a tab.  |
| \f        | Matches a formfeed.   |
| \b        | Matches a backspace.  |
| \0        | Matches a null character.                                   |
| \000      | Also matches a null character because of the following:     |
| \nnn      | Matches an ASCII character of that octal value.             |
| \xnn      | Matches an ASCII character of that hexadecimal value.       |
| \cX       | Matches an ASCII control character.                         |

| Pattern    | Description   |
|------------|---|
| \metachar  | Matches the meta-character (e.g., \, .,  ).   |
| (abc)      | Used to create subexpressions. Remembers the match for later backreferences. Referenced by replacement patterns that use \1, \2, etc. |
| \1, \2,... | Matches whatever first (second, and so on) of parens matched.   |
| x?         | Matches 0 or 1 x's, where x is any of above.  |
| x*         | Matches 0 or more x's.  |
| x+         | Matches 1 or more x's.  |
| x{m,n}     | Matches at least m x's, but no more than n.   |
| abc        | Matches all of a, b, and c in order.  |
| a b c      | Matches one of a, b, or c.  |
| \b         | Matches a word boundary (outside [] only).  |
| \B         | Matches a non-word boundary.  |
| ^          | Anchors match to the beginning of a line or string.   |
| \$         | Anchors match to the end of a line or string.   |

## Replacement Patterns

The following expressions can only apply to the replacement pattern:

| Pattern  | Description   |
|----------|---|
| \$`      | Replaced with the entire target string before match.  |
| \$&      | The entire matched area; this is identical to \0 and \$0.   |
| \$'      | Replaced with the entire target string following the matched text.                                |
| \$0-\$50 | \$0-\$50 evaluate to nothing if the the subexpression corresponding to the number doesn't exist.  |
| \0-\50   |   |
| \xnn     | Replaced with the character represented by nn in Hex, e.g., \xAA is ™.                            |
| \nnn     | Replaced with the character represented by nn in Octal.   |
| \cX      | Replaced with the character that is the control version of X, e.g., \cP is DLE, data line escape. |

## Double-byte Systems

If you are working with a double-byte system such as Japanese, RegEx cannot operate on the characters directly. You should first convert all double-byte text to UTF8 using REALbasic's built-in Text Converter functions. See, for example, the [TextConverter](#) class for an example of how to use the Text Converter functions.

All text that will be processed by RegEx should be converted. This includes *SearchPattern*, *ReplacementPattern*, and *TargetString*. The result of the Search or Search and Replace will be a UTF8 string, so you will need to convert it back to its original form using the Text Converter functions. Both Search and Search and Replace operations work on all platforms, provided that this conversion takes place.

## RegEx Class

---

### Regular Expression Examples

The basic idea of regular expressions is that it enables you to find and replace text that matches the set of conditions you specify. It extends normal Search and Replace with pattern searching.

### Wildcards

Some special characters are used to match a class of characters:

| Wildcard | Matches  |
|----------|--|
| .        | Any single character except a line break, including a space. |

If you use the “.” as the search pattern, you will select the first character in the target string and, if you repeat the search, you will find each successive character, except for Return characters

The following wildcards match by position in a line:

| Wildcard | Matches   | Example   |
|----------|---|---|
| ^        | Beginning of a line (unless used in a character class; see below) | ^Phone: Finds lines that begin with “Phone”:      |
| \$       | End of a line (unless used in a character class)                  | \$: Finds the last character in the current line. |

### Character Classes

A character class allows you to specify a set or range of characters. You can choose to either match or ignore the character class. The set of characters is enclosed in brackets.

If you want to ignore the character class instead of match it, precede it by a caret (^). Here are some examples:

| Character Class | Matches  |
|-----------------|--|
| [aeiou]         | Any one of the characters a, e, i, o, u.   |
| [^aeiou]        | Any character except a, e, i, o, u.  |
| [a-e]           | Any character in the range a-e, inclusive  |
| [a-zA-Z0-9]     | Any alphanumeric character. Note: Case-sensitivity is controlled by the CaseSensitive property of the <a href="#">RegExOptions</a> class.                          |
| []              | Finds a [.   |
| []              | Finds a ]. To find a a closing bracket, place it immediately after the opening bracket.  |
| [a-e^]          | Finds a character in the range a-e or the caret character. To find the caret character, place it anywhere except as the first character after the opening bracket. |
| [a-c-]          | Finds a character in the range a-c or the - sign. To match a -, place it at the beginning or end of the set.   |

### Non-printing Characters

You can use the following notation to find non-printing characters:

| Special Character | Matches             |
|-------------------|---------------------|
| \r                | Line break (return) |
| \n                | Newline (line feed) |

| Special Character | Matches               |
|-------------------|-----------------------|
| \t                | Tab                   |
| \f                | Formfeed (page break) |
| \xNN              | Hex code NN.          |

**Other Special Characters** The following patterns are wildcards for the following special characters:

| Special Character | Matches  |
|-------------------|--|
| \s                | Any whitespace character (space, tab, return, linefeed, form feed) |
| \S                | Any non-whitespace character.                                      |
| \w                | Any “word” character (a-z, A-Z, 0-9, and _)                        |
| \W                | Any “non-word” character (All characters not included by \w).      |
| \d                | Any digit [0-9].   |
| \D                | Any non-digit character.   |

**Repetition Characters** Repetition characters are modifiers that allow you to repeat a specified pattern.

| Repetition Character | Matches                  | Examples   |
|----------------------|--------------------------|--|
| *                    | Zero or more characters. | d* finds no characters, or one or more consecutive “d”s.<br>. finds an entire line of text, up to but not including the return character.              |
| +                    | One or more characters.  | d+ finds one or more consecutive “d”s.<br>[0-9]+ finds a string of one or more consecutive numbers, such as “90404”, “1938”, the “32” in “Win32”, etc. |
| ?                    | Zero or one characters.  | d? finds no characters or one “d”.   |

Please note that, since \* and ? match zero instances of the pattern, they always succeed but may not select any text. You can use them to specify an optional character, as in the examples in the following section.

### “Greediness”

REALbasic supports the “?” as a “greediness” modifier for a subpattern in a regular expression. By default, greediness is controlled by the Greedy property of the [RegExOptions](#) class, but can be overridden using the “?”. You can place a “?” directly after a \* or + to reverse the “greediness” setting. That is, if Greedy is True, using the ? after a \* or + causes it to match the minimum number of times possible: For example, consider the following.

| Target String | Greedy | Regular Expression | Result         |
|---------------|--------|--------------------|----------------|
| aaaa          | True   | (a+?) (a+)         | \$1=a, \$2=aaa |
| aaaa          | False  | (a+?) (a+)         | \$1=aaa, \$2=a |

## RegEx Class

---

### Extension Mechanism

We also support the regular expression extension mechanism used in Perl. For instance:

| (?#text)     | Comment   |
|--------------|---|
| (?:pattern)  | For grouping without creating backreferences  |
| (?=pattern)  | A zero-width positive look-ahead assertion. For example, \w+(?=\\t) matches a word followed by a tab, without including the tab in \$&.   |
| (?!pattern)  | A zero-width negative look-ahead assertion. For example foo(?!bar)/matches any occurrence of "foo" that isn't followed by "bar".  |
| (?<=pattern) | A zero-width positive look-behind assertion. For example, (?<=\\t)\\w+ matches a word that follows a tab, without including the tab in \$&. Works only for fixed-width look-behind. |
| (?<!pattern) | A zero-width negative look-behind assertion. For example (?<!bar)foo matches any occurrence of "foo" that does not follow "bar". Works only for fixed-width look-behind.            |

**Subexpressions** You can use parentheses within your search patterns to isolate portions of the matched string. You do this when you need to refer to subsections of the matched in your replacement string. For example you would do this if you need to replace only a portion of the matched string or insert other text into the matched string.

Here is an example. If you want to match any date followed by the letters "B.C." you can use the pattern "\d+\sB\\.C\\." (Any number of digits followed by a space character, followed by the letters "B.C.") This will match dates such as 33 B.C., 1742 B.C., etc. However, if you wanted your replacement pattern to leave the year alone but replace the letters with something else, you would use parens. The search pattern "(\\d+)\\s(B\\.C\\.)" does this.

When you write your replacement pattern, you can refer to the year only with the variable \1 and the letters with \2.

If you write "(\\d+)\\s(B.C.|A.D.|BC|AD)", then \2 would contain the matched letters.

### Combining Patterns

Much of the power of regular expressions comes from combining these elementary patterns to make up complex searches. Here are some examples:

| Pattern             | Matches  |
|---------------------|--|
| \\$?[0-9,]+\\.?\\d* | Matches dollar amounts with an optional dollar sign.       |
| \\d+\\sB\\.C\\.     | one or more digits followed by a space, followed by "B.C." |

### The Alternation Operator

The alternation operator (|) allows you to match any of a number of patterns using the logical "or" operator. Place it between two existing patterns to match either pattern. You can use more than one alternation operator in a pattern:

| Pattern              | Matches                   |
|----------------------|---------------------------|
| \\she\\s   \\sshe\\s | " he " or " she "         |
| cat dog possum       | "cat", "dog", or "possum" |

| Pattern   | Matches   |
|---|---|
| ([0-9,]+\\sB\\.C\\.)([0-9,]+\\sA\\.D\\.) or<br>[0-9,]+\\s((B\\.C\\.)(A\\.D\\.)) | Years of the form "yearNum B.C. or A.D." e.g., "2,175 B.C." or "215 A.D." |

## Search and Replace

You use special patterns to represent the matched pattern. Using replacement patterns, you can append or prepend the matched pattern with other text.

| Pattern      | Description   |
|--------------|---|
| \$&          | Contains the entire matched pattern.<br>If "d\\dd\\dsB\\.C\\." finds "1541 B.C.", then the replacement pattern "the year \$&" results in "the year 1541 B.C.", as the \$& contains the string "1541 B.C".   |
| \1, \2, etc. | Contains the matched subpatterns, defined by use of parentheses in the search string.<br>The search pattern "(\\d+)\\s(B\\.C\\. A\\.D\\. BC AD)" looks for any number of digits followed by a space character, followed by either "B.C.", "BC", "A.D.", or "AD". The \1 variable contains the match to the "\\d+" portion of the expression and \2 contains the match to the "B\\.C\\. A\\.D\\. BC AD" portion. |

## Credits

REALbasic uses a modified version of the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England.

The source to this library is available at:

<ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/>

## Examples

The following [PushButton's](#) Action event handler allows you to search the text in EditField1 using the search pattern entered into [EditField2](#) and display the results of the search in a [StaticText](#):

```
Dim rg as New RegEx
Dim myMatch as RegExMatch
rg.SearchPattern=EditField2.text
myMatch=rg.search(EditField1.text)
if myMatch <> Nil then
  StaticText1.text=myMatch.SubExpressionString(0)
else
  StaticText1.text="Text not found!"
End if
exception err as RegExException
MsgBox err.message
```

## See Also

[RegExMatch](#), [RegExOptions](#) classes; [RegExException](#) Error.

## RegExException Error

The RegEx engine found a runtime error in your code. Currently, the only type of runtime exception is a [RegExSearchPatternException](#), but future versions of REALbasic may trap other types of errors.

**Super Class** [RuntimeException](#)

**See Also** [RuntimeException](#) class; [RegExSearchPatternException](#) error; [Exception](#), [Try](#) blocks.

---

## RegExMatch Class

Used to extract the matched string when doing a search with regular expressions.

**Super Class** [Object](#)

### Properties

| Name                       | Parameters                                      | Description   |
|----------------------------|---|---|
| <b>SubExpressionCount</b>  |   | Number of SubExpressions that are available with the search just performed. REALbasic's Regular Expressions support both \number and \$number syntax for SubExpressions. SubExpressions allow replacement of parts of the pattern. Returns an <a href="#">Integer</a> .           |
| <b>SubExpressionString</b> | matchNumber as <a href="#">Integer</a>          | Returns the SubExpression as a <a href="#">String</a> for the given matchNumber. 0 returns the entire MatchString (the implicit 0 <sup>th</sup> SubExpression), and 1 is the first real SubExpression. Returns a <a href="#">String</a> .   |
| <b>SubExpressionStartB</b> | matchNumber as <a href="#">Integer</a>          | Returns the starting byte position of the SubExpression given by matchNumber. Returns an <a href="#">Integer</a> .  |
| <b>Replace</b>             | [ReplacementPattern as <a href="#">String</a> ] | Substitutes the matched result in a manner specified by the given ReplacementPattern. If no ReplacementPattern is specified, it uses the ReplacementPattern which was specified in the <a href="#">RegEx</a> object at the time of the search. Returns a <a href="#">String</a> . |

**See Also** [RegEx](#) class.

# RegExOptions Class

Used to specify options when doing a search using regular expressions.

**Super Class** [Object](#)

## Properties

| Name          | Type                    | Description  |
|---------------|-------------------------|--|
| CaseSensitive | <a href="#">Boolean</a> | Specifies if case is to be considered when matching a string. The default is <a href="#">False</a> .   |
| DotMatchAll   | <a href="#">Boolean</a> | Normally the period matches everything except a new line, this option allows it to match new lines. The default is <a href="#">False</a> .   |
| Greedy        | <a href="#">Boolean</a> | <p>Greedy means the search finds everything from the beginning of the first delimiter to end of the last delimiter and everything in-between. For example, Say you want to match the following bold-tagged text in HTML:</p> <p>The &lt;b&gt;quick&lt;/b&gt; brown &lt;b&gt;fox&lt;/b&gt; jumped If you use this pattern:</p> <pre>&lt;b&gt;.+&lt;/b&gt;</pre> <p>You end up matching "&lt;b&gt;quick&lt;/b&gt; brown &lt;b&gt;fox&lt;/b&gt;", which isn't what you wanted.</p> <p>So, you can turn Greedy off or use this syntax:</p> <pre>&lt;b&gt;.+?&lt;/b&gt;</pre> <p>and you will match "&lt;b&gt;quick&lt;/b&gt;", which is exactly what you wanted.) Default is <a href="#">True</a>.</p> |
| LineEndType   | <a href="#">Integer</a> | <p>This is in effect for the current Regular Expression "session". Has no effect on SearchPatterns if <a href="#">TreatTargetAsOneLine</a> is <a href="#">True</a>. Changes the way \n is expanded for ReplacementPatterns</p> <p>0 = any line ending (Windows, Macintosh, or Unix)<br/>     1 = default for platform (if running on Macintosh, the same as 2) (if running on Windows, the same as 3, if running on Linux, the same as 4)<br/>     2 = Mac ASCII 13 or \r<br/>     3 = Win32 ASCII 10 or \n<br/>     4 = Unix ASCII 10 or \n<br/>     Default is 0.</p>  |

## RegExSearchPatternException Error

---

| Name                   | Type                    | Description  |
|------------------------|-------------------------|--|
| MatchEmpty             | <a href="#">Boolean</a> | Indicates whether patterns are allowed to match the empty string. Default is <a href="#">True</a> .                        |
| ReplaceAllMatches      | <a href="#">Boolean</a> | Indicates whether all occurrences of the pattern are to be replaced. The default is <a href="#">False</a> .                |
| StringBeginIsLineBegin | <a href="#">Boolean</a> | Indicates whether a string's beginning should be counted as the beginning of a line. The default is <a href="#">True</a> . |
| StringEndIsLineEnd     | <a href="#">Boolean</a> | Indicates whether a string's end should be counted as the end of a line. The default is <a href="#">True</a> .             |
| TreatTargetAsOneLine   | <a href="#">Boolean</a> | Ignores internal newlines for purposes of matching against '^' and '\$'. The default is <a href="#">False</a> .            |

**See Also** [RegEx](#), [RegExMatch](#) classes.

---

## RegExSearchPatternException Error

You used an invalid regular expression search pattern.

**Super Class** [RegExException](#) error.

**Example** The following exception block, placed at the end of a method that does a regular exception search, traps for invalid search patterns and prints the Message property of the RegExSearchPatternException. You could also use [RegExException](#), since RegExSearchPatternException is subclassed from [RegExException](#).

```
Exception err as RegExSearchPatternException  
MsgBox err.message
```

**See Also** [RegExException](#), [RuntimeException](#) classes [Try](#), [Exception](#) blocks, [Catch](#) statement.

---

## Rem Statement

Used to add comments to your code.

**Syntax** **Rem** *text string*

**Notes** REALbasic's compiler will ignore any comments you enter using the **Rem** statement. Comments are not included in stand-alone applications.

The ' (apostrophe) or two forward slashes (`//`) can be used to comment code as well. Comments can appear on separate lines or on the same line as executable code, provided they are to the right of any code that will execute. Valid comments appear in the Code Editor in red by default. You can change the color of comments in Options (Preferences on Mac OS X).

The Ctrl+' keyboard shortcut (Command-' on Macintosh) toggles the selected lines of code between commented out and 'live' states. The Comment and Uncomment buttons in the Code Editor toolbar do the same thing.

**Example**

The following code uses comments to document the results of [Boolean](#) comparisons:

```
Dim a, b, c, d As Boolean
a=True
b=False
d=a Or b Rem Evaluates to True
d=b And c Rem Evaluates to False
```

**See Also**

[//](#), [!](#) operators.

## Remove Method

Removes an element from an array.

**Syntax**

`array.Remove index`

| Part  | Type                    | Description                                     |
|-------|-------------------------|---|
| array | Any data type           | Required. The array to remove the element from. |
| index | <a href="#">Integer</a> | Required. The element number to be removed.     |

**Notes**

The **Remove** method removes the *index* element from the *array*.

The **Remove** method works with one-dimensional arrays only.

**Example**

This example removes element 4 from the *aNames* array:

```
aNames.Remove 4
```

**See Also**

[Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Redim](#), [Pop](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

# Replace Function

Replaces the first occurrence of a [string](#) with another [string](#).

**Syntax**    *result=Replace(sourceString, oldString, newString)*

OR

*result=stringVariable.Replace(oldString, newString)*

| Part           | Type                   | Description   |
|----------------|------------------------|---|
| result         | <a href="#">String</a> | A copy of sourceString with any changes made by the Replace function. |
| sourceString   | <a href="#">String</a> | The original string.  |
| oldString      | <a href="#">String</a> | The characters to be replaced.  |
| newString      | <a href="#">String</a> | The replacement characters.   |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .                         |

**Notes**    Replaces the first occurrence of oldString in sourceString with newString. **Replace** is case-insensitive.

If newString is an empty string (""), the **Replace** function deletes the first occurrence of the oldString in the sourceString.

If oldString is an empty string (""), the **Replace** function returns an unchanged copy of the sourceString.

**Examples**    Below are some examples that show the results of the **Replace** function:

```
Dim result As String  
result=Replace("The quick fox", "fox", "rabbit") //returns "The quick rabbit"  
result=Replace("The quick fox", "f", "b") //returns "The quick box"  
result=Replace("The quick fox", "quick", "") //returns "The fox"
```

Using the second syntax:

```
Dim result,s As String  
s="The quick fox"  
result=s.Replace("fox", "rabbit") //returns "The quick rabbit"  
MsgBox result
```

**See Also**    [ReplaceAll](#), [ReplaceAllB](#), [ReplaceB](#), [ReplaceLineEndings](#) functions.

# ReplaceB Function

Replaces the first occurrence of a [string](#) with another [string](#).

**Syntax** `result=ReplaceB(sourceString, oldString, newString)`

OR

`result=stringVariable.ReplaceB(oldString, newString)`

| Part           | Type                   | Description   |
|----------------|------------------------|---|
| result         | <a href="#">String</a> | A copy of sourceString with any changes made by the Replace function. |
| sourceString   | <a href="#">String</a> | The original string.  |
| oldString      | <a href="#">String</a> | The characters to be replaced.  |
| newString      | <a href="#">String</a> | The replacement characters.   |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .                         |

## Notes

Replaces the first occurrence of oldString in sourceString with newString. **ReplaceB** is the binary version of [Replace](#).

If newString is an empty string (""), the **ReplaceB** function deletes the first occurrence of the oldString in the sourceString.

If oldString is an empty string (""), the **ReplaceB** function returns an unchanged copy of the sourceString.

**ReplaceB** is case-sensitive; it treats sourceString as a series of raw bytes. It should be used instead of [Replace](#) when the string represents binary data or when your application will run in a one-byte character set (such as the US system) and you want case-sensitivity.

## Examples

Below are some examples that show the results of the **ReplaceB** function:

```
Dim result As String
result=ReplaceB("The quick fox","fox","rabbit") //returns "The quick rabbit"
result=ReplaceB("The quick fox","f","b") //returns "The quick box"
result=ReplaceB("The quick fox","quick","","") //returns "The fox"
```

Using the second syntax:

```
Dim result,s As String
s="The quick fox"
result=s.ReplaceB("fox","rabbit") //returns "The quick rabbit"
MsgBox result
```

## ReplaceAll Function

---

**See Also** [ReplaceAll](#), [ReplaceAllB](#), [Replace](#), [ReplaceLineEndings](#) functions.

## ReplaceAll Function

Replaces all occurrences of a [string](#) with another [string](#).

**Syntax** *result=ReplaceAll(sourceString, oldString, newString)*

OR

*result=stringVariable.ReplaceAll(oldString, newString)*

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | A copy of sourceString with any changes made by the ReplaceAll function. |
| sourceString   | <a href="#">String</a> | The original string.   |
| oldString      | <a href="#">String</a> | The characters to be replaced.   |
| newString      | <a href="#">String</a> | The replacement characters.  |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .                            |

### Notes

The **ReplaceAll** function replaces all occurrences of *oldString* in *sourceString* with *newString*. **ReplaceAll** is case-insensitive.

If *newString* is an empty string (""), the **ReplaceAll** function deletes every occurrence of the *oldString* in the *sourceString*.

If *oldString* is an empty string (""), the **ReplaceAll** function returns an unchanged copy of the *sourceString*.

### Examples

Below are some examples that show the results of the **ReplaceAll** function:

```
Dim result As String  
result=ReplaceAll("xyxyxy","x","z") //returns "zyzyzy"  
result=ReplaceAll("The quick fox",""," ") //returns "Thequickfox"
```

### See Also

[Replace](#), [ReplaceAll](#), [ReplaceAllB](#), [ReplaceLineEndings](#) functions.

---

## ReplaceAllB Function

Replaces all occurrences of a [string](#) with another [string](#).

**Syntax** *result=ReplaceAllB(sourceString, oldString, newString)*

**OR*****result=stringVariable.ReplaceAllB(oldString, newString)***

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | A copy of sourceString with any changes made by the ReplaceAll function. |
| sourceString   | <a href="#">String</a> | The original string.   |
| oldString      | <a href="#">String</a> | The characters to be replaced.   |
| newString      | <a href="#">String</a> | The replacement characters.  |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .                            |

**Notes**

The **ReplaceAllB** function replaces all occurrences of oldString in sourceString with newString. **ReplaceAllB** is case-sensitive because it treats the source string as a series of raw bytes.

If newString is an empty string (""), the **ReplaceAllB** function deletes every occurrence of the oldString in the sourceString.

If oldString is an empty string (""), the **ReplaceAllB** function returns an unchanged copy of the sourceString.

**ReplaceAllB** is case-sensitive; it treats sourceString as a series of raw bytes. It should be used instead of [ReplaceAll](#) when the string represents binary data or when your application will run in a one-byte character set (such as the US system) and you want case-sensitivity.

**Examples**

Below are some examples that show the results of the **ReplaceAll** function:

```
Dim result As String
result=ReplaceAllB("xyxyxy","x","z") //returns "zyzyzy"
result=ReplaceAllB("The quick fox","","") //returns "Thequickfox"
```

**See Also**

[Replace](#), [ReplaceB](#), [ReplaceAll](#), [ReplaceLineEndings](#) functions.

---

## ReplaceLineEndings Function

Replaces the line endings in the passed [String](#) with the specified line ending.

**Syntax*****result=ReplaceLineEndings(SourceString,LineEnding)***

| Part         | Type                   | Description   |
|--------------|------------------------|---|
| result       | <a href="#">String</a> | A copy of sourceString with the new line endings replacing the original line endings. |
| SourceString | <a href="#">String</a> | The original string.  |

| Part       | Type                   | Description   |
|------------|------------------------|---|
| LineEnding | <a href="#">String</a> | The line ending used to replace the line endings in <i>SourceString</i> . |

### Notes

**ReplaceLineEndings** does a global search-and-replace for the end of line characters in *SourceString* using the specified line ending as the replacement string. The search automatically recognizes Windows, Macintosh, and Unix line endings. Use this function to make multiline (or multi-paragraph) text compatible across platforms. The easiest way to specify the replacement line ending is with the [EndOfLine](#) class.

### Example

This example replaces the line endings in the text in an [EditField](#) with Windows line endings.

```
Dim s as String  
s=ReplaceLineEndings(EditField1.Text,EndOfLine.Windows)
```

### See Also

[EndOfLine](#) class.

---

## ResourceFork Class

ResourceFork Class objects are used to read and write data to resources. You can add resource files to the Project Editor (files of type “rsrc”).

### Super Class

[Object](#)

### Properties

| Name              | Parameters  | Description   |
|-------------------|---|---|
| GetIcl            | ID as <a href="#">Integer</a>                                     | Loads large (32 x 32) icl icon specified by ID.<br>Returns type <a href="#">Picture</a> . |
| GetIcs            | ID as <a href="#">Integer</a>                                     | Loads small (16 x 16) ics icon specified by ID.<br>Returns type <a href="#">Picture</a> . |
| ResourceLocked    | Type as <a href="#">String</a> ,<br>ID as <a href="#">Integer</a> | Used to get and set the locked attribute of the resource.                                 |
| ResourcePreload   | Type as <a href="#">String</a> ,<br>ID as <a href="#">Integer</a> | Used to get and set the Preload attribute of the resource.                                |
| ResourceProtected | Type as <a href="#">String</a> ,<br>ID as <a href="#">Integer</a> | Used to get and set the Protected attribute of the resource.                              |

| Name              | Parameters   | Description  |
|-------------------|--|--|
| ResourcePurgeable | Type as <a href="#">String</a> , ID as <a href="#">Integer</a> | Used to get and set the Purgeable attribute of the resource. |
| ResourceSysHeap   | Type as <a href="#">String</a> , ID as <a href="#">Integer</a> | Used to get and set the SysHeap attribute of the resource.   |
| TypeCount         | <a href="#">Integer</a>  | Number of resources types present in the resource fork.      |

## Methods

| Name             | Parameters   | Description   |
|------------------|--|---|
| AddPicture       | Pict as <a href="#">Picture</a> , ID as <a href="#">Integer</a> , Name as <a href="#">String</a>                                 | Creates a PICT resource using the Name and ID specified and fills it with the Pict specified.   |
| AddResource      | Data as <a href="#">String</a> , Type as <a href="#">String</a> , ID as <a href="#">Integer</a> , Name as <a href="#">String</a> | Adds a resource of the Type specified, using the Name and ID specified and fills it with the Data specified.  |
| Close            |  | Closes the open resource fork. The resource fork will be closed automatically when the instance is destroyed.   |
| GetCicn          | ID as <a href="#">Integer</a>  | Returns the CICN (color icon) resource as a <a href="#">Picture</a> based on the ID passed.   |
| GetCursor        | ID as <a href="#">Integer</a>  | Returns the CURS resource corresponding to ID as an object of type <a href="#">MouseCursor</a> .  |
| GetHandle        | Type as <a href="#">String</a> , ID as <a href="#">Integer</a>   | Returns a resource handle as an <a href="#">Integer</a> , given the type and ID of the desired resource. Intended for use with <a href="#">Declare</a> statements. The resource will be released when the resource fork is closed or when you call ReleaseHandle. |
| GetNamedPicture  | Name as <a href="#">String</a>   | Returns the PICT resource as a <a href="#">Picture</a> based on the Name passed.  |
| GetNamedResource | Type as <a href="#">String</a> , Name as <a href="#">String</a>  | Returns the specified resource as a <a href="#">string</a> .  |
| GetPicture       | ID as <a href="#">Integer</a>  | Returns the PICT resource as a <a href="#">Picture</a> based on the ID passed.  |
| GetResource      | Type as <a href="#">String</a> , ID as <a href="#">Integer</a>   | Returns the specified resource as a <a href="#">string</a> .  |
| GetSound         | ID as <a href="#">Integer</a>  | Returns the snd resource as a <a href="#">Sound</a> object based on the ID passed.  |
| ReleaseHandle    | Handle as <a href="#">Integer</a>  | Releases a resource handle obtained from GetHandle.   |

| Name           | Parameters  | Description   |
|----------------|---|---|
| RemoveResource | Type as <a href="#">String</a> , ID as <a href="#">Integer</a>    | Removes the specified resource from the resource fork.  |
| ResourceCount  | Type as <a href="#">String</a>                                    | Returns the number of resources of the specified type.  |
| ResourceID     | Type as <a href="#">String</a> , Index as <a href="#">Integer</a> | Returns the resource ID as an <a href="#">Integer</a> based on the Type and Index passed. Note: This list begins at zero. |
| ResourceName   | Type as <a href="#">String</a> , ID as <a href="#">Integer</a>    | Returns the resource name as a <a href="#">string</a> based on the Type and ID passed. Note: This list begins at zero.    |
| ResourceType   | Index as <a href="#">Integer</a>                                  | Returns the resource type as a <a href="#">string</a> based on the Index passed. Note: This list begins at zero.          |

### Notes

With one exception, access to the resourcefork is supported only on Macintosh. Check the value of the [TargetMacOS](#) constant before attempting to access the resourcefork.

The one exception is that you can add custom cursors to a project using the CURS resource and access them from a Windows build. See the Notes for the [MouseCursor](#) class and the section “Custom Cursors in Windows Applications” in the User’s Guide.

In built applications, all of the fields in the Get Info area of the Build Application dialog box are saved in a standard ‘vers’ resource. You can access this information — such as version number, region code, and release level — using the GetResource method of the [ResourceFork](#) class.

If you use resource numbers that conflict with internal REALbasic resources, unpredictable results may occur. One way to avoid resource ID conflicts is to choose high values, like 1128 instead of 128.

An alternate way of adding custom icons to your built application is to put your icon data (including ‘icns’ 32-bit icons) in a resources file that you add to your project. These icons will overwrite the icon resources supplied by REALbasic itself.

More information on the resource attribute properties (ResourceLocked, ResourcePreLoad, ResourceProtected, ResourcePurgeable, ResourceSysHeap) can be found in Apple Computer’s Inside Macintosh series.

### Examples

This example sets the locked attribute of a PICT resource whose ID is 128:

```
Dim rf as ResourceFork  
rf=GetFolderItem("MyApp").OpenResourceFork  
rf.ResourceLocked("PICT",128)=True
```

This example checks to see if the purgable property is set:

```
Dim rf as ResourceFork
rf=GetFolderItem("MyApp").OpenResourceFork
If rf.ResourcePurgeable("PICT",128) Then
    Beep
    MsgBox "This resource is purgeable."
End If
```

This example opens the project's resource fork. If you have added a resource file to your project, you can use this technique to access your resource file (assuming that the compiled application is running on Macintosh):

```
Dim rf as ResourceFork
rf=App.ResourceFork
```

This example gets the desktop application icon for Photoshop® and displays it in a [Canvas](#) control.

```
Dim rf as ResourceFork
Dim f as FolderItem
Dim p as Picture
f=GetFolderItem("Photoshop")
If f <> Nil then
    rf=f.OpenResourceFork
    If rf <> Nil then
        p=rf.GetIcon(128)
        Canvas1.backdrop=p
    Else
        MsgBox "Resource not found!"
    End if
Else
    MsgBox "File not found!"
End if
```

## Return Keyword

---

This example loads a cursor from the Photoshop application and uses it as the application's cursor.

```
Dim rf as ResourceFork
Dim m as MouseCursor
Dim f as FolderItem
f=GetFolderItem("Photoshop")
If f <> Nil then
  rf=f.OpenResourceFork
  If rf <> Nil then
    m=rf.GetCursor(1525)
    App.MouseCursor=m
  Else
    MsgBox "Resource not found!"
  End if
Else
  MsgBox "File not found!"
End if
```

**See Also** [App](#), [Cursors](#), [System](#) objects; [MouseCursor](#), [Window](#) class.

---

## Return Keyword

Returns flow of execution to the calling method.

### Syntax

**Return [value]**

| Part  | Type | Description  |
|-------|------|--|
| value | Any  | Value returned to the calling function. The data type must be specified in the Function declaration. |

### Notes

Use the **Return** keyword to immediately return control to the statement that called the method or function and to pass back a value. If the **Return** keyword is called in a method rather than a function, control is returned to the calling method without returning any value.

The value that a function returns can be an array or a single value. If you want to return an array, write empty parentheses after the name of the data type in the Return Type area in the Add Method area. For example, if you want to return an array of integers, write **Integer()** as the Return Type. If you need to return a multi-dimensional array, place one fewer commas in the parentheses than dimensions. For example, to return a two-dimensional array of Doubles, write **Double()** as the Return Type.

### See Also

[Function](#), [Sub](#) statements.

# RGB Function

Returns a [Color](#) based on the red, green, and blue values specified. You can also specify a color using the [HSV](#) or [CMY](#) function or the & color constant, [&cRRRBGG](#).

## Syntax

**result=RGB(red, green, blue)**

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Color</a>   | An object that represents the color based on the red, green and blue values. |
| red    | <a href="#">Integer</a> | The amount of red in the color (0-255).                                      |
| green  | <a href="#">Integer</a> | The amount of green in the color (0-255).                                    |
| blue   | <a href="#">Integer</a> | The amount of blue in the color (0-255).                                     |

## Notes

The **RGB** function returns a color object based on the amounts of red, green, and blue passed. These amounts are represented by integers between 0 and 255.

You can also specify a color using the RGB model using the [&c](#) literal. With the [&c](#) literal, you specify the amounts of red, green, and blue in hexadecimal rather than decimal.

## Examples

This example uses the **Rgb** function to assign various colors to the ForeColor property of a [Canvas](#) control.

```
canvas1.graphics.forecolor=Rgb(0,0,0)
//set to black
canvas1.graphics.forecolor=Rgb(255,0,0)
//set to red
canvas1.graphics.forecolor=Rgb(255,255,255)
//set to white
```

## See Also

[Color](#) data type; [CMY](#), [HSV](#), [SelectColor](#) functions; [&c](#) literal.

# RGBSurface Object

Used for direct-color pixel manipulations. The **RGBSurface** property of a [Picture](#) object allows you to manipulate the picture at the pixel level. Can be used only for pictures created by [NewPicture](#) with a pixel depth of 16 or 32.

Pixel manipulations using the **RGBSurface** property are faster than the same manipulations using the [Graphics](#) class methods.

## Methods

| Name      | Parameters   | Description   |
|-----------|--|---|
| FloodFill | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a><br>FillColor as<br><a href="#">Color</a>  | Performs a “floodfill” (the action performed by a PaintBucket tool in an image editing program) on an RGBSurface. The pixel at x, y and all adjacent pixels of the same color are changed to the passed <i>FillColor</i> .  |
| Pixel     | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a>   | Returns the color of the object at location specified by the x, y parameters. Returns a <a href="#">Color</a> .   |
| Transform | Map() as<br><a href="#">Integer</a><br>OR<br>RedMap() as<br><a href="#">Integer</a><br>GreenMap()<br>as <a href="#">Integer</a><br>BlueMap()<br>as <a href="#">Integer</a> | Applies a one-to-one pixel transformation to all pixels of an RGBSurface. You specify the transformation via either three lookup tables, RedMap, GreenMap, and BlueMap, or one lookup table to apply to all three channels.<br>Each map parameter is an 256 element array of integers.<br>The transformation works as follows: For each pixel, the pixel’s RGB value is used as an index into the map arrays and the value found becomes the new R, G, or B value for the pixel.<br>For example, if you set up a map such that map(i)=255-i, then calling Transform(Map) will invert the image. |

## Example

This example replaces all the black pixels in a [Picture](#), somePicture, with white:

```
Dim surf As RGBSurface
Dim x, y As Integer
surf = somePicture.RGBSurface
For y = somePicture.height - 1 downTo 0
  For x = somePicture.width - 1 downTo 0
    If surf.Pixel(x,y) = &c000000 then surf.Pixel(x,y) = &cFFFFFF
  next
next
```

The following example uses the FloodFill method to paint an **RGBSurface** object in a random color. It is in the MouseDown event of a [Canvas](#) control. The variable p is a [Picture](#) object.

```
Dim randomColor as Color
randomColor=RGB(Rnd*255,Rnd*255,Rnd*255) //create random color
p.RGBSurface.FloodFill x,y,randomColor //fill the area with the color
Me.Graphics.DrawPicture p,0,0 //repaint the canvas control
```

The following example uses the Transform method to invert an image:

```
Dim map(255), i as Integer
For i = 0 to 255
    map(i) = 255 - i
next
somePicture.RGBSurface.Transform map
```

**Note** 32-bit **RGBSurface** objects are mapped to 24-bit objects on Windows.

**See Also** [RGBSurface](#) property of the [Picture](#) object.

## Right Function

Returns the last n characters from the [string](#) specified.

**Syntax** *result=Right(source, count)*

OR

*result=stringVariable.Right(count)*

| Part           | Type                    | Description   |
|----------------|-------------------------|---|
| result         | <a href="#">String</a>  | The rightmost count characters of source.   |
| source         | <a href="#">String</a>  | The source string from which to get the characters.   |
| count          | <a href="#">Integer</a> | The number of characters you wish to get from the source. If count is greater than the length of source, all characters in source are returned. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .   |

**Notes** The **Right** function returns characters from the source string starting from the right side (as the name implies).

If you need to read bytes rather than characters, use the [RightB](#) function.

**Examples** This example uses the **Right** function to return the last 5 characters from a [string](#).

```
Dim s As String
s=Right("Hello World", 5) //returns "World"
```

**See Also** [Asc](#), [Chr](#), [InStr](#), [Left](#), [Len](#), [Mid](#) functions; [String](#) data type.

## RightB Function

Returns the last n bytes from the [string](#) specified.

**Syntax**    ***result=RightB(source, count)***

**OR**

***result=stringVariable.RightB(count)***

| Part           | Type                    | Description   |
|----------------|-------------------------|---|
| result         | <a href="#">String</a>  | The rightmost count bytes of source.  |
| source         | <a href="#">String</a>  | The source string from which to get the bytes.  |
| count          | <a href="#">Integer</a> | The number of bytes you wish to get from the source. If count is greater than the length of source, all bytes in source are returned. |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .   |

**Notes**

The **RightB** function returns bytes from the source string starting from the right side (as the name implies).

**RightB** treats *source* as a series of bytes rather than a series of characters. It should be used when *source* represents binary data. If you need to read characters rather than bytes, use the [Right](#) function.

**Examples**

This example uses the **RightB** function to return the last 5 bytes from a [string](#).

```
Dim s As String  
s=RightB("Hello World", 5) //returns "World"
```

**See Also**

[AscB](#), [ChrB](#), [InStrB](#), [LeftB](#), [LenB](#), [MidB](#), [Right](#) functions; [String](#) data type.

---

## Rnd Function

Returns a randomly generated number in the range  $0 \leq \text{Rnd} < 1$ . The equivalent functionality is provided by the [Random](#) class as a special case. The [Random](#) class also provides additional options, such as a random number selected from a Normal distribution.

**See Also**

[Random](#) class.

# Round Function

Returns the value specified rounded to the nearest [Integer](#).

## Syntax

**result=Round (value)**

| Part   | Type                   | Description                  |
|--------|------------------------|------------------------------|
| result | <a href="#">Double</a> | The rounded value.           |
| value  | <a href="#">Double</a> | The value you want to round. |

## Notes

The **Round** function returns the value passed to it rounded to the nearest [Integer](#). Fractional values greater than or equal to .5 are rounded up; those below .5 are rounded down, as shown in the example.

## Examples

This example uses the **Round** function to return a rounded number.

```
Dim d as Double
d=Round(1.499) //returns 1
d=Round(1.500) //returns 2
```

## See Also

[Ceil](#), [Floor](#) functions.

# RoundRectangle Control

Draws a rectangle with rounded corners.

## Super Class

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| BorderWidth | <a href="#">Integer</a> | The width of the round rectangle's border in pixels. |
| FillColor   | <a href="#">Color</a>   | The color of the interior of the RoundRectangle.     |
| OvalHeight  | <a href="#">Integer</a> | The height of the rounded corners.                   |
| OvalWidth   | <a href="#">Integer</a> | The width of the rounded corners.                    |

## RoundRectShape Class

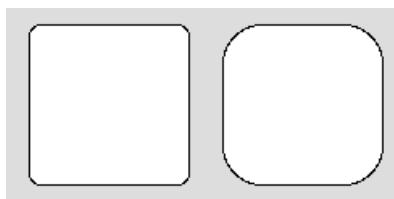
---

### Events

| Name      | Parameters   | Description   |
|-----------|--|---|
| MouseDown | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Round Rectangle region at the location passed in to x,y. <a href="#">Return True</a> if you are going to handle the MouseDown.  |
| MouseDrag | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Round Rectangle and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |
| MouseUp   | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the Round Rectangle region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.                                  |

### Notes

Increasing the OvalWidth and OvalHeight creates rounded corners that are have a more gradual arc. In the figure below, the round rectangle on the left has an OvalWidth and OvalHeight of 16. The round rectangle on the right has an OvalWidth and OvalHeight of 50.



### Example

The following code defines a custom **RoundRectangle**:

```
roundrectangle1.fillColor=RGB(255,0,0) //Red  
roundrectangle1.ovalheight=30  
roundrectangle1.ovalwidth=16  
roundrectangle1.borderwidth=1
```

### See Also

[Rectangle](#) control; [RectControl](#) class.

---

## RoundRectShape Class

Draws a (two-dimensional) rounded rectangle in a vector graphics environment.

### SuperClass

[RectShape](#)

## Properties

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| Cornerheight | <a href="#">Double</a>  | Height of the inset corner oval.   |
| Cornerwidth  | <a href="#">Double</a>  | Width of the inset corner oval.  |
| Segments     | <a href="#">Integer</a> | The number of polygon sides to use when approximation is needed. Zero means it's up to the renderer. |

## Example

The following method in the Paint event of a window draws a square with rounded corners.

```
Dim r as New RoundRectShape
r.width=75
r.height=75
r.border=100
r.bordercolor=RGB\(0,0,0\)
r.fillcolor=RGB\(255,102,102\)
r.cornerHeight=15
r.cornerRadius=15
r.borderWidth=2.5
g.DrawObject r,150,150
```

## See Also

[ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Group2D](#), [Graphics](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#), [StringShape](#) classes.

## RTrim Function

Returns the [string](#) data type passed with trailing (right side) whitespaces removed.

### Syntax

**result=RTrim(*SourceString*)**

OR

**result=*stringVariable*.RTrim**

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | SourceString with trailing whitespaces removed.                                |
| SourceString   | <a href="#">String</a> | The source, a copy of which, to be returned with trailing whitespaces removed. |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .                                  |

### Notes

**Rtrim** uses the list of unicode “whitespace” characters at <http://www.unicode.org/Public/UNIDATA/PropList.txt>.

**Examples** This example removes the whitespaces from the right side of the string passed:

```
Dim s as String  
s=RTrim(" Hello World ") //Returns " Hello World"
```

**See Also** [Asc](#), [Chr](#), [InStr](#), [Len](#), [Left](#), [LTrim](#), [Mid](#), [Trim](#) functions; [String](#) data type.

## Runtime Object

Returns information about the current state of the runtime environment. This information can be useful for debugging purposes.

### Properties

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| ObjectCount | <a href="#">Integer</a> | Current number of objects in existence.  |
| ObjectClass | <a href="#">String</a>  | Takes ObjectNumber ( <a href="#">Integer</a> ) as parameter, where ObjectNumber is greater than or equal to zero and less than ObjectCount. Returns the class of the passed object                 |
| ObjectRefs  | <a href="#">Integer</a> | Takes ObjectNumber ( <a href="#">Integer</a> ) as parameter, where ObjectNumber is greater than or equal to zero and less than ObjectCount. Returns the number of references to the passed object. |
| ObjectID    | <a href="#">Integer</a> | Takes ObjectNumber ( <a href="#">Integer</a> ) as parameter, where ObjectNumber is greater than or equal to zero and less than ObjectCount. Returns a unique identifier for the passed object.     |
| MemoryUsed  | <a href="#">Integer</a> | Returns the total amount of memory used (in bytes) by the allocated objects.   |

### Example

The following code displays the properties related to individual objects in a multicolumn [ListBox](#) and the total amount of memory used in an [EditField](#):

```
Dim i, total as Integer  
total=Runtime.ObjectCount  
For i=0 to total-1  
    Listbox1.addrow Str(Runtime.ObjectID(i))  
    Listbox1.Cell(i,1)=Runtime.ObjectClass(i)  
    Listbox1.Cell(i,2)=Str(Runtime.ObjectRefs(i))  
Next  
EditField1.text=Str(Runtime.memoryUsed)
```

### See Also

[System](#) object.

# RuntimeException Class

Super Class [Object](#)

## Properties

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| ErrorNumber | <a href="#">Integer</a> | Used to contain an error number that describes the runtime error.  |
| Message     | <a href="#">String</a>  | Used to contain descriptive text to display when the runtime exception is encountered.   |
| Stack       | <a href="#">String</a>  | A <a href="#">String</a> array that contains a list of all of the methods in the stack from the main entrypoint to the point at which the exception was invoked. The first element (0) contains the current function and the methods in the stack continue from there. This feature only works if the Include Function Names option is enabled in the 'blessed' App class's properties.. |

## Notes

Use runtime exceptions to trap errors so they can be handled. For example, reading from or writing to an array element that does not exist will generate an [OutOfBoundsException](#). Exceptions have no methods or events.

| Name                                      | Description   |
|---|---|
| <a href="#">FunctionNotFoundException</a> | A function declared using the <a href="#">Declare</a> statement's "Soft" keyword could not be loaded.   |
| <a href="#">IllegalCastException</a>      | You cast an object to a different class and sent it a message its real class can't accept.  |
| <a href="#">InvalidParentException</a>    | You tried to get the parent of a control using the Parent property of the <a href="#">Control</a> class, but its parent is in a different window. |
| <a href="#">KeyChainException</a>         | A method of the <a href="#">KeyChain</a> or <a href="#">KeyChainItem</a> classes failed.  |
| <a href="#">KeyNotFoundException</a>      | An attempt was made to access a <a href="#">Dictionary</a> item with a key that the dictionary does not contain.                                  |
| <a href="#">NilObjectException</a>        | An attempt was made to access an object that does not exist.  |
| <a href="#">OLEException</a>              | An OLE-related runtime exception occurred. Handle errors in Office Automation code via the <a href="#">OLEException</a> class.                    |
| <a href="#">OutOfBoundsException</a>      | An attempt was made to read from or write to a value, character, or element outside the bounds of the object or data type.                        |

## RuntimeException Class

---

| Name  | Description  |
|---|--|
| <a href="#">OutOfMemoryException</a>            | Raised in certain cases when an operation cannot be completed due to insufficient memory.  |
| <a href="#">RBScriptAlreadyRunningException</a> | The user tried to modify an <a href="#">RBscript</a> that is already executing or tried to modify the context of the script while it is running.   |
| <a href="#">RegExException</a>                  | The <a href="#">RegEx</a> engine issued a runtime exception. Currently this means that you used an invalid search pattern in a Regular Expression. In the future, other types of regular expression exceptions may be added.   |
| <a href="#">RegistryAccessErrorException</a>    | You tried to use the <a href="#">RegistryItem</a> class without proper access privileges or tried to use it under any Macintosh OS or Linux. It is a Windows-only feature.   |
| <a href="#">ShellNotRunningException</a>        | You tried to access an asynchronous or interactive shell session, but the shell was not running.   |
| <a href="#">SOAPException</a>                   | SOAPExceptions can be raised when using a WSDL to define your SOAP function. If the method name does not exist or the parameters passed do not match the WSDL specifications, a SOAPException runtime error will be raised.  |
| <a href="#">SpotlightException</a>              | An error related to a <a href="#">SpotlightQuery</a> was encountered, such as an invalid query.  |
| <a href="#">StackOverflowException</a>          | When one routine (method/event handler/menu handler) calls another, memory is used to keep track of the place in each routine where it was called along with the values of its local variables. The purpose of this is to return (when the routine being called finishes) to the previous routine with all local variables as they were before. The memory set aside for tracking this is called the Stack (because you are "stacking" one routine on top of another). If your application runs out of stack space, a StackOverflowException will occur. You should be able to test your application thoroughly enough to prevent this error from occurring. |
| <a href="#">ThreadAlreadyRunningException</a>   | You tried to change the stack size of a <a href="#">Thread</a> while it was running.   |
| <a href="#">TypeMismatchException</a>           | You tried to assign to an object the wrong data type.  |
| <a href="#">UnsupportedFormatException</a>      | You used a string expression that does not evaluate to a number or tried to open or save an unsupported picture format.  |

| Name                               | Description   |
|------------------------------------|---|
| <a href="#">XMLDOMException</a>    | This exception may occur during the creation of a DOM document.     |
| <a href="#">XMLException</a>       | There was an error in parsing XML.                                  |
| <a href="#">XMLReaderException</a> | There was an error in parsing XML using <a href="#">XMLReader</a> . |

If you fail to trap a runtime error in an exception block, it will trigger the `UnhandledException` event of the [Application](#) class. You can then handle all runtime exceptions generically within that event handler.

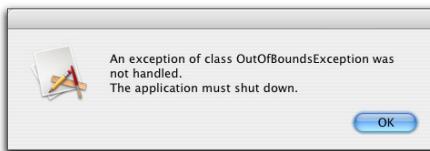
When using the Office Automation plug-ins, you can include an exception block that references these classes. For example:

```
Exception err as OLEException
MsgBox err.message+ " Error No.: "+Str(err.code)
```

Third-party plug-ins may also incorporate their own types of runtime exceptions.

When you are testing your application in the IDE, execution will stop at the offending line and you will see an error message in your Code Editor. When the error occurs in a built application, the user will see a generic error message (as shown in the Notes section below) and quit — unless you include an exception handler.

Runtime Exceptions are used to trap errors in programming. Without an exception handler, the user will see a generic message box from REALbasic that a particular type of exception has occurred.



The application will quit when the user accepts this message box.

Including exception handlers in your code allows you to display information that will help you determine the cause of the problem and prevents your application from quitting automatically.

Say, for example, you are getting an [OutOfBoundsException](#) when trying to access an element of an array. You could put in an exception handler to display the number of items in the array and the item number you were attempting to change along with any other useful information that would help you to track down the source of the bug. See the section on [OutOfBoundsException](#) for an example of this.

There is another example of exception handling in the section on [IllegalCastException](#).

### Examples

This example includes an [If](#) statement that checks for the type of exception and displays a message box indicating the type of exception that has occurred. The [Exception](#) block goes after the last line in the method. Note that its indentation is at the same level as the [Sub](#) statement.

```
Sub Action
    Dim a(3) as String
    a(4)="geoff"
    Exception err
    If err IsA NilObjectException then
        MsgBox "Nil Object Exception"
    elseif err IsA OutOfBoundsException then
        MsgBox "Out of Bounds"
    elseif err IsA TypeMismatchException then
        MsgBox "Type Mismatch"
    End if
```

Instead of the [If](#) statement, you can use several [Exception](#) blocks, each of which handles a specific type of runtime exception.

```
Dim f as FolderItem
f.delete
Exception err as NilObjectException
MsgBox "Nilobjectexception"
Exception err as OutOfBoundsException
MsgBox "Out of Bounds"
Exception err as TypeMismatchException
MsgBox "typemismatch"
```

The following example checks for the type of exception and handles the exception if it is a [NilObjectException](#). If the exception is not a [NilObjectException](#), it lets execution continue through the calling chain by calling [Raise](#).

```
Dim f as FolderItem
f.Delete
Exception err
If err IsA NilObjectException then
    err.message="Drat!"
    MsgBox err.message
else
    Raise err
end if
```

### See Also

[Function](#), [Raise](#), [Sub](#) statements; [FunctionNotFoundException](#), [IllegalCastException](#), [InvalidParentException](#), [KeyChainException](#), [KeyNotFoundException](#), [NilObjectException](#), [OLEException](#), [OutOfBoundsException](#), [OutOfMemoryException](#), [RbScriptAlreadyRunningException](#), [RegExException](#),

[RegExSearchPatternException](#), [ShellNotRunningException](#), [SOAPException](#), [StackOverFlowException](#), [TypeMismatchException](#), [XMLErrorException](#), [XMLException](#), [XMLReaderException](#) errors; [Exception](#), [Try](#) blocks; [Nil](#) keyword.

---

## SaveAsDialog Class

Used to create customized Save As dialogs. Non-customizable Save As dialogs are created by the [GetSaveFolderItem](#) function.

**Super Class** [FolderItemDialog](#)

**Notes** Using the properties of the [FolderItemDialog](#) class, you can customize the following aspects of a save-file dialog box:

- Position (Left and Top properties)
- Default directory (Initial Directory property)
- Valid file types to show (Filter property). If you specify more than one file type (separate the file type names with semicolons), **SaveAsDialog** will add a Format pop-up menu to the dialog, allowing the user to specify the desired file type to use. Examine the extension of the file name returned to see which format was chosen.
- Default filename (SuggestedFileName property)
- Text of the Validate and Cancel buttons (ActionButtonCaption and CancelButtonCaption properties)
- Text that appears in the Title bar of the dialog (Title property)
- Text that appears in the body of the dialog (PromptText property)

On Mac OS X, a Hide Filename Extension checkbox appears in the save-file dialog. The [FolderItem](#) returned has its ExtensionVisible property set according to the user's use of this checkbox.

## Screen Class

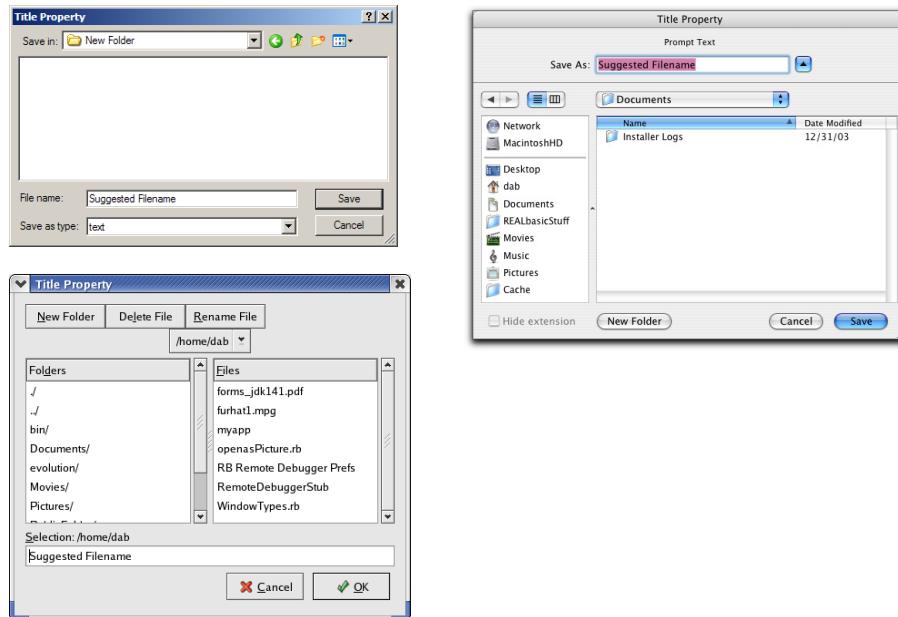
---

### Example

The following example opens a customized save-file dialog box and displays the contents of the “Documents” directory in the browser area.

```
Dim dlg as New SaveAsDialog
Dim f as FolderItem
dlg.InitialDirectory=Volume(0).Child("Documents")
dlg.promptText="Prompt Text"
dlg.SuggestedFileName="Suggested Filename"
dlg.Title="Title Property"
dlg.Filter=TextTypes.text //defined as a file type in TextTypes File Type Set
f=dlg.ShowModal()
If f <> Nil then
    //file saved
Else
    //user canceled
End if
```

This is what it looks like on the Windows, Macintosh, and Linux platforms:



### See Also

[FileType](#), [FolderItem](#), [FolderItemDialog](#), [OpenDialog](#), [SelectFolderDialog](#) classes;  
[GetSaveFolderItem](#) function.

---

## Screen Class

Used to get information about the monitors connected to the user's computer.

Super Class [Object](#)

## Properties

| Name                   | Type                    | Description   |
|------------------------|-------------------------|---|
| <b>AvailableHeight</b> | <a href="#">Integer</a> | The available height in pixels of the specified screen, taking into account the menubar and/or dock, if present.  |
| <b>AvailableLeft</b>   | <a href="#">Integer</a> | The number of usable pixels from the left edge of the specified screen. This would be zero unless, for example, the user had placed the Dock on the left.   |
| <b>AvailableTop</b>    | <a href="#">Integer</a> | The number of usable pixels from the top edge of the specified screen. It takes into account the height of the menubar, if present on the specified screen. |
| <b>AvailableWidth</b>  | <a href="#">Integer</a> | The available width of the screen in pixels, taking into account the Dock, if placed on the left or right of the specified screen.                          |
| <b>Depth</b>           | <a href="#">Integer</a> | The color depth of the screen.  |
| <b>Height</b>          | <a href="#">Integer</a> | The height of the screen.   |
| <b>Left</b>            | <a href="#">Integer</a> | The left coordinate of the screen relative to the main screen. For the main screen, it is zero.   |
| <b>Top</b>             | <a href="#">Integer</a> | The top coordinate of the screen relative to the main screen. For the main screen, it is zero.  |
| <b>Width</b>           | <a href="#">Integer</a> | The width of the screen.  |

## Notes

Although you cannot create an object of type **Screen**, you can access screen objects through the [Screen](#) function. See the [Screen](#) and [ScreenCount](#) functions for more information.

## Example

The following reports on the values of AvailableLeft, AvailableHeight, AvailableTop, and AvailableWidth for the main screen. If the user has the Hide the Dock option on, the size of the dock when it is temporarily made visible is not taken into account.

```
Dim s as String
s="Left=" +Str(Screen(0).availableleft)+EndOfLine
s=s+"Width=" +Str(Screen(0).availablewidth)+EndOfLine
s=s+"Top=" +Str(Screen(0).availabletop)+EndOfLine
s=s+"Height=" +Str(Screen(0).availableheight)+EndOfLine
MsgBox s
```

## Screen Function

---

The following example reports on the characteristics of the user's main screen. The code is placed in the Open event of a window and displays its results in a series of EditFields in the window.

```
HeightField.text=Str(Screen(0).height)
WidthField.text=Str(Screen(0).width)
AvailableHeightField.text=Str(Screen(0).availableHeight)
AvailableWidthField.text=Str(Screen(0).availablewidth)
AvailableTopField.text=Str(Screen(0).availableTop)
AvailableLeftField.text=Str(Screen(0).availableLeft)
DepthField.text=Str(Screen(0).depth)
```

**See Also** [Screen](#), [ScreenCount](#) functions.

## Screen Function

Used to access the properties of a [Screen](#) object. Returns a reference to the screen passed.

### Syntax

**Screen(index).property**

| Part     | Type                            | Description   |
|----------|---------------------------------|---|
| index    | <a href="#">Integer</a>         | The number of the screen whose property you wish to read. 0=the main screen (the screen with the menu bar). |
| property | Any screen object property name | The property you wish to read. See the description of the <a href="#">Screen</a> class.                     |

### Notes

You can use the **Screen** function to access the screen properties for the monitors attached to the user's computer. Screen 0 is the main screen (the one with the menu bar). You can use the [ScreenCount](#) function to determine how many screens exist.

### Examples

This example displays the size of the user's main screen.

```
MsgBox "Your main screen is "+Str(Screen(0).width)+" by "+Str(Screen(0).height)+"."
```

### See Also

[Screen](#) class; [ScreenCount](#) function.

## ScreenCount Function

Used to determine the number of screens connected to the user's computer.

**Syntax****result=ScreenCount**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Integer</a> | Any container expecting an <a href="#">Integer</a> value. |

**Notes**

The **ScreenCount** function returns the number of screens (monitors) connected to the user's computer.

**Example**

This example reports on the number of monitors attached to the user's computer.

```
Dim myscreens as Integer
myscreens=ScreenCount
If (myscreens=1) then
  MsgBox "You've only got one monitor."
else
  MsgBox "You've got "+Str(myscreens)+" screens!"
end if
```

**See Also**

[Screen](#) class; [Screen](#) function.

## Scrollbar Control

The scrollbar control.

**Super Class**

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

**Properties**

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| AcceptFocus | <a href="#">Boolean</a> | If <a href="#">True</a> , the Scrollbar can accept the focus. The default is <a href="#">False</a> . The default is <a href="#">False</a> . This property is Windows-only.                        |
| LineStep    | <a href="#">Integer</a> | The amount the value changes when a scroll arrow is clicked. The default is 1   |
| LiveScroll  | <a href="#">Boolean</a> | If <a href="#">True</a> , a ValueChanged event occurs as the user drags the thumbnail in the scrollbar. Otherwise, a single ValueChanged event occurs when the user stops dragging the thumbnail. |
| Maximum     | <a href="#">Integer</a> | The maximum value of the scrollbar. The default is 100. For Mac OS 8.5 and above only, the Maximum value is much larger.  |
| Minimum     | <a href="#">Integer</a> | The minimum value of the scrollbar. The default is 0. See note above regarding Mac OS.  |

## Scrollbar Control

| Name     | Type                    | Description   |
|----------|-------------------------|---|
| PageStep | <a href="#">Integer</a> | The amount the value changes when the user clicks in the Scrollbar track. |
| Value    | <a href="#">Integer</a> | The current value of the scrollbar.                                       |

## Events

| Name         | Parameters   | Description   |
|--------------|--|---|
| GotFocus     |  | (Windows and Linux) The scrollbar has received the focus. On Linux, the scrollbar is not in the tab order but it gets the focus when the user clicks on it. Scrollbars do not get the focus on Macintosh even when the user is manipulating them.   |
| LostFocus    |  | (Windows and Linux) The scrollbar has lost the focus.   |
| MouseDown    | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the scrollbar at the location passed in to x,y. Returns a <a href="#">Boolean</a> . <a href="#">Return True</a> if you are going to handle the MouseDown. The coordinates x, and y are local to the control. Point 0,0 is the top-left corner of the entire control (to the left of the label and above the box that is drawn). |
| MouseDrag    | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the Scrollbar and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |
| MouseUp      | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was released inside the scrollbar at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |
| ValueChanged |  | The scrollbar value has changed.  |

## Examples

Changing the maximum value of a **Scrollbar** at runtime:

```
ScrollBar1.maximum=200
```

Setting the text of a [StaticText](#) control to the value of the **Scrollbar** when the user scrolls:

```
Sub ValueChanged()  
    staticText1.text=Str(scrollbar1.value)
```

See also the discussion of scrolling [ListBox](#) controls horizontally using a **Scrollbar** control.

## See Also

[Slider](#) control; [RectControl](#) class.

# Select Case Statement

Executes one of several groups of statements, depending on the value of an expression.

## Syntax

**Select Case testExpression**

[**Case [Is] expression-n**

**statements-n]**

[**Else or Case Else**

**elseStatements]**

**End [Select]**

| Part           | Description  |
|----------------|--|
| testExpression | Any expression that evaluates to a value. The expression can be of any data type or an object.                   |
| expression-n   | Any expression or list of expressions. You can use a function that evaluates to the data type of testExpression. |
| statements-n   | Statements to be executed if expression-n is true.   |
| elseStatements | Statements to be executed if no expressions are <a href="#">True</a> .   |

## Notes

The **Select Case** statement is useful when there are several possible conditions that must be checked. Unlike an [If](#) statement, the **Select Case** statement will exit as soon as it finds a matching **Case** expression and executes any statements that follow the **Case** expression up to the next **Case** expression. If there are no **Case** expressions that match, the **elseStatements** are executed.

The expression **Case Else** can be used as a synonym for **Else**.

The **Case** statement can accept several types of expressions. The expression can be a single value, a comma-delimited list of values, a function that returns a value, a range of values specified with the “To” keyword, or an expression that uses the “Is” keyword to do an equality or inequality test. You can combine types of expressions, separating them by commas

Here are some examples:

```
Case 2, 4, 6, 8 //several values
Case 2 To 5 //range of values using To
Case 2 To 5, 7,9,11 //Both separate values and range
Case myFunction(x) // a Function
Case Is >= 42 // greater than/equal to operator
Case Is <19 //less than operator
```

## Select Case Statement

---

### Example

This example uses the **Select Case** statement to determine which day of the week the user has entered into an [EditField](#) and displays a message based on that information.

```
Dim dayNumber As Integer
Dim msg As String
dayNumber=Val(EditField1.text)
Select Case dayNumber
Case 2
    msg="It's Monday"
Case 3
    msg="It's Tuesday"
Case 4
    msg="It's Wednesday"
Case 5
    msg="It's Thursday"
Case 6
    msg="It's Friday"
Else
    msg="It's the weekend."
End Select
MsgBox msg
```

The following example uses the **Select Case** statement to determine which button was pressed in a [MessageDialog](#). Notice that it compares objects of type [MessageDialogButton](#).

```
Dim d as New MessageDialog //declare the MessageDialog object
Dim b as MessageDialogButton //for handling the result
d.icon=MessageDialog.GraphicCaution //display warning icon
d.ActionButton.Caption="Save"
d.CancelButton.Visible=True //show the Cancel button
d.CancelButton.Cancel=True //esc key works for Cancel
d.AlternateActionButton.Visible=True //show the "Don't Save" button
d.Message="Save changes before closing?"
d.Explanation="If you don't save your changes, you will lose "
    +"all that important work you did since your last coffee break."
b=d.ShowModal //display the dialog
Select Case b //b is a MessageDialogButton
Case d.ActionButton //determine which button was pressed.
    //user pressed Save
Case d.AlternateActionButton
    //user pressed Don't Save
Case d.CancelButton
    //user pressed Cancel
End Select
```

### See Also

[If...Then...Else](#) statement.

## SelectColor Function

Displays the standard Color picker, allowing the user to choose a color.

### Syntax

**result=SelectColor(ByRef col, prompt)**

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if the user clicks OK; <a href="#">False</a> if the user cancels.   |
| col    | <a href="#">Color</a>   | <a href="#">Color</a> passed to SelectColor and the color the user selects. Note usage of <a href="#">ByRef</a> in the parameter list. |
| prompt | <a href="#">String</a>  | Prompt to be displayed in the Color Picker. Shown only on Macintosh.   |

### Notes

You control the default choice of color displayed by the Color Picker by passing a value of *col* to **SelectColor**. If *col* is not assigned a value, black is used as the default color. The color that the user chooses is returned in *col*. If the user clicks Cancel without making a choice, *result* is set to [False](#); otherwise it is set to [True](#).

### Example

The following example allows the user to select a color that will be used as the fillcolor in a [Rectangle](#) control.

```
Dim c as Color
Dim b as Boolean
c=CMY(.35,.9,.6) //choose the default color shown in color picker
b=SelectColor(c, "Select a Color")
Rectangle1.FillColor=c
```

### See Also

[Color](#) data type; [CMY](#), [RGB](#), [HSV](#) functions; [&c](#) literal.

## SelectFolder Function

Displays an open-file dialog box allowing the user to select a folder rather than a file.

### Syntax

**result=SelectFolder**

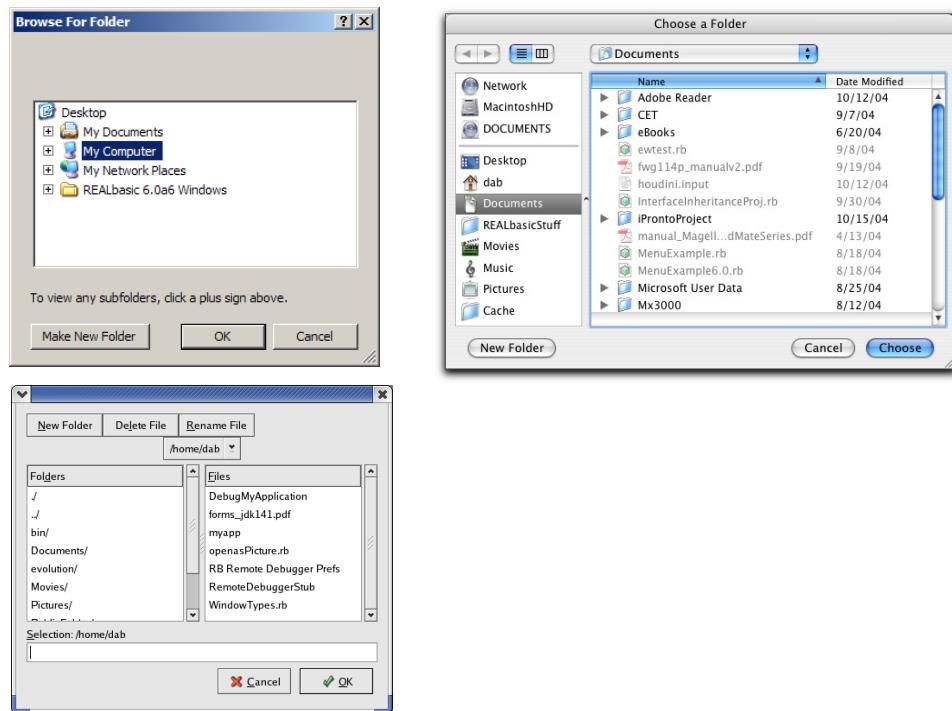
| Part   | Type                       | Description                   |
|--------|----------------------------|-------------------------------|
| result | <a href="#">FolderItem</a> | The folder the user selected. |

### Notes

The **SelectFolder** function displays a dialog box similar to the open-file dialog box displayed by the [GetOpenFolderItem](#) function. The difference is that the dialog box displayed by the **SelectFolder** function allows the user to choose a folder rather than a file.

## SelectFileDialog Class

This is what the dialog looks like on Windows, Macintosh, and Linux.



### Examples

This example displays the name of the folder the user chose.

```
Dim f as FolderItem
f=SelectFolder
If f<> Nil Then
  MsgBox f.name
End If
```

### See Also

[GetOpenFolderItem](#), [GetSaveFolderItem](#) functions; [SelectFileDialog](#) class.

## SelectFileDialog Class

Used to create and present customized Select Folder dialog boxes. The non-customized version of this function is provided by the [SelectFolder](#) function.

**Super Class** [FolderInputDialog](#) class.

**Notes** Using the properties of the [FolderInputDialog](#) class, you can customize the following aspects of a select-folder dialog box:

- Position (Left and Top properties)
- Default directory (Initial Directory property)
- Valid file types to show (Filter property)
- Text of Validate and Cancel buttons (ActionButtonCaption and CancelButtonCaption properties).
- Text that appears in the Title bar of the dialog (Title property)
- Text that appears in the body of the dialog (PromptText property)

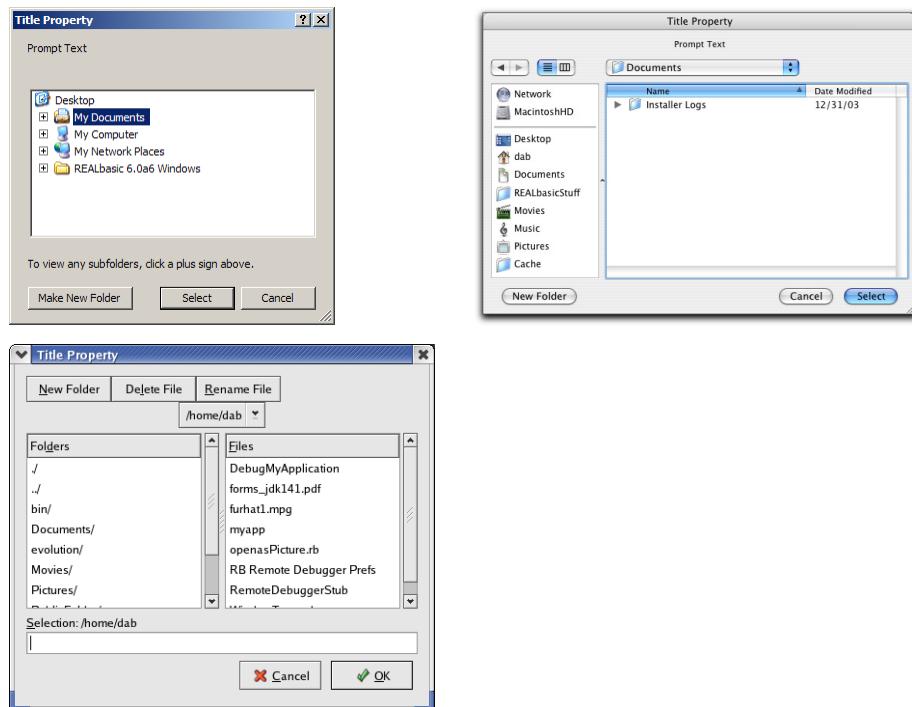
**Example** The following example opens a select folder dialog box and presents the contents of the “Documents” directory on the user’s startup volume in the browser:

```
Dim dlg as New SelectFileDialog
Dim f as FolderItem
dlg.ActionButtonCaption="Select"
dlg.Title="Title Property"
dlg.PromptText="Prompt Text"
dlg.InitialDirectory=Volume(0).Child("Documents")
f=dlg.ShowModal()
if f <> Nil then
    //use the folderitem here
else
    //user cancelled
end if
```

## Self Keyword

---

This is what it looks like on Windows, Macintosh, and Linux:



## See Also

[FileType](#), [FolderItem](#), [FolderItemDialog](#), [OpenDialog](#), [SaveAsDialog](#), classes; [SelectFolder](#) function.

---

## Self Keyword

**Self** is a reference to an object's parent object.

### Notes

When called within the method of an object on a form, **Self** will always be a reference to the object's parent window. This means that in a [PushButton](#) control's Action event handler:

`Left = 100` and `Self.Left = 100`

mean the same thing because, in the absence of an object identifier, the parent object (in this case, the window the PushButton is a part of) is assumed.

For code for a control in a window, "Self" refers to the window, and [Me](#) refers to the control.

When **Self** is called within the method of a class, **Self** will be a reference to the instance of the class in use. This is usually not necessary since, as stated, in the absence of an object identifier, the parent or super object is assumed.

**See Also** [Me](#) keyword.

---

## Self Cannot be used in the Method of a Module Error

You can't use [Self](#) or [Me](#) in a method or function in a module because there is no parent window or control.

**See Also** [Self](#) keyword.

---

## Semaphore Class

Used to control access to resources in a multithreaded environment.

**Super Class** [Object](#)

**Constructor** The **Semaphore** has a constructor that takes the initial count of the number of resources the semaphore is protecting as an optional parameter.

| Name      | Parameters                              | Description  |
|-----------|---|--|
| Semaphore | NumResources as <a href="#">Integer</a> | The number of resources the Semaphore is intended to protect. This defaults to 1. If you need to manage access to a single resource, you can instead use a <a href="#">CriticalSection</a> . |

Every time you successfully obtain a lock on the resource, the **Semaphore** will decrement its internal count of available resources. When there are no more, threads that request locks will begin to block and wait for resources. This is why you are allowed to pass the initial count of resources, to give you more control over the behavior of the **Semaphore**.

The Semaphore class is different from the [CriticalSection](#) and [Mutex](#) classes in this way: calling Signal in the same thread will cause the counter to decrement. If you call Signal recursively, you will cause the application to hang.

### Methods

| Name      | Parameters | Description  |
|-----------|------------|--|
| Signal    |            | Call Signal to obtain a lock on a resource. When you obtain a lock, you can use the resource without fear that another thread will try to use it simultaneously. If the call succeeds, this function returns immediately and your code continues to execute. If the lock is not available, then the thread that called this method will wait until the lock becomes available. When the call returns, you will have the lock, but you may have to wait until the lock becomes available. When you are finished using the resource, call the Release method to give it back to the Semaphore.                                     |
| Release   |            | Call Release to give a locked resource back to the Semaphore. You must call Release when you are finished using a locked resource in order to make it available to other requests.   |
| TrySignal |            | Returns a <a href="#">Boolean</a> . This is a more friendly version of the Signal method that gives you a sneak-peek to determine whether there is a lock available. If there is a lock available, you are granted the lock and TrySignal returns <a href="#">True</a> . When you are finished with the resource, call Release to give it back to the Semaphore. If the lock is not available, you do not wait for it. Instead, TrySignal returns <a href="#">False</a> to let you know. Do not attempt to use the resource after it returns <a href="#">False</a> because you are likely to collide with another thread. If you |

### Notes

A **Semaphore** is an object that can be used to coordinate access to a shared resource.

To acquire the ownership of a semaphore, a thread calls the Signal or TrySignal methods. If the Semaphore isn't owned, the thread acquires ownership, otherwise the thread will be forced to wait until the Semaphore is released via the Release method by the owning thread.

### See Also

[CriticalSection](#), [Mutex](#), [Thread](#) classes.

---

## Separator Control

Used to add a separator to a window.

### Super Class

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name       | Type                    | Description  |
|------------|-------------------------|--|
| Height     | <a href="#">Integer</a> | The height of the separator.   |
| Left       | <a href="#">Integer</a> | The distance in pixels from the left of the window to the left of the separator.   |
| LockBottom | <a href="#">Boolean</a> | Determines whether the bottom edge of the control should stay at a set distance from the bottom edge of the owning window.   |
| LockLeft   | <a href="#">Boolean</a> | Determines whether the left edge of the control should stay at a set distance from the left edge of the owning window. LockLeft has no effect unless LockRight is <a href="#">True</a> . |
| LockRight  | <a href="#">Boolean</a> | Determines whether the right edge of the control should stay at a set distance from the right edge of the owning window.   |
| LockTop    | <a href="#">Boolean</a> | Determines whether the top edge of the control should stay at a set distance from the top edge of the owning window. LockTop has no effect unless LockBottom is <a href="#">True</a> .   |
| Top        | <a href="#">Integer</a> | The distance in pixels from the top of the window to the top of the separator.   |
| Width      | <a href="#">Integer</a> | The width of the separator.  |

## Notes

If the width is greater than the height, the separator is horizontal; otherwise it is vertical. In either case, the smaller value controls the thickness of the **Separator**, not the thickness of the visible line in the center of the control. Mouse movement events are detected over the entire area of the **Separator**, not just the visible line.

The following illustration shows a very thick separator.



## See Also

[RectControl](#) class.

# Serial Control

Serial controls are used to perform serial communications.

**Super Class** [Object](#)

The Serial control can be instantiated via code since it is not a subclass of [Control](#). This allows you to easily write code that does communications without adding the control to a window.

### Properties

| Name              | Type                    | Description   |
|-------------------|-------------------------|---|
| Baud              | <a href="#">Integer</a> | The rate at which data will be sent and received (see the Baud Rate chart in the Notes section, below). The Baud class constants can be used to get or set the Baud rate. To do this, check the value of Baud against one or more class constants. See the table in the section “Baud Rates,” following the Methods table. On Mac OS X 10.4 and above, you can also specify arbitrary baud rates. |
| Bits              | <a href="#">Integer</a> | 0=5 data bits, 1=6 data bits, 2=7 data bits, 3=8 data bits. The following class constants are available:<br>0 - Bits5<br>1 - Bits6<br>2- Bits7<br>3- Bits8  |
| BytesAvailable    | <a href="#">Integer</a> | Contains the number of bytes left to receive.   |
| BytesLeftToSend   | <a href="#">Integer</a> | Contains the number of bytes left to send.  |
| ClearToSend       | <a href="#">Boolean</a> | Use to read the state of the ClearToSend line.  |
| CTS               | <a href="#">Boolean</a> | Enables CTS flow control.   |
| DataCarrierDetect | <a href="#">Boolean</a> | Enables you to read the state of the DataCarrierDetect line.  |
| DataSetReady      | <a href="#">Boolean</a> | Enables you to read the state of the DataSetReady line.   |
| DataTerminalReady | <a href="#">Boolean</a> | Sets the state of the data terminal line.   |
| DTR               | <a href="#">Boolean</a> | Enables DTR flow control.   |
| Index             | <a href="#">Integer</a> | The number of the control when it's part of a control array.  |
| LastErrorCode     | <a href="#">Integer</a> | Contains the last known error in the Serial control. Some of these values are currently OS errors. REALbasic error class constants are as follows:<br>100 - AccessDenied<br>101 - PortNotFound<br>102 - InvalidOptions<br>Available on Windows and Mac OS X.  |
| Left              | <a href="#">Integer</a> | The left side of the control in local coordinates (relative to the window).   |

| Name                 | Type                       | Description  |
|----------------------|----------------------------|--|
| MacInDriverRefNumber | <a href="#">Integer</a>    | Macintosh In Driver Reference Number. Of use to API programmers.   |
| MacOutDriverRefNum   | <a href="#">Integer</a>    | Macintosh Out Driver Reference Number. Of use to API programmers.  |
| Name                 | <a href="#">String</a>     | The name of the object.  |
| Parity               | <a href="#">Integer</a>    | The parity that is being used. You can use the following class constants instead of the numbers to check or set the parity. They are:<br>0- ParityNone<br>1- ParityOdd<br>2 - ParityEven   |
| RequestToSend        | <a href="#">Boolean</a>    | Sets the state of the RequestToSend line.  |
| RingIndicator        | <a href="#">Boolean</a>    | Enables you to read the state of the RingIndicator line.   |
| SerialPort           | <a href="#">SerialPort</a> | Used to identify the serial port to which the Serial control will communicate  |
| Stop                 | <a href="#">Integer</a>    | The number of stop bits being used. You can use the following class constants instead of the numbers to check or set the number of stop bits. They are:<br>0 - StopBits1:1 stop bits,<br>1 - StopBits15:1.5 stop bits,<br>2 - StopBits2: 2 stop bits |
| Top                  | <a href="#">Integer</a>    | The top of the control in local coordinates (relative to the window).  |
| Win32DriverHandle    | <a href="#">Integer</a>    | Windows Driver Handle.   |
| XON                  | <a href="#">Boolean</a>    | Enables XON flow control.  |

## Events

| Name             | Parameters                                | Description  |
|------------------|---|--|
| DataAvailable    |   | New incoming data is now available in the buffer.  |
| Error            |   | Occurs when there is an error with the Serial control. Available on Windows and Mac OS X.                          |
| LineStateChanged | ChangedLines() as <a href="#">Integer</a> | When a line state changes, the LineStateChanged event occurs and passes an array of lines whose state has changed. |

## Methods

| Name                   | Parameters   | Description  |
|------------------------|--|--|
| ClearBreak             |  | Clears the break signal on the control immediately, without the need to call the Reset method. Available on Windows and Mac OS X.  |
| Close                  |  | Closes the serial port.  |
| Flush                  |  | Clears all data from the serial port buffer.   |
| LeaveDTROnClose        |  | Tells the serial control not to negate DTR on close. The serial port must be open for this method to function.   |
| LineChangeNotification | <a href="#">ParamArray</a><br>Lines as<br><a href="#">Integer</a>                      | When a line state changes, the LineStateChanged event occurs and passes an array of lines whose state has changed. To select which lines to watch or see which lines have changed, please use the new constants added to the Serial class for this purpose:<br>ClearToSend<br>DataCarrierDetect<br>DataSetReady<br>DataTerminalReady<br>RequestToSend<br>RingIndicator |
| LookAhead              | [Encoding as<br><a href="#">TextEncoding</a> ]   | Returns a <a href="#">String</a> . Returns all the unread characters in the buffer without deleting them from the buffer. The optional <i>Encoding</i> parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify a text encoding.   |
| Open                   |  | Attempts to open the serial port. Returns <a href="#">True</a> if the port was opened and <a href="#">False</a> if it was not.   |
| Poll                   |  | Causes the control's properties to update and causes the DataAvailable event to execute if any new data is available.  |
| Read                   | Bytes as<br><a href="#">Integer</a><br>[,Encoding as<br><a href="#">TextEncoding</a> ] | Reads Bytes of data from the buffer and returns them as a <a href="#">string</a> . The optional <i>Encoding</i> parameter enables you to specify the text encoding of the data to be read. Use the <a href="#">Encodings</a> object to specify a text encoding.  |

| Name     | Parameters                                  | Description   |
|----------|---|---|
| ReadAll  | [Encoding as <a href="#">TextEncoding</a> ] | Returns all incoming data available in the buffer. The optional <i>Encoding</i> parameter enables you to specify the text encoding of the data to be read. Use the <a href="#">Encodings</a> object to specify a text encoding. |
| Reset    |   | Resets the port's baud and byte format. The serial port must be open for this method to function.   |
| SetBreak |   | Sets the break signal on the control immediately, without the need to call the Reset method. Available on Windows and Mac OS X.   |
| Write    | Data as <a href="#">String</a>              | Asynchronously writes Data to the serial port.  |
| XmitWait |   | Waits until all data sent to the serial port with the Write method has been sent.   |

**Baud Rates** To set the Baud Rate, assign the desired class constant to the Baud property. To get the Baud Rate, compare the value of the Baud property to the constants in this table.

| Baud Rate | Value | Constant   |
|-----------|-------|------------|
| 300       | 0     | Baud300    |
| 600       | 1     | Baud600    |
| 1200      | 2     | Baud1200   |
| 1800      | 3     | Baud1800   |
| 2400      | 4     | Baud2400   |
| 3600      | 5     | Baud3600   |
| 4800      | 6     | Baud4800   |
| 7200      | 7     | Baud7200   |
| 9600      | 8     | Baud9600   |
| 14400     | 9     | Baud14400  |
| 19200     | 10    | Baud19200  |
| 28800     | 11    | Baud28800  |
| 38400     | 12    | Baud38400  |
| 57600     | 13    | Baud57600  |
| 115200    | 14    | Baud115200 |
| 230400    | 15    | Baud230400 |

Setting nonstandard baud rates is supported only on Windows and Mac OS X 10.4 and above. On OS X 10.4 and higher, the system now supports arbitrary baud rates by

passing the request along to the driver. If the driver supports the passed baud rate, then it is set (or approximated).

On Linux, some non-standard baud rates are possible to achieve by using the Setserial system call and setting your baud rate to a special value.

### Notes

The Serial control can be used to communicate via multiple serial ports at once. You can use the properties of the [System](#) object to determine the number of serial ports on the computer and get an array of those ports. You should create an interface to allow the end user to choose the desired port, since serial ports are different on different machines and platforms.

When data is received by a serial port connection, the DataAvailable event handler will automatically execute. In this event handler, you would then use the Read or ReadAll functions to access the data in the serial port buffer. These functions remove the data from the serial port buffer as they return the data. If you need to read the data from the serial port buffer without removing it from the buffer, use the LookAhead property. This buffer will use as much memory as it needs from the memory available so there is no need for it to be resized.

Because the Write method is handled asynchronously, you may need to use the XmitWait method to force REALbasic to wait until REALbasic has finished sending the data out the serial port.

On Mac OS X and Linux, REALbasic gets exclusive rights to the serial port when opening it. This means that another application cannot also open the serial port after REALbasic has opened it, unless the user is running as root.

The **Serial** control implements the [Readable](#) and [Writable](#) class interfaces. If you implement either or both of these interfaces, you must provide methods that those interfaces specify.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

### Example

The following code opens the serial port. It assumes that the Serial1 control has been added to a window.

```
If Serial1.Open then
    MsgBox "The serial port is open."
Else
    MsgBox "The serial port could not be opened."
End if
```

When the serial device sends data back to the Serial control that is open, the Serial control’s DataAvailable event fires. The data is transferred into a memory buffer. In the DataAvailable event handler, you use the Read or ReadAll methods to get some or all of the data from the buffer. Choose the Read method when you want to get a specific

number of bytes (characters) from the buffer. Choose the ReadAll method to get all of the data in the buffer. In both cases, the data returned from the buffer is removed from the buffer to make room for more incoming data. If you need to examine the data in the buffer without removing it from the buffer, you can call the LookAhead method.

This example appends any incoming data to an [EditField](#):

```
Sub DataAvailable()
    EditField1.Text=EditField1.Text+Me.ReadAll\(\)
```

Both the Read and ReadAll methods of the Serial class take an optional parameter that enables you to specify the encoding. Use the Encodings object to get the desired encoding and pass it as a parameter. For example, the code above has been modified to specify that the incoming text uses the ANSI encoding, a standard on Windows:

```
Sub DataAvailable()
    EditField1.Text=EditField1.Text+Me.ReadAll\(Encodings.WindowsANSI\)
```

You can send data to the serial device at any time as long as you have opened the serial port with the Serial control's Open method. You send data using the Serial control's Write method. The data you wish to send must be a string, as the Write method accepts only a string as a parameter.

```
If Serial1.Open then
    Serial1.Write(EditField1.Text)
Else
    MsgBox "The serial port could not be opened."
End if
```

The Write method is performed asynchronously. This means that the next line of code following the Write method can already be executing before all the data has actually been sent to the serial device. If you need your code to wait for all data to be sent to the serial device before continuing, call the Serial control's XmitWait method immediately following a call to the Write method.

The following code directs the Serial control to communicate using Serial port zero (Modem port).

```
Serial.SerialPort=System.SerialPort\(0\)
```

**See Also** [SerialPort](#) class; [System](#) object; [Readable](#), [Writable](#) class interfaces.

## SerialPort Class

Used to get the properties of any serial port.

## ServerSocket Class

---

Super Class [Object](#)

### Properties

| Name             | Type                    | Description              |
|------------------|-------------------------|--------------------------|
| Name             | <a href="#">String</a>  | Name of the serial port. |
| InputDriverName  | <a href="#">String</a>  | Name of input driver.    |
| OutputDriverName | <a href="#">String</a>  | Name of output driver.   |
| RatedSpeed       | <a href="#">Integer</a> | Rated speed (baud)       |
| MaximumSpeed     | <a href="#">Integer</a> | Maximum speed (baud)     |

**Example** See the example for the [System](#) object. You get the values of serialport properties by first creating a [System](#) object.

**See Also** [System](#) object.

---

## ServerSocket Class

Used to support multiple connections on the same port. The **ServerSocket** is available only in the Professional version of REALbasic.

Super Class [Object](#)

### Properties

| Name                    | Type                    | Description  |
|-------------------------|-------------------------|--|
| Handle                  | <a href="#">Integer</a> | This is the ServerSocket's internal descriptor and it can be used with <a href="#">Declare</a> statements. The descriptor is platform-specific. If <i>Handle</i> is less than zero, the descriptor is not available. |
| IsListening             | <a href="#">Boolean</a> | <a href="#">True</a> when the <b>ServerSocket</b> is listening for incoming connections.   |
| LocalAddress            | <a href="#">String</a>  | The local address that the <b>ServerSocket</b> is using.   |
| MaximumSocketsConnected | <a href="#">Integer</a> | The maximum number of connections the <b>ServerSocket</b> will allow to connect. When a socket disconnects from the <b>ServerSocket</b> , the server will allow one more connection.                                 |
| MinimumSocketsAvailable | <a href="#">Integer</a> | The smallest number of sockets available in the server's pool of socket connections. If the <b>ServerSocket</b> falls below this number, it will call the AddSocket event to replenish its supply of sockets.        |

| Name | Type                    | Description                        |
|------|-------------------------|------------------------------------|
| Port | <a href="#">Integer</a> | The port to bind to for listening. |

## Events

| Name      | Parameters                           | Description   |
|-----------|--------------------------------------|---|
| AddSocket |                                      | The <b>ServerSocket</b> is requesting that you add a socket to its internal pool. It is called when the <b>ServerSocket</b> first begins listening. Initially it will be called <i>MaximumSocketsConnected</i> times to fill its internal pool of sockets. This socket must stay resident (cannot be a local variable) and must be non- <a href="#">Nil</a> . Returns a <a href="#">TCPSocket</a> . Since UDP is a connectionless protocol, it does not make sense for a <b>ServerSocket</b> to deal with UDPSockets. |
| Error     | ErrorCode as <a href="#">Integer</a> | Occurs when the socket has an error. Standard operating system error codes or REALbasic errors are returned.  |

## Methods

| Name              | Parameters | Description  |
|-------------------|------------|--|
| ActiveConnections |            | Returns an array of <a href="#">TCPSockets</a> . Gets the array of active sockets managed by the <b>ServerSocket</b> . These are the <a href="#">TCPSockets</a> that were added via the AddSocket event and are now connected. |
| Listen            |            | Begins the listening process for the <b>ServerSocket</b> with the properties you specified.  |
| StopListening     |            | Terminates listening on the <b>ServerSocket</b> . Does not terminate any established connections.  |

## Notes

A **ServerSocket** is a permanent socket that listens on a single port for multiple connections. When a connection attempt is made on that port, the **ServerSocket** hands the connection off to another socket, and continues listening on the same port. Without the **ServerSocket**, it is difficult to implement this functionality due to the latency between a connection coming in, being handed off, creating a new listening socket, and restarting the listening process. If you had two connections coming in at about the same time, one of the connections may be dropped because there was no listening socket available on that port.

You can change the *MinimumSocketsAvailable* and *MaximumSocketsConnected* properties after establishing the listening socket. If you change the *MaximumSocketsConnected* property, it will not kill any existing connections (it just may not allow more connections until the existing connections have been released). If you change the *MinimumSocketsAvailable* property, it may fire the *AddSocket* event to replenish its internal buffer.

## ServiceApplication Class

---

Binding a **ServerSocket** to a port below 1024 requires the proper privileges on all operating systems.

The **ServerSocket** and **SSLocket** are available only in the Professional version of REALbasic.

**See Also** [SocketCore](#), [SSLocket](#), [TCPSocket](#) classes.

---

## ServiceApplication Class

Used to create a service application, which runs in the background without a user interface.

**Super Class** [ConsoleApplication](#)

### Events

| Name     | Parameters                              | Description  |
|----------|---|--|
| Continue |   | Occurs when the user selects Continue from the Service Control Manager on Windows, assuming that a Pause event has already occurred. The Continue event will not fire unless a Pause event has fired previously. The user expects the application to resume when this event occurs.  |
| Pause    |   | Occurs when the user selects Pause from the Service Control Manager on Windows. When the user selects Pause, he expects the service to stop performing whatever action it was doing. It should continue to halt execution until the Continue event is fired.   |
| Stop     | ShuttingDown as <a href="#">Boolean</a> | Occurs in two situations: When the user selects Stop from the Service Control Manager on Windows. In this case, <i>ShuttingDown</i> is <a href="#">False</a> . This event also occurs when the computer is shutting down. In this case <i>ShuttingDown</i> is <a href="#">True</a> . This event occurs when you call the <a href="#">Quit</a> method on any platform or when the service terminates. The <i>ShuttingDown</i> parameter is supported only on Windows. |

### Notes

A Service application is a special type of [ConsoleApplication](#) that is designed to run without any interface and even if no users are logged in. A Service application should not require any user interaction since it's possible that no user is logged in while your application is running. Typical examples of service applications are FTP servers, HTTP servers, and other types of servers that have no user interface.

Console applications rely on the [Print](#) and [Input](#) commands for communicating with the user. Services behave much the same (depending on how the service is installed, and on what operating system). For example, if your service is run as a daemon on

Mac OS X or Linux using the inet.d scripts, then [Print](#) and [Input](#) will actually correspond to a socket that the system has already attached for you. This means that the Write method of the [StandardOutputStream](#) class will actually behave just like a socket and the Read method of the [StandardInputStream](#) class will as well.

It is worth noting that this behavior is not guaranteed for all operating systems. If you plan on deploying your service on multiple OS's, it is best to use a [TCPSocket](#) directly instead of relying on standard input and output being hooked up to a socket for you.

To create a service application, you must first create a regular console application. Choose File ▶ New Project and choose the Console Application item. A console application, has a single project item called App whose super class is [ConsoleApplication](#). Change App's super to [ServiceApplication](#).

The **ServiceApplication** class is a subclass of [ConsoleApplication](#) class, so all the [ConsoleApplication](#) events and methods are available. In addition to the [ConsoleApplication](#)'s events, there are three new events for a **ServiceApplication**. Some of these events will be fired only on operating systems that support them—and the only OS that supports them currently is Windows.

|                 |  |
|-----------------|--|
| <b>See Also</b> | <a href="#">ConsoleApplication</a> , <a href="#">StandardInputStream</a> , <a href="#">StandardOutputStream</a> classes; <a href="#">Input</a> , <a href="#">Print</a> , <a href="#">StdErr</a> , <a href="#">StdIn</a> , <a href="#">StdOut</a> methods; <a href="#">TargetHasGUI</a> constant. |
|-----------------|--|

## Shell Class

Used to execute Unix or DOS shell commands under Windows, Mac OS X, or Linux.

**Super Class** [Object](#)

### Properties

| Name      | Type                    | Description   |
|-----------|-------------------------|---|
| ErrorCode | <a href="#">Integer</a> | Returns 0 if the Execute method was executed successfully. Otherwise, it returns a system-supplied error code. On Windows, it returns -1 if execution fails.  |
| IsRunning | <a href="#">Boolean</a> | <a href="#">True</a> if an asynchronous or interactive shell process is running.  |
| Mode      | <a href="#">Integer</a> | Controls the operation of the shell. Interactive mode is supported only on Mac OS X.<br>0-Synchronous. The shell executes its command and returns the result.<br>1-Asynchronous. The shell executes its command and returns data via the DataAvailable event.<br>2-Interactive. Data can be sent to a running shell session with the Write method and data is returned via the DataAvailable event. |
| PID       | <a href="#">Integer</a> | ID of the interactive shell process.  |

## Shell Class

---

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Result  | <a href="#">String</a>  | Contains the contents of output buffer if you are in Synchronous mode (mode 0) without clearing the buffer.   |
| TimeOut | <a href="#">Integer</a> | (Windows only) Time (milliseconds) that specifies how long a process can run before it is automatically terminated. The default is 2000 or 2 seconds. |

## Events

| Name          | Parameters | Description   |
|---------------|------------|---|
| Completed     |            | Fired only in modes 1 and 2. Triggered whenever the executed command is finished.   |
| DataAvailable |            | Fired only in modes 1 and 2. Triggered whenever the Shell object returns data— just like the <a href="#">TCPsocket</a> class. |

## Methods

| Name      | Parameters   | Description  |
|-----------|--|--|
| Close     |  | Shuts down the running command.  |
| Execute   | Command as <a href="#">String</a><br>[Parameters as <a href="#">String</a> ] | Executes a one-line Unix or DOS shell command. Used in Synchronous mode. If you are in Synchronous mode (mode 0), the Result property will contain the results. The <i>Command</i> parameter is the path/name of the executable to run and the second parameter contains the arguments to pass to the executable. You can specify the executable without passing a second parameter. On Windows, if the path/name of the executable contains spaces and you want to pass arguments to it, it is safer to separate the two. |
| Poll      |  | Looks for data returned from the running shell and may trigger a DataAvailable event.  |
| ReadAll   |  | Returns the contents of the output buffer and clears the buffer.   |
| Write     | Text as <a href="#">String</a>   | Sends a string to the shell's input buffer.  |
| WriteLine | Text as <a href="#">String</a>   | Sends a string ending in a linefeed to the shell's input buffer.   |

## Notes

Use the **Shell** class to execute DOS or Unix commands and get the results. It has a single method, Execute, which executes a one-line command in Synchronous mode. This causes two properties of the **Shell** object to change: ErrorCode, which is a system-supplied error code or 0 for no error; and Result, which is a [string](#) containing the output of the command. The TimeOut property specifies how long (in milliseconds) a process can run before it is automatically terminated. A value of -1 means the process can run indefinitely. This property currently applies only to Windows.

**Examples**

Using the synchronous mode, the following code lists the current directory's files (Mac OS X or Linux).

```
Dim s As Shell
s=New Shell
#if TargetWin32
  s.execute "dir"
#elseif (TargetMacOS or TargetLinux) and Not(TargetMacOSClassic)
  s.execute "ls -la"
#else
  MsgBox "This doesn't run on Mac OS Classic!"
#endif

If s.errorCode = 0 then
  EditField1.text = s.result
else
  Msgbox "Error code: " + Str(s.errorCode)
end if
```

The following example gets the names and values of the environment variables on the user's computer and displays the information in an [EditField](#). You can use the [EnvironmentVariable](#) method of the [System](#) object to get or set individual environment variables.

```
Dim s as New Shell
Dim cmd as String
#if TargetMacOS or TargetLinux and Not(TargetMacOSClassic)
  cmd="env"
#elseif TargetWin32
  cmd="set"
#endif

  s.execute cmd
if s.errorCode=0 then
  EditField1.text=s.Result
else
  EditField1.text="Error "+Str(s.ErrorCode)
end if
```

The following terminal application allows you to submit Unix commands using the interactive mode. The interface consists of two [EditFields](#), InputField, in which the user can enter a command, and OutputField that displays the results.

The Open event for the window initializes the shell object (declared as a property of the window).

```
mShell = New Shell
mShell.Mode = 2
```

## ShellNotRunningException Error

---

The user can type a unix command into the EditField. When he presses Return, the following code in the [EditField's](#) KeyDown event runs. The Write method sends the command to the Shell's input buffer.'

```
If Key = Chr(13) Then
  If Not mShell.IsRunning Then
    mShell.Execute "sh"
  End If
  mShell.Write InputField.Text
  mShell.Write Chr(13)
  InputField.Text = ""
  Return True
Else
  Return False
End If
```

A [Timer](#) calls the ReadAll method in its Action event:

```
If mShell <> Nil Then
  OutputField SelText = mShell.ReadAll
End If
```

**See Also** [TargetCarbon](#), [TargetWin32](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetMacOS](#), [TargetLinux](#) constants.

---

## ShellNotRunningException Error

You tried to pass a string to a [Shell](#) that is not running.

**Super Class** [RuntimeException](#)

**Notes** The **ShellNotRunningException** error will occur if you try to pass a string for the [Shell](#) to execute but it is not running. This can happen if you are using the [Shell](#) interactively and it has been closed prior to passing the string.

**See Also** [Shell](#) class.

---

## Short Data Type

A 16-bit signed integer. Its value can range from -32768 to 32767.

The Short can be used only in [Declare](#) statements and cannot be used when declaring REALbasic variables, properties, or methods.

**See Also**

[Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [UByte](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

## ShowURL Method

Uses the appropriate web browser application (based on the user's Internet settings) to go to the URL (web address, ftp address, etc.) specified.

**Syntax**

**ShowURL URL**

| Part | Type                   | Description                              |
|------|------------------------|--|
| URL  | <a href="#">String</a> | The URL, ftp, or email address to go to. |

**Notes**

URL stands for Uniform Resource Locator. URL's are addresses for Internet locations such as ftp and web sites. The **ShowURL** method uses the user's Internet preferences to determine which application to launch based on the type of URL specified. For example, passing a URL that begins with "http" will launch the user's web browser. Some browsers have limits on the length of a URL.

**Examples**

This example will launch the user's chosen web browser and go to the REAL Software web page.

```
ShowURL "http://www.realsoftware.com"
```

This example will launch the user's chosen ftp client application and go to an ftp site.

```
ShowURL "ftp://ftp.realsoftware.com/current_release"
```

This example will launch the user's chosen e-mail client application and create a new e-mail message, with the recipient's e-mail address in the "To" area:

```
ShowURL "mailto:batman@batcave.org"
```

## Shuffle Method

Shuffles (rearranges randomly) the elements of an array.

**Syntax**

**array.Shuffle**

| Part  | Type                | Description                                |
|-------|---------------------|--|
| array | Any valid data type | The array whose elements will be shuffled. |

## ShutDownItemsFolder Function

---

### Notes

**Shuffle** works on arrays of any data type and any number of dimensions. It is based on a random number. An element of the original array has a roughly equal chance of appearing in any cell of the array after the shuffle, regardless of the size and number of dimensions of the array.

### Example

The following example shuffles a one-dimensional array and shows the result in a **ListBox**.

```
Dim aTest() as Integer
Dim i as Integer
aTest=Array(0,1,2,3,4,5,6,7,8,9)
aTest.Shuffle
For i = 0 to Ubound(aTest)
    ListBox1.Addrow Str(aTest(i))
Next
```

The following example shuffles a two-dimensional array.

```
Dim myArray(5,5) as String
Dim i, j as Integer
For i=0 to 5
    For j=0 to 5
        myArray(i,j)=Str(i)+Str(j)
    Next
Next
myArray.Shuffle
```

### See Also

[Dim statement](#); [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Redim](#), [Remove](#), [Pop](#), [Sort](#) methods; [ParamArray](#) keyword.

---

## ShutDownItemsFolder Function

Used to access the ShutDownItems folder in the Mac OS “classic” System folder. It returns [Nil](#) on other platforms.

### Syntax

**result=ShutDownItemsFolder**

| Part   | Type                       | Description   |
|--------|----------------------------|---|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the ShutDown Items folder. |

### Notes

Use the **ShutDownItemsFolder** function to access the ShutDownItems folder.

### Windows

On Windows, **ShutDownItemsFolder** returns [Nil](#).

**Macintosh** On Mac OS X, **ShutdownItemsFolder** returns [Nil](#); on Mac OS “classic”, it returns a reference to the ShutdownItems folder in the System folder.

**Linux** On Linux, **ShutdownItemFolder** returns [Nil](#).

The [SpecialFolder](#) module enables you to access many other special folders that are maintained by the OS.

**Examples** This example places in a [ListBox](#) all the names of the items in the Shut Down Items folder. It first checks that the user is running Mac OS “classic.”

```
Dim i as Integer
Dim f as FolderItem
#if TargetMacOSClassic //classic Mac OS
f=ShutdownItemsFolder
for i=1 to f.count
    ListBox1.addrow f.item(i).name
next
#endif
```

**See Also** [ApplicationSupportFolder](#), [DesktopFolder](#), [FontsFolder](#), [PreferencesFolder](#), [StartupItemsFolder](#), [SystemFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

## Sign Function

Returns the sign of the number passed to it.

**Syntax** **result=Sign(value)**

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Integer</a> | The sign of <i>value</i> . Returns -1 if <i>value</i> is negative, 0 if <i>value</i> is zero, and 1 if <i>value</i> is positive. If you pass a <a href="#">string</a> or some other incorrect data type, <i>result</i> is 0. |
| value  | <a href="#">Double</a>  | The number being passed to the function.   |

## Sin Function

---

**Example** The following example determines the sign of the number passed to it.

```
Dim d as Double
Dim result as Integer
If EditField1.text <> "" then
    result=Sign(Val(EditField1.text))
    MsgBox Str(result)
Else
    MsgBox "Please enter a number!"
End if
```

**See Also** [Abs](#), [Val](#) functions.

## Sin Function

---

Returns the sine of the value specified.

**Syntax** **result=Sin (value)**

| Part   | Type                   | Description                     |
|--------|------------------------|---------------------------------|
| value  | <a href="#">Double</a> | The value you want the sine of. |
| result | <a href="#">Double</a> | The sine of <i>value</i> .      |

**Notes** The **Sin** function returns the sine of the angle (in radians) passed to it. If value is in degrees, multiply it by PI/180 to convert it to radians.

**Examples** This example uses the **Sin** function to return the sine of a number.

```
Dim d As Double
Const PI=3.14159265358979323846264338327950
d=Sin(0.5) //returns 0.4794255
d=Sin(30*PI/180) //returns .5
```

**See Also** [Asin](#) function.

## Single Data Type

---

A **Single** is an intrinsic data type in REALbasic. A **Single** is a number that can contain a decimal value, i.e., a real number. It can take on a value between -1.175494 e-38 and 3.402823 e+38. In other languages, REALbasic's **Single** may be referred to as a single precision real number. If you need more precision, you should use a [Double](#) instead. Because **Singles** are numbers, you can perform mathematical calculations on them.

**Singles** use 4 bytes of memory. Other languages refer to a REALbasic **Single** as a Float. The default value of a **Single** is 0.0.

The [VarType](#) function returns a value of 5 when passed a **Single**.

## Example

The **Single** and [Double](#) data types allow you to store and manage floating point numbers.

```
Dim s as Single
s=3.1416
```

## See Also

[Boolean](#), [Color](#), [Double](#), [Integer](#), [String](#), data types; [+, +](#), [\\*](#), [/](#), [≤](#), [≤≤](#), [≡](#), [≥≡](#), [≥](#), [≥≥](#), [\](#), [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [VarType](#) functions; [Dim](#) statement.

# Slider Control

Implements the slider control.

## Super Class

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name       | Type                    | Description  |
|------------|-------------------------|--|
| LineStep   | <a href="#">Integer</a> | On Windows and Linux, LineStep is the amount a slider moves when the slider has the focus and the user uses the arrow keys. On Macintosh it has no effect.                                     |
| LiveScroll | <a href="#">Boolean</a> | If <a href="#">True</a> , a ValueChanged event occurs as the user drags the thumbnail in the slider. Otherwise, a single ValueChanged event occurs when the user stops dragging the thumbnail. |
| Maximum    | <a href="#">Integer</a> | The maximum value of the slider. The default is 100.   |
| Minimum    | <a href="#">Integer</a> | The minimum value of the slider.   |
| PageStep   | <a href="#">Integer</a> | The amount the value changes when the user clicks in the Slider track.   |
| Value      | <a href="#">Integer</a> | The current value of the slider.   |

## Events

| Name      | Parameters | Description   |
|-----------|------------|---|
| GotFocus  |            | (Windows and Linux) The slider has received the focus and has a focus ring. |
| LostFocus |            | (Windows and Linux) The slider has lost the focus.                          |

| Name         | Parameters   | Description   |
|--------------|--|---|
| MouseDown    | x as<br><a href="#">Integer</a><br>y as<br><a href="#">Integer</a> | The mouse button was pressed inside the Slider at the location passed in to x,y. Returns a <a href="#">Boolean</a> . <a href="#">Return True</a> if you are going to handle the MouseDown. The coordinates x and y are local to the control. Point 0,0 is the top-left corner of the entire control (to the left of the label and above the box that is drawn). |
| MouseDrag    | x as<br><a href="#">Integer</a><br>y as<br><a href="#">Integer</a> | The mouse button was pressed inside the Slider and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.  |
| MouseUp      | x as<br><a href="#">Integer</a><br>y as<br><a href="#">Integer</a> | The mouse button was released inside the Slider at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.  |
| ValueChanged |  | The slider's value has changed.   |

**Examples**

Changing the maximum value of a **Slider** at runtime:

```
Slider1.maximum=200
```

Setting the text of staticText1 to the value of the slider when the user scrolls:

```
Sub ValueChanged()  
StaticText1.text=Str(Slider1.value)
```

**See Also**

[RectControl](#) class; [Scrollbar](#) control.

## SMTPSecureSocket Class

Used to send secure email via the SMTP protocol using SSL or TLS encryption.

**Super Class** [SSLocket](#)**Properties**

| Name     | Type                         | Description   |
|----------|------------------------------|---|
| Messages | <a href="#">EmailMessage</a> | The queue of messages that are waiting to be sent. This is an array of which the 0 <sup>th</sup> element is the next message to be sent. When a message is sent, it is removed from the queue and passed to the MessageSent event. To append a message to the queue, use the <a href="#">Append</a> method. To remove a message, use the <a href="#">Remove</a> method. |

## Methods

| Name                 | Parameters | Description  |
|----------------------|------------|--|
| DeleteAllMessages    |            | Deletes all messages from the send queue.                        |
| DisconnectFromServer |            | Closes the connection with the mail server.                      |
| SendMail             |            | Connects to the mail server and sends the messages in the queue. |

## Events

| Name                  | Parameters  | Description   |
|-----------------------|---|---|
| ConnectionEstablished | Greeting as <a href="#">String</a>  | Executes when the connection has been established with the mail server and passes the greeting string that the server returned.   |
| Error                 |   | An error has occurred.  |
| MailSent              |   | Executes when all of the messages in the queue have been sent.  |
| MessageSent           | Email as <a href="#">EmailMessage</a>   | Executes when a message has been sent; <i>Email</i> contains the message that was sent.   |
| ServerError           | ErrorID as <a href="#">Integer</a> ,<br>ErrorMessage as <a href="#">String</a> ,<br>Email as <a href="#">EmailMessage</a> | Executes when an error occurred while sending a message. <i>ErrorID</i> and <i>ErrorMessage</i> are sent by the mail server and <i>Email</i> is the message that was being sent when the error occurred. <i>Email</i> is removed from the queue at the time the event executes. |

## Notes

The **SMTPSecureSocket** class is the same as the [SMTPSocket](#) class except that it is derived from [SSLocket](#) instead of [TCPsocket](#). As a result, you can use the **Secure** property of the [SSLocket](#) class to provide secure communications.

If you use a constructor in a subclass of a **SMTPSecureSocket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

## See Also

[EmailMessage](#), [HTTPSSecureSocket](#), [HTTPSSocket](#), [POP3SecureSocket](#), [TCPsocket](#), [POP3Socket](#), [SMTPSocket](#), [SocketCore](#) classes.

# SMTPSocket Class

Used to send email via the SMTP protocol.

## SMTPSocket Class

---

Super Class [TCPSocket](#)

### Properties

| Name     | Type                         | Description   |
|----------|------------------------------|---|
| Messages | <a href="#">EmailMessage</a> | The queue of messages that are waiting to be sent. This is an array of which the 0 <sup>th</sup> element is the next message to be sent. When a message is sent, it is removed from the queue and passed to the MessageSent event. To append a message to the queue, use the <a href="#">Append</a> method. To remove a message, use the <a href="#">Remove</a> method. |

### Methods

| Name                 | Parameters | Description  |
|----------------------|------------|--|
| DeleteAllMessages    |            | Deletes all messages from the send queue.                        |
| DisconnectFromServer |            | Closes the connection with the mail server.                      |
| SendMail             |            | Connects to the mail server and sends the messages in the queue. |

### Events

| Name                  | Parameters                             | Description  |
|-----------------------|--|--|
| ConnectionEstablished | Greeting as <a href="#">String</a>     | Executes when the connection has been established with the mail server and passes the greeting string that the server returned.  |
| Error                 |  | An error occurred.   |
| MailSent              |  | Executes when all of the messages in the queue have been sent.   |
| MessageSent           | Email as <a href="#">EmailMessage</a>  | Executes when a message has been sent; <i>Email</i> contains the message that was sent.  |
| SendComplete          | UserAborted as <a href="#">Boolean</a> | All of the data in the socket buffer has been written (sent). The <i>UserAborted</i> parameter enables you to determine whether the user cancelled the transfer by returning <a href="#">True</a> from the SendProgress event. If the user aborted, this value is <a href="#">True</a> , otherwise it is <a href="#">False</a> . |

| Name         | Parameters  | Description  |
|--------------|---|--|
| SendProgress | BytesSent as <a href="#">Integer</a><br>BytesLeft as <a href="#">Integer</a>  | Occurs when your network provider queues your data in 'chunks' and is about to send the next chunk. The parameters indicate the amount of progress that has been made during the send. Returns a <a href="#">Boolean</a> . Returning <a href="#">True</a> from this event causes the send to be aborted. |
| ServerError  | ErrorID as <a href="#">Integer</a> ,<br>ErrorMessage as <a href="#">String</a> ,<br>Email as <a href="#">EmailMessage</a> | Executes when an error occurred while sending a message. <i>ErrorID</i> and <i>ErrorMessage</i> are sent by the mail server and <i>Email</i> is the message that was being sent when the error occurred. <i>Email</i> is removed from the queue at the time the event executes.                          |

## Notes

The [POP3Socket](#) and **SMTPSocket** classes are used together to form the basis of an email client. POP3 is the standard internet protocol for receiving messages and SMTP (Simple Mail Transfer Protocol) is the standard internet protocol for sending emails.

SMTPSocket supports the 'login' authentication type as well as 'plain.' This increases the authentication compatibility with some servers.

If you use a constructor in a subclass of **SMTPSocket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

## Example

The following code in the Action event of a [PushButton](#) sends an email message. It works with a form with fields for each part of the email. [EditFields](#) are used for the Username and Password of the email account, the From, To, and CC addresses, the subject of the email, and the body in plain text and HTML form. Since the user can enter multiple To and CC addresses, the example parses these strings and adds the individual To and CC addresses to the [EmailMessage](#) using the AddRecipient and AddCCRecipient methods.

An attachment can be specified in the form of a full path to the file to be attached. If an attachment exists, it is added to the EmailMessage's Attachments property.

## SMTPSocket Class

---

An **SMTPSocket** named SMTPSocket1, has been added to the window.

```
Dim mail as EmailMessage
Dim file as EmailAttachment
Dim i as Integer
Dim s as String

SMTPSocket1.address = "mail.mySMTPServer.com"
SMTPSocket1.port = 25

SMTPSocket1.username = usernameFld.text //get username from editfield
SMTPSocket1.password = passwordFld.text //get password from Editfield

mail = New EmailMessage
mail.fromAddress=fromAddressFld.text //get From address from EditField
mail.subject=subjectFld.text //get Subject of mail from EditField
mail.bodyPlainText = bodyFld.text //get body in plain text from EditField
mail.bodyHTML = htmlFld.text //get body in HTML from EditField
mail.headers.appendHeader "X-Mailer","REALbasic SMTP Demo"

//multiple To addresses separated by commas
//each to address added to Recipient array
s = ReplaceAll(toAddressFld.text, ", ", Chr(13))
For i = 1 to CountFields(s,Chr(13))
    mail.addRecipient Trim(NthField(s,Chr(13),i))
next

//multiple CC addresses separated by commas
//each to address added to CCRecipient array

s = ReplaceAll(ccAddressFld.text, ", ",Chr(13))
For i = 1 to CountFields(s,Chr(13))
    mail.addCCRecipient Trim(NthField(s,Chr(13),i))
next

//add attachments, if any
If fileFld.text <> "" then //is there a path to an attachment file?
    file = New EmailAttachment
    file.loadFromFile GetFolderItem(fileFld.text)
    mail.attachments.append file //add file to attachments array
end
//send the mail
SMTPSocket1.messages.append mail //add email to list of messages
SMTPSocket1.SendMail //send message
```

### See Also

[EmailAttachment](#), [EmailHeader](#), [EmailMessage](#), [HTTPSecureSocket](#), [HTTPSocket](#), [POP3SecureSocket](#), [POP3Socket](#), [SMPSecureSocket](#), [SocketCore](#), [TCPSocket](#) classes.

# SOAPException Error

**SOAPExceptions** can be raised when using a WSDL to define your SOAP function. If the method name does not exist or the parameters passed do not match the WSDL specifications, a **SOAPException** runtime error will be raised.

**Super Class** [RuntimeException](#) class.

**Example** This example raises an exception of type **SOAPException** because the method name ValidEmail does not exist in the WSDL. The method should be isValidateEmail. A **SOAPException** can also be raised by passing in an incorrect number of parameters.

```
Dim sm as SoapMethod

// create new method and load WSDL document from URL
sm = New SoapMethod("http://www.webservicex.net/ValidateEmail.asmx?WSDL")

// execute and display function result
MsgBox sm.validEmail(emailFld.text)

// catch exceptions
Exception err as SOAPException
MsgBox "SOAP Error: " + err.message
```

**See Also** [RuntimeException](#), [SOAPMethod](#), [SOAPResult](#) classes; [Exception](#), [Try](#) blocks.

# SOAPMethod Class

Used to call a SOAP remote method or function.

**Super Class** [Object](#)

## Constructor

| Name       | Parameters                       | Description   |
|------------|----------------------------------|---|
| SOAPMethod | [URL as <a href="#">String</a> ] | Creates a SOAPMethod object, optionally with the URL to the WSDL document. Otherwise, set the URL and WSDL using the URL and WSDL properties. |

## Properties

| Name            | Type                        | Description  |
|-----------------|-----------------------------|--|
| Action          | <a href="#">String</a>      | SOAP action for the method being called.   |
| MethodNamespace | <a href="#">String</a>      | Namespace for the method being called.   |
| Parameter       | <a href="#">String</a>      | SOAP parameter whose value you are requesting.   |
| Timeout         | <a href="#">Integer</a>     | Number of seconds to wait for the response from the SOAP service. Setting this to zero will wait indefinitely until the service responds or the connection is dropped. |
| URL             | <a href="#">String</a>      | URL for the SOAP service being queried.  |
| WSDL            | <a href="#">XMLDocument</a> | WSDL document to use for defining the SOAP service. A WSDL document automatically sets up the necessary properties that are required to call the SOAP service.         |

## Methods

| Name            | Parameters   | Description  |
|-----------------|--|--|
| ClearParameters |  | Clears the values of all parameters that have been set.  |
| Invoke          | Name as <a href="#">String</a>   | Executes the SOAP method and returns the result as a <a href="#">SOAPResult</a> .  |
| LoadWSDLFromURL | URL as <a href="#">String</a>  | Loads a WSDL document from a URL.  |
| UseSocket       | Socket as <a href="#">HTTPSocket</a> or <a href="#">HTTPSecureSocket</a> | This method tells the <b>SOAPMethod</b> which socket to use when communicating with the SOAP service. If no socket is set, then one will be created and used when necessary. This method accepts either an <a href="#">HTTPSocket</a> or an <a href="#">HTTPSecureSocket</a> . Use this feature when you want to connect via a proxy or SSL connection by setting up the properties of the socket and then assigning it to the <b>SOAPMethod</b> . |

## Notes

To execute a SOAP service, a WSDL must be set in order to pass the correct parameter values. For example, if the SOAP service provides a function called GetTemp, which takes a Zip Code, you call it as follows:

```
SOAPMethod1.GetTemp("60615")
```

## Examples

The following example creates a SOAPMethod and loads its WSDL from the URL specified in the constructor. It assumes that a valid Zip Code has been entered into an

[EditField](#) called ZipCodeFld. It then calls the GetTemp function and displays the result.

```
Dim sm as SOAPMethod
```

```
sm = New SOAPMethod("http://www.xmethods.net/sd/2001/TemperatureService.wsdl")
MsgBox sm.GetTemp(zipcodeFld.text)
```

The following example gets information about a Zip Code. It uses the Result function of the [SOAPResult](#) class to look for a particular tag within the result for easy data retrieval.

```
Dim sm as SoapMethod
```

```
Dim sr as SoapResult
```

```
// create the soap method and set the parameter(s)
```

```
sm = New SoapMethod
```

```
sm.parameter("USZip") = zipcodeFld.text
```

```
// set soap method properties
```

```
sm.methodNamespace = "http://www.webserviceX.NET"
```

```
sm.action = "http://www.webserviceX.NET/GetInfoByZIP"
```

```
sm.url = "http://www.webserviceX.NET/uszip.asmx"
```

```
// execute the method
```

```
sr = sm.invoke("GetInfoByZIP")
```

```
// display the Area_Code portion of the result
```

```
MsgBox sr.result("Area_Code")
```

## SOAPResult Class

---

The following example will generate an error from the SOAP service because the method namespace is incorrect. A “B” has been appended to the end of it. The [SOAPResult](#) class will provide the error information received from the SOAP service.

```
Dim sm as SoapMethod
Dim sr as SoapResult
// create SOAPMethod and define parameters
sm = New SoapMethod
//Assume there's an EditField named emailFld in the window
sm.parameter("EmailAddress") = emailFld.text

// set method properties
sm.methodNamespace = "http://www.webserviceX.NETB" //incorrect
sm.action = "http://www.webserviceX.NET/isValidEMail"
sm.url = "http://www.webserviceX.NET/ValidateEmail.asmx"
// execute function
sr = sm.invoke("IsValidEMail")

// check for error
If sr.error = True then
    Beep
    MsgBox "ERROR: " + sr.errorMessage
else
    // display result
    MsgBox sr.result("IsValidEMailResult")
end
```

**See Also** [SOAPResult](#) class; [SOAPException](#) error.

---

## SOAPResult Class

Returns the value found for the specified XML tag.

**Super Class** [Object](#)

### Properties

| Name        | Type                        | Description  |
|-------------|-----------------------------|--|
| Body        | <a href="#">XMLNode</a>     | The body portion of the SOAP response.                     |
| Document    | <a href="#">XMLDocument</a> | Response returned from the SOAP service.                   |
| Envelope    | <a href="#">XMLNode</a>     | Envelope portion of the SOAP response.                     |
| Error       | <a href="#">Boolean</a>     | <a href="#">True</a> if the SOAP service returned a fault. |
| ErrorCode   | <a href="#">Integer</a>     | Fault code returned from the SOAP service.                 |
| ErrorString | <a href="#">String</a>      | Fault message returned from the SOAP service.              |

## Methods

| Name   | Parameters                        | Description   |
|--------|-----------------------------------|---|
| Result | Name as<br><a href="#">String</a> | Returns the value found in the tag name specified. This function will search through the Body portion of the response and look for the tag that was passed. When it finds this tag, it returns its value. |

## Example

The following example uses **SOAPResult** to get the temperature in the passed Zip Code.

```

Dim sm as SoapMethod
Dim sr as SoapResult

// create soap method and define parameters
sm = New SoapMethod
sm.parameter("zipcode") = zipcodeFld.text

// setup method properties
sm.MethodNamespace = "urn:xmethods-Temperature"
sm.url = "http://services.xmethods.net:80/soap/servlet/rpcrouter"

// call the method
sr = sm.invoke("getTemp")

// display the result
MsgBox sr.result("return")

```

## See Also

[SOAPMethod](#) class; [SOAPException](#) error.

# SocketCore Class

The base class for [TCPSocket](#), and [UDPSocket](#). The [SSLocket](#) class is derived from [TCPSocket](#). **SocketCore** is an abstract class and cannot be instantiated directly.

**Super Class** [Object](#)

## Properties

| Name   | Type                    | Description   |
|--------|-------------------------|---|
| Handle | <a href="#">Integer</a> | This is the socket's internal descriptor and it can be used with <a href="#">Declare</a> statements. The descriptor is platform-specific. If Handle is less than zero, the descriptor is not available. |

| Name             | Type                             | Description  |
|------------------|----------------------------------|--|
| IsConnected      | <a href="#">Boolean</a>          | Indicates whether the socket is currently connected.<br>For <a href="#">TCP Sockets</a> , a connection means you can send and receive data, and are connected to a remote machine. For <a href="#">UDP Sockets</a> , this means that you are bound to the port and are able to send, receive, join or leave multicast groups, or set socket options.   |
| LastErrorCode    | <a href="#">Integer</a>          | The last error code for the socket.  |
| LocalAddress     | <a href="#">String</a>           | The local IP address of the computer.  |
| NetworkInterface | <a href="#">NetworkInterface</a> | Specifies which network interface the socket should use when binding. Leaving this property set to <a href="#">Nil</a> will use the currently selected interface. You can get the network interface(s) of the user's computer by calling the <a href="#">GetNetworkInterface</a> method of the <a href="#">System</a> object.  |
| Port             | <a href="#">Integer</a>          | The port to bind on or connect to. On Mac OS X and Linux, attempting to bind to a port less than 1024 causes a <a href="#">SocketCore.Error</a> event to fire with an error number 107 unless the application is running with root permissions. This is due a security feature built into the underlying Unix-based OS.<br>Setting Port to zero will allow the underlying network layer to determine a random port for you when you call the <a href="#">Connect</a> method or, in the case of the <a href="#">TCP Socket</a> , the <a href="#">Listen</a> method. |

## Events

| Name          | Parameters                             | Description  |
|---------------|--|--|
| DataAvailable |  | Occurs when additional data has come into the internal receive buffer.   |
| Error         |  | Occurs when an error occurs with the socket.   |
| SendComplete  | UserAborted as <a href="#">Boolean</a> | Occurs when a send has completed. Use this to determine when all your data has been sent. <i>UserAborted</i> will be True if the user aborted the send by returning <a href="#">True</a> from the <a href="#">SendProgress</a> event. You can use this information to update different status variables or to alert the user the success or failure of the transfer. If the send was completed, this value is <a href="#">False</a> . <i>UserAborted</i> will always be <a href="#">False</a> for UDP sockets. |

## Methods

| Name    | Parameters | Description   |
|---------|------------|---|
| Close   |            | Closes the socket's connection, closes any connections the socket may have, and resets the socket. The only information that is retained after calling Close is the socket's port, address (in the case of <a href="#">TCP Sockets</a> ), and LastErrorCode properties, and data left in the socket's receive buffer. All other information is discarded. |
| Connect |            | Attempts to connect. For <a href="#">TCP Sockets</a> , the address and port properties must be set. For <a href="#">UDP Sockets</a> , the port property must be set. The Connect method binds a socket to a port. After calling Connect, the Port property will report the actual port you are bound to.  |
| Poll    |            | Polls the socket manually, which allows a socket to be used synchronously.  |
| Purge   |            | Removes all data from the socket's internal receive buffer. It does not affect the socket's internal send buffer.   |

## Error Codes

The LastErrorCode property contains an integer value specifying what the last error code is. These error codes provide you with key information about your socket, and it is not advisable to ignore them.

Each error code is associated with a class constant. When you need to check whether a particular error occurred, you can check the value of the LastErrorCode property against these class constants.

Use the constants in the table below to compare to the LastErrorCode property.

| Error Code | Class Constant  | Description   |
|------------|-----------------|---|
|            | NoError         | No error occurred.  |
| 100        | OpenDriverError | There was an error opening and initializing the drivers. It may mean that either Open Transport (on the Macintosh), or WinSock (on Windows) is not installed, or the version is too early.  |
| 101        |                 | This error code is no longer used.  |
| 102        | LostConnection  | This code means that you lost your connection. You will get this error if the remote side disconnects (whether it's forcibly—by pulling their ethernet cable out of the computer), or gracefully (by calling SocketCore's Close method). This may or not be a true error situation. If the remote side closed the connection, then it is not truly an error; it's just a status indication. But if they pulled the ethernet cable out of the computer, then it really is an error; but the results are the same. The connection was lost. You will also get this error if you call the Disconnect method of TCP Socket. |

| Error Code | Class Constant      | Description   |
|------------|---------------------|---|
| 103        | NameResolutionError | REALbasic was unable to resolve the address that was specified. A prime example of this would be a mistyped IP address, or a domain name of an unknown or unreachable host.   |
| 104        |                     | This error code is no longer used.  |
| 105        | AddressInUseError   | The address is currently in use. This error will occur if you attempt to bind to a port that you have already bound to. An example of this would be setting up two listening sockets to try to listen on the same port.   |
| 106        | InvalidStateError   | This is an invalid state error, which means that the socket is not in the proper state to be doing a certain operation. An example of this is calling the Write method before the socket is actually connected.   |
| 107        | InvalidPortError    | This error means that the port you specified is invalid. This could mean that you entered a port number less than 0, or greater than 65,535. It could also mean that you do not have enough privileges to bind to that port. This happens under Mac OS X and Linux if you are not running as root and try to bind to a port below 1024. You can only bind to ports less than 1024 if you have root privileges. A normal "Admin" user does not have root privileges. |
| 108        | OutOfMemoryError    | This error indicates that your application has run out of memory.   |

These are not the only errors that are returned by LastErrorCode. If REALbasic cannot adequately map the provider's error code to one of the above codes, it will pass you the provider's error code. For Windows, these error codes are usually positive numbers in the range [10004, 11004]. For the Macintosh, these error codes are negative numbers in the range [-3211, -3285]. For a description of the Macintosh error codes, please see MacErrors.h. For Windows error codes, see WinSock.h.

## Notes

There are some instances where you would like the system to pick a port for you. These needs can range from needing a port for passive FTP file transfers, or perhaps you wrote a class to auto-discover other applications on the network and you would like to negotiate a port to connect over using TCP/IP. If you specify a Port of 0 and then call [TCPSocket.Listen](#) or [UDPSocket.Connect](#), REALbasic will pick a random open port for you. Most often, these ports will be in the range of 49512 to 65535 (inclusive).

If you use this method to obtain an open port, then you probably need to use the following procedure to determine which port was chosen. After calling the Connect method or the Listen method of the [TCPSocket](#) class, you can check the Port property

to see which port was assigned. This means that the port number will change from 0 to the port number that you are bound to.

There's another benefit checking the Port property. There is a hacking technique called port hijacking where the hacker steals a port out from under you. If this happens, checking the Port property will tell you if someone has hijacked the port. It can be a good idea (though paranoid) to periodically check to make sure the Port property returns a port that you expect to see. For instance, if you were listening on port 80 for HTTP connections, but the Port property says you're listening on port 2113, then something may be wrong.

The SendComplete event's parameter lets you know whether the transfer has completed or has been cancelled by returning [True](#) from the SendProgress event of the [TCPSocket](#) class. You can use this information to update different status variables, or alert the user of transfer success or failure. If the user aborted, this parameter is [True](#), and if the send was completed, this value is [False](#). The SendComplete event's parameter is always [False](#) for UDPSockets since there is not a SendProgress event for that class.

## Endianess

There are two different endian standards. Windows and Linux use “little” endianess and Macintosh uses “big” endianess. Sockets work with streams of data and do not change the endianness of the data you are transferring. This is fine in many cases if you are transferring strings from one platform to another (like from Mac OS X to Windows XP). But if you are transferring binary data, such as from a [BinaryStream](#) or a [MemoryBlock](#), you will want to ensure that the endianess matches from server to client regardless of what platform you are on. You can do this by setting the LittleEndian property on [MemoryBlocks](#) or [BinaryStreams](#) to the same value for your client and your server. Failure to ensure that the endianess is consistent will result in a possible byte-order conflict in your application. You can use the [TargetBigEndian](#) and [TargetLittleEndian](#) constants to determine the endian standard for the current computer.

## See Also

[Datagram](#), [ServerSocket](#), [TCPSocket](#), [UDPSocket](#) classes.

---

## Sort Method

Sorts the elements of a one-dimensional array in ascending order.

## Syntax

**array.Sort**

| Part  | Description                       |
|-------|-----------------------------------|
| array | Required. The array to be sorted. |

## Notes

The **Sort** method works with [Integer](#), [string](#), [single](#), and [double](#) arrays only. It accepts only one-dimensional arrays.

## Sortwith Method

---

**Examples** This example sorts the aNames array.

```
aNames.Sort
```

**See Also** [Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sortwith](#) methods; [ParamArray](#) keyword.

## Sortwith Method

---

Sorts one or more additional arrays in the same order as the base array. The sort is in ascending order.

**Syntax** **array.Sortwith(array1[,...arrayN])**

| Part                | Description   |
|---------------------|---|
| array               | Required. The array to be sorted.   |
| array1              | Required. Array to be sorted in the order determined by sorting the base array. Array1 must have the same number of elements as the base array but can be of another data type. |
| array2...<br>arrayN | Optional. Additional arrays to be sorted with the base array. These arrays must also have the same number of elements as the base array but can be of different data types.     |

### Notes

The base array used with the **Sortwith** method works with [Integer](#), [string](#), [single](#), and [double](#) arrays only. It accepts only one-dimensional arrays.

**Sortwith** is ideal for sorting all the columns of a data table by one of its columns. For example, if you have a set of three arrays that store Names, Addresses, and Phone numbers of a group of people, you can sort the data table by Name by specifying the Names array as the base array and pass the Address and Phone number arrays as the parameters.

**Examples** This example sorts the columns of the data table by the values in the aNames array.

```
Dim aNames(), aAddresses(),aPhones() as String
aNames=array("Mozart", "Bing", "Jackson", "Flintstone")
aAddresses=array("34 Pickwick", "3424 Kenwood", "432 Marlboro", "1 Hardrock")
aPhones=array("234-3700", "697-5300", "753-2500", "None")

aNames.Sortwith(aAddresses,aPhones)

//display the sorted arrays in a Listbox to verify the sort
For i as Integer=0 to ubound(aNames)
    ListBox1.addrow aNames(i)
    ListBox1.cell(i+1,1)=aAddresses(i)
    ListBox1.cell(i+1,2)=aPhones(i)
Next
```

## See Also

[Dim](#) statement; [Array](#), [Join](#), [Split](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#) methods; [ParamArray](#) keyword.

# Sound Class

Used to play sounds.

Super Class [Object](#)

## Methods

| Name        | Parameters | Description   |
|-------------|------------|---|
| Clone       |            | Returns a clone of the sound as a <b>Sound</b> , but can be played, stopped, and modified independently of the original sound. Use Clone instead of opening the same sound file twice, because Clone is likely to be substantially more efficient.                              |
| IsPlaying   |            | Returns <b>True</b> if the sound is playing.  |
| Pan         |            | Specifies the relative volume (balance) between the left and right speakers. Range is from -100 to +100, with 0 indicating equal volume in the left and right channels. -100 plays sound in the left channel only; 100 plays sound in the right channel only. The default is 0. |
| Play        |            | Plays the sound.  |
| PlayLooping |            | Plays the sound in an infinite loop.  |
| Stop        |            | Stops a sound that is playing.  |

## Speak Method

---

| Name   | Parameters | Description  |
|--------|------------|--|
| Volume |            | Controls the volume of the sound. Range is from 0 to 100; 0 mutes the sound and 100 plays the sound at the “normal” volume set in the computer’s volume setting. Default is 100. |

### Notes

Sounds that have been added to the Project Editor can be accessed via their object name. Sounds can also be accessed by calling the OpenAsSound method of a [FolderItem](#). There are no properties or events for sound objects.

**Sound** can play any number of sounds simultaneously, limited only by system resources.

### Examples

This example plays a sound called “Sledgehammer” which has been dragged into the Project Editor.

```
sledgehammer.play
```

This example loads a sound file called “Doh!” from the current directory (folder) into a sound object and plays it.

```
Dim f as FolderItem  
Dim s as Sound  
f = GetFolderItem("Doh!")  
s = f.OpenAsSound  
s.play
```

The following example plays the sound “Giggle”, which has been added to the Project Editor, in an endless loop.

```
Giggle.playlooping
```

This example stops “Giggle.”

```
Giggle.stop
```

### See Also

[NotePlayer](#) control.

---

## Speak Method

Uses the computer’s speech synthesizer to pronounce the passed text string (Windows and Macintosh only).

**Syntax****Speak (*phrase [,Interrupt]*)**

| Part      | Type  | Description  |
|-----------|---|--|
| phrase    | <a href="#">String</a> or <a href="#">Variant</a> | Text string to be passed to the speech synthesizer. Speak will accept any <a href="#">variant</a> that can be expressed as a <a href="#">string</a> .        |
| Interrupt | <a href="#">Boolean</a>                           | Optional interrupt flag. If set to <a href="#">True</a> , the call will terminate the previous calls to Speak. If omitted, <a href="#">False</a> is assumed. |

**Notes**

**Speak** takes a [string](#) (or any [variant](#) that can be expressed as a [string](#)) and uses the Windows or Macintosh text-to-speech engine to speak the text. The speech is asynchronous, allowing normal program flow to continue. By default, subsequent calls to **Speak** before the first call has finished will queue up and speak after the completion of the previous call. If the optional interrupt flag is used and set to [True](#), the previous **Speak** calls will be stopped immediately.

**Speak** is not supported on Linux.

**Example**

The following code in a [PushButton](#) pronounces the phrase entered into an [EditField](#).

```
If EditField1.text <> "" then
    Speak EditField1.text
Else
    Speak "Please enter some text in the field!"
End if
```

## SpecialFolder Object

Used to get a [FolderItem](#) to a specific folder or directory managed by the host operating system.

**Syntax****result=SpecialFolder.FolderName**

| Part       | Type                            | Description  |
|------------|---------------------------------|--|
| result     | <a href="#">FolderItem</a>      | If successful, a <a href="#">FolderItem</a> for the specified folder/directory. SpecialFolder returns <a href="#">Nil</a> if the call is unsuccessful. |
| FolderName | <a href="#">String</a> constant | The requested folder or directory. See the table in the Notes section for the possible values.   |

**Notes**

Not all types of folders are supported on all operating systems. After a call to **SpecialFolder**, check that the result not [Nil](#).

## SpecialFolder Object

---

Here is the list of possible values for *FolderName* and what is returned on each platform. If the path includes the name of the current user, the word “user” appears in italics.

| FolderName                    | Windows XP                                     | Mac OS X  | Linux               |
|-------------------------------|--|---|---------------------|
| AppleMenu                     | \User\StartMenu\Programs                       | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| Applications                  | \Program Files\                                | :Applications:  | <a href="#">Nil</a> |
| ApplicationData               | \User\Application Data\                        | user:Library:Application Support:   | <a href="#">Nil</a> |
| Bin                           | <a href="#">Nil</a>                            | :bin:   | /bin                |
| ControlPanels                 | <a href="#">Nil</a>                            | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| Cookies                       | \User\Cookies\                                 | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| Desktop                       | \User\Desktop\                                 | :Users: <i>user</i> :Desktop:   | <a href="#">Nil</a> |
| Documents                     | \User\My Documents\                            | Users: <i>user</i> :Document:   | <a href="#">Nil</a> |
| Etc                           | <a href="#">Nil</a>                            | :private:etc:   | /etc/               |
| Extensions                    | \Windows\System32\                             | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| Favorites                     | \User\Favorites\                               | Users: <i>user</i> :Library:Favorites:  | <a href="#">Nil</a> |
| Fonts                         | \WindowsFonts\                                 | :System:Library:Fonts:  | <a href="#">Nil</a> |
| GetFromCode ( <i>string</i> ) |  | Pass a four-character code in <i>String</i> for a specific <a href="#">FolderItem</a> . |                     |
| History                       | \User\Local Settings.History\                  | Users: <i>user</i> :Sites:  | <a href="#">Nil</a> |
| Home                          | <a href="#">Nil</a>                            | :Users:   | /home/              |
| InternetCache                 | \User\Local Settings\Temporary Internet Files\ | :Library:Caches:  | <a href="#">Nil</a> |
| Library                       | <a href="#">Nil</a>                            | :Library:   | /lib/               |
| Mount                         | <a href="#">Nil</a>                            | :Volumes:   | /mnt/               |
| Music                         | \User\My Documents\My Music\                   | Users: <i>user</i> :Music:  | <a href="#">Nil</a> |
| NetworkPlaces                 | \User\NetHood\                                 | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| Pictures                      | \User\My Documents\My Pictures\                | Users: <i>user</i> :Pictures:   | <a href="#">Nil</a> |
| Preferences                   | \User\Application Data\                        | Users: <i>user</i> :Library:Preferences:  | <a href="#">Nil</a> |
| Printers                      | \User\PrintHood\                               | :System:Library:Printers:   | <a href="#">Nil</a> |
| RecentItems                   | \User\Recent\                                  | Users: <i>user</i> :Library:Recent Documents:   | <a href="#">Nil</a> |
| SBin                          | <a href="#">Nil</a>                            | :sbin:  | /sbin/              |
| SendTo                        | \User\SendTo\                                  | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| SharedApplicationData         | \All Users\Application Data\                   | <a href="#">Nil</a>   | <a href="#">Nil</a> |
| SharedApplications            | \Program Files\Common Files\                   | <a href="#">Nil</a>   | <a href="#">Nil</a> |

| FolderName         | Windows XP                         | Mac OS X                             | Linux               |
|--------------------|------------------------------------|--------------------------------------|---------------------|
| SharedDesktop      | \All Users\Desktop\                | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| SharedDocuments    | \All Users\Documents\              | :Users:Shared:                       | <a href="#">Nil</a> |
| SharedFavorites    | All Users.Favorites\               | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| SharedPreferences  | \All Users\Application Data\       | :Library:Preferences                 | <a href="#">Nil</a> |
| SharedStartupItems | \Start Menu\Programs\Startup\      | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| SharedTemplates    | \All Users\Templates\              | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| ShutdownItems      | <a href="#">Nil</a>                | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| StartupItems       | \user\Start Menu\Programs\Startup\ | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| System             | \Windows\System32\                 | :System:                             | <a href="#">Nil</a> |
| Templates          | \user\Templates\                   | <a href="#">Nil</a>                  | <a href="#">Nil</a> |
| Temporary          | \user\Local Setings\Temp\          | :private:tmp:501:<br>TemporaryItems: | <a href="#">Nil</a> |
| Trash              | \user\Desktop\Recycle Bin\         | :Users:user:.Trash                   | <a href="#">Nil</a> |
| UserBin            | <a href="#">Nil</a>                | :usr:bin:                            | /usr/bin/<br>/      |
| UserHome           | <a href="#">Nil</a>                | :Users:user:                         | /home/<br>user/     |
| UserLibrary        | <a href="#">Nil</a>                | <a href="#">Nil</a>                  | /usr/lib/           |
| UserSBin           | <a href="#">Nil</a>                | :usr:sbin:                           | /usr/sbi<br>n/      |
| Var                | <a href="#">Nil</a>                | :private:var:                        | /var/               |
| VarLog             | <a href="#">Nil</a>                | :private:var:log:                    | /var/log/<br>/      |
| Windows            | \Windows\                          | <a href="#">Nil</a>                  | <a href="#">Nil</a> |

## Example

The following code gets a [FolderItem](#) for the Application Data folder and displays its absolute path.

```
Dim f as FolderItem
f=SpecialFolder.ApplicationData
If f <> Nil then
  MsgBox f.Absolutepath
Else
  MsgBox "There is no Application Data folder on this computer"
End if
```

## See Also

[FolderItem](#) class.

# Split Function

Creates a one-dimensional array from the [String](#) passed.

### Syntax

**result=Split(source [,delimiter])**

OR

**source.Split([delimiter])**

| Part      | Type                            | Description  |
|-----------|---------------------------------|--|
| result    | <a href="#">String</a><br>array | Array resulting from breaking <i>source</i> into elements using <i>delimiter</i> as the field delimiter.   |
| source    | <a href="#">String</a>          | Source string to be parsed into an array.  |
| delimiter | <a href="#">String</a>          | Optional field delimiter used to parse <i>Source</i> into array elements. If <i>delimiter</i> is omitted, then a space is used as the delimiter. |

### Notes

Use the **Split** function to create a new [String](#) array from a list of elements (or fields) that are separated by a delimiter. If the optional parameter, *delimiter*, is not passed, a single space is assumed as the delimiter. If the delimiter is an empty string, the source string is split into characters.

### Examples

The first example specifies the comma delimiter and the second example uses the default delimiter. They place each field into an array element, producing a three-element array. The last example parses the string into individual characters.

```
Dim anArray(-1) as String  
anArray=Split("Adam,Aardvark,Accountant","","")  
anArray=Split("Adam Aardvark Accountant")  
anArray=Split("Adam","")  
//First two using the alternate syntax:  
Dim s as String  
s="Adam,Aardvark,Accountant"  
anArray=s.Split(",") //produces 3-element array  
anArray=s.Split("")//produces array of individual characters
```

### See Also

[String](#) data type; [Dim](#) statement; [Array](#), [Join](#), [NthField](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

---

# SplitB Function

Creates a one-dimensional array from the [String](#) passed. **SplitB** is identical to [Split](#), except that it treats the source as binary data.

**Syntax** `result=SplitB(source[,delimiter])`

OR

`source.SplitB([delimiter])`

| Part      | Type                            | Description  |
|-----------|---------------------------------|--|
| result    | <a href="#">String</a><br>array | Array resulting from breaking <i>source</i> into elements using <i>delimiter</i> as the field delimiter.   |
| source    | <a href="#">String</a>          | Source string to be parsed into an array.  |
| delimiter | <a href="#">String</a>          | Optional field delimiter used to parse <i>Source</i> into array elements. If <i>delimiter</i> is omitted, then a space is used as the delimiter. |

## Notes

Use the **SplitB** function to create a new [String](#) array from a list of elements (or fields) that are separated by a delimiter. If the optional parameter, *delimiter*, is not passed, a single space is assumed as the delimiter. If the delimiter is an empty string, the source string is split into characters.

## Examples

The first example specifies the comma delimiter and the second example uses the default delimiter. They place each field into an array element, producing a three-element array. The last example parses the string into individual characters.

```
Dim anArray(-1) as String
anArray=SplitB("Adam,Aardvark,Accountant","","")
anArray=SplitB(" Adam Aardvark Accountant ")
anArray=SplitB("Adam"," ")
//First two using the alternate syntax:
Dim s as String
s="Adam,Aardvark,Accountant"
anArray=s.SplitB(",") //produces 3-element array
anArray=s.SplitB(" ")//produces array of individual characters
```

## See Also

[String](#) data type; [Dim](#) statement; [Array](#), [CountFieldsB](#), [Join](#), [NthFieldB](#), [Ubound](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

# SpotlightException Class

**Super Class** [RuntimeException](#)

Used to handle exceptions related to Spotlight queries, such as a invalid query passed to the *Query* property of a [SpotlightQuery](#) control.

## See Also

[Exception](#), [Try](#) blocks, [RuntimeException](#), [SpotlightItem](#), [SpotlightQuery](#) classes.

## SpotlightItem Class

Used to manage results from a [SpotlightQuery](#). Supported only on Mac OS X 10.4 and above. It does nothing on earlier versions of Mac OS X and all other operating systems.

### Properties

| Name          | Type                       | Description   |
|---------------|----------------------------|---|
| <b>File</b>   | <a href="#">FolderItem</a> | References the file that this <b>SpotlightItem</b> represents.  |
| <b>Value</b>  | <a href="#">Variant</a>    | Parameter is PropertyName as <a href="#">String</a> . Returns the value of the specified property. The properties can be any of the ones listed in MDItem.h or any custom properties. Value returns <a href="#">Nil</a> if the specified property is not present. |
| <b>Handle</b> | <a href="#">Integer</a>    | Returns a handle to an MDItemRef. This can be used in <a href="#">Declare</a> statements to access features that are not provided by <a href="#">SpotlightQuery</a> .   |

**Examples** See the examples for [SpotlightQuery](#) which use the File property to display attributes of the each search result.

**See Also** [SpotlightException](#), [SpotlightQuery](#) classes.

---

## SpotlightQuery Class

Used to perform Spotlight searches on Mac OS X 10.4 and above. It does nothing on earlier versions of Mac OS X and all other operating systems.

**Super Class** [Object](#)

**SpotlightQuery** appears in the list of Built-in controls in the IDE, but since it is not subclassed from [Control](#), you can instantiate it via code.

### Constructor

| Name           | Parameters                         | Description  |
|----------------|------------------------------------|--|
| SpotlightQuery | [query as <a href="#">String</a> ] | The optional query parameter specifies the query. The default is no query. |

### Properties

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| <b>Completed</b> | <a href="#">Boolean</a> | <a href="#">True</a> if the query has finished running and <a href="#">False</a> otherwise. |

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| Handle      | <a href="#">Integer</a> | Returns the handle to the MDQueryRef. This can be used in <a href="#">Declares</a> to access Spotlight features that are not supported directly by this class. It may not be available until after calling the Run method. |
| Query       | <a href="#">String</a>  | The query string that will be passed to the Spotlight engine. You search on the metadata attribute keys provided by Apple using their format. See the notes for the format of the query.                                   |
| Synchronous | <a href="#">Boolean</a> | If <a href="#">True</a> , the Run method will be synchronous. If it is synchronous, then the events will not fire.   |

## Events

| Name      | Parameters  | Description  |
|-----------|---|--|
| Changed   | ItemsAdded() as <a href="#">SpotlightItem</a> ,<br>ItemsChanged() as <a href="#">SpotlightItem</a> ,<br>ItemsRemoved() as <a href="#">SpotlightItem</a> | This event is fired when the <b>SpotlightQuery</b> has either found more items, determined that some no longer match, or both. When doing the initial gathering, ItemsAdded() is not populated for speed reasons since it can be called frequently.  |
| Completed |   | Fired when the <b>SpotLightQuery</b> has finished gathering the initial items. Call the Stop method inside this event if you would not like to know when new documents match the query. If you don't call Stop, the Changed event will fire if items no longer match or new items match the query. |

## Methods

| Name   | Parameters                       | Description   |
|--------|----------------------------------|---|
| Count  |                                  | Gets the number of results of the query.  |
| Item   | Index as <a href="#">Integer</a> | Returns a <a href="#">SpotlightItem</a> . Returns the <a href="#">SpotlightItem</a> specified by Index. Index is zero-based.                                      |
| Pause  |                                  | Temporarily pauses the query from posting any updates. This means that the Count and Item functions will not change until you resume the query by calling Resume. |
| Resume |                                  | Resumes reporting more results after the query has been paused.   |
| Run    |                                  | Runs the query specified by the Query property.   |
| Stop   |                                  | Stops the query.  |

## SpotlightQuery Class

---

### Notes

An overview of the Spotlight API is at:

<http://developer.apple.com/documentation/Carbon/Conceptual/SpotlightQuery/index.html>

Spotlight works by extracting metadata attributes from files on the user's hard disk. By default, this extraction is done in the background by Spotlight *Importers*. When an end-user does a Spotlight search, he is actually doing a search on the attributes that have been extracted via the importers. When you use the **SpotlightQuery** class, you must specify the attribute or attributes you are searching on using Spotlight keywords and syntax.

In other words, you will need to become familiar with Apple's MDQuery language. Each simple query is in the format of *attribute=Value*, where attribute is a Spotlight metadata attribute and Value is the target value.

Apple's list of searchable metadata attributes is at:

[http://developer.apple.com/documentation/Carbon/Reference/MetadataAttributesRef/index.html#/apple\\_ref/doc/uid/TP40001689](http://developer.apple.com/documentation/Carbon/Reference/MetadataAttributesRef/index.html#/apple_ref/doc/uid/TP40001689)

For example, `kMDItemContentType = "*audio*"` would find all files that had a content type containing "audio" (case insensitive). The "\*" is the wildcard character. You can combine expressions with "`&&`" (logical "And") and "`||`" (logical "Or"). For example, to find a file that was Audio and had a artist of Lifehouse, it would look like this: `kMDItemContentType = "*audio*" && kMDItemArtist = "Lifehouse".`

The complete description of MDQuery syntax is at:

[http://developer.apple.com/documentation/Carbon/Conceptual/SpotlightQuery/index.html#/apple\\_ref/doc/uid/TP40001841](http://developer.apple.com/documentation/Carbon/Conceptual/SpotlightQuery/index.html#/apple_ref/doc/uid/TP40001841)

### Examples

The following synchronous query populates a **ListBox** with the list of audio files on the user's computer and the absolute path to each file. You can put the code in a **PushButton**.

```
Dim query as New SpotlightQuery("kMDItemContentType=""*audio*""")  
query.Synchronous=True  
query.Run  
  
For i as Integer = 0 to query.Count-1  
    ListBox1.AddRow query.Item(i).File.DisplayName  
    ListBox1.Cell(i,1)=query.Item(i).File.AbsolutePath  
Next  
  
Exception err as SpotlightException  
MsgBox "A Spotlight error occurred."
```

The following asynchronous query uses the search string that the user enters into an **EditField** and displays the filename and its absolute path in a **Listbox**. It uses a **SpotlightQuery** control named "Query" that has been added to the window.

First, add the following method “UpdateList” to the window:

```
ListBox1.DeleteAllRows
Query.Pause

For i as Integer = 0 to Query.Count-1
    ListBox1.AddRow Query.Item(i).File.DisplayName
    ListBox1.Cell(i,1)=Query.Item(i).File.AbsolutePath
Next
Query.Resume
```

In the **SpotLightQuery**’s Changed and Completed event handlers, call the UpdateList method.

In a PushButton, enter the following code in its Action event handler.

```
If EditField1.Text <> "" Then
    Query.Query = "kMDItemDisplayName = \"*"+EditField1.Text+"*\""
    Query.Run
Else
    MsgBox "Please enter a file name to search for."
End If

Exception err as SpotlightException
MsgBox "A Spotlight error occurred."
```

The following If statement checks to see if the user is running Mac OS X 10.4 or higher:

```
Dim sysversion as Integer
If System.Gestalt("sysv",sysversion) and sysversion >= &h1040 then
    //you can call SpotlightQuery here
Else
    //Don't bother calling SpotlightQuery
End if
```

**See Also** [SpotlightException](#), [SpotlightItem](#) classes.

## Sprite Class

A picture that can be animated by a [SpriteSurface](#) control.

**Super Class** [Object](#)

## Properties

| Name     | Type                    | Description  |
|----------|-------------------------|--|
| Group    | <a href="#">Integer</a> | Determines whether or not the <a href="#">SpriteSurface</a> control's Collision event handler will be called when the Sprite comes in to contact with another Sprite. Sprites with a negative Group value will collide with any other Sprites (except those with a Group value of 0). A Group value of 0 means the Sprite will not collide with any other Sprites. A positive Group value means that the Sprite will collide with Sprites of any other Group value (except 0) but not those with the same Group value. |
| Image    | <a href="#">Picture</a> | The picture that will be drawn when the SpriteSurface's Run method is called.  |
| Priority | <a href="#">Integer</a> | Determines the order the Sprite is drawn in during the NextFrame event handler.  |
| ScreenX  | <a href="#">Integer</a> | The sprite's distance in pixels from the left side of the <a href="#">SpriteSurface</a> .  |
| ScreenY  | <a href="#">Integer</a> | The sprite's distance in pixels from the top of the <a href="#">SpriteSurface</a> .  |
| X        | <a href="#">Integer</a> | The location of the left side of the Image within the <a href="#">SpriteSurface</a> area.  |
| Y        | <a href="#">Integer</a> | The location of the top of the Image within the <a href="#">SpriteSurface</a> area.  |

## Methods

| Name  | Parameters | Description  |
|-------|------------|--|
| Close |            | Removes the Sprite from the <a href="#">SpriteSurface</a> it was created in. |

## Notes

A **Sprite** object is a picture that can be animated using a [SpriteSurface](#) control. Moving the Sprite across the screen is simply a matter of changing the Sprite's x and y properties in the NextFrame event handler of the [SpriteSurface](#) control that created the sprite. Should two sprites touch each other during the animation, there is the potential for a collision. A collision means that the [SpriteSurface](#)'s Collision event handler will be executed and the two Sprites that have collided will be passed to it. This gives you the opportunity to decide what should happen when the two sprites come into contact. Whether or not a collision occurs is based on the values of the two Sprites' Group properties:

| Group Values                                | Collision? |
|---|------------|
| 0 for either Sprite                         | No         |
| Both Sprites have same positive value       | No         |
| Both Sprites have different positive values | Yes        |
| One Sprite has a negative value             | Yes        |

Sprites are drawn during the [SpriteSurface](#) NextFrame event handler's execution in order based on their Priority property. This property is assigned by default based on the order the Sprites were created. The first Sprite's Priority is 1, the second Sprite's

Priority is 2, etc. In this example, when the first and second Sprites pass over each other, the second Sprite will appear to be in front of the first Sprite.

All Sprite object properties are changeable at runtime. This means you can change the Sprite's image, location, group, and priority during the animation.

A Sprite can be created using the [New](#) operator and attached to a [SpriteSurface](#) using the Attach method of the [SpriteSurface](#) class.

#### **Comparing ScreenX and X (and ScreenY and Y)**

ScreenX and X (and ScreenY and Y) will be the same for a SpriteSurface that has not been scrolled. When it is scrolled, the sprite's ScreenX and ScreenY values change because their positions on the screen have changed. Their X and Y values remain the same because their position in "world" coordinates has not changed.

#### **Mask and Transparency Properties**

Sprites ignore the Mask property of the [Picture](#), as well as the value of the Transparent property. White pixels are always transparent and non-white pixels are always opaque.

#### **Examples**

See the [SpriteSurface](#) control's example.

#### **See Also**

[SpriteSurface](#) control.

## SpriteSurface Control

Used to create sprite animation.

**Super Class** [RectControl](#)

#### **Properties**

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| BackDrop    | <a href="#">Picture</a> | A picture that will be drawn within the area defined by the Height, Left, Top, and Width properties when the SpriteSurface's Run method is called. If the picture is too small, it will be tiled. to fill out the entire area. <a href="#">Sprite</a> objects will be drawn in front of the BackDrop. |
| ClickToStop | <a href="#">Boolean</a> | If <a href="#">True</a> , clicking the mouse will stop the SpriteSurface animation.   |
| Depth       | <a href="#">Integer</a> | Bit depth of the spritesurface area. May take on values of 0, 8,16, or 32. Specifying 0 causes the spritesurface to automatically match its bit depth to the depth of the monitor at the time the spritesurface was opened. The Depth property can be changed on the fly.                             |

## SpriteSurface Control

---

| Name          | Type                     | Description  |
|---------------|--------------------------|--|
| FrameSpeed    | <a href="#">Integer</a>  | The number of vertical retraces per frame. Zero is the fastest the computer can redraw. Compute FrameSpeed by dividing 60 by the number of frames per second you want and round to the next <a href="#">Integer</a> . Each frame will cause the NextFrame event to execute.<br>The definition of FrameSpeed has changed in version 2 of REALbasic. Please update version 1 applications accordingly. |
| Graphics      | <a href="#">Graphics</a> | Used for drawing to the screen when the SpriteSurface is running. Avoid drawing into the area defined by the Height, Left, Top, and Width properties since the SpriteSurface handles this area.  |
| ScrollX       | <a href="#">Integer</a>  | Distance the SpriteSurface has been scrolled in the horizontal direction.  |
| ScrollY       | <a href="#">Integer</a>  | Distance the SpriteSurface has been scrolled in the vertical direction.  |
| Super         |                          | The class of object the control is based on.   |
| SurfaceHeight | <a href="#">Integer</a>  | Now set equal to Height.   |
| SurfaceLeft   | <a href="#">Integer</a>  | Now set equal to Left.   |
| SurfaceTop    | <a href="#">Integer</a>  | Now set equal to Top.  |
| SurfaceWidth  | <a href="#">Integer</a>  | Now set equal to Width.  |

## Events

| Name      | Parameters  | Description  |
|-----------|---|--|
| Collision | S1 as <a href="#">Sprite</a> ,<br>S2 as <a href="#">Sprite</a>  | Two <a href="#">sprites</a> with different Group property values (or negative Group property values) have collided (one has touched the other).                  |
| NextFrame |   | The SpriteSurface is ready to draw the next frame of animation. <a href="#">Sprite</a> objects will be redrawn at the end of this event handler.                 |
| PaintTile | g as <a href="#">Graphics</a><br>xpos as<br><a href="#">Integer</a><br>ypos as<br><a href="#">Integer</a> | Occurs when the Scroll method is called.<br>g is a <a href="#">Graphics</a> object and xpos and ypos are the coordinates of the square that needs to be redrawn. |

## Methods

| Name   | Parameters                  | Description   |
|--------|-----------------------------|---|
| Attach | S as <a href="#">Sprite</a> | Attaches the passed <a href="#">Sprite</a> to a SpriteSurface.                                    |
| Close  |                             | Stops the animation and returns the screen to the window that contains the SpriteSurface control. |

| Name      | Parameters   | Description  |
|-----------|--|--|
| KeyTest   | KeyCode as <a href="#">Integer</a>   | Returns <a href="#">True</a> if the key represented by the KeyCode passed has been pressed. See the <a href="#">KeyCode</a> chart for acceptable values.   |
| NewSprite | Image as <a href="#">Picture</a> ,<br>X as <a href="#">Integer</a> ,<br>Y as <a href="#">Integer</a> ) | Returns a new <a href="#">Sprite</a> object and places its Image within the SpriteSurface at the coordinates passed.   |
| PaintTile | X as <a href="#">Integer</a><br>Y as <a href="#">Integer</a>   | Forces the background tile specified by x,y to be repainted, triggering the PaintTile event. If you need to change part of the background during a game, this is the best way to force the background to update. (Note that the update will not actually appear on the screen until the next frame.)   |
| Run       |  | Begins the animation. Calling Run starts the animation in a modal state: it locks up the user interface and prevents music from playing. A safer method of running the animation is to add a <a href="#">Timer</a> to the window that contains the Sprite surface and set its mode to 2 (multiple calls). In the <a href="#">Timer</a> 's Action event, call the SpriteSurface's Update method. This may not achieve as high a frame rate as calling Run but avoids the problems of a modal interface. |
| Scroll    | dx as <a href="#">Integer</a><br>dy as <a href="#">Integer</a>   | Scroll the SpriteSurface the amount given by dx and dy.  |
| Stop      |  | Stops the animation.   |
| Update    |  | Runs the animation one step per call. It redraws the sprites that have moved or changed, fires collision events, and calls the NextFrame event once.   |

## Notes

### Sizing and Positioning the SpriteSurface Area

The SpriteSurface control can be added to a window and resized by dragging or by assigning values to the The Left, Top, Width, and Height properties. These properties are read/write and can be used to change the size and location of the SpriteSurface on-the-fly.

You can also use two properties of the parent window, [FullScreen](#) and [MenuBarVisible](#), to allow the SpriteSurface object to take over the user's entire screen area. Set [FullScreen](#) to [True](#) and [MenuBarVisible](#) to [False](#) and use the SpriteSurface's Lock... properties to lock the SpriteSurface control to the parent window so that the SpriteSurface area expands automatically. This sets the Left, Top, Width, and Height properties accordingly.

### Creating Animation with Sprites

The SpriteSurface control is used to create animation where pictures can be moved around the screen with all redrawing handled automatically by the SpriteSurface control. Each picture is a [Sprite](#) object. [Sprite](#) objects have x and y properties that

determine their current location on the screen when the SpriteSurface is running the sprite animation.

### Causing Sprites to Move and Change Images

The NextFrame event handler is called each time the SpriteSurface is ready to draw the next frame of animation. If you want a Sprite to change position in the next frame, change its x and/or y properties in the NextFrame event handler. If you want the Sprite's picture to change in the next frame of animation, change its Image property in the NextFrame event handler. Use the Update method to run the animation one step at a time. It calls the NextFrame event handler only once.

To remove a Sprite from the animation, simply call the Sprite's Close method.

### Frame Redrawing

The speed at which frames are redrawn is based on the FrameSpeed property. The higher the FrameSpeed, the faster the animation will run.

The safest way to run the animation is to update it via a [Timer](#). Add a [Timer](#) to the window that contains the **SpriteSurface**. Adjust its Period property according to how fast you want the animation to go and set its Mode to 2 (Multiple). In its Action event, call the **SpriteSurface's** Update method.

You can also call the SpriteSurface's Run method, but this puts the application in a modal state, which can cause other problems. Call the Update method via the [Timer](#) if you want to allow the user to use the mouse and keyboard during the animation.

To stop the animation, call the SpriteSurface's Close method.

### Current Implementation

The size and position of the SpriteSurface area are controlled by the Left, Top, Height, and Width properties or by setting the parent window's [FullScreen](#) property to [True](#) and using the LockLeft, LockTop, LockRight, and LockBottom properties to automatically expand the SpriteSurface area as the window expands.

Once the Run method is called, NextFrame events will continue to be called until the user clicks the mouse button if the ClickToStop property is [True](#). If ClickToStop is [False](#), the animation will continue until the window that the SpriteSurface is on closes.

**KeyCodes**

The following graphic shows the keycodes for an Apple standard keyboard and the French keyboard. This graphic was provided courtesy of Apple Computer, Inc.:

**Examples**

In this example, a [Sprite](#) is moved back and forth horizontally across the screen. The PushButton's Action event handler creates a [Sprite](#) and stores it in a window property of type [Sprite](#) named LisaSprite. The image for the LisaSprite object is a pict image called LisaPict that has been dragged into the Project Editor. The SpriteSurface control named SpriteSurface1 will control the animation. The [PushButton](#) then begins the animation by calling the Run method of the SpriteSurface1 control. Another window property of type [Integer](#) named LisaDirection is used to keep track of the direction of the LisaSprite. Here are the event handlers:

PushButton1:

[Sub Action\(\)](#)

```
LisaSprite=SpriteSurface1.NewSprite(LisaPict,200,200)
```

```
LisaDirection=0
```

SpriteSurface1:

```
Sub NextFrame()
If LisaSprite.x>=640 Then
    LisaDirection=1
ElseIf lisaSprite.x<=0 then
    lisaDirection=0
End if
Select Case lisaDirection
Case 0
    lisaSprite.x=LisaSprite.x+1
Case 1
    lisaSprite.x=lisaSprite.x-1
End Select
```

The [Timer](#) in the window calls the SpriteSurface's Update method in its Action event. It's mode is set to 2.

**SpriteSurface1.Update**

This example closes (using the Close method) the running SpriteSurface control when the user presses the Return key:

```
Sub NextFrame()
If Me.KeyTest(&h24) Then
    Me.Close
End if
```

A more sophisticated example called “REALSimpleSprites” that uses a new class to control sprites is available at our ftp site and on the REALbasic CD.

The following example uses the PaintTile event to animate the screen image when the user presses an arrow key. The PaintTile event occurs when the scroll method is called.

The SpriteSurface's NextFrame event has the following code:

```
If spriteSurface1.keytest(&h7E) then //up
    me.scroll 0 , -10
elseif spriteSurface1.keytest(&h7D) then //down
    me.scroll 0 , 10
elseif spriteSurface1.keytest(&h7B) then //left
    me.scroll -10,0
elseif spriteSurface1.keytest(&h7C) then //right
    me.scroll 10,0
elseif spriteSurface1.keytest(&h24) then //quit
    me.close
end if
```

Each condition tests whether the user pressed an arrow key and calls the scroll method with appropriate x and y values. The PaintTile event has the following code:

```
Sub PaintTile (g as Graphics, xpos as Integer, ypos as Integer)
    g.foreColor=rgb(0,0,0) //set the forecolor to black
    g.fillRect 0,0,64,64 //erase the tile
    //set the forecolor to white
    g.foreColor=rgb(255,255,255)
    g.drawString str(xpos)+"/"+str(ypos), 0,32
```

**See Also** [Sprite](#) object.

## Sqrt Function

Returns the square root of the value specified.

**Syntax** **result=Sqrt (value)**

| Part   | Type                   | Description                            |
|--------|------------------------|--|
| value  | <a href="#">Double</a> | The value you want the square root of. |
| result | <a href="#">Double</a> | The square root of the <i>value</i> .  |

**Notes** The **Sqrt** function returns the square root of the value passed to it.

**Example** This example uses the **Sqrt** function to return the square root of a number.

```
Dim d As Double
d=Sqrt(16) //returns 4
```

**See Also** [^](#), [\\*](#) operators.

## SSLocket Control

Use the **SSLocket** control to do secure communications via TCP/IP using secure sockets layer (SSL) technology. The **SSLocket** control is available only in the Professional version of REALbasic. The Standard version of REALbasic does not compile references to the **SSLocket** control.

**Super Class** [TCPSocket](#) class.

## Properties

| Name                 | Type                    | Description  |
|----------------------|-------------------------|--|
| ConnectionType       | <a href="#">Integer</a> | Specifies the type of SSL connection. ConnectionType can take the following values:<br>0 - SSLv2: SSL (Secure Sockets Layer) version 2.<br>1 - SSLv23: SSL version 3, but can roll back to 2 if needed.<br>2- SSLv3: SSL version 3.<br>3- TLSv1: TLS (Transport Layer Security) version 1. The default value is 1, SSL version 23. Once you have called the SSLocket.Connect method, you cannot change the connection type without calling SSLocket.Close first, but you can always query the connection type. If you are unsure whether the server supports SSLv23, you can try connecting multiple times. If an attempt results in a 102 error, try again with a different value for ConnectionType. |
| Secure               | <a href="#">Boolean</a> | Set to <a href="#">True</a> to specify an SSL connection. If Secure is <a href="#">False</a> , the SSLocket transmits data just like a <a href="#">TCPSocket</a> .   |
| <b>SSLConnected</b>  | <a href="#">Boolean</a> | <a href="#">True</a> if you have an SSL connection.  |
| <b>SSLConnecting</b> | <a href="#">Boolean</a> | <a href="#">True</a> if REALbasic is in the process of doing a handshake to establish an SSL connection.   |

## Events

| Name          | Parameters | Description  |
|---------------|------------|--|
| Connected     |            | A connection has been established using the Address and Port properties. |
| DataAvailable |            | New incoming data is now available in the buffer.                        |
| Error         |            | An error occurred.   |
| SendComplete  |            | All of the data in the socket buffer has been written (sent).            |

## Notes

To establish an SSL connection, set the Secure property to [True](#) and use the Connect method. **SSLocket** does not support secure listening sockets.

The **SSLocket** control is not listed in the Controls pane in the Window Editor. There are two ways to add an **SSLocket** control to a window:

- Drag a [TCPSocket](#) control to a window and then change its Super class to **SSLocket**.
- Display the window's contextual menu by right+clicking (Windows and Linux) or control-clicking on the window (Macintosh) and then choosing Add ▶ SocketCore ▶ TCPSocket ▶ SSLocket.

The **SSLocket** control can be instantiated via code since it is not a subclass of [Control](#). This allows you to easily write code that does communications without adding the control to a window.

Writing to a socket is done asynchronously. This means each time the Write method is called, the data passed goes into a buffer in memory before actually being sent and then removed from the buffer. Once the socket has finished sending the data in the buffer to the computer at the other end of the socket connection, the SendComplete event handler is executed. This allows you to know when all of the data has really been sent.

Calling Read, ReadAll, or Lookahead may not fetch all of the data in the internal buffer. This is because SSL needs to read data in blocks (due to the cryptography), and it may not have a complete block in the buffer. For example, there may be 700 bytes available in the buffer, but SSL can only decrypt 512 bytes due to the remainder being an incomplete block. What occurs in this case is some data may remain stagnant in the buffer. When more data comes in, REALbasic executes a DataAvailable event. If there are no more DataAvailable events, then upon disconnection, REALbasic will execute additional DataAvailable events to let you pick up any stagnant data that SSL can give us back. There are two things to watch out for because of this:

- 1) If there is not sufficient data for SSL to decrypt, you may get a DataAvailable event but no data.
- 2) Calling **SSLocket.Close** may execute DataAvailable events.

If you are subclassing an **SSLocket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

**See Also** [SocketCore](#), [TCPSocket](#) classes.

---

## StackOverflowException Error

Occurs when the calling chain becomes too long.

**Super Class** [RuntimeException](#)

**Notes** Not surprisingly, a **StackOverflowException** occurs when the stack overflows. This happens when the calling chain gets too long. This can easily happen when your code makes a recursive call without providing a way to terminate the recursion—or the condition that terminates the recursive call takes too many calls to occur.

**Example** The following method calls itself until the stack overflows:

```
Function Square (value as Integer) as Integer
    Return Square(value)
```

## StandardInputStream Class

---

You can handle the function with the following simple [Exception](#) handler:

```
Function Square (value as Integer) as Integer
    Return Square(value)
Exception err
If err IsA StackOverflowException then
    MsgBox "The stack has overflowed!"
end if
```

### See Also

[RuntimeException](#) class; [Function](#), [Raise](#), [Sub](#) statements, [Nil](#) keyword; [Exception](#), [Try](#) blocks.

## StandardInputStream Class

---

Used for reading text input in a [ConsoleApplication](#).

Super Class [Object](#)

### Methods

| Name     | Parameters   | Description   |
|----------|--|---|
| Read     | Count as <a href="#">Integer</a> ,<br>[enc as <a href="#">TextEncoding</a> ] | Returns a <a href="#">String</a> . Reads <i>count</i> bytes from the input stream. The optional parameter <i>enc</i> specifies the encoding of the text.                              |
| ReadLine |  | Returns a <a href="#">String</a> . Reads from the input stream until the user presses the Return or Enter key. ReadLine does not include the Newline character as part of the string. |

### Notes

Since there is only one **StandardInputStream** for the system, you do not need to create a **StandardInputStream** object to use; you just use [StdIn](#).

The **StandardInputStream** class implements the “Readable” class interface. If you implement this interface, you must provide a Read method with the parameters as shown above.

**StandardInputStream** incorporates a conversion operator so that you can use [StdIn](#) as a [TCPSocket](#). This is useful only for services that are started for you by xinetd on Mac OS X or Linux. Here is an example of how to use this:

```
Dim Incoming as TCPSocket = Stdin
```

### See Also

[ConsoleApplication](#), [StandardOutputStream](#) classes; [Input](#), [Print](#), [StdErr](#), [StdIn](#), [StdOut](#) methods; [TargetHasGUI](#) constant.

# StandardOutputStream Class

Used for writing data to the output or error streams in console applications.

**Super Class** [Object](#)

## Methods

| Name      | Parameters                     | Description   |
|-----------|--------------------------------|---|
| Write     | data as <a href="#">String</a> | Writes <i>data</i> to the output stream.  |
| WriteLine | data as <a href="#">String</a> | Writes <i>data</i> to the output stream. WriteLine appends the NewLine character to <i>data</i> . |

**Notes** Since there is only one **StandardOutputStream** for the system, you do not need to create a **StandardOutputStream** object to use; you just use [StdOut](#) or [StdErr](#) for the Error stream.

The **StandardOutputStream** class implements the “Writable” class interface. If you implement this interface, you must provide a Write method with the parameter as shown above.

**StandardOutputStream** incorporates a conversion operator so that you can use [StdOut](#) and [StdErr](#) as [TCPSockets](#). This is useful only for services that are started for you by xinetd on Mac OS X or Linux. Here is an example of how to use this:

```
Dim Outgoing as TCPSocket = StdOut
Dim ErrMsg as TCPSocket = StdErr
```

**See Also** [ConsoleApplication](#) class; [Input](#), [Print](#), [StandardInputStream](#), [StdErr](#), [StdIn](#), [StdOut](#), methods; [TargetHasGUI](#) constant.

---

# StandardToolbarItem Control

Used to create a toolbar in a window. Supported on Mac OS X 10.2 and above only. StandardToolbarItems can be placed in windows for other platforms for layout purposes only.

**Super Class** [Control](#)

## StartupItemsFolder Function

---

### Properties

| Name            | Type                    | Description   |
|-----------------|-------------------------|---|
| ControlOrder    | <a href="#">Integer</a> | Determines the left to right placement of the item in the toolbar.  |
| Enabled         | <a href="#">Boolean</a> | <a href="#">True</a> if the item is enabled.  |
| Handle          | <a href="#">Integer</a> | A ToolbarItemRef, used in <a href="#">Declare</a> statements, of interest only to Macintosh toolbox programmers.  |
| IdentifierOther | <a href="#">String</a>  | A toolbaritem identifier that REALbasic hasn't provided. (e.g. another way to make a separator is to set this property to "com.apple.hitoolbox.toolbar.separator").   |
| Left            | <a href="#">Integer</a> | The distance (pixels) from the left side of the window in the Window Editor. In the runtime application, the ordering of toolbar items is strictly based left to right on control order.  |
| Top             | <a href="#">Integer</a> | The distance (pixels) from the top of the window in the Window Editor. In the runtime application, the ordering of toolbar items is strictly based left to right on control order.  |
| Type            | <a href="#">Integer</a> | Type - One of the Apple pre-defined toolbar types or a custom toolbar type:<br>0 - Other (you need to use IdentifierOther to specify the identifier to use)<br>1 - Separator (cosmetic widget only)<br>2 - Space, separates items by a small space<br>3 - Flexible Space, pushes items after it to be right justified |

### Events

| Name  | Parameters | Description                            |
|-------|------------|--|
| Close |            | Occurs when the item is destroyed      |
| Open  |            | Occurs when the item is first created. |

### Note

If at least one [ToolbarItem](#) or [StandardToolbarItem](#) is on the window, a toolbar will be created, and the toolbar button will be added to the window. Each ToolbarItem will be added in Control Order from Left to Right.

### See Also

[ToolbarItem](#) control.

---

## StartupItemsFolder Function

Used to access the Startup folder.

**Syntax****result=StartupItemsFolder**

| Part   | Type                       | Description  |
|--------|----------------------------|--|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the Startup Items Folder. |

**Notes**

Items in the Startup Items folder are opened automatically when the user starts the computer. If you want your application to give the user the option to automatically open it at system startup, you will need to place it (or an alias to it) in the Startup Items folder.

The **StartupItemsFolder** function provides a way to access the Startup Items folder that will work under different language systems, and will continue to work even if the operating system is reorganized.

The [SpecialFolder](#) module enables you to access many special folders that are managed by the OS.

**Windows**

**StartupItemsFolder** accesses the Startup folder for the current user: \Documents and Settings\user\Start Menu\Programs\Startup.

**Macintosh**

**StartupItemsFolder** access the Startup Items folder on Mac OS “classic” and returns [Nil](#) on Mac OS X.

**Linux**

**StartupItemsFolder** returns [Nil](#) on Linux.

**Example**

This example displays the absolute path to the Startup Items folder, if it exists on the user’s machine.

```
Dim f As FolderItem
f=StartupItemsFolder
if f <> Nil Then
  MsgBox f.AbsolutePath
Else
  MsgBox "The folderItem does not exist."
End if
```

**See Also**

[ApplicationSupportFolder](#), [DesktopFolder](#), [FontsFolder](#), [PreferencesFolder](#), [SystemFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

## Static Statement

Creates a local variable or local array with the name and size (in the case of an array) and data type specified. A variable declared with the **Static** statement and assigned a value retains its value from one invocation of the method to the next. In contrast, variables declared with the [Dim](#) statement are completely local to the method and are destroyed when each invocation of the method goes out of scope.

## Static Statement

---

### Syntax

The **Static** statement has two syntaxes:

#### **Static variableName [,variableNameN] As [New] dataType [= initialValue]**

| Part          | Type                  | Description   |
|---------------|-----------------------|---|
| variableName  | Variable name         | The name of the new variable.   |
| variableNameN | Variable name         | Optional. The names of other variables you wish to create with the same data type. Each name should be separated with a comma.  |
| dataType      | Data type name        | The data type of the variable.  |
| InitialValue  | Same type as dataType | Initial value for variableName. If the datatype is an object that requires instantiation via the <a href="#">New</a> operator, you can do that instead of assigning an initial value. |

#### **Static arrayName(size [,sizeN]) as dataType**

| Part      | Type                    | Description  |
|-----------|-------------------------|--|
| arrayName | Variable name           | The name of the new array.   |
| size      | <a href="#">Integer</a> | The size (number of elements) for the array.   |
| sizeN     | <a href="#">Integer</a> | Optional. The size of the next dimension of the array if you are creating a multi-dimensional array. |
| dataType  | Data type name          | The data type of the array.  |

### Notes

Variables created with the **Static** statement are local in scope. The value of a local variable can be accessed only from within the method. Unlike the [Dim](#) statement, variables created with the **Static** statement retain their values from one invocation of the method or function to the next. In other words, they are *persistent* local variables. In contrast, the values assigned to local variables created by the [Dim](#) statement are lost when the method goes out of scope.

The two syntaxes for **Static** are used for creating variables and arrays, respectively. In the first syntax, the **Static** statement is used to create new variables. A variable is an object stored in RAM that can hold a value. In the second syntax, the **Static** statement is used to create a new array of the size specified. An array is a variable that can contain multiple values that are all the same data type. Passing more than one size parameter creates a multi-dimensional array, with the number of dimensions equal to the number of size parameters passed.

You can optionally provide an initial value to a variable declared with a **Static** statement. If the variable is one of REALbasic's built-in data types ([String](#), [Integer](#),

[Single](#), [Double](#), [Boolean](#), or [Color](#)) you can use the assignment statement to assign the initial value. Here are some examples:

```
Static a as Integer =5  
Static b as Double =87.87  
Static s as String="Howdy"  
Static c as Color = &cFF4B51
```

The following statement initializes three variables to an initial value:

```
Static x, y, z As Integer = 10
```

If the data type is an object, you can optionally instantiate the object with the [New](#) operator in the **Static** statement. This code declares and instantiates a [Date](#) object.

```
Static d as New Date  
MsgBox d.shortdate
```

Otherwise, you need a second line of code to instantiate the variable, *d*, as in the following:

```
Static d as Date  
d=New Date  
MsgBox d.shortdate
```

If the call to the object's constructor takes parameters, you can pass the parameters as well. Here is an example:

```
Static mb as New MemoryBlock(512)
```

The **Static** statement can be placed anywhere in a method or function, including inside a conditional structure (an [If](#) or [Case](#) statement).

If you don't know the size of the array you need at the time you declare it, you can declare it as a null array, i.e., an array with no elements, and use the [Redim](#) command to resize it later. You do this by giving it an index of -1. This means "an array of no elements." For example, the statement:

```
Static aNames (-1) as String
```

creates the array *aNames* with no elements. If your program needs to load a list of names that the user enters, you can wait to size the array until you know how many names the user has entered. You can also accomplish this by leaving out the -1. The following statement is equivalent:

```
Static aNames () as String
```

## StaticText Control

---

When you assign values to an array variable, such as with the [Split](#) function, you don't need to know the number of elements ahead of time.

You can also assign an array to another array. For example, the following is valid:

```
Static myArray() as String
Static newArray() as String
myArray.Append "Anthony"
myArray.Append "Aardvark"
myArray.Append "Accountant"
newArray()=myArray //newArray set equal to myArray
```

### Examples

This example uses the **Static** statement to create an [integer](#) variable called "age" and assigns it the value 33.

```
Static age As Integer
age=33
```

This example uses the **Static** statement to create two [string](#) variables.

```
Static FirstName, LastName As String
```

This example uses the **Static** statement to create an array called aNames with 11 elements (remember, arrays have a zero element).

```
Static aNames(10) As String
```

This example uses the **Static** statement to create an array called aNames with one element.

```
Static aNames(0) As String
```

### See Also

[Append](#), [Insert](#), [Redim](#), [Remove](#), [Sort](#), [Sortwith](#) methods; [UBound](#) function; [Collection](#), [Dictionary](#) classes; [Dim](#) statement.

---

## StaticText Control

Displays text that is not editable by the user.

### Super Class

[RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

## Properties

| Name       | Type                        | Description  |
|------------|-----------------------------|--|
| Bold       | <a href="#">Boolean</a>     | Applies the bold style to the text.  |
| Caption    | <a href="#">String</a>      | The text displayed. Same as the Text property. This property was added for Visual Basic compatibility.   |
| DataField  | <a href="#">String</a>      | Relevant only if the StaticText is used in conjunction with a <a href="#">DataControl</a> to display the contents of fields in a database table. Name of a field in the table referenced by <a href="#">DataControl</a> .  |
| DataSource | <a href="#">DataControl</a> | The <a href="#">DataControl</a> that references a table in a database whose fields are displayed (TableName as <a href="#">String</a> ). If assigned in code, you use the Name property of a <a href="#">DataControl</a> . Use the DataField property to link a field to be displayed to a StaticText. In the IDE, a popup menu of field names will appear. Database fields can also be displayed using <a href="#">EditFields</a> , <a href="#">PopupMenu</a> s and <a href="#">ListBoxes</a> . |
| Italic     | <a href="#">Boolean</a>     | Applies the italic style to the text.  |
| Multiline  | <a href="#">Boolean</a>     | If <a href="#">True</a> , wrap text to multiple lines.   |
| Text       | <a href="#">String</a>      | The text displayed. Same as the Caption property.  |
| TextAlign  | <a href="#">Integer</a>     | The alignment of the text: Use the following class constants:<br>AlignLeft (0): Left alignment<br>AlignCenter1 = Center alignment<br>AlignRight (2): Right alignment   |
| TextColor  | <a href="#">Color</a>       | The color of the text.   |
| TextFont   | <a href="#">String</a>      | Name of the font used to display the text.   |
| TextSize   | <a href="#">Integer</a>     | Size of the font used to display the text.   |
| Underline  | <a href="#">Boolean</a>     | Applies the underline style to the text.   |

## Events

| Event          | Parameters   | Description  |
|----------------|--|--|
| AcceleratorKey |  | The accelerator key for the <b>StaticText</b> was pressed. This event should call the SetFocus event of the next control in the Tab order. This is usually the control to its right or below.                                  |
| MouseDown      | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the <b>StaticText</b> region at the location passed in x and y. <a href="#">Return True</a> if you are going to handle the MouseDown.  |
| MouseDrag      | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was pressed inside the <b>StaticText</b> and moved (dragged) at the location local to the control passed to x and y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |

## StdErr Method

---

| Event   | Parameters   | Description  |
|---------|--|--|
| MouseUp | x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | The mouse button was released inside the <b>StaticText</b> region at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event. |

### Notes

You can define an Accelerator key for a **StaticText**. In the Text property, precede the accelerator character with an ampersand. For example, the entry “&Home” signifies that the H is the accelerator. Pressing Alt+H on Windows triggers the AcceleratorKey event.

### See Also

[RectControl](#) class; [EditField](#) control.

---

## StdErr Method

Retrieves the global reference to the standard error stream, which is typically the same as the standard output stream. This is for [ConsoleApplications](#) only.

### Notes

The **StdErr** method implements the [Writable](#) class interface. If you implement this interface, you must implement the specified methods and parameters.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

### Syntax

**result=StdOut**

| Part   | Type                                 | Description   |
|--------|--------------------------------------|---|
| result | <a href="#">StandardOutputStream</a> | The standard error stream is typically the Terminal application and is of type <a href="#">StandardOutputStream</a> . |

### See Also

[ConsoleApplication](#), [StandardInputStream](#), [StandardOutputStream](#) classes;  
[TargetHasGUI](#) constant; [Input](#), [Print](#), [StdIn](#), [StdErr](#) methods; [Writable](#) class interface.

---

## StdIn Method

Retrieves the global reference to the standard input stream, used in [ConsoleApplications](#).

**Syntax****result=StdIn**

| Part   | Type                                | Description  |
|--------|-------------------------------------|--|
| result | <a href="#">StandardInputStream</a> | The standard input stream is typically the keyboard as used for the Terminal application. It could be a file, serial control, socket or other such input device on some systems. |

**Notes**

The **Stdin** method implements the [Readable](#) class interface. If you implement this interface, you must implement the methods and parameters specified by the class interface.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

**See Also**

[ConsoleApplication](#), [StandardInputStream](#), [StandardOutputStream](#) classes; [TargetHasGUI](#) constant; [Input](#), [Print](#), [StdOut](#), [StdErr](#) methods; [Readable](#) class interface.

## StdOut Method

Retrieves the global reference to the standard output stream in [ConsoleApplications](#), which is typically the terminal window.

**Syntax****result=StdOut**

| Part   | Type                                 | Description   |
|--------|--------------------------------------|---|
| result | <a href="#">StandardOutputStream</a> | The standard output stream is typically the Terminal application. |

**Notes**

The **StdOut** method implements the [Writable](#) class interface. If you implement this interface, you must implement the specified methods and parameters. For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

**See Also**

[ConsoleApplication](#), [StandardInputStream](#), [StandardOutputStream](#) classes; [TargetHasGUI](#) constant; [Input](#), [Print](#), [StdIn](#), [StdErr](#) methods; [Writable](#) class interface.

## Str Function

Returns the [string](#) form of the value passed.

## StrComp Function

---

### Syntax

**result=Str(number)**

| Part   | Type   | Description  |
|--------|--|--|
| result | <a href="#">String</a>   | The <a href="#">string</a> version of the number passed. |
| number | <a href="#">Integer</a> , <a href="#">Single</a> , or <a href="#">Double</a> | Any numeric expression.                                  |

### Notes

Use the [Format](#) function when you want to convert numeric values into formatted strings such as dates, times, currency etc.

**Str** returns seven (7) significant digits. It uses scientific notation when it needs to return more than 6 digits. **Str**(123456789) returns 1.234568e+8.

**Str** assumes the period (.) is the decimal separator. If your application needs to recognize other decimal separators, use the [CStr](#) function.

### Examples

This example uses the **Str** function to return the [string](#) form of several numbers.

```
Dim s As String  
s=Str(123) //returns "123"  
s=Str(-123.44) //returns "-123.44"  
s=Str(123.0045) //returns "123.0045"  
Const Pi=3.14159265358979323846264338327950  
s= Str(pi) // returns "3.141593"
```

### See Also

[CDbl](#), [CStr](#), [Format](#), [Val](#) functions.

## StrComp Function

Makes a binary (case-sensitive) or text (lexicographic) comparison of the two [strings](#) passed and returns the result.

### Syntax

**result=StrComp(string1, string2, mode)**

OR

**result=stringVariable.StrComp(string2, mode)**

| Part    | Type                    | Description  |
|---------|-------------------------|--|
| result  | <a href="#">Integer</a> | If string1 < string2 the function returns -1<br>If string1 = string2 the function returns 0<br>If string1 > string2 the function returns 1 |
| string1 | <a href="#">String</a>  | The first string for comparison.   |
| string2 | <a href="#">String</a>  | The second string for comparison.  |

| Part           | Type                    | Description   |
|----------------|-------------------------|---|
| mode           | <a href="#">Integer</a> | The comparison mode:<br>0=binary (case-sensitive)<br>1=text (lexicographic) |
| stringVariable | <a href="#">String</a>  | Any variable of type <a href="#">String</a> .                               |

**Notes**

The text comparison is lexicographic, not case-insensitive. This means that case will be taken into account if the strings are the same (other than case).

When using Mode 0 (binary), **StrComp** always compares the bytes of the strings as they are, doing no encoding conversions. It's possible that two strings might look the same on screen but be reported as different in **StrComp** Mode 0 if they are in different encodings (and, therefore, contain different bytes).

**Examples**

The following example returns -1 because the two strings are the same in every way except in case.

```
StrComp("Spam", "spam", 1)
```

The following example returns -1 because in a text comparison of the two strings, string2 is greater than string1. The ASCII value of "s" is greater than the ASCII value of "S".

```
StrComp("Spam", "spam", 0)
```

**See Also**

[InStr](#) function; [>](#), [>=](#), [<](#), [<=](#), [=](#), [<>](#) operators.

## String Data Type

A **String** is an intrinsic data type in REALbasic. It is a series of numeric or alphabetic characters enclosed in quotes. Any kind of alphabetic or numeric information can be stored as a **string**. "Jean Marie", "3/17/98", "45.90" are all examples of strings. When you place quotes around information in your code, you are telling REALbasic to look at the data as just a series of characters and nothing more. The maximum length of a **string** is limited only by available memory. The default value of a String is "".

**Notes**

All computers use encoding schemes to store character strings as a series of bytes. The oldest and most familiar encoding scheme is the ASCII encoding. It defines character codes only for values 0-127. These values include only the upper and lowercase English alphabet, numbers, some symbols, and invisible control codes used in early computers.

Many extensions to ASCII have been introduced which handle additional symbols, accented characters, non-Roman alphabets, and so forth. In particular, the Unicode encoding is designed to handle any language and a mixture of languages in the same

## StringProvider Class Interface

---

string. REALbasic supports two different Unicode formats, UTF-8 and UTF-16. All of your constants, string literals, and so forth are stored internally using UTF-8 encoding.

If the strings you work with are created, saved, and read within REALbasic, you shouldn't have to worry about encoding issues because REALbasic stores the encoding it uses along with the content of the string.

Since character codes above 127 represent different characters in different encoding schemes, it is important that you understand the encoding that is used for strings that were generated outside of REALbasic. When you read in a text string using the [TextInputStream](#) class, set the Encoding property to the correct encoding or use the optional Encoding parameter of the Read, ReadLine, or ReadAll methods. You can determine the encoding of a string using the [Encoding](#) function.

If you need to save string data in a particular encoding, use the [ConvertEncoding](#) function. Specify the desired encoding using the [Encodings](#) object.

If you need to get a character that corresponds to the value of a character code, use the [Chr](#) function only if it is an ASCII code. In other cases, it is best to use the Chr method of the [TextEncoding](#) class. This enables you to specify both the encoding scheme and the desired character code.

REALbasic includes an extensive library of functions for comparing and manipulating strings, which are listed in the See Also section.

The [VarType](#) function returns 8 when passed a [String](#).

### Example

```
Dim s as String  
s="How now brown cow"
```

### See Also

[+, <, <=, <>, =, >, >=, Asc, AscB, Chr, ChrB, ConvertEncoding, CStr, Encoding, InStr, InStrB, IsNumeric, Left, LeftB, Len, LenB, Lowercase, LTrim, Mid, MidB, NthField, Replace, ReplaceAll, Right, RightB, RTrim, Str, StrComp, Titlecase, Trim, Uppercase, Val, VarType](#) functions; [Dim, Static](#) statements; [Encodings](#) object; [TextEncoding, TextInputStream](#) classes; [Boolean, Color, Double, Integer, Single, Variant](#) data types.

---

## StringProvider Class Interface

Used to program custom object bindings.

### Methods

| Name      | Parameters | Description                        |
|-----------|------------|------------------------------------|
| getString |            | Returns a <a href="#">String</a> . |

# StringShape Class

Draws a text string in a vector graphics environment.

**Super Class** [Object2D](#)

## Properties

| Name             | Type                    | Description   |
|------------------|-------------------------|---|
| <b>Bold</b>      | <a href="#">Boolean</a> | Set to <a href="#">True</a> to use the Bold style.  |
| <b>Italic</b>    | <a href="#">Boolean</a> | Set to <a href="#">True</a> to use the Italic style.  |
| <b>Text</b>      | <a href="#">String</a>  | The text to draw. The text can be only one line and in one text style (font, font size, style) specified by the other StringShape properties. |
| <b>TextFont</b>  | <a href="#">String</a>  | Name of the font to use.  |
| <b>FontSize</b>  | <a href="#">Double</a>  | Font Size in points.  |
| <b>Underline</b> | <a href="#">Boolean</a> | Set to <a href="#">True</a> to use the Underline style.   |

## Notes

The X,Y properties specify the center of the string's baseline. Strings that contain line breaks are not supported. **StringShapes** can be rotated, but doing so is memory intensive, especially for large strings.

## Example

This example draws a line of text when the user presses the mouse button. Place the following code in the MouseDown event of a [window](#).

```
Dim s as New StringShape
s.Text="Hello World"
s.TextFont="Helvetica"
s.Bold=True
Graphics.DrawObject s,x,y
```

## See Also

[ArcShape](#), [CurveShape](#), [FigureShape](#), [FolderItem](#), [Group2D](#), [Graphics](#), [Object2D](#), [OvalShape](#), [Picture](#), [PixmapShape](#), [RectShape](#), [RoundRectShape](#) classes.

# StyledText Class

Used to managed styled text independently of its display in an [EditField](#). For an [EditField](#), you can access the properties and methods of this class via the StyledText property of that class.

**Super Class** [Object](#)

## StyledText Class

---

**Properties** None

## Methods

| Name               | Parameters  | Description   |
|--------------------|---|---|
| AppendStyleRun     | Run as <a href="#">StyleRun</a>                                       | Appends the <a href="#">StyleRun</a> passed to the end of <i>Text</i> .   |
| Bold               | Start as <a href="#">Integer</a><br>Length as <a href="#">Integer</a> | Gets or sets the Bold style to the selected text in <i>Text</i> . <i>Start</i> is the starting position in <i>Text</i> . <i>Start</i> starts at zero. The default value for <i>Length</i> is 1. Returns a <a href="#">Boolean</a> . Use assignment syntax to set the value.   |
| Font               | Start as <a href="#">Integer</a><br>Length as <a href="#">Integer</a> | Gets or sets the font for the selected text in <i>Text</i> . <i>Start</i> is the starting position in <i>Text</i> . The default value for <i>Length</i> is 1. Returns a <a href="#">String</a> . Use assignment syntax to set the value.  |
| InsertStyleRun     | Run as <a href="#">StyleRun</a><br>Index as <a href="#">Integer</a>   | Inserts the <a href="#">StyleRun</a> passed at the insertion point designated by <i>Index</i> . <i>Index</i> is zero-based. You can get the total number of StyleRuns in <i>Text</i> from the <a href="#">StyleRunCount</a> method.   |
| Italic             | Start as <a href="#">Integer</a><br>Length as <a href="#">Integer</a> | Gets or sets the Italic style to the selected text in <i>Text</i> . <i>Start</i> is the starting position in <i>Text</i> . The default value for <i>Length</i> is 1. Returns a <a href="#">Boolean</a> . Use assignment syntax to set the value.  |
| Paragraph          | Index as <a href="#">Integer</a>                                      | Returns as a <a href="#">Paragraph</a> the paragraph in <i>Text</i> indicated by <i>Index</i> . <i>Index</i> is zero-based. The properties of the <a href="#">Paragraph</a> class enable you to access its position and alignment.  |
| ParagraphAlignment | Index as <a href="#">Integer</a>                                      | Aligns the paragraph indicated by <i>Index</i> . <i>Index</i> is zero-based. Assign an alignment to ParagraphAlignment using one of the <a href="#">EditField's</a> or the <a href="#">Paragraph's</a> class constants. The constants are:<br>AlignDefault (0): Default alignment<br>AlignLeft (1): Left aligned<br>AlignCenter (2): Centered<br>AlignRight (3): Right aligned<br>See the example for an illustration of this syntax.<br>Not supported with Linux EditFields. |
| ParagraphCount     |   | Returns as an <a href="#">Integer</a> the number of paragraphs in <i>Text</i> .   |
| RemoveStyleRun     | Index as <a href="#">Integer</a>                                      | Removes the <a href="#">StyleRun</a> indicated by <i>Index</i> . <i>Index</i> is zero-based. You can get the total number of StyleRuns in <i>Text</i> from the <a href="#">StyleRunCount</a> method.  |

| Name          | Parameters  | Description  |
|---------------|---|--|
| Size          | Start as <a href="#">Integer</a><br>Length as <a href="#">Integer</a> | Gets or sets the font size for the selected text in <i>Text</i> . <i>Start</i> is the starting position in <i>Text</i> . The default value for <i>Length</i> is 1. Returns an <a href="#">Integer</a> . Use assignment syntax to set the value.  |
| StyleRun      | Index as <a href="#">Integer</a>                                      | Returns a <a href="#">StyleRun</a> specified by <i>Index</i> . <i>Index</i> is zero-based.   |
| StyleRunCount |   | Returns as an <a href="#">Integer</a> the number of style runs in <i>Text</i> . Use this in conjunction with the StyleRun and StyleRunRange methods to get information about each style run.   |
| StyleRunRange | Index as <a href="#">Integer</a>                                      | Returns a <a href="#">Range</a> for the specified <a href="#">StyleRun</a> . <i>Index</i> is zero-based. Properties of the <a href="#">Range</a> class give you access to the <a href="#">StyleRun</a> 's starting and ending positions, and length.   |
| Text          |   | The text whose style attributes are managed via the instance of this class. Returns a <a href="#">String</a> . The <i>Start</i> and <i>Index</i> parameters of the methods of this class are zero-based and referenced from the start of <i>Text</i> . Set the value of <i>Text</i> using the assignment operator. |
| TextColor     | Start as <a href="#">Integer</a><br>Length as <a href="#">Integer</a> | Gets or sets the text color for the selected text in <i>Text</i> . <i>Start</i> is the starting position in <i>Text</i> . The default value for <i>Length</i> is 1. Returns a <a href="#">Color</a> . Use assignment syntax to set the value.  |
| Underline     | Start as <a href="#">Integer</a><br>Length as <a href="#">Integer</a> | Gets or sets the Underline style to the selected text in <i>Text</i> . <i>Start</i> is the starting position in <i>Text</i> . The default value for <i>Length</i> is 1. Returns a <a href="#">Boolean</a> . Use assignment syntax to set the value.  |

## Notes

The **StyledText** class enables you to apply style attributes to text independently of an [EditField](#) and its properties. Also, the **StyledText** class gives you the ability to align paragraphs (left, center, and right). The properties and methods of the **StyledText** class can be accessed from an [EditField](#) via its **StyledText** property, but you can also open, manage, and save styled text that is not displayed in any control.

The only operations that will update the contents of the [EditField](#) are the methods of the **StyledText** class. This means that obtaining a [StyleRun](#) (using **StyleRun()**) and setting the style attributes for that run will not be reflected in the [EditField](#). If you want to operate on a [StyleRun](#), you must remove the old run and replace it with the new run. Of course, the easier way to do it would be to make your changes using the **StyledText** methods instead of the [StyleRun](#) properties.

Because the [EditField](#) already has selection style attributes, the **StyledText** class honors that information. This means that you can set the style information using the selection methods that are available within the [EditField](#) class, and they will be reflected when

getting [StyleRun](#) information (and vice versa). Not all of the style attributes for selections in the [EditField](#) class are available in the [StyleRun](#) class (such as Outline, Condense, Shadow, Extend and Plain). Of course, the first four are supported only on Mac OS “classic”.

### Example

The following example uses the methods of the **StyledText** class to mark up and align text and display it in an [EditField](#).

```
Dim Text as String // text to be displayed in EditField
Dim st,ln as Integer // start and length values of a paragraph
//define four paragraphs in Text
Text = "This is the text that we are going to save " _
+ "into our file from the EditField." +EndofLine _
+ "Isn't that interesting?" +EndofLine _
+ "Man, I sure do love using RB to take care of my projects." +EndofLine _
+ "That's because the RS Engineering staff is just so awesome!" +EndofLine
EditField1.StyledText.Text = text //four paragraphs in Text

EditField1.StyledText.Bold(5, 2) = True
EditField1.StyledText.TextColor(5, 2) = &cFF0000 //bold and red

EditField1.StyledText.Size( 7, 10) = 16
EditField1.StyledText.Underline(12, 4) = True //16 pt underline

EditField1.StyledText.Size(100, 4) = 18
EditField1.StyledText.TextColor(100, 4) = &cFF00FF //18 pt and Magenta

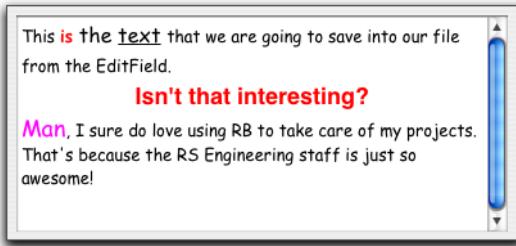
EditField1.StyledText.Font(0, Len(text) ) = "Comic Sans MS"

//center aliign second paragraph
EditField1.StyledText.ParagraphAlignment(1)=Paragraph.AlignCenter

//set this paragraph in Helvetica, 18 pt bold, red
//first get the start and length values for this paragraph...
st=EditField1.StyledText.Paragraph(1).StartPos
ln=EditField1.StyledText.Paragraph(1).Length+1

//next apply attributes...
EditField1.StyledText.Bold(st,ln)=True
EditField1.StyledText.Font(st,ln)="Helvetica"
EditField1.StyledText.Size(st,ln)=18
EditField1.StyledText.TextColor(st,ln)=&cFF0000
```

The resulting [EditField](#) looks like this:



**See Also** [EditField](#), [StyleRun](#), [Paragraph](#), [Range](#) classes.

## StyledTextPrinter Class

Used to print styled text stored in an [EditField](#).

**Super Class** [Object](#)

### Properties

| Property | Type                    | Description   |
|----------|-------------------------|---|
| EOF      | <a href="#">Boolean</a> | True when there is no more text to print in the <a href="#">EditField</a> . Use it to determine when to stop calling DrawBlock. See examples. |
| Width    | <a href="#">Integer</a> | Width of the text to be printed.  |

### Methods

| Method    | Parameters  | Description   |
|-----------|---|---|
| DrawBlock | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a><br>Height as <a href="#">Integer</a> | Used to print the styled text. x and y are coordinates of the point at which you want to begin printing the text. x and y are offset from the top-left corner of the printable area as defined in Page Setup. |

### Examples

The following example prints the styled text in an [EditField](#). The EditField's MultiLine and Styled properties must be set to [True](#) to accept and print styled text.

```
Dim stp as StyledTextPrinter
Dim g as Graphics
g=OpenPrinterDialog()
If g <> Nil then
    stp=EditField1.StyledTextPrinter(g,72*7.5)
    stp.DrawBlock 0,0,72*10
End if
```

This example prints the contents of EditField1 in two columns with a quarter-inch of spacing between them. The EOF property is used to determine if all the text has been printed. When EOF is `False`, it prints the second column or prints another page.

```
Dim g as Graphics
Dim stp as StyledTextPrinter
Dim ColumnWidth, SpaceBetweenColumns, PageHeight as Integer
Dim ColumnToPrint as Integer

ColumnWidth = 261
//7.5 inches minus 1/4 inch for space
//divided by 2
SpaceBetweenColumns = 18
//18 pixels is 1/4 inch
PageHeight = 10 * 72
//10 inches * 72 pixels per inch

g=OpenPrinterDialog\(\)
If g <> Nil then
    stp=EditField1.StyledTextPrinter(g,540)
    stp.width = ColumnWidth

ColumnToPrint = 1
Do until stp.eof
    stp.DrawBlock (columnWidth+SpaceBetweenColumns)*
        (columnToPrint-1),0,PageHeight
    If ColumnToPrint = 2 then //printing last column
        If Not stp.eof then // more text to print
            g.nextPage
            ColumnToPrint = 1
        End if
    Else // more columns to print on this page
        ColumnToPrint = columnToPrint + 1
    End if
Loop
End if
```

### See Also

[EditField](#) control; [OpenPrinter](#), [OpenPrinterDialog](#) functions; [Graphics](#), [PrinterSetup](#) classes.

---

## StyleRun Class

Used to manage a style run within [StyledText](#).

### Super Class [Object](#)

## Properties

| Name      | Type                    | Description                                       |
|-----------|-------------------------|---|
| Bold      | <a href="#">Boolean</a> | Gets or sets the Bold style.                      |
| Font      | <a href="#">String</a>  | Gets or sets the Font of the StyleRun.            |
| Italic    | <a href="#">Boolean</a> | Gets or sets the Italic style.                    |
| Size      | <a href="#">Integer</a> | Gets or sets the font size of the StyleRun.       |
| Text      | <a href="#">String</a>  | Gets or sets the text that makes up the StyleRun. |
| TextColor | <a href="#">Color</a>   | Gets or sets the color of the text.               |
| Underline | <a href="#">Boolean</a> | Gets or sets the Underline style.                 |

## Notes

A **StyleRun** is a series of consecutive characters that have the same style attributes, as indicated by the **StyleRun**'s properties. The Text property of a [StyledText](#) object is a series of StyleRuns.

You can get the total number of StyleRuns via the StyleRunCount method of the [StyledText](#) class and reference a **StyleRun** via the StyleRun method. You can get the starting and ending character positions (and, therefore, the length) of a StyleRun via the StyleRunRange method. The [Range](#) class gives you access to the **StyleRun's** start position, length, and end position.

If you are displaying the styled text in an [EditField](#), changes to a StyleRun's properties are not reflected in the [EditField](#). If you want to operate on a StyleRun, you must remove the old run and replace it with the new run. Of course, the easier way to update styled text in an EditField is to make your changes using the [StyledText](#) methods instead of the **StyleRun** properties.

- Example** The following example saves styled text displayed in an [EditField](#) as a series of StyleRuns that are appended to one another.

```
Dim out as BinaryStream
out = DesktopFolder.Child("TestFile").CreateBinaryFile(" ")

//If we couldn't create the file, then bail out
If out = Nil then Return

//Now we want to loop over all the StyleRuns
//and dump them out to the file.
Dim mb as MemoryBlock
Dim sr as StyleRun
Dim i, Count as Integer
Dim TextLen, FontLen as Integer
Dim StaticInfo as Integer

//We already know that a style run takes up a certain amount of space
//3 bytes for the booleans, 2 for size and 4 for color
staticInfo = 1 + 1 + 1 + 2 + 4

//Get the number of styles
count = EditField1.StyledText.StyleRunCount
For i = 0 to Count-1 //get the StyleRuns
    sr = EditField1.StyledText.StyleRun( i )

    textLen = Len(sr.Text) + 1
    fontLen = Len(sr.Font) + 1

    //Stuff the style run content and style info into a memory block
    mb = New MemoryBlock( staticInfo + fontLen + textLen )
    mb.BooleanValue(0) = sr.Bold
    mb.BooleanValue(1) = sr.Italic
    mb.BooleanValue(2) = sr.Underline
    mb.Short(3) = sr.Size
    mb.ColorValue(5, 32) = sr.TextColor
    mb.CString(9) = sr.Font
    mb.CString(9 + fontLen) = sr.Text

    //Now we can write that information to the file
    out.Write(mb)
    Next

    out.Close //close the file
```

- See Also** [EditField](#), [Paragraph](#), [Range](#), [StyledText](#) classes.

# Sub Statement

Declares the name, parameters, and code that form the body of a subroutine (method).

## Syntax

**Sub** *name*(*[parameterList]*)

**[local variable declarations]**

**[statements]**

**[Return]**

**[statements]**

**[exception handlers]**

**[Finally]**

| Part                 | Description   |
|----------------------|---|
| <b>name</b>          | Required. The name of the subroutine (method); follows standard variable naming conventions.  |
| <b>parameterList</b> | Optional. List of values representing parameters that are passed to the Subroutine when it's called. Multiple parameters are separated by commas. |

## Notes

The **Sub** statement is used to define a method. The word **Sub** is used because a subroutine is the equivalent of a method in the BASIC language. Methods are usually associated with an object (exceptions are global methods or functions that are part of a module). All executable code must be in a **Sub** or [Function](#) statement. A **Sub** can not be defined inside another **Sub** or [Function](#). A **Sub** executes each line of code from the top down, assuming that you have not used the [GoTo](#) statement. Once the last line of code is executed, control returns to the line following the statement that called the **Sub**.

You call a **Sub** by using its name followed by any parameters in parentheses.

Variables used in a **Sub** can be global, public, protected, private, or local in scope. Global variables can be accessed from within any **Sub** or [Function](#). They exist from the moment the application runs to the time it quits. Global variables are created by creating properties of a module and declaring their scope Global. Public variables (a.k.a. properties) work like global properties except that the name of the owning window, class, or module must be used when referring to them, e.g., "module1.publicProperty". Private and protected properties are available only within the owning object and are called using the name of the owning window, class, or module.

Local variables are variables that are created each time the **Sub** is run and destroyed when the **Sub** finishes. Consequently, they can only be accessed by the statements within the **Sub**. They are created by using the [Dim](#) statement from within a **Sub** or [Function](#). A [Dim](#) statement can appear anywhere within the **Sub**.

The [Return](#) statement can be used to immediately return control to the statement that called the **Sub**.

Exception handlers are statements that handle runtime errors. See the [RuntimeException](#) class and the [Exception](#) and [Try](#) blocks for more information.

Sometimes a method needs to do some cleanup work whether it is finishing normally or aborting early because of an exception. The optional [Finally](#) block at the end of the method runs after the exception handlers, if it has any. Code in this block will be executed even if an exception has occurred, whether the exception was handled or not.

If you need your method to return a value, you will want to declare it as a [Function](#). A [Function](#) is a method that can return a value. The value can be a one-dimensional array. See the [Function](#) statement for more information.

By default, parameter passing is done by value and the value cannot be modified by the method. You can also pass a parameter by reference. When you pass a parameter by reference, a pointer to the object is passed. This allows you to return a value using the parameter.

If you want to pass a parameter by reference, precede it by the keyword '[ByRef](#)' in the parameter list, e.g., '[ByRef](#) MyInt as [Integer](#)).

A **Sub** method can call itself resulting in recursion. Too much recursion can lead to [StackOverFlowException](#) errors.

### See Also

[Break](#), [Catch](#), [Exit](#), [Finally](#), [Function](#), [Return](#) statements; [RuntimeException](#) class; [Exception](#), [Try](#) blocks.

---

## Super Cannot be used in a Module Method Error

The concept of Super applies only to the Class hierarchy, so it has no meaning in a method of a module.

---

## Syntax Error

Any number of miscellaneous errors that were not identified by more specific error messages has occurred.

**Examples** An omitted closing parenthesis:

```
If (d > 0 then
.
end if
```

An incorrect Dim statement:

```
Dim d Double //omitting 'as'
Dim f //omitting 'as' and data type
Dim As Boolean //no variable name
Dim Double as Double //'Double' is a reserved word
Dim a b as Integer //should be 'a,b'
Dim c as //type missing
Dim myFlag as True //illegal type and use of reserved word
```

The 'Then' keyword is missing from an If statement:

```
Dim a as Integer
a=5
If a >0 //needs to be If a>0 then
    MsgBox "Boo"
end if
```

The Elseif statement was not preceded by an If statement.

```
Dim i,j,k as Integer
// do something
elseif j>0 then
//do something else
end if

//Correct
If j=0 then
.
Elseif j>0 then
.
End if
```

The End If keyword is missing from an If statement:

```
Dim dayNum as Integer
if dayNum=1 then
    MsgBox "It's Monday"
// end if needed
```

## Syntax Error

---

#else used instead of Else with a [#If](#) statement (conditional compilation).

```
Dim c as Color
#If TargetMacOS then
  b=SelectColor( c , "Select a Color")
else // should be #else
  Beep
#EndIf
```

The Next keyword is missing from a [For](#) loop:

```
Dim i,j as Integer
Dim alnts(2,2) as Integer
For i=0 to 2
  For j=0 to 2
    alnts(i,j)=i*j
  Next
//final 'Next' missing here
```

One too many Next keywords are used in this example:

```
Dim i,j as Integer
Dim alnts(2,2) As Integer
For i=0 to 2
  For j=0 to 2
    alnts(i,j)=i*j
  Next
Next
Next //too many "nexts"
```

Omitting the Wend keyword from a [While](#) loop:

```
While i<10
  Beep
  Next //should be Wend
```

Omitting the [While](#) statement:

```
Dim i as Integer
i=1
//While statement missing here
  Beep
  i=i+1
Wend
```

The Loop keyword is missing from a [Do](#) statement:

```
Dim i as Integer
Do until i=5
Beep
i=i+1
//missing Loop statement
```

A Loop keyword was used without a preceding (matching) [Do](#) statement

```
Dim x as Integer
// Do statement missing
x=x+1
Loop Until x<100
```

The End Select keyword is missing from a [Select Case](#) statement:

```
Select case x
case 1
MsgBox "The party of the first part."
case 2
MsgBox "The party of the second part."
//no matching End Select statement
```

The [Select Case](#) statement is missing:

```
Dim Day as String
case 1 //Select Case statement missing
Day = "Monday"
case 2
Day = "Tuesday"
end select
```

The [Sub](#) and [Function](#) statements cannot appear inside a method.

```
Sub myMethod(x as Integer,y as Integer)
Sub myMethod (x as Integer,y as Integer)
```

A comma indicates that the second, required parameter is missing:

```
ListBox1.insertRow 2,
```

Using an equals sign when it is not necessary.

```
Return=x*y //equals sign should be a space
```

## System Object

---

A keyword was misspelled:

```
If dayNum=1 then  
  MsgBox "It's Monday"  
end fi //should be end if
```

A space rather than a comma is used to separate variable names in a [Dim](#) statement:

```
Dim a b as Integer //should be 'a,b'
```

---

# System Object

The properties of a **System** object are used to get information about the user's computer.

Super Class [Object](#)

## Properties

| Name                          | Type                        | Description   |
|-------------------------------|-----------------------------|---|
| <a href="#">AddressBook</a>   | <a href="#">AddressBook</a> | Returns the user's Address Book (Mac OS X only).  |
| <a href="#">CommandLine</a>   | <a href="#">String</a>      | This property contains the parameters passed to the application. For example, if the application "Foo.exe" was executed with the call "Foo.exe a b c", then CommandLine would contain "Foo.exe a b c". This is valid for Windows, Mac OS X built using the Mach-O format, and Linux applications.   |
| <a href="#">Cursors</a>       | <a href="#">MouseCursor</a> | Provides access to the cursors in the <a href="#">Cursors</a> object. Use these cursors to change the current mouse pointer to any of the standard cursor shapes.   |
| <a href="#">Keychain</a>      | <a href="#">Keychain</a>    | (Mac OS X) Returns the default <a href="#">Keychain</a> . You can assign the default <a href="#">Keychain</a> by setting this to another <a href="#">Keychain</a> . If no <a href="#">Keychain</a> exists, it will try to create one. If you don't want it to create one, you should first call System.KeyChainCount and check against 0. |
| <a href="#">KeyChainCount</a> | <a href="#">Integer</a>     | (Mac OS X) Returns the number of available <a href="#">Keychains</a> . This number includes all <a href="#">Keychains</a> within the Keychains folder as well as any other <a href="#">Keychains</a> known to the <a href="#">Keychain</a> 's manager.  |

| Name                  | Type                      | Description   |
|-----------------------|---------------------------|---|
| Keyscript             | <a href="#">Integer</a>   | Allows you to get and set the keyboard script. The values are the values for script systems, e.g., 0 for MacRoman, 1 for MacJapanese, etc. In addition, you can assign negative numbers to get various special effects, e.g., assign -1 to switch to the next layout in the Keyboard menu. For more information, see: <a href="http://developer.apple.com/techpubs/mac/Text-386.html">http://developer.apple.com/techpubs/mac/Text-386.html</a> |
| MouseDown             | <a href="#">Boolean</a>   | <a href="#">True</a> if the mouse is down.  |
| MouseX                | <a href="#">Integer</a>   | The X coordinate of the mouse (pixels). Measured from the top-left corner of the screen.  |
| MouseY                | <a href="#">Integer</a>   | The Y coordinate of the mouse (pixels). Measured from the top-left corner of the screen.  |
| Network               | <a href="#">Network</a>   | Provides access to the methods of the <a href="#">Network</a> object. They enable you to verify whether the computer is connected to a network and, if so, to look up DNS and IP addresses.   |
| NetworkInterfaceCount | <a href="#">Integer</a>   | Returns the number of network interface cards that are installed in the user's computer.  |
| PPPStatus             | <a href="#">Integer</a>   | Provides the current status of the PPP connection. Please refer to the table in the section "PPPStatus values" for a list of the PPPStatus constants.   |
| QuickTime             | <a href="#">QuickTime</a> | (Windows and Macintosh) Provides access to the <a href="#">QuickTime</a> object which returns certain properties of the QuickTime software installed on the user's computer.  |
| SerialPortCount       | <a href="#">Integer</a>   | Number of serial ports available.   |

## Methods

| Name     | Parameters                    | Description   |
|----------|-------------------------------|---|
| DebugLog | msg as <a href="#">String</a> | Writes <i>msg</i> in the following ways, depending on your platform. On Windows, it logs to the debugger, so programs like DebugView can be used to view the string. On Macintosh, it uses DebugStr; it will drop you into MacsBug on Mac OS Classic and log to the Console on Mac OS X. On Linux, it prints the message to StdErr. |

## System Object

---

| Name                | Parameters  | Description   |
|---------------------|---|---|
| EnvironmentVariable | Name as <a href="#">String</a>  | Used to get and set Mac OS X and Linux environment variables. <i>Name</i> is case-sensitive. Returns a <a href="#">String</a> .<br>For example,<br><code>System.EnvironmentVariable("HOME")</code> gets the value of the HOME environment variable, the path to the user's home directory.<br><code>System.EnvironmentVariable("HOME") = "/Users/Joe"</code><br>sets the path to the Home directory.<br>To get the list of environment variables and their values (Mac OS X and Linux), execute the "env" command from a terminal window or via the <a href="#">Shell</a> class. On Windows, use the "set" command instead of "env". See the Examples section.    |
| Gestalt             | Code as <a href="#">String</a><br><a href="#">ByRef</a> Result as <a href="#">Integer</a> | Macintosh only. Returns characteristic of system specified by Code. Values of Code are found in Inside Macintosh. Returns a <a href="#">Boolean</a> , which is <a href="#">True</a> if successful.  |
| GetNetworkInterface | [Index as <a href="#">Integer</a> ]   | Returns a <a href="#">NetworkInterface</a> object. Use the <a href="#">NetworkInterface</a> object to obtain networking information about the user's computer. Use the optional parameter to specify the network interface card. <i>Index</i> is zero-based. NetworkInterfaceCount gives you the number of network interfaces in the computer.  |
| IsFunctionAvailable | FunctionName as <a href="#">String</a> ,<br>LibraryName as <a href="#">String</a>         | Returns a <a href="#">Boolean</a> . IsFunctionAvailable will attempt to load the passed function from the specified library. If the function can be loaded, IsFunctionAvailable returns <a href="#">True</a> . If the operation fails, it returns <a href="#">False</a> . Calling IsFunctionAvailable or using the Soft keyword in a <a href="#">Declare</a> statement will cache both the library loading code as well as the function itself. Multiple calls to the same function will suffer almost no overhead costs in comparison to regular ("hard") Declares. See the description of Soft Declares in the notes for the <a href="#">Declare</a> statement. |

| Name            | Parameters   | Description  |
|-----------------|--|--|
| Log             | Level as <a href="#">Integer</a> , msg as <a href="#">String</a> | Writes <i>msg</i> to the logging mechanism. On Mac OS X and Linux, it writes to the system logger, which is typically at /var/logs. On Windows NT/XP it writes to the Event Logger, which can be viewed by the Event Viewer. On Mac OS 9 and Windows 98, REALbasic creates a log file on the user's desktop. Use the LogLevel class constants to set the value of <i>Level</i> . See the section "LogLevel Class Constants" for the list of constants.   |
| PenButtonPushed | Index as <a href="#">Integer</a>                                 | Returns a <a href="#">Boolean</a> indicating whether the button passed as <i>Index</i> is currently pushed. Mac OS X only.   |
| PenPressure     |  | Returns as a <a href="#">Double</a> the current pen pressure (meaningful only if a graphics tablet is connected). Mac OS X only. The value ranges from zero (not touching the tablet) to 100 (full pressure).  |
| PenType         |  | Returns an <a href="#">Integer</a> that describes the type of pen being used (meaningful only if a graphics tablet is connected). Mac OS X only. You can use the following class constants to determine the pen type:<br>PenTypeNone: Mouse (no graphics tablet connected)<br>PenTypeTip: Tip end of pen device<br>PenTypeEraser: Eraser end of pen device   |
| Pixel           | X as <a href="#">Integer</a> , Y as <a href="#">Integer</a>      | Returns the color of a screen pixel (as <a href="#">Color</a> ), specified in global coordinates. If the specified coordinates are not on any screen, then the color returned is black.  |
| PPPConnect      | [UserIntervention as <a href="#">Boolean</a> ]                   | Connects a dialup connection. The optional parameter gives you the option of allowing user intervention during the dial-up process. Some of your users may have more than one ISP they can select, and the default may not always be who they want to call. If this is the case, set the optional parameter to <a href="#">True</a> to allow user intervention. This feature is not available on Macintosh, as there are no standard dialogs provided for this feature. If the user is running Windows 98/ME, or you pass <a href="#">False</a> or omit the parameter, the system will attempt to connect using the first phonebook entry it can find. |
| PPDisconnect    |  | Disconnects a PPP connection.  |

| Name       | Parameters   | Description   |
|------------|--|---|
| SerialPort | Index as <a href="#">Integer</a><br>or<br>Path as <a href="#">String</a> | Gets the specified serial port on the user's machine. Specify the serial port either by its index or by path. On Mac OS X and Linux, the path is the path to the device file corresponding to the serial device (e.g., /dev/ttyS0). On Windows, this can be the COM port you want to open, e.g., Serial1.SerialPort=System.SerialPort("COM1") |

### PPP

The point-to-point protocol (PPP) is used to gain a connection to the Internet with a dial-up modem. It is system-wide functionality that you can use to get the modem to dial out to an ISP, and upon successful connection, your [TCPSocket](#) or [UDPSocket](#) code will be able to communicate with the outside world.

If you call the PPPDisconnect method, it will terminate the computer's connection to the Internet. This means that you will kill other applications' connections as well as REALbasic's, so be sure to ask the user if they want the connection terminated before happily killing all connections to the Internet.

The PPPConnect method behaves slightly differently, depending on how you use it. If you pass [True](#) on Macintosh, there will be no intervention. There are no standard dialogs provided for the user to choose which connection to use on Macintosh. If you pass [True](#) on Windows and the user is running Windows NT 4 or later, then the standard RAS Manager dialogs will appear and ask the user for connection information. If the user is running Windows 98/ME or you pass [False](#) or do not pass a parameter, then the system will attempt the connection using the first phonebook entry it can find.

### PPPStatus values

The following table summarizes the values returned by the PPPStatus property. You can determine the status by comparing the PPPStatus property to the class constants.

| Value | Class Constant       | Description   |
|-------|----------------------|---|
| 0     | PPPNocConnection     | There is no connection present. This means that you haven't called the PPPConnect method or the connection process failed. It could also mean that you have called the PPPDisconnect method and the connection has been disconnected. |
| 1     |                      | Not used.   |
| 3     | PPPConnectionClosing | The connection is being closed. This means that you have called the PPPDisconnect method and the disconnection process has begun. It does not mean that the disconnect is complete.   |
| 4     | PPPConnectionOpening | A connection is being attempted. This does not mean that you have a valid internet connection. Calling your <a href="#">TCPSocket</a> or <a href="#">UDPSocket</a> code before retesting for a valid connection may cause an error.   |

| Value | Class Constant | Description  |
|-------|----------------|--|
| 5     | PPPConnected   | You have a valid connection. This means that you can execute your <a href="#">TCPsocket</a> or <a href="#">UDPSocket</a> code because you are fully connected. |

**LogLevel Class Constants**

Use the following class constants when setting the value of the Level parameter in calls to the Log method.

| Constant            |
|---------------------|
| LogLevelAlert       |
| LogLevelCritical    |
| LogLevelDebug       |
| LogLevelEmergency   |
| LogLevelError       |
| LogLevelInformation |
| LogLevelNotice      |
| LogLevelSuccess     |
| LogLevelWarning     |

For example, `Log(System.LogLevelError, "my Error message")` writes your error message with the LogLevelError constant.

**Notes**

MouseX and MouseY return coordinates with respect to the top-left corner of the user's monitor. The [Window](#) and [Control](#) classes contain MouseX and MouseY properties that are referenced to the window (in the case of the Control class, the control's parent window).

The SerialPort and SerialPortCount properties work normally in carbonized applications running in Mac OS X. However, carbon applications running under Mac OS "classic" versions cannot access the serial ports. For those machines, build a separate "classic" version of your application.

The Point-to-Point protocol (PPP) is the protocol used for establishing an internet connection via a dial-up modem. Once a connection is established, you can use objects derived from the [TCPsocket](#) and/or [UDPSocket](#) classes to manage the connection.

**Multiple Interface Support**

Multiple network interface support enables you to write applications that can bind to different NIC cards installed on a user's machine. For example, you can use this to write tunneling applications. To see what interfaces are installed on the user's machine, you can use the GetNetworkInterface method and assign the obtained interface object to a NetworkInterface property of the [SocketCore](#) class.

- Examples** The following displays the number of serial ports on the user's machine and their names. On a 'normal' Macintosh, serialport zero is the Modem port and serialport 1 is the Printer port:

```
EditField1.text=Str(System.serialportcount)
EditField2.text=System.serialport(0).Name
EditField3.text=System.serialport(1).Name
```

The following returns the version of QuickTime installed on the user's computer:

```
Dim s as String
Dim b as Boolean
Dim i,resp as Integer
#If TargetMacOS
b=System.gestalt("qtim",resp)
If b then
s=Hex(resp)
For i =Len(s) downto 1
s=Left(s,i)+". "+Mid(s,i+1)
Next
MsgBox "QuickTime version "+s
End if
#Endif
```

The result is:



This example checks whether the application is running on Mac OS 8.5 or higher:

```
// check the system version, since these
//functions are only available
// in MacOS 8.5 and higher
Dim SysVersion as Integer
If System.Gestalt("sysv", sysVersion) and sysVersion >= &h0850 then
// do something cool here
end if
```

To determine whether the application is running under Mac OS X, use the same code but with a constant of [&h1000](#).

The following gets the system environment variables for any supported platform.

```
Dim sh as New Shell
Dim cmd as String

#If TargetWin32
cmd = "set"
#else
cmd = "env"
#endif

sh.Execute cmd
MsgBox sh.Result
```

**See Also** [Application](#), [MouseCursor](#), [SerialPort](#) classes; [App](#), [Cursors](#), [NetworkInterface](#) objects.

## SystemFolder Function

Used to access the System folder.

**Syntax** *result=SystemFolder*

| Part   | Type                       | Description   |
|--------|----------------------------|---|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the System Folder. |

**Notes** Use the **SystemFolder** function to access the user's System Folder.

The **SystemFolder** function provides a way to access the System folder that will work under different language systems, and will continue to work even if the operating system is reorganized.

The [SpecialFolder](#) module enables you to access many special folders that are managed by the OS.

**Windows** On Windows, **SystemFolder** accesses the \Windows\System32 directory.

**Macintosh** On Macintosh, **SystemFolder** accesses the :System folder.

**Linux** On Linux, **SystemFolder** accesses the /usr directory.

**Example** This example displays the absolute path to the system folder on the user's machine.

```
Dim f As FolderItem  
f=SystemFolder  
if f <> Nil Then  
    MsgBox f.AbsolutePath  
Else  
    MsgBox "The folderItem does not exist."  
End if
```

**See Also** [ApplicationSupportFolder](#), [DesktopFolder](#), [FontsFolder](#), [PreferencesFolder](#), [StartupItemsFolder](#), [TemporaryFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

---

## TabPanel Control

The standard tab control.

**Super Class** [PagePanel](#)

### Properties

| Name             | Type                    | Description  |
|------------------|-------------------------|--|
| <b>Facing</b>    | <a href="#">Integer</a> | The side of the tab panel on which the tabs appear. Use the following class constants:<br>FacingNorth (0): North<br>FacingSouth (1): South<br>FacingEast (2): East<br>FacingWest (3): West<br>For example:<br>TabPanel1.Facing=TabPanel.FacingWest |
| <b>SmallTabs</b> | <a href="#">Boolean</a> | If <a href="#">True</a> , decreases the vertical dimension of the tab panel (on North/South tabs). See the Notes for an illustration. Not supported on Windows or Linux.   |

### Methods

| Name    | Parameters  | Description   |
|---------|---|---|
| Append  | Caption as <a href="#">String</a>                                       | Appends a page after the last page of the TabPanel and uses the passed string as its caption.   |
| Caption | Index as <a href="#">Integer</a>  | Allows you to set the caption of the specified tab at runtime. The first tab is numbered zero.  |
| Insert  | Index as <a href="#">Integer</a> ,<br>Caption as <a href="#">String</a> | Inserts a page in the position specified by <i>Index</i> . The first page is numbered zero. The passed string is used as its caption. |

| Name   | Parameters                       | Description  |
|--------|----------------------------------|--|
| Remove | Index as <a href="#">Integer</a> | Removes the page specified by <i>Index</i> . All of the controls on the page will also be deleted. |

## Events

| Name      | Parameters   | Description  |
|-----------|--|--|
| Change    |  | The frontmost tab has changed.   |
| GotFocus  |  | (Windows only) The TabPanel has received the focus. If the user selected the control by tabbing into it, the frontmost tab has a selection marquee.  |
| LostFocus |  | (Windows only) The TabPanel has lost the focus.  |
| MouseDown | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the TabPanel at the location passed in to x,y. Returns a <a href="#">Boolean</a> . <a href="#">Return True</a> if you are going to handle the MouseDown. The coordinates x, and y are local to the control. Point 0,0 is the top-left corner of the entire control (to the left of the label and above the box that is drawn). |
| MouseDrag | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was pressed inside the TabPanel and moved (dragged) at the location local to the control passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |
| MouseUp   | x as <a href="#">Integer</a><br>y as <a href="#">Integer</a> | The mouse button was released inside the TabPanel at the location passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.   |

## Notes

### Getting and Setting the displayed Panel

Use the Value property to get and set the tab panel being displayed. The first panel is numbered zero. The **TabPanel** control is subclassed from [PagePanel](#); the Value property is described there.

You get or set the panel on which a control is located via the PanelIndex property of the [Control](#) class. The first panel is numbered zero.

The following line of code moves a ListBox from its current panel to the second panel of a [PagePanel](#).

```
ListBox1.PanelIndex=1
```

### Adding, Labelling, and Renaming Tabs

Click the value of the Panels property of the TabPanel to display the Tab Panel editor. From this screen, you can:

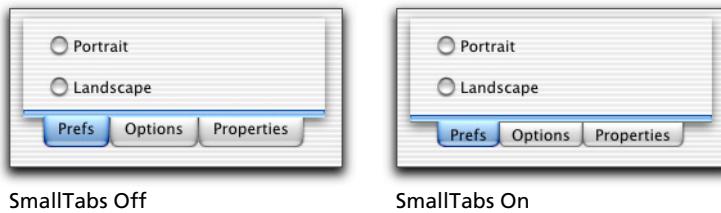
- Rename the existing tab labels by clicking twice on its name and editing the name.
- Add tabs and their labels by clicking the Add button.
- Delete any tab by highlighting it and clicking Delete.

- Rearrange the tabs by highlighting a tab and clicking the Up or Down buttons.
- You add controls to a particular panel by clicking its tab and dragging controls to that panel.

### Facings and SmallTabs

The Facings property lets you place the tabs on any side of the TabPanel. The SmallTabs property reduces the vertical (North or South) or horizontal (East or West) size of the tabs.

The following comparison illustrates the effect of the SmallTabs property on a TabPanel that uses the South facing:



### Placing Controls on Tab Panels

You can place other controls on the surface of any tab panel. Such controls are visible only when the user clicks on that panel's tab to bring it to the front. You can use the TabPanelIndex property of the [Control](#) class to read the panel on which the control is located. You cannot move a control to another panel via code. Please note that TabPanelIndex may be superseded by another way of reading a control's 'parent' object and code that uses TabPanelIndex may have to be rewritten.

Placing TabPanels within other TabPanels, [PagePanels](#) within [PagePanels](#), TabPanels within PagePanels, and vice versa are not officially supported.

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. This is done to make applications on all platforms behave consistently. The ampersand character is used to indicate the keyboard accelerator on Windows.

### See Also

[RectControl](#) class; [PagePanel](#) control.

---

## Tan Function

Returns the tangent of the angle specified.

### Syntax

***result=Tan (value)***

| Part   | Type                   | Description                                     |
|--------|------------------------|---|
| result | <a href="#">Double</a> | The tangent of value.                           |
| value  | <a href="#">Double</a> | The value (in radians) you want the tangent of. |

**Notes** The **Tan** function returns the tangent of the angle (in radians) passed to it. If the angle is in degrees, multiply it by PI/180 to convert it to radians.

**Examples** This example uses the **Tan** function to return the tangent of a number.

```
Dim d As Double
Const PI=3.14159265358979323846264338327950
d=Tan(45*PI/180) //returns 1.0
```

**See Also** [Atan](#) function.

## TargetBigEndian Constant

Used to indicate that you are compiling for an OS that uses the Big Endian byte order.

**Syntax** *result=TargetBigEndian*

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if are compiling for an OS that uses the Big Endian byte order. |

**Notes** Big Endian means that the high-order byte is stored in memory at the lowest address and the low-order byte at the highest address. Macintosh uses the Big Endian byte order.

**See Also** [DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; [#If](#) statement.

## TargetCarbon Constant

Used to indicate that you are compiling Carbon code.

**Syntax** *result=TargetCarbon*

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if are compiling Carbon code, either as a standalone application or within REALbasic. |

**Notes** The Carbon version of REALbasic compiles as Carbon while running in the IDE.

[TargetMacOSClassic](#), and **TargetCarbon** are mutually exclusive subsets of [TargetMacOS](#). **TargetCarbon** is [True](#) if you are compiling for Macintosh in either the

## TargetHasGUI Constant

---

PEF or MachO formats. [TargetMacOSClassic](#) is [True](#) only if PPC code is running on a PowerPC under a ‘classic’ Mac OS.

### See Also

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; [#If](#) statement.

## TargetHasGUI Constant

---

Used to indicate whether the application has a GUI. [ConsoleApplications](#) do not.

### Syntax

**result=TargetHasGUI**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if are compiling REALbasic code that supports a GUI. |

### Example

Use **TargetHasGUI** to isolate code for a version of the application that runs as a console application. The following example writes an error message to the standard error stream if the application has no GUI.

```
Sub DisplayError(err as String)
#If TargetHasGUI
    MsgBox err
#else
    StdErr.WriteLine err
#endif
```

### See Also

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; [#If](#) statement; [ConsoleApplication](#), [ServiceApplication](#) classes.

## TargetLinux Constant

---

Used to indicate that you are compiling code for the Linux OS.

### Syntax

**result=TargetLinux**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if are compiling Linux code. |

**See Also**

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; #If statement.

## TargetLittleEndian Constant

Used to indicate that you are compiling for an OS that uses the Little Endian byte order.

**Syntax**

***result=TargetLittleEndian***

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if are compiling for an OS that uses the little Endian byte order. |

**Notes**

Little Endian means that the low-order byte is stored in memory at the lowest address and the high-order byte is stored at the highest address. PCs use the Little Endian byte order.

**See Also**

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; #If statement.

## TargetMachO Constant

Used to indicate that you are compiling code for the Mac OS X Mach-O runtime architecture. A Mach-O build of a REALbasic application runs only on Mac OS X. The code is in the format for the Mach kernel of OS X.

**Syntax**

***result=TargetMachO***

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if are compiling Mach-O code. |

**See Also**

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetWin32](#) constants; #If statement.

## TargetMacOS Constant

Used to indicate that you are compiling code for any Macintosh OS, “classic” or Mac OS X in either the Mach-O or PEF formats.

**Syntax****result=TargetMacOS**

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if you are compiling code for any Mac OS, either as a standalone application or within REALbasic. |

**Notes**

[TargetMacOSClassic](#), and [TargetCarbon](#) are mutually exclusive subsets of [TargetMacOS](#). [TargetMacOSClassic](#) is [True](#) only if PPC code is running on a PowerPC under a ‘classic’ Mac OS.

**See Also**

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#), [TargetWin32](#) constants; [#If](#) statement.

## TargetMacOSClassic Constant

Used to indicate that you are compiling MacOS 8/9 code on a PowerPC.

**Syntax****result=TargetMacOSClassic**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if you are compiling MacOS 8/9 code. |

**Notes**

[TargetMacOSClassic](#), and [TargetCarbon](#) are mutually exclusive subsets of [TargetMacOS](#). [TargetMacOSClassic](#) is [True](#) only if your code is running on a PowerPC under a ‘classic’ Mac OS. Macintoshes that use the 68K family of processors are no longer supported.

**See Also**

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetWin32](#) constants; [#If](#) statement.

## TargetWin32 Constant

Used to indicate that you are compiling code for a Windows OS. REALbasic supports Windows 98 and above.

**Syntax*****result=TargetWin32***

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Boolean</a> | Returns <a href="#">True</a> if you are compiling Windows code. |

**See Also**

[DebugBuild](#), [RBVersion](#), [RBVersionString](#), [TargetBigEndian](#), [TargetCarbon](#), [TargetHasGUI](#), [TargetLinux](#), [TargetLittleEndian](#), [TargetMachO](#), [TargetMacOS](#), [TargetMacOSClassic](#), [TargetMachO](#) constants; [#If](#) statement.

# TCPSocket Control

Used for TCP/IP communication.

**Super Class** [SockerCore](#) class.

**Properties**

| Name            | Type                    | Description  |
|-----------------|-------------------------|--|
| Address         | <a href="#">String</a>  | The TCP/IP address to try to connect to.   |
| BytesAvailable  | <a href="#">Integer</a> | The number of bytes of data are available in the internal receive buffer.  |
| BytesLeftToSend | <a href="#">Integer</a> | The number of bytes left in the queue remaining to be sent. This enables you to create a synchronous socket without needing to subclass the <b>TCPSocket</b> .                 |
| RemoteAddress   | <a href="#">String</a>  | The remote address of the remote machine you are connected to. Use this instead of the Address property to determine the address of the machine you are actually connected to. |

## Events

| Name          | Parameters   | Description  |
|---------------|--|--|
| Connected     |  | Occurs when a connection has been established.   |
| DataAvailable |  | Occurs when additional data has come into the internal receive buffer.   |
| Error         |  | An error occurred or the connection was either dropped or deliberately ended. The LastErrorCode property will hold a value that indicates what happened. Please refer to the list of error codes and corresponding class constants for the <a href="#">SocketCore</a> class. Use the class constants to test against the value in LastErrorCode.   |
| SendComplete  | UserAborted as <a href="#">Boolean</a>                                       | All of the data in the socket buffer has been written (sent). The UserAborted parameter enables you to determine whether the user cancelled the transfer by returning <a href="#">True</a> from the SendProgress event. If the user aborted, this value is <a href="#">True</a> , otherwise it is <a href="#">False</a> .  |
| SendProgress  | BytesSent as <a href="#">Integer</a><br>BytesLeft as <a href="#">Integer</a> | Occurs when your network provider queues your data in 'chunks' and is about to send the next chunk. The parameters indicate the amount of progress that has been made during the send. Returns a <a href="#">Boolean</a> . Returning <a href="#">True</a> from this event causes the send to be aborted. This does not close the TCPSocket's connection; it only clears the buffer. After all of the data has been transferred you will get a final SendProgress event followed by a SendComplete event. |

## Methods

| Name       | Parameters   | Description  |
|------------|--|--|
| Disconnect |  | Disconnects the socket, resets it, and fires a <a href="#">SocketCore</a> Error event with a 102 error to let you know that the socket has been disconnected.  |
| Listen     |  | Attempts to listen for incoming connections on the currently specified port. After calling Listen, the Port property will report the actual port you are bound to.   |
| Lookahead  | [Encoding as <a href="#">TextEncoding</a> ]                                      | Returns a <a href="#">String</a> , containing the data that is available in the internal queue without removing it. The optional Encoding parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify an encoding.      |
| Read       | Bytes as <a href="#">Integer</a><br>[,Encoding as <a href="#">TextEncoding</a> ] | Reads the amount of bytes specified from the internal receive buffer. Returns the data as a <a href="#">String</a> . The optional Encoding parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify a text encoding. |

| Name    | Parameters                                  | Description  |
|---------|---|--|
| ReadAll | [Encoding as <a href="#">TextEncoding</a> ] | Reads all the data from the internal buffer. Returns the data as a <a href="#">String</a> . The optional <i>Encoding</i> parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify a text encoding. |
| Write   | Data as <a href="#">String</a>              | Writes out <i>Data</i> to the remote connection.   |

## Notes

The *Transmission Control Protocol*, or TCP, is the basis for most internet traffic. It is a connection-oriented protocol that provides a reliable way to transfer data across a network.

To establish a connection between two computers using TCPSockets, one computer must be set up to listen on a specific port. The other computer (called the client) then attempts to connect by specifying the network address (or IP address) of the remote machine and the port on which to attempt the connection. In order to send and receive data with a remote machine, both machines must have some indication that this connection will be established. That happens by either picking a well-defined port for the listener (or server) to listen on, or by some prior arrangement (e.g., you are the author of both the server and the client program).

When a server receives a connection attempt on the port it is listening on, it accepts the incoming connection, and sends an acknowledgement back to the remote machine. Once both machines have reached an agreement (or are “connected”), then you can begin sending and receiving data. When you close your connection with the remote machine, there is a similar handshake process that goes on, so both computers know that the connection is being terminated.

Because of the amount of error checking and handshakes, TCP is an extremely reliable protocol. When you send a packet of information, it is guaranteed to make it to the remote machine unless you have been disconnected, either abortive or orderly. But this feature comes at the cost of high overhead. A typical TCP packet that is sent over the network has around a 40-byte header that goes with it. This header is checked, and changed by all the various machines en route to its destination. This overhead makes TCP a slower protocol to use. The trade-off is speed versus security, in favor of security. If it is speed you are looking for (for example to support a networked game), then you should look into the UDP protocol.

The **TCPSocket** more gracefully handles the disconnection process. It tries to do an orderly disconnect when possible (which allows for all data transfers to finish), and will only fall back on an abortive disconnect when it is not possible to do an orderly one.

**TCPSockets** now handle flow control, so sending large amount of data will not drop data during the send or the receive process.

It is possible for packets of data to come in before the socket has finished the connection process. If this happens, the **TCPSocket** is prepared to handle this, so your initial

packets will not be lost. This is a Macintosh-only feature; Windows sockets were already prepared to handle these situations.

The SendProgress event allows you to determine send speeds, and tells you how many bytes of data you have sent since your last SendProgress event, as well as how many bytes are left to send. If you return [True](#) from the send progress event, you cancel the current transfer. This does not close the socket's connection; it only clears the send buffer. You can use this event to determine if a connection is too slow, and cancel it. Once all of the data has been transferred, you will get a last SendProgress event, followed by a SendComplete event.

The RemoteAddress property for a connecting socket returns the IP address of the remote host connection for Windows sockets. One useful thing you can do with the RemoteAddress property is to test to make sure the IP address you wanted to connect to is the same as the IP address you are currently connected to.

**TCPSockets** can be orphaned. An orphaned socket will keep a lock on itself after you call either the Connect or Listen methods. The socket gets unlocked when it has been disconnected. This means that an orphaned socket will continue to work so long as it is connected. This also means that you need to call the Close method before its lock will be released.

If you orphan a socket, you need to be certain that the socket will get a disconnect message from its connection at some point. For example, some web servers will not release a socket once the data has been transferred to it (for efficiency reasons). When you are done with the socket, unless you explicitly call the Close method of the [SocketCore](#) class, it will remain active and connected. If you think your socket could be in a state in which it is left open, keep a reference to the socket around somewhere and use the Close method to terminate the connection. (Note that calling the Disconnect method also applies instead of calls to the Close method).

If you use a constructor in a subclass of **TCPSocket**, you must call the Super class's constructor in your subclass's constructor. The subclass will not work unless this is done.

Binding a **TCPSocket** to well-known ports below 1024 requires proper privileges on all operating systems.

The **TCPSocket** class implements the [Readable](#) and [Writable](#) class interfaces. If you implement either or both of these interfaces, you must provide the methods and parameters specified by these class interfaces.

For more information about class interfaces and how to implement them, see the section "Class Interfaces" on page 458 of the *User's Guide*.

## See Also

[EasyTCPSocket](#), [ServerSocket](#), [SocketCore](#), [SSLSocket](#), [UDPSocket](#) classes; [Readable](#), [Writable](#) class interfaces.

# TemporaryFolder Function

Used to access the Temporary folder.

## Syntax

**result=TemporaryFolder**

| Part   | Type                       | Description  |
|--------|----------------------------|--|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the Temporary folder. |

## Notes

TemporaryFolder returns a reference to a folder on the user's startup volume. Use it when you need to store a file in a place that the user can't see on the desktop.

[GetTemporaryFolderItem](#) creates a [FolderItem](#) in this folder.

The **TemporaryFolder** function provides a way to access the Temporary folder that will work under different language systems.

The [SpecialFolder](#) module enables you to access many special folders that are managed by the OS.

## Windows

On Windows, **TemporaryFolder** returns a reference to \user\Local Settings\Temp.

## Macintosh

On Macintosh, **TemporaryFolder** returns a reference to :private:tmp:501:TemporaryItems.

## Linux

On Linux, **TemporaryFolder** returns a reference to the /tmp directory.

## Examples

This example displays the absolute path to the user's Temporary folder.

```
Dim f As FolderItem
f=TemporaryFolder
if f <> Nil Then
  MsgBox f.AbsolutePath
Else
  MsgBox "The folderItem does not exist."
End if
```

## See Also

[ApplicationSupportFolder](#), [DesktopFolder](#), [FontsFolder](#), [GetTemporaryFolderItem](#), [PreferencesFolder](#), [StartupItemsFolder](#), [SystemFolder](#), [TrashFolder](#), [SpecialFolder](#) functions.

# TextColor Function

The currently selected operating system text color.

## TextConverter Class

---

### Syntax

**result=TextColor**

| Part   | Type                  | Description                      |
|--------|-----------------------|----------------------------------|
| result | <a href="#">Color</a> | The color used for drawing text. |

### Notes

This value is useful when you are using [Canvas](#) controls to create custom controls. When drawing objects, use this color as the ForeColor value when drawing text.

This value can be changed by the user, so you should access this value in the Paint event handler rather than storing the value.

### See Also

[DarkBevelColor](#), [DarkTingeColor](#), [FillColor](#), [FrameColor](#), [HighlightColor](#), [LightBevelColor](#), [LightTingeColor](#) functions; [Color](#) data type.

---

## TextConverter Class

Used to convert text from one encoding system to another.

**Super Class** [Object](#)

### Methods

| Name    | Parameters                          | Description   |
|---------|-------------------------------------|---|
| Clear   |                                     | Clears the state of the TextConverter object for converting a new string      |
| Convert | InputText as <a href="#">String</a> | Converts Input text. Returns the converted text as a <a href="#">String</a> . |

### Example

The following example converts the text in an [EditField](#):

```
Dim c as TextConverter  
c=GetTextConverter(GetTextEncoding(&h500), GetTextEncoding(0))  
EditField2.text=c.convert(EditField1.text)
```

### Text Encoding Base and Variant Codes

The following tables give the values of the **Base** and **Variant** parameters that are used in several of the functions in the Text Encoding theme. Table 2 gives the codes for **Base** and Table 3 gives the codes for **Variant**.

**Table 2: Text Encoding codes for Base**

| Base           | Value |
|----------------|-------|
| MacRoman       | 0     |
| MacJapanese    | 1     |
| MacChineseTrad | 2     |

**Table 2: Text Encoding codes for Base (Continued)**

| Base               | Value |
|--------------------|-------|
| MacKorean          | 3     |
| MacArabic          | 4     |
| MacHebrew          | 5     |
| MacGreek           | 6     |
| MacCyrillic        | 7     |
| MacDevanagari      | 9     |
| MacGurmukhi        | 10    |
| MacGujarati        | 11    |
| MacOriya           | 12    |
| MacBengali         | 13    |
| MacTamil           | 14    |
| MacTelugu          | 15    |
| MacKannada         | 16    |
| MacMalayalam       | 17    |
| MacSinhalese       | 18    |
| MacBurmese         | 19    |
| MacKhmer           | 20    |
| MacThai            | 21    |
| MacLaotian         | 22    |
| MacGeorgian        | 23    |
| MacArmenian        | 24    |
| MacChineseSimp     | 25    |
| MacTibetan         | 26    |
| MacMongolian       | 27    |
| MacEthiopic        | 28    |
| MacCentralEurRoman | 29    |
| MacVietnamese      | 30    |
| MacExtArabic       | 31    |
| MacSymbol          | 33    |
| MacDingbats        | 34    |
| MacTurkish         | 35    |
| MacCroatian        | 36    |
| MacIcelandic       | 37    |
| MacRomanian        | 38    |
| MacCeltic          | 39    |

**Table 2: Text Encoding codes for Base (Continued)**

| Base              | Value  |
|-------------------|--------|
| MacGaelic         | 40     |
| MacUnicode        | &h7E   |
| MacFarsi          | &h8C   |
| MacUkrainian      | &h98   |
| MacInuit          | &hEC   |
| MacVT100          | &hFC   |
| MacHFS            | &hFF   |
| UnicodeDefault    | &h0100 |
| UnicodeV1_1       | &h0101 |
| ISO10646_1933     | &h0101 |
| UnicodeV2_0       | &h0103 |
| Unicodev2_1       | &h0103 |
| ISOLatin1         | &h0201 |
| ISOLatin2         | &h0202 |
| ISOLatin3         | &h0203 |
| ISOLatin4         | &h0204 |
| ISOLatinCyrillic  | &h0205 |
| ISOLatinArabic    | &h0206 |
| ISOLatinGreek     | &h0207 |
| ISOLatinHebrew    | &h0208 |
| ISOLatin5         | &h0209 |
| DOSLatinUS        | &h0400 |
| DOSGreek          | &h0405 |
| DOSBalticRim      | &h0405 |
| DOSLatin1         | &h0410 |
| DOSGreek1         | &h0411 |
| DOSLatin2         | &h0412 |
| DOSCyrillic       | &h0413 |
| DOSTurkish        | &h0414 |
| DOCPortuguese     | &h0415 |
| DOSIslandic       | &h0416 |
| DOSHebrew         | &h0417 |
| DOSCanadianFrench | &h0418 |
| DOSArabic         | &h0419 |
| DOSNordic         | &h041A |

**Table 2: Text Encoding codes for Base (Continued)**

| <b>Base</b>        | <b>Value</b> |
|--------------------|--------------|
| DOSRussian         | &h041B       |
| DOSGreek2          | &h041C       |
| DOSThai            | &h041D       |
| DOSJapanese        | &h0420       |
| DOSChineseSimplif  | &h0421       |
| DOSKorean          | &h0422       |
| DOSChineseTrad     | &h0423       |
| WindowsLatin1      | &h0500       |
| WindowsANSI        | &h0500       |
| WindowsLatin2      | &h0501       |
| WindowsCryillic    | &h0502       |
| WindowsGreek       | &h0503       |
| WindowsLatin5      | &h0504       |
| WindowsHebrew      | &h0505       |
| WindowsArabic      | &h0506       |
| WindowsBalticRim   | &h0507       |
| WindowsVietnamese  | &h0508       |
| WindowsKoreanJohab | &h0510       |
| US_ASCII           | &h0600       |
| JIS_X0201_76       | &h0620       |
| JIS_X0208_83       | &h0621       |
| JIS_X0208_90       | &h0622       |
| JIS_X0212_90       | &h0623       |
| JIS_C6226_78       | &h0624       |
| GB_2312_80         | &h0630       |
| GBK_95             | &h0631       |
| KSC_5601_87        | &h0640       |
| KSC_5601_92_Johab  | &h0641       |
| CNS_11643_92_P1    | &h0651       |
| CNS_11643_92_P2    | &h0652       |
| CNS_11643_92_P3    | &h0653       |
| ISO_2022_JP        | &h0820       |
| ISO_2022_JP_2      | &h0821       |
| ISO_2022_CN        | &h0830       |
| ISO_2022_CN_EXT    | &h0831       |

**Table 2: Text Encoding codes for Base (Continued)**

| Base            | Value  |
|-----------------|--------|
| ISO_2022_KR     | &h0840 |
| EUC_JP          | &h0920 |
| EUC_CN          | &h0930 |
| EUC_TW          | &h0931 |
| EUC_KR          | &h0940 |
| ShiftJIS        | &h0A01 |
| KOI8_R          | &h0A02 |
| Big5            | &h0A03 |
| MacRomanLatin1  | &h0A04 |
| HZ_GB_2312      | &h0A05 |
| NextStepLatin   | &h0B01 |
| EBCDIC_US       | &h0C01 |
| EBCDIC_CP037    | &h0C02 |
| MultiRun        | &h0FFF |
| MacTradChinese  | 2      |
| MacRSSymbol     | 8      |
| MacSimpChinese  | 25     |
| MacGeez         | 28     |
| MacEastEurRoman | 29     |
| MacUninterp     | 32     |

**Table 3: Text Encoding codes for Variant.**

| <b>Variant</b>                        | <b>Value</b> |
|---------------------------------------|--------------|
| Text Encoding Default Variant         | 0            |
| Mac Roman Default Variant             | 0            |
| Mac Roman Currency Sign Variant       | 1            |
| Mac Roman Euro Sign Variant           | 2            |
| Mac Icelandic Std Default Variant     | 0            |
| Mac Icelandic TT Default Variant      | 1            |
| Mac Icelandic Std Euro Sign Variant   | 4            |
| Mac Icelandic TT Euro Sign Variant    | 5            |
| Mac Croatian Default Variant          | 0            |
| Mac Croatian Currency Sign Variant    | 1            |
| Mac Croatian Euro Sign Variant        | 2            |
| Mac Romanian Default Variant          | 0            |
| Mac Romanian Currency Sign Variant    | 1            |
| Mac Romanian Euro Sign Variant        | 2            |
| Mac Japanese Standard Variant         | 0            |
| Mac Japanese StdNoVerticals Variant   | 1            |
| Mac Japanese Basic Variant            | 2            |
| Mac JapanesePostScript Scrn Variant   | 3            |
| Mac Japanese PostScript Print Variant | 4            |
| Mac Japanese VartAtKuPlusTen Variant  | 5            |
| Mac Arabic Standard Variant           | 0            |
| Mac Arabic TrueType Variant           | 1            |
| Mac Arabic Thuluth Variant            | 2            |
| Mac Arabic AlBayan Variant            | 3            |
| Mac Farsi Standard Variant            | 0            |
| Mac Farsi TrueType Variant            | 1            |
| Mac Hebrew Standard Variant           | 0            |
| Mac Hebrew FigureSpace Variant        | 1            |
| Mac VT100 Default Variant             | 0            |
| Mac VT100 CurrencySign Variant        | 1            |
| Mac VT100 EuroSign Variant            | 2            |
| Unicode No Subset                     | 0            |
| Unicode Canonical Decomp Variant      | 2            |

**Table 3: Text Encoding codes for Variant. (Continued)**

| Variant                             | Value |
|-------------------------------------|-------|
| Big5_Basic Variant                  | 0     |
| Big5_Standard Variant               | 1     |
| Big5_ETen Variant                   | 2     |
| Unicode No Compatibility Variant    | 1     |
| Unicode No Composed Variant         | 3     |
| Unicode No Corporate Variant        | 4     |
| Mac Roman Standard Variant          | 0     |
| Mac Icelandic Standard Variant      | 0     |
| Mac Icelandic TrueType Variant      | 1     |
| Japanese Standard Variant           | 0     |
| Japanese Std No Verticals Variant   | 1     |
| Japanese Basic Variant              | 2     |
| Japanese PostScript Scrn Variant    | 3     |
| Japanese PostScript Print Variant   | 4     |
| Japanses VertAtKuPlusTen Variant    | 5     |
| Hebrew Standard Variant             | 0     |
| Hebrew Figure Space Variant         | 1     |
| Unicode Max Decomposed Variant      | 2     |
| Japanese NoOneByteKanaOption        | 0020  |
| Japanese Use ASCII Backslash Option | &h40  |

**See Also**

[ConvertEncoding](#), [DefineEncoding](#), [Encoding](#), [GetTextConverter](#) functions;  
[TextEncoding](#) class; [Encodings](#) object.

---

## TextEncoding Class

Used to specify the text encoding of a [String](#).

**Super Class** [Object](#)**Properties**

| Name | Type                    | Description   |
|------|-------------------------|---|
| Base | <a href="#">Integer</a> | The type of encoding. The entry for <a href="#">TextConverter</a> contains the possible values of <i>Base</i> . |

| Name         | Type                    | Description   |
|--------------|-------------------------|---|
| Code         | <a href="#">Integer</a> | (Mac OS) The Mac OS <b>TextEncoding</b> value, useful for declares. You can also use it to compare two encodings: If the Code properties of two TextEncoding objects are equal, then they represent the same text encoding (including base, variation, and format). |
| Variant      | <a href="#">Integer</a> | A variant of the Base text encoding. The entry for <a href="#">TextConverter</a> contains the possible values of Variant.   |
| Format       | <a href="#">Integer</a> | A format for the <i>Base</i> text encoding. Used by Unicode for defining which format of Unicode you wish to use.   |
| InternetName | <a href="#">String</a>  | Internet Text Encoding name.  |

## Methods

| Name   | Parameters                                    | Description  |
|--------|---|--|
| Chr    | CodePoint as <a href="#">Integer</a>          | Returns the character in the given encoding specified by <i>CodePoint</i> . The “code point” of the character is the same as the value that <a href="#">Asc</a> returns. In general, it is safest to use this rather than the <a href="#">Chr</a> function unless you are dealing with ASCII codes only (0-127). |
| Equals | otherEncoding as <a href="#">TextEncoding</a> | Compares the given encoding to the passed encoding. Returns a <a href="#">Boolean</a> .  |

## Notes

When a computer stores text, it encodes each character as a numeric value and stores the byte (or bytes) associated with that number. When it needs to display or print that character, it consults the encoding scheme to determine which character the number represents.

The first computers used the encoding scheme called “ASCII”, which stands for American Standard Code for Information Exchange. It specified 128 values and includes codes for upper and lower case letters, numbers, the common symbols on a keyboard, and some “invisible” control codes that were heavily used in early computers.

As computers became more sophisticated and were introduced in non-English speaking countries, the limitations of the ASCII encoding scheme became apparent. It didn’t include codes for accented characters and had no chance of handling idiographic languages, such as Japanese or Chinese, which require thousands of characters.

As a result, extensions to the ASCII encoding scheme were developed. Outside the range of 0-127, the schemes, in general, do not agree. For example, in the US Mac OS and Windows computers use different encodings for codes 128-255. Many other encoding schemes for handling languages that use non-ASCII characters have been developed.

The most general solution to the problem is an encoding called Unicode. It is designed to handle every character in every language. It also enables you to represent a mixture of

languages within one text stream. However, not all strings that you may encounter use Unicode.

When you encounter a string, you need to know its encoding in order to interpret the sequence of bytes (or double-bytes) that make up the string's content. In REALbasic, every string contains both the bytes (content) and the encoding (if it is known; it is [Nil](#) if not known). REALbasic supports two different formats of Unicode, UTF-8 and UTF-16. All strings in a REALbasic project are compiled as UTF-8. This is a Unicode encoding that uses one byte for ASCII characters and up to four bytes for non-ASCII characters.

If you work only with strings that are created and managed within your REALbasic application, you probably don't need to deal with encodings directly, as REALbasic takes care of the issues via UTF-8. However, if you receive strings from an outside source such as via the internet, an external database (that is, not the REALSQLdatabase engine), or a text file, you should let REALbasic know what encoding is used. If the string is a [Memoryblock](#), the encoding will be [Nil](#).

You can assign an encoding to a string in several ways. For example, if you are reading the string using the [TextInputStream](#) class, you use the Encoding property. The [Encodings](#) object gives you access to all known encodings. Here is an example that reads a text file that uses the MacRoman encoding:

```
Dim f As FolderItem
Dim t As TextInputStream
f=GetOpenFolderItem("text") //file type defined as as File Type
If f <> Nil then
    t=f.OpenAsTextFile
    t.Encoding=Encodings.MacRoman //specify encoding of input stream
    EditField1.text=t.ReadAll
    t.Close
End if
```

Also, the Read, ReadLine, and ReadAll methods take an optional parameter that lets you specify the encoding.

If you need to output a string in a specific encoding, you can use the [ConvertEncoding](#) function to do so. For example, this code converts the text in an [EditField](#) to the WindowsANSI encoding:

```
Dim s as String
s=EditField1.Text.ConvertEncoding(Encodings.WindowsANS)
```

You will find text encoding helpful if you develop:

- Internet applications, such as web browsers or e-mail applications
- Applications that transfer text across different platforms
- Applications based in Unicode

The [Encoding](#) function makes it easy to obtain the **TextEncoding** of any string. Use the [Encodings](#) object to obtain a specified text encoding. Some of the most useful are UTF8, UTF16, UCS4, ASCII, MacRoman, MacJapanese, and WindowsLatin1. Use the Autocomplete feature of the Code Editor to view the complete list.

## ASCII Codes

The following table presents the ASCII character codes. It presents the Decimal, Hex, and Octal values for ASCII codes (0 to 127).

**Table 4: Standard ASCII codes.**

| Decimal | Hex | Octal | Result | Decimal | Hex | Octal | Result |
|---------|-----|-------|--------|---------|-----|-------|--------|
| 0       | 0   | 0     | NUL    | 32      | 20  | 40    | SP     |
| 1       | 1   | 1     | SOH    | 33      | 21  | 41    | !      |
| 2       | 2   | 2     | STX    | 34      | 22  | 42    | "      |
| 3       | 3   | 3     | ETX    | 35      | 23  | 43    | #      |
| 4       | 4   | 4     | EOT    | 36      | 24  | 44    | \$     |
| 5       | 5   | 5     | ENQ    | 37      | 25  | 45    | %      |
| 6       | 6   | 6     | ACK    | 38      | 26  | 46    | &      |
| 7       | 7   | 7     | BEL    | 39      | 27  | 47    | '      |
| 8       | 8   | 10    | BS     | 40      | 28  | 50    | (      |
| 9       | 9   | 11    | HT     | 41      | 29  | 51    | )      |
| 10      | A   | 12    | LF     | 42      | 2A  | 52    | *      |
| 11      | B   | 13    | VT     | 43      | 2B  | 53    | +      |
| 12      | C   | 14    | FF     | 44      | 2C  | 54    | ,      |
| 13      | D   | 15    | CR     | 45      | 2D  | 55    | -      |
| 14      | E   | 16    | SO     | 46      | 2E  | 56    | .      |
| 15      | F   | 17    | SI     | 47      | 2F  | 57    | /      |
| 16      | 10  | 20    | DLE    | 48      | 30  | 60    | 0      |
| 17      | 11  | 21    | DC1    | 49      | 31  | 61    | 1      |
| 18      | 12  | 22    | DC2    | 50      | 32  | 62    | 2      |
| 19      | 13  | 23    | DC3    | 51      | 33  | 63    | 3      |
| 20      | 14  | 24    | DC4    | 52      | 34  | 64    | 4      |
| 21      | 15  | 25    | NAK    | 53      | 35  | 65    | 5      |
| 22      | 16  | 26    | SYN    | 54      | 36  | 66    | 6      |
| 23      | 17  | 27    | ETB    | 55      | 37  | 67    | 7      |
| 24      | 18  | 30    | CAN    | 56      | 38  | 70    | 8      |
| 25      | 19  | 31    | EM     | 57      | 39  | 71    | 9      |
| 26      | 1A  | 32    | SUB    | 58      | 3A  | 72    | :      |
| 27      | 1B  | 33    | ESC    | 59      | 3B  | 73    | ;      |
| 28      | 1C  | 34    | FS     | 60      | 3C  | 74    | <      |
| 29      | 1D  | 35    | GS     | 61      | 3D  | 75    | =      |
| 30      | 1E  | 36    | RS     | 62      | 3E  | 76    | >      |
| 31      | 1F  | 37    | US     | 63      | 3F  | 77    | ?      |

**Table 4: Standard ASCII codes. (Continued)**

| Decimal | Hex | Octal | Result | Decimal | Hex | Octal | Result |
|---------|-----|-------|--------|---------|-----|-------|--------|
| 64      | 40  | 100   | @      | 96      | 60  | 140   | `      |
| 65      | 41  | 101   | A      | 97      | 61  | 141   | a      |
| 66      | 42  | 102   | B      | 98      | 62  | 142   | b      |
| 67      | 43  | 103   | C      | 99      | 63  | 143   | c      |
| 68      | 44  | 104   | D      | 100     | 64  | 144   | d      |
| 69      | 45  | 105   | E      | 101     | 65  | 145   | e      |
| 70      | 46  | 106   | F      | 102     | 66  | 146   | f      |
| 71      | 47  | 107   | G      | 103     | 67  | 147   | g      |
| 72      | 48  | 110   | H      | 104     | 68  | 150   | h      |
| 73      | 49  | 111   | I      | 105     | 69  | 151   | i      |
| 74      | 4A  | 112   | J      | 106     | 6A  | 152   | j      |
| 75      | 4B  | 113   | K      | 107     | 6B  | 153   | k      |
| 76      | 4C  | 114   | L      | 108     | 6C  | 154   | l      |
| 77      | 4D  | 115   | M      | 109     | 6D  | 155   | m      |
| 78      | 4E  | 116   | N      | 110     | 6E  | 156   | n      |
| 79      | 4F  | 117   | O      | 111     | 6F  | 157   | o      |
| 80      | 50  | 120   | P      | 112     | 70  | 160   | p      |
| 81      | 51  | 121   | Q      | 113     | 71  | 161   | q      |
| 82      | 52  | 122   | R      | 114     | 72  | 162   | r      |
| 83      | 53  | 123   | S      | 115     | 73  | 163   | s      |
| 84      | 54  | 124   | T      | 116     | 74  | 164   | t      |
| 85      | 55  | 125   | U      | 117     | 75  | 165   | u      |
| 86      | 56  | 126   | V      | 118     | 76  | 166   | v      |
| 87      | 57  | 127   | W      | 119     | 77  | 167   | w      |
| 88      | 58  | 130   | X      | 120     | 78  | 170   | x      |
| 89      | 59  | 131   | Y      | 121     | 79  | 171   | y      |
| 90      | 5A  | 132   | Z      | 122     | 7A  | 172   | z      |
| 91      | 5B  | 133   | [      | 123     | 7B  | 173   | {      |
| 92      | 5C  | 134   | \      | 124     | 7C  | 174   |        |
| 93      | 5D  | 135   | ]      | 125     | 7D  | 175   | }      |
| 94      | 5E  | 136   | ^      | 126     | 7E  | 176   | ~      |
| 95      | 5F  | 137   | -      | 127     | 7F  | 177   | DEL    |

## TextInputStream Class

---

- Examples** The following example obtains the TextEncoding of the string passed to the [Encoding](#) function.

```
Dim t as TextEncoding
t=Encoding(Editfield1.text)
If t <> Nil then
    staticText1.text="Base="+Str(t.base)
    staticText2.text="Format="+Str(t.format)
    staticText3.text="Variant="+Str(t.variant)
end if
```

The following statement uses the **Encodings** object to obtain the UTF8 text encoding for text in an [EditField](#).

```
Editfield2.text=DefineEncoding(EditField1.text,Encodings.UTF8)
```

The following example uses the Chr method to obtain the character corresponding to the code point of 165 for the MacRoman encoding, the bullet character (•):

```
Dim s as String
s=Encodings.MacRoman.Chr(165)
```

- See Also** [Chr](#), [ConvertEncoding](#), [DefineEncoding](#), [Encoding](#), [GetInternetTextEncoding](#), [GetTextConverter](#), [GetTextEncoding](#) functions; [Encodings](#) object.

---

## TextInputStream Class

In order to read text from a file, you need to create a **TextInputStream** object.

**TextInputStreams** have methods that allow to read from a file, check to see if you are at the end of the file, and close the file when you are done reading from it. They are created by calling the OpenAsTextFile method of a [FolderItem](#) object. If you are working with a file with an encoding that is not UTF-8, you should set the value of the Encoding property.

- Super Class** [Object](#)

## Constructor

| Name            | Parameters   | Description   |
|-----------------|--|---|
| TextInputStream | Handle <a href="#">as Integer</a> ,<br>Type as <a href="#">Integer</a> | Type is one of the HandleType class constants and Handle is the appropriate handle type specified by the Type parameter. The HandleType class constants are as follows:<br>1- HandleTypeWin32Handle. A Windows32 OS handle.<br>2- HandleTypeFilePointer. A file pointer.<br>3- HandleTypeFileNumber. A file descriptor.<br>4- HandleTypeMacFileRefNum. A file reference number.<br>5- HandleTypeMacFileSpecPointer. An FSSpec. For instance, you can use a <a href="#">Declare</a> to open a file with whatever permissions that you wish, and then pass the Handle to a stream object's constructor. |

## Properties

| Name          | Type                         | Description   |
|---------------|------------------------------|---|
| Encoding      | <a href="#">TextEncoding</a> | Specifies the encoding to be defined for a string returned by ReadLine or ReadAll. It does not actually convert the bytes, but only assigns them an encoding, as if you had called <a href="#">DefineEncoding</a> . Use the <a href="#">Encoding</a> object to specify the <a href="#">TextEncoding</a> . It defaults to UTF-8, but you can assign it a different encoding to match your file, or even assign <a href="#">Nil</a> if you want the string to have an undefined encoding. |
| Handle        | <a href="#">Integer</a>      | Parameter is Type as <a href="#">Integer</a> . The class constants given below can be passed as the parameter.<br>1- HandleTypeWin32Handle. A Windows32 OS handle.<br>2- HandleTypeFilePointer. A file pointer.<br>3- HandleTypeFileNumber. A file descriptor.<br>4- HandleTypeMacFileRefNum. A file reference number.<br>5- HandleTypeMacFileSpecPointer. An FSSpec.<br>Handle returns a handle of the Type passed or -1 if the requested Type cannot be retrieved.                    |
| LastErrorCode | <a href="#">Integer</a>      | Reports the last file I/O error. Error numbers are given as original file system error codes.   |
| PositionB     | <a href="#">Integer</a>      | Indicates the byte position of the file pointer, not the character position.  |

## Methods

| Name  | Parameters | Description      |
|-------|------------|------------------|
| Close |            | Closes the file. |

| Name     | Parameters                                  | Description   |
|----------|---|---|
| ReadAll  | [Encoding as <a href="#">TextEncoding</a> ] | Returns all of the text (as a <a href="#">string</a> ) from the TextInputStream. The optional <i>Encoding</i> parameter enables you to specify the encoding of the text. If you pass <a href="#">Nil</a> , the default encoding is used. This is usually UTF-8, unless it was set to another encoding via an assignment statement. If you want to set the encoding to <a href="#">Nil</a> , use the <i>Encoding</i> property instead.       |
| ReadLine | [Encoding as <a href="#">TextEncoding</a> ] | Returns the next line of text (as a <a href="#">string</a> ) from the TextInputStream. The optional <i>Encoding</i> parameter enables you to specify the encoding of the text. If you pass <a href="#">Nil</a> , the default encoding is used. This is usually UTF-8, unless it was set to another encoding via an assignment statement. If you want to set the encoding to <a href="#">Nil</a> , use the <i>Encoding</i> property instead. |

### Notes

The **TextInputStream** class implements the [Readable](#) class interface. If you implement this interface, you must provide the methods with the parameters as specified by this class interface.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

When you read a text file that is from another operating system or in another language (or a mixture of languages) you may need to specify the text encoding that was used when the file was written. If you know the encoding, use the [Encodings](#) object to get the encoding and use it to set the value of the *Encoding* property of the **TextInputStream** object. Here is an example that reads a text file that uses the MacRoman encoding:

```
Dim f As FolderItem
Dim t As TextInputStream
f=GetOpenFolderItem("text") //file type defined in File Type Sets Editor
If f <> Nil then
  t=f.OpenAsTextFile
  t.Encoding=Encodings.MacRoman //specify encoding of input stream
  EditField1.text=t.ReadAll
  t.Close
End if
```

To specify the encoding, you can instead use optional parameter of the *ReadAll* method:

```
EditField1.Text=t.ReadAll(Encodings.MacRoman)
```

instead of

```
t.Encoding=Encodings.MacRoman
```

**Examples** This example reads the rows and columns of data from a tab-delimited text file into a [ListBox](#):

```
Dim f As FolderItem
Dim textInput As TextInputStream
Dim rowFromFile,oneCell As String
Dim i As Integer

f=GetOpenfolderItem\("text/plain"\) //defined as a FileType

If f <> Nil Then
    textInput = f.OpenAsTextFile
    textInput.Encoding=Encodings.MacRoman //strings are MacRoman
    Do
        rowFromFile=textInput.ReadLine
        If ListBox1.ColumnCount < CountFields\(rowFromFile,Chr\(9\)\) Then
            ListBox1.ColumnCount=CountFields\(rowFromFile,Chr\(9\)\)
        End If
        ListBox1.AddRow NthField\(rowFromFile,Chr\(9\),1\)
        For i=1 to CountFields\(rowFromFile,Chr\(9\)\)
            oneCell=NthField\(rowFromFile,Chr\(9\),i\)
            ListBox1.Cell(ListBox1.ListCount-1,i-1)=oneCell
        Next
    Loop Until textInput.EOF
    textInput.Close
End If
```

## See Also

[ConvertEncoding](#), [DefineEncoding](#), [Encoding](#) functions; [BinaryStream](#), [TextEncoding](#), [TextOutputStream](#) classes; [Encodings](#) object; [Readable](#) class interface.

---

## TextOutputStream Class

In order to write text to a file, you need to create a **TextOutputStream** object. TextOutputStreams have methods that allow to write to a file and close the file when you are done writing to it. They are created by calling the CreateTextFile and AppendToTextFile methods of a [FolderItem](#) object. If you need to specify the encoding of the text, call [ConvertEncoding](#) prior to writing the file.

**Super Class** [Object](#)

## TextOutputStream Class

---

### Constructor

| Name             | Parameters   | Description   |
|------------------|--|---|
| TextOutputStream | Handle <a href="#">as Integer</a> ,<br>Type as <a href="#">Integer</a> | Type is one of the HandleType class constants and Handle is the appropriate handle type specified by the Type parameter. The HandleType class constants are as follows:<br>1- HandleTypeWin32Handle. A Windows32 OS handle.<br>2- HandleTypeFilePointer. A file pointer.<br>3- HandleTypeFileNumber. A file descriptor.<br>4- HandleTypeMacFileRefNum. A file reference number.<br>5- HandleTypeMacFileSpecPointer. An FSSpec. For instance, you can use a <a href="#">Declare</a> to open a file with whatever permissions that you wish, and then pass the Handle to a stream object's constructor. |

### Properties

| Name          | Type                    | Description  |
|---------------|-------------------------|--|
| Delimiter     | <a href="#">String</a>  | The character used to mark the end of a line of text written to the file. The default is a carriage return.  |
| Handle        | <a href="#">Integer</a> | Parameter is Type as <a href="#">Integer</a> . The class constants given below can be passed as the parameter.<br>1- HandleTypeWin32Handle. A Windows32 OS handle.<br>2- HandleTypeFilePointer. A file pointer.<br>3- HandleTypeFileNumber. A file descriptor.<br>4- HandleTypeMacFileRefNum. A file reference number.<br>5- HandleTypeMacFileSpecPointer. An FSSpec.<br>Handle returns a handle of the Type passed or -1 if the requested Type cannot be retrieved. |
| LastErrorCode | <a href="#">Integer</a> | Reports the last file I/O error. Error numbers are given as original file system error codes.  |

### Methods

| Name      | Parameters                     | Description  |
|-----------|--------------------------------|--|
| Close     |                                | Closes the file.   |
| Write     | Text as <a href="#">String</a> | Writes the text passed to the TextOutputStream.  |
| WriteLine | Text as <a href="#">String</a> | Writes the text passed to the textOutputStream and appends the Delimiter to the end of the line. |

### Notes

The **TextOutputStream** class implements the [Writable](#) class interface. If you implement this interface, you must provide the methods with the parameters as specified by the class interface.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

If you need to write the file using a particular encoding, use the [ConvertEncoding](#) function to first convert the encoding of the text to the desired encoding before passing the text to the Write or WriteLine methods. Here is an example:

```
Dim f As FolderItem
Dim t as TextOutputStream
f = GetFolderItem("Sample.txt")
t = f.CreateTextFile
t.Write ConvertEncoding(Editfield1.text, Encodings.WindowsANSI)
t.close
```

### Example

This example displays the Save As dialog box. A text file is then created and the text properties of three [EditFields](#) are written to the new file. Finally the file is closed.

```
Dim file As FolderItem
Dim fileStream As TextOutputStream
file=GetSaveFolderItem("application/text","My Info")
fileStream=file.CreateTextFile
fileStream.WriteLine namefield.text
fileStream.WriteLine addressfield.text
fileStream.WriteLine phonefield.text
fileStream.Close
```

### See Also

[BinaryStream](#), [FolderItem](#), [TextInputStream](#) classes; [ConvertEncoding](#) function; [Encodings](#) object; [Writeable](#) class interface.

## The Counter variable and the Variable Following Next Must be the Same Error

In a [For](#) loop, if you use both a counter variable and a variable in the Next statement, they must match. The variable on the Next statement can be omitted.

### Example

In the following [For](#) loop, the counter variable is i, so the variable on the Next statement must also be i.

```
Dim i,j as Integer
For i=1 to 10
    j=j+1
    Next j
```

### See Also

[For](#) statement.

## The keyword 'Then' is expected after this If statement's condition Error

You neglected to type the 'Then' keyword after the [Boolean](#) condition in an [If](#) statement. To eliminate the possibility of this error, type Shift+Enter after typing

**IF condition**

REALbasic will then add the "Then" after the *condition* and then add the End If statement.

Another shortcut is to write only the statements that meet the condition (inside the If...End if), select them, and then choose Wrap in If...End If from the Code Editor contextual menu. This will surround your statements with IF condition Then above and End If below.

### Example

```
If error=-123 //needs a 'Then' right here
    Beep
    MsgBox "Whoops! An error occurred."
End If
```

### See Also

[If](#) statement.

---

## The Size of an Array Must Be a Constant or a Number Error

When you declare and size an array using a [Dim](#) statement, the size of the array must be indicated by either a constant or a number.

### Example

Trying to size an array using a variable:

```
Dim j as Integer
j=5
Dim a(j) as Integer
```

### Notes

If you need a variable length array, declare the array as having no bounds and then use the [Append](#) method to add elements, such as:

```
Dim myArray (-1) as Integer
myArray.Append(5)
```

**See Also** [Dim](#) statement.

## There is no Class with this Name Error

You used a nonexistent class name.

**Examples** Using nonexistent class names on a [Dim](#) statement.

```
Dim Fred as Husband // 'Husband' not a data type
Dim f as Folder // should be FolderItem
Dim d as longdouble // no such thing; programmer is lost
```

Using a nonexistent class with the [New](#) operator.

```
Dim f as FolderItem
f=New Folder
```

**See Also** [Dim](#) statement; [New](#) operator.

## This Array Has Fewer Dimensions than You Have Provided Error

You tried to execute a built-in array method that requires a one-dimensional array on a multi-dimensional array.

**Example** You can't sort a two-dimensional array.

```
Dim alnts(3,3) as Integer
Dim i,j as Integer
For i=0 to 2
  For j=0 to 2
    alnts(i,j)=i*j
  Next
Next
alnts.Sort // doesn't work on 2D arrays
```

Using syntax that indicates that you are referencing an element in a two-dimensional array when it has been declared as a one-dimensional array:

```
Dim alnts(3) as Integer
alnts(0,0)=3 // incorrect
```

## This Array Has More Dimensions than You Have Provided Error

---

Trying to change a one-dimensional array to a two-dimensional array with a [Redim](#) statement:

```
Dim alnts(3) as Integer
```

```
Redim alnts(5,5)
```

**See Also** [Append](#), [Insert](#), [Redim](#), [Remove](#), [Sort](#) methods; [Dim](#) statement.

## This Array Has More Dimensions than You Have Provided Error

---

In referring to an element of a multidimensional array, you didn't provide subscripts for all the dimensions.

### Example

```
Dim a(5,4) as String  
a(1)="Bob"
```

**See Also** [Dim](#) statement.

## This Array Method Works for only One-dimensional Arrays

---

You tried to execute a method that operates only on one-dimensional arrays on a multi-dimensional array.

**Example** The following code tries to use the [Append](#) method with a two-dimensional array.

```
Dim aTest(3,3) as String  
Dim i,j as Integer  
For i=0 to 3  
  For j=0 to 3  
    aTest(i,j)=Str(i)+Str(j)  
  Next  
Next  
aTest.Append "Frank" //can't use Append here.
```

**See Also** [Append](#), [Array](#), [IndexOf](#), [Insert](#), [Join](#), [Pop](#), [Remove](#), [Sort](#), [Split](#) methods.

## This #else Does Not Have a Matching #If Preceding It Error

An extra #else keyword was used in an [#If](#) statement (conditional compilation) or an #else statement is used where no #if preceded it.

**Example** An #else keyword was used instead of an Else keyword in an [If](#) statement.

```
Dim theNumber As Integer
Dim digits As Integer
theNumber=33
If theNumber<10 Then
    digits=1
#else //should be else
    digits=3
End If
```

**See Also** [#If](#), [If](#) statements.

---

## This #endif Does Not Have a Matching #If Preceding It Error

An [#If](#) statement was omitted from an #if...#endif structure.

**Example** The [If](#) statement in this example should be [#If](#).

```
If TargetMacOS then
    Beep
#endif
```

**See Also** [#If](#) statement.

---

## This Class is Missing One or More Methods of an Interface it Implements Error

One or more of the methods of the class interface that this class implements could not be found in this class, even though it claims to implement the class interface.

When you specify the properties of a class in its Properties pane, you can optionally specify a class interface for the class. When you do so, the class must have its own

## This Global Variable has the Same Name as a Class Error

---

methods of the same names as the names of the methods in this class interface. The matching methods must use the same parameter types and return types as the corresponding method in the class interface.

If you omit one or more of these methods, then you will receive this error. When you attempt a debug build, a dialog box will appear that gives this error message and contains a list of all the methods that the class is required to have.

## This Global Variable has the Same Name as a Class Error

---

You created a global variable in a module that uses the same name as one of the classes in your project.

## This Global Variable has the Same Name as a Global Function Error

---

You reused the name of a method or function in a module as the name of a property in a module.

**Example** Defining “myMessageBox” as a method in a module and also defining “myMessageBox” as a property in the module.

## This Global Variable has the Same Name as a Module Error

---

You reused the name of a module as the name of a module property whose scope is Global.

**Example** Defining a module named “Arthur” and creating a property in the module that’s also named “Arthur” and has Global scope.

## This is not An Array but you are using it as One Error

You passed an object to a method or function that expects an array but did not get one. It also occurs if you are trying to assign a value to an element of an alleged array but it is not an array.

### Examples

```
Dim alnts(4) as Integer  
Dim f as FolderItem  
Redim f(10) //not an array
```

```
Dim a as String  
a(1)="Ted" //not an array
```

**See Also**    [Dim](#) statement; [Redim](#) operator.

---

## This Item Conflicts with Another Item of the Same Name Error

Occurs when you are calling an overloaded method and REALbasic cannot determine which version you meant to call. This happens when two methods have the same name, number of parameters, and the data types of the parameters match.

This error will also occur if a method name conflicts with another type of project item such as a window, menuitem, control, and so forth. For example, if you have a global function named Foo and a window called Foo.

**Example**    The method myOverLoadedMethod takes three integer parameters, but it is defined twice (performing different functions) in a window's Code Editor. A call to myOverLoadedMethod produces the error. The solution is to rename or eliminate the second instance of myOverLoadedMethod.

### Instance 1

```
Sub myOverLoadedMethod(a as Integer, b as Integer, c as Integer)  
    Dim d as Integer  
    d=a*b/c  
    MsgBox str(d)
```

## This Kind of Array Cannot be Sorted Error

---

### Instance2

```
Sub myOverLoadedMethod(x as Integer, y as Integer, z as Integer)
    Dim d as Integer
    d=x/z*y
    MsgBox str(d)
```

#### Calling method

```
Dim a,b,c as Integer
a=5
b=10
c=15
myOverloadedMethod(a,b,c)
```

---

## This Kind of Array Cannot be Sorted Error

You tried to sort a [Boolean](#) array or an array of objects.

### Example

You tried to sort an array of objects instead of an array of alphas or numbers. For example, an array of [Date](#) objects cannot be sorted, but an array of TotalSeconds properties of [Date](#) objects can be sorted, since the TotalSeconds property is a [Double](#).  
This example array cannot be sorted.

```
Dim d(2) as Date
d(0)=New Date
d(1)=New Date
d(2)=New Date
d(0).year=1965
d(1).year=1970
d(2).year=1950
d.Sort
```

The last line of this example produces the error message.

```
Dim theTruth(3) as Boolean
theTruth(0)=True
theTruth(1)=True
theTruth(2)=False
theTruth(3)=False
theTruth.Sort
```

### See Also

[Sort](#) method.

## This Local Variable or Constant has the Same Name as a Declare in this Method Error

You used the name of a local variable or local constant in a [Declare](#) statement in a method.

**See Also** [Declare](#) statement.

## This Local Variable or Parameter has the Same Name as a Constant Error

You reused the name of a constant in a [Dim](#) statement or a parameter declaration.

**Examples** Reusing the name of a constant in a local variable.

```
Const PI=3.14  
Dim PI as Double  
pi=31.416
```

Reusing the name of a parameter as a constant:

```
Sub myGreatGlobalMethod (PI as Double)  
Const PI=3.1416
```

**See Also** [Const](#), [Dim](#) statements.

## This Local Variable or Parameter has the Same Name as a Constant Error

You declared a local variable in a method or a parameter in a [Sub](#) or [Function](#) that has the same name as a constant.

**Example** You cannot use the same name for a constant and a local variable:

```
Const Version="10B"  
Dim Version as String
```

**See Also** [Const](#), [Dim](#) statements.

## This Method Doesn't Return a Value Error

You used a method in an expression as though it returned a value, but it doesn't.

- Example** The user-written method myColor is not a function, so this syntax in a calling method is not correct:

```
Dim c as Color  
c=myColor //does not return a color
```

- See Also** [Function](#), [Return](#), [Sub](#) statements.

---

## This Method is Protected. It can only be Called From Within its Class Error

You tried to access a protected method from outside the object that owns it. A protected method is accessible only from within its parent object—its class, window, or module.

You can unprotect a method by choosing either the Public or Global scope in the Add Method declaration area.

---

## This Method is too Long Error

The method is longer than the compiler can handle. It is very unlikely that you will see this error.

---

## This Method or Property Does Not Exist Error

You used a syntax that implies that a term is a method or property, but it does not exist. It also occurs when you use a variable in an assignment statement but omit the equals sign, making it appear that it is a method or property.

- Examples** Calling a method that has not been defined:

```
Dim aFiles(3) as FolderItem  
aFiles.List //no such thing
```

Assigning a value to a variable that has not been declared using a [Dim](#) statement:

```
i=10
```

This is the Action event handler of a [PushButton](#); the user wants to display the caption property of the [PushButton](#).

```
MsgBox caption //should be me.caption*
```

Trying to set in code a property that can only be set in the Properties pane. The following line of code tries to set the value of the Bold property of a [MenuItem](#), but REALbasic does not recognize the Bold property in code.

```
SearchFind.Bold=True
```

Using [Me](#) in a method or function that is not attached to an object. The following method tries to return the caption property of the calling [PushButton](#), but this is not permitted.

```
Function myCaption() as String  
Return Me.Caption
```

You can only access the calling object's properties using [Me](#) within an event handler belonging to that object.

The programmer intended to declare aNames as an array in the [Dim](#) statement but forgot to do so.

```
Dim aNames as String  
aNames.Append "Walter Kerr"
```

Trying to assign a value to a Private or Protected property that is out of scope. For example, the following line tries to assign a value to a Private property of a module from a window:

```
myPrivateProperty=56
```

### See Also

[Dim](#) statement.

---

## This Method Requires Fewer Parameters than were Passed Error

You passed too many parameters to a method. Another possibility is that you passed the required number of parameters but did not separate them with commas.

## This Method Requires More Parameters than were Passed Error

---

**Examples** Passing the required number of parameters but omitting a comma:

```
ListBox1.AddRow 2 "Chicago"
```

Passing too many parameters:

```
listBox1.insertRow 2,"Charlie","Dallas"
```

The user-written function, myFunction, takes two parameters:

```
Dim d as Double  
d=myFunction(5,6,8)
```

---

## This Method Requires More Parameters than were Passed Error

You didn't pass all the required parameters to a method. Another possibility is that you passed the required number of parameters but did not separate them with commas.

**Example** Passing the required number of parameters but omitting a comma:

```
ListBox1.insertRow 2 "Chicago"
```

Passing too few parameters:

```
ListBox1.insertRow "Chicago"
```

```
Dim c as Color  
c=RGB(100,100) //returns "RGB takes 3 parameters"t
```

In the following examples, myFunction takes two parameters and returns a [Double](#):

```
d=myFunction(5,6) //correct  
d=myFunction (5,6) //also correct  
d=myFunction 5,6 //error  
d=myFunction //also an error
```

## This Name is Already in Use Error

You used a variable name in a [Dim](#) statement that has already been declared in another [Dim](#) statement or as a property. Or, you tried to declare a property with the same name as another property.

### Example

```
Dim a,b as Double  
Dim b as Double //already in use
```

### See Also

[Dim](#) statement.

## This Property is Protected. It can only be Used From Within its Class Error

You tried to access a property from outside the class that owns it. A Protected property can be used only within its parent class, window, or module.

## This Property Has the Same Name as a Class Method Error

You reused a name of a method in a class as the name of a property in the class.

## This Property Has the Same Name as an Event Error

You used the name of an event as the name of a property. You need to rename the property.

### Example

Using 'Open' as both the name of an event and a property in a class.

## This Property Has the Same Name as a Method Error

You reused the name of a method as the name of a property.

## This Type Conversion is only implemented for one-dimensional Arrays Error

You tried to convert the data type for a multi-dimensional array.

---

## Thread Class

Threads execute code in the background, independent of the user interface.

Super Class [Object](#)

### Events

| Name | Parameters | Description  |
|------|------------|--|
| Run  |            | The Run method has been called. The threaded code is placed in this event handler. |

### Properties

| Name      | Type                    | Description  |
|-----------|-------------------------|--|
| Priority  | <a href="#">Integer</a> | <p>Gets and sets the relative priority of the thread. The application's main thread has a priority of 5 and this is also the default value of any programmatically created threads. If you don't modify this value, your application will behave normally, in the sense that all the threads will share the CPU's resources equally. Increasing the value of <i>Priority</i> above 5 gives that thread more processing cycles than other threads. Doubling the value of <i>Priority</i> will double the number of processing cycles the thread gets. A thread with a <i>Priority</i> of 1 will get one-fifth the processing cycles as the main thread. The range is 1 to 2^31-1, but very high values of <i>Priority</i> will make it difficult for other threads to run at all. The value of <i>Priority</i> changes the way in which threads get processing cycles only if their values are not all equal. You can use the following class constants to assign to <i>Priority</i> or to compare its value:</p> <p>1:LowestPriority<br/>5:NormalPriority<br/>10: HighPriority</p> <p>You do not need to use these constants; you can use any legal integer value instead.</p> |
| StackSize | <a href="#">Integer</a> | Size of the stack, in bytes. Can be set when the thread is not running. The default stack size is 64K on Macintosh and Linux and 904K on Windows. A StackSize of zero indicates the default.   |

| Name     | Type                    | Description  |
|----------|-------------------------|--|
| State    | <a href="#">Integer</a> | Indicates the state of the thread. A thread can be in one of four states, which can be identified via the following class constants:<br>0 - Running.<br>1 - Waiting. The thread has been blocked by a call to Signal or Enter from one of the locking mechanisms, <a href="#">CriticalSection</a> , <a href="#">Mutex</a> , or <a href="#">Semaphore</a> .<br>2- Suspended. It will not execute because of a call to the Suspend method.<br>3- Sleeping. It has been put to sleep by a call to the Sleep method.<br>4- NotRunning. The thread will be in this state prior to a call to Run or after the thread has finished running. |
| ThreadId | <a href="#">Integer</a> | ID of the thread, assigned at runtime.   |

## Methods

| Name    | Parameters   | Description  |
|---------|--|--|
| Kill    |  | Kills the current thread. Internally, this method will cause your thread's stack to unwind, destructors for local objects to be called, and so forth. It's a graceful exit which can cause code to execute. It may also cause context switches.  |
| Resume  |  | Wakes up a thread that is suspended because of a call to Suspend or sleeping because of a call to Sleep. When you call Resume, the thread reacquires its ability to receive processing cycles according to its value of <i>Priority</i> . Calling Resume will not cause a context switch to take place, that is, a switch to the next thread in the queue.   |
| Run     |  | Executes the thread's Run event handler.   |
| Sleep   | Milliseconds as <a href="#">Integer</a> , WakeEarly as <a href="#">Boolean</a> | Puts the thread to sleep for the specified amount of time. By default, <i>WakeEarly</i> is <a href="#">False</a> . If it is not set to <a href="#">True</a> , the thread will be put to sleep for the full amount of time specified by <i>Milliseconds</i> . If you set <i>WakeEarly</i> to <a href="#">True</a> , the thread can be resumed early. It will wake up early if there are no other threads able to execute. |
| Suspend |  | Puts the thread in a suspended state. The thread will not be allocated any processing cycles, regardless of its assigned <i>Priority</i> . It is "asleep." To wake it up, call the Resume method. If the thread is executing when you call Suspend, it will immediately cause a context switch.  |

## Notes

By default, a REALbasic application runs in the main thread. This thread is not accessible via methods of the **Thread** class; it is managed internally. All other threads are accessible and share execution time among each other and the main thread.

Threads are managed by REALbasic's Thread Scheduler. Its task is to allocate the CPU's processing cycles among all of the application's threads. The *Priority* property

determines how many or how few processing cycles the thread gets, relative to the main thread and any other threads that you have created.

When the Thread Scheduler decides to stop execution of the current thread and allow another thread to run, it is called a *context switch*. The amount of time a thread runs is called the *time slice* for the thread.

**Thread** class objects are used to execute code that needs to run independently of the user interface. The Thread Scheduler allocates processing cycles between the main thread and any threads you create via this class.

For example, if you were drawing a complex image that takes several minutes to render, this code could be executed in a thread so that the user could continue using the user interface without having to wait for the rendering to finish.

A thread can be used to allow a very time-consuming loop to run in the background. Since unthreaded loops take over the interface, preventing the user from making menu selections or interacting with a window's controls, this is a good way to allow the user to continue working with the application while a lengthy process is taking place.

Threads yield time to other threads and other applications each time they execute a looping construct such as [For](#), [While](#), and [Do](#).

To create a thread, create a new class and make **Thread** its super class. Add the new class to the Project. Put the code you wish to execute in the thread's Run event handler. To execute the thread, simply create a new instance of this object and call its Run method. You can do this by adding the instance of the **Thread** class to a window, as shown in the example, or instantiate it in the usual way.

You instead use the [Semaphore](#), [CriticalSection](#), or [Mutex](#) classes to manage access to shared resources among multiple threads. For example, if several threads may need to access the same file, you can use a [CriticalSection](#) to be sure that two or more threads are not trying to access the same file simultaneously.

Using the DoEvents method of the [Application](#) class from a multithreaded application is not supported and will most likely generate extremely hard to track down errors and crashes in your applications.

**Example**

To set up this example, first add a new class to the Project called *LongProcessThread* and set its Super Class to **Thread**. In the Run event, place the following code:

```
Dim i as Integer
Dim s as String
Const c = 186000
Const bigNum = 100000000

For i = 1 to bigNum
  If i / 1000 = i \ 1000 then
    //do something really tedious here...anything you want
    s = Str(i/c*c*c) + " for " + Str(bigNum)
  //just try to update the interface
  Window1.ListBox1.AddRow s //add a row to the ListBox
End if
Next
Msgbox "Finished"
```

Add an instance of the LongProcessThread class to the default window, Window1. Add a ListBox, ListBox1, to the default window and a PushButton to the window with the following Action event handler:

**LongProcessThread1.Run**

This event handler does the calculations as a new thread. To see how the user-created thread makes a difference, create a second PushButton in the window and use the code in the Thread's Run event as the PushButton's Action event. When this button is clicked, REALbasic will do the same calculations, but in the main thread rather than in the thread you created.

When you do the calculation in its own thread, you will see that REALbasic is able to update the ListBox while the calculation is in progress and allows the user to manipulate interface items, such as scroll the ListBox and choose menu items. However, when you do the calculation inside the main thread, the interface is locked up until the calculation is finished.

**See Also**

[Application](#), [CriticalSection](#), [Mutex](#), [Semaphore](#) classes; [#Pragma](#) directives; [ThreadAlreadyRunningException](#).

**ThreadAlreadyRunningException Error**

You tried to set the stack size of a thread that is already running or tried to call the Run method for a thread that is already running.

**Super Class** [RuntimeException](#)

**See Also** [CriticalSection](#), [Semaphore](#), [Mutex](#), [RuntimeException](#), [Thread](#) classes; [Exception](#), [Try](#) blocks; [Catch](#) statement.

---

## Ticks Function

Returns the number of ticks (60th of a second) that have passed since the user's computer was started.

**Syntax**

**result=Ticks**

| Part   | Type                    | Description   |
|--------|-------------------------|---|
| result | <a href="#">Integer</a> | The number of ticks that have passed while the computer is operating. |

**Notes**

Because modern operating systems can stay running for so long, it's possible for the machine's internal counters to "roll over." This means that if you are using this function to determine how much time has elapsed between two events, you may encounter a case where it appears that the stop time is prior to the start time.

**Examples**

This example displays in message box the number of minutes the computer has been on.

```
Dim minutes As Integer  
minutes=Ticks/60/60  
MsgBox "Your computer has been on for"+Str(minutes)+" minutes."
```

**See Also**

[Microseconds](#) function.

---

## Timer Object

A **Timer** is an object that can execute code after a specified period of time or at a repeated interval. If added to a window via the Window Editor, it is not visible in the built application since it is not a control.

**Super Class** [Object](#)

Because it is subclassed from [Object](#) rather than [RectControl](#), you can instantiate it via code with the [New](#) operator.

## Properties

| Name    | Type                    | Description   |
|---------|-------------------------|---|
| Enabled | <a href="#">Boolean</a> | Enables you to turn the Timer on or off. When Enabled is <a href="#">True</a> , the Timer is on.  |
| Index   | <a href="#">Integer</a> | The number of the control when it is part of a control array.   |
| Left    | <a href="#">Integer</a> | The left side of the control in local coordinates (relative to the window).   |
| Mode    | <a href="#">Integer</a> | The interval at which the Action event will be executed. The following class constants are available:<br>ModeOff (0) - Off,<br>ModeSingle (1) - Single,<br>ModeMultiple (2) - Multiple<br>For example: timer1.Mode=Timer.ModeSingle<br>The default is 2, Multiple.  |
| Name    | <a href="#">String</a>  | The name of the object.   |
| Period  | <a href="#">Integer</a> | The time (in milliseconds) between executions. Periods of less than or equal to zero default to a value of 1 millisecond. The default value is 1000. The rate that a Timer can actually fire depends on the speed of the host computer, the operating system, and other tasks the computer is doing. It is possible to set Period to a value that cannot be achieved by the computer that is running the application. |
| Top     | <a href="#">Integer</a> | The top of the control in local coordinates (relative to the window).   |

## Methods

| Name  | Parameters | Description                       |
|-------|------------|-----------------------------------|
| Reset |            | Resets the Timer and restarts it. |

## Events

| Name   | Parameters | Description  |
|--------|------------|--|
| Action |            | The code that will execute after the Period specified and in the Mode specified. |

## Notes

Although the **Timer** appears in the list of Built-in controls in the Window Editor, this is done only as a convenience to programmers. In terms of the object hierarchy, the **Timer** is not a control. This means you can create Timers in your code via the [New](#) operator, just as with other objects.

The Mode property controls the interval used to execute the **Timer**'s Action event handler. If the Mode is not 0 (ModeOff), the Timer waits for the Period to pass before executing the Action event handler. If the Mode is set to ModeOff at Runtime, the

## Titlecase Function

---

The **Timer** will immediately cease waiting and the Action event handler will no longer be executed.

The **Timer** will continue to execute its Action event handler (assuming the Mode property is not set to ModeOff) regardless of whether the window is frontmost or not. The visibility of the window also has no impact on the execution of a **Timer**'s Action event handler. The **Timer** has been designed so that it yields time to other applications running on the computer so as not to bog down the machine.

### Examples

The safest way to run the animation is to update it via a [Timer](#). Add a [Timer](#) to the window that contains the **SpriteSurface**. Adjust its Period property according to how fast you want the animation to go and set its Mode to 2 (Multiple). In its Action event, call the **SpriteSurface's** Update method.

The [Timer](#) in the window calls the SpriteSurface's Update method in its Action event. Its mode is set to 2.

#### SpriteSurface1.Update

The same approach is recommended for animating [RB3DSpace](#) objects. With the Timer's Mode property set to 2, call the [RB3DSpace](#)'s Update method from the Action event of a **Timer** object.

The following code in the Action event of a **Timer** detects whether the Up, Down, Left, or Right arrow keys are pressed.

```
If Keyboard.AsyncKeyDown(123) then
    StaticText1.text="left arrow key"
end if
If Keyboard.AsyncKeyDown(124) then
    StaticText1.text="right arrow key"
end if
If Keyboard.AsyncKeyDown(125) then
    StaticText1.text="down arrow key"
end if
If StaticText1.AsyncKeyDown(126) then
    StaticText1.text="Up arrow key"
end if
```

### See Also

[Object](#) class.

---

## Titlecase Function

Returns the [string](#) passed to it with all alphabetic characters in Titlecase.

### Syntax

**result=Titlecase(value)**

OR

**result=stringVariable.Titlecase**

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | A copy of the original string with all characters converted to their titlecase equivalent. |
| value          | <a href="#">String</a> | The original string.   |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .  |

## Notes

Converts all characters in a [string](#) to lowercase characters and then converts the first character of each word to uppercase. Numbers are not affected.

## Examples

The example below converts the values passed to **Titlecase**.

```
Dim s As String
s=Titlecase("tHe Quick fOX") //returns "The Quick Fox"
s=Titlecase("THE LAZY DOG") //returns "The Lazy Dog"
```

## See Also

[Lowercase](#), [Uppercase](#) functions.

# ToolbarItem Control

Used to create a toolbar in a window. Supported on Mac OS X 10.2 and above only. ToolbarItems can be placed in windows for other platforms for layout purposes only.

Super Class [Control](#)

## Properties

| Name         | Type                    | Description  |
|--------------|-------------------------|--|
| Caption      | <a href="#">String</a>  | The text label below the ToolbarItem. See Notes.   |
| ControlOrder | <a href="#">Integer</a> | Determines the left to right placement of the icons on the Toolbar.  |
| Enabled      | <a href="#">Boolean</a> | <a href="#">True</a> if the ToolbarItem is enabled.  |
| Handle       | <a href="#">Integer</a> | A toolbarItemRef, used in <a href="#">Declare</a> statements, of use only to Macintosh toolbox programmers.                |
| HelpTag      | <a href="#">String</a>  | Standard help tag for the item.  |
| HelpTagLong  | <a href="#">String</a>  | Help tag that appears when the user holds down the Command key.  |
| Image        | <a href="#">Picture</a> | The picture to be displayed by the ToolbarItem. If no picture is associated with the ToolbarItem, a "?" icon is displayed. |

## ToolbarItem Control

---

| Name | Type                    | Description   |
|------|-------------------------|---|
| Left | <a href="#">Integer</a> | The distance (pixels) from the left of the window in the Window Editor. In the runtime application, the ordering of toolbar items is strictly based left to right on control order. |
| Top  | <a href="#">Integer</a> | The distance (pixels) from the top of the window in the Window Editor. In the runtime application, the ordering of toolbar items is strictly based left to right on control order.  |

## Events

| Name   | Parameters | Description                                     |
|--------|------------|---|
| Action |            | Occurs when the user clicks on the ToolbarItem. |
| Close  |            | Occurs when the ToolbarItem is destroyed.       |
| Open   |            | Occurs when the ToolbarItem is first created.   |

## Methods

| Name             | Parameters  | Description   |
|------------------|---|---|
| SetImageFromIcon | Creator as <a href="#">String</a><br>Type as <a href="#">String</a> | Assigns a picture to the Image property using an icon registered with the system using the Type and Creator passed. |

The following are popular icons for use with SetImageFromIcon:

| Creator | Type | Description |
|---------|------|-------------|
| macs    | tdel | Delete      |
| macs    | tfav | Favorites   |
| macs    | thom | Home        |
| macs    | burn | Burn        |
| macs    | ejec | Eject       |
| macs    | CLIP | Clipboard   |
| macs    | desk | Desktop     |
| macs    | FNDR | Finder      |
| macs    | trsh | Trash       |
| macs    | ftrh | Full Trash  |
| macs    | docu | Document    |
| macs    | gurl | URL         |
| macs    | udsk | iDisk       |
| macs    | ilht | HTTP        |
| macs    | ilft | FTP         |
| macs    | gnet | Network     |

For a complete list of Apple-supplied icons, see Icons.h in the Universal Headers.

#### Note

If the Caption property contains an ampersand character, it will display only if it is preceded by another ampersand character. For example, if you set the Caption property to “Bob & Ray”, only “Bob Ray” will display. You need to set it to “Bob && Ray”. The ampersand character is used on Windows to signify the keyboard accelerator.

#### See Also

[StandardToolbarItem](#) control.

---

## TrashFolder Function

Used to access the items in the Trash Can or Recycle Bin.

#### Syntax

**result=TrashFolder**

| Part   | Type                       | Description   |
|--------|----------------------------|---|
| result | <a href="#">FolderItem</a> | A <a href="#">FolderItem</a> that represents the Trash folder or Recycle Bin. |

|                  |   |
|------------------|---|
| <b>Notes</b>     | The Trash folder is represented on desktop by the Recycle Bin or Trash Can icons. When a user drags items in or out of the Recycle Bin/Trash Can, they are in fact, dragging those items in or out of the Trash folder.<br><br>The <b>TrashFolder</b> function provides a way to access the Trash folder that will work under different languages and operating systems.<br><br>The <a href="#">SpecialFolder</a> module enables you to access many special folders that are managed by the OS. |
| <b>Windows</b>   | On Windows, a call to TrashFolder returns a <a href="#">FolderItem</a> that references the current user's Recycle Bin.  |
| <b>Macintosh</b> | On Mac OS X, TrashFolder returns a <a href="#">FolderItem</a> that references the invisible ".Trash" folder for the current user.   |
| <b>Linux</b>     | On Linux, TrashFolder returns a <a href="#">FolderItem</a> that references the invisible ".Trash" folder for the current user. If that directory does not exist, it returns <a href="#">Nil</a> .   |
| <b>Example</b>   | This example displays the absolute path to the user's Trash folder, if it exists on the user's machine.   |

```
Dim f As FolderItem
f=TrashFolder
if f <> Nil Then
    MsgBox f.AbsolutePath
Else
    MsgBox "The folderItem does not exist."
End if
```

|                 |   |
|-----------------|---|
| <b>See Also</b> | <a href="#">ApplicationSupportFolder</a> , <a href="#">DesktopFolder</a> , <a href="#">FontsFolder</a> , <a href="#">PreferencesFolder</a> , <a href="#">StartupItemsFolder</a> , <a href="#">SystemFolder</a> , <a href="#">TemporaryFolder</a> , <a href="#">SpecialFolder</a> functions. |
|-----------------|---|

---

## TrayItem Class

Used to add items to the System Tray. Windows only.

### Properties

| Name    | Type                    | Description                         |
|---------|-------------------------|-------------------------------------|
| Icon    | <a href="#">Picture</a> | The icon displayed as the TrayItem. |
| HelpTag | <a href="#">String</a>  | Help associated with the TrayItem.  |

## Events

| Name   | Parameters                       | Description  |
|--------|----------------------------------|--|
| Action | Cause as <a href="#">Integer</a> | The user has clicked on the TrayItem. The parameter Cause indicates the type of click. To detect the kind of click, use the following class constants:<br>DoubleClick<br>LeftMouseButton<br>RightMouseButton |

## Example

Add an instance of the **TrayItem** class to your project and then instantiate it in the Open event of the [App](#) object, i.e.,

```
myTrayItem = New MyAppTrayItem
Me.AddTrayItem( myTrayItem )
```

The following code in the Action event of a **TrayItem** handles mouse clicks and double-clicks.

```
If cause = TrayItem.LeftMouseButton then
  MsgBox "Left Mouse Button Down"
elseif cause = TrayItem.RightMouseButton then
  dim mnu as new MenuItem

  mnu.Append( new MenuItem( "&About" ) )
  mnu.Append( new MenuItem( MenuItem.TextSeparator ) )
  mnu.Append( new MenuItem( "E&xit" ) )
  dim results as MenuItem
  results = mnu.Popup

  if results <> nil then
    select case results.Text
      case "E&xit"
        Quit
      case "&About"
        MsgBox "The REALbasic TrayItem demo."
    end select
  end if

elseif cause = TrayItem.DoubleClick then
  MsgBox "Double Click"
end if
```

When the application quits, you call the RemoveTrayItem method of the [Application](#) class. This goes in the Close event handler of the [App](#) object.

```
Me.RemoveTrayItem(myTrayItem)
```

Super Class [Object](#)

## TriangleList Class

Used to store triangle definitions. A triangle is referred to by the values of A, B, and C in an [RB3DSpace](#). Each is an index of a vertex in the vertex list.

Super Class [Object](#)

### Properties

| Name | Type                    | Description   |
|------|-------------------------|---|
| A    | <a href="#">Integer</a> | The index of the A vertex. The parameter is i as <a href="#">Integer</a> , where index is the ith triangle in the list. |
| B    | <a href="#">Integer</a> | The index of the B vertex. The parameter is i as <a href="#">Integer</a> , where index is the ith triangle in the list. |
| C    | <a href="#">Integer</a> | The index of the C vertex. The parameter is i as <a href="#">Integer</a> , where index is the ith triangle in the list. |

### Methods

| Name   | Parameters   | Description   |
|--------|--|---|
| GetABC | Index as <a href="#">Integer</a> ,<br><a href="#">ByRef</a> A as <a href="#">Integer</a> ,<br><a href="#">ByRef</a> B as <a href="#">Integer</a> ,<br><a href="#">ByRef</a> C as <a href="#">Integer</a> | Gets the triangle indexes for the triangle in the list specified by the Index parameter and returns the values of A, B, and C in the passed parameters. |
| SetABC | Index as <a href="#">Integer</a> ,<br>A as <a href="#">Integer</a> ,<br>B as <a href="#">Integer</a> ,<br>C as <a href="#">Integer</a>   | Sets the triangle indexes for the triangle in the list specified by the Index parameter.  |
| Copy   | other as<br><a href="#">TriangleList</a>   | Copies the passed TriangleList to the current <a href="#">TriangleList</a> .  |

### See Also

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

# Trim Function

Returns the [string](#) passed with leading and trailing whitespaces removed.

**Syntax** ***result=Trim(SourceString)***

**OR**

***result=stringVariable.Trim***

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | SourceString with leading and trailing whitespaces removed.                                |
| SourceString   | <a href="#">String</a> | The source, a copy of which, to be returned with leading and trailing whitespaces removed. |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .  |

**Notes** Trim uses the list of unicode “whitespace” characters at <http://www.unicode.org/Public/UNIDATA/PropList.txt>.

**Examples** This example removes the whitespaces from either side of the string passed:

```
Dim s as String
s=Trim(" Hello World ") //Returns "Hello World"
```

**See Also** [Asc](#), [Chr](#), [InStr](#), [Left](#), [Len](#), [LTrim](#), [Mid](#), [Right](#), [RTrim](#) functions.

# Trimesh Class

Used to represent a single triangular mesh in a 3D space. Used within an [RB3DSpace](#).

**Super Class** [Element3D](#)

## Properties

| Name             | Type                         | Description   |
|------------------|------------------------------|---|
| HasVertexColors  | <a href="#">Boolean</a>      | If <a href="#">True</a> , the Trimesh has vertex colors.  |
| HasVertexNormals | <a href="#">Boolean</a>      | If <a href="#">True</a> , the Trimesh has vertex normals  |
| HasVertexUVs     | <a href="#">Boolean</a>      | If <a href="#">True</a> , the Trimesh has vertex UV data. |
| Material         | <a href="#">Material</a>     | The material of the Trimesh.                              |
| TriangleCount    | <a href="#">Integer</a>      | The number of triangles in the Trimesh.                   |
| Triangles        | <a href="#">TriangleList</a> | The list of triangles that define the Trimesh.            |
| VertexColors     | <a href="#">ColorList</a>    | The colors of each vertex.                                |

| Name            | Type                       | Description   |
|-----------------|----------------------------|---|
| VertexCount     | <a href="#">Integer</a>    | Number of vertices in the Trimesh.                                |
| VertexNormals   | <a href="#">VectorList</a> | The vertex normals, if HasVertexNormals is <a href="#">True</a> . |
| VertexPositions | <a href="#">VectorList</a> | The vertex positions of the Trimesh.                              |
| VertexUVs       | <a href="#">UVList</a>     | The vertex UVs, if HasVertexUVs is <a href="#">True</a> .         |

**Notes**

A **Trimesh** represents a triangular mesh that contains information about its vertices and faces. For each vertex, it contains information about its normals and colors. The Trimesh class lets you create or modify triangle meshes on-the-fly. With the Material class, you can modify textures and/or colors on-the-fly as well.

Trimesh uses four “helper” classes, [TriangleList](#), [UVList](#), [VectorList](#), and [ColorList](#), to set or get Trimesh data.

**Example**

This example creates one triangle with different vertex colors and places it in the 3D space.

```
Dim tm as New Trimesh

// define the vertices
tm.VertexCount = 3
tm.VertexPositions.SetXYZ(0, 5, 5, 0 )
tm.VertexPositions.SetXYZ(1, 5, 10, 0 )
tm.VertexPositions.SetXYZ(2, 10, 10, 0 )
tm.HasVertexColors=True
tm.VertexColors.Item(0) = &cFF0000
tm.VertexColors.Item(1) = &c00FF00
tm.VertexColors.Item(2) = &c0000FF

// define the triangles
tm.TriangleCount = 1
tm.Triangles.SetABC(0, 0, 1, 2)

// add it to our display!
Rb3DSpace1.objects.Append tm
```

In order to see the **Trimesh**, you need to move the camera back away from the origin. This code in the [RB3DSpace's](#) Open event moves the camera so that you can get a good look at the multicolored **Trimesh**.

```
If Me.Objects = Nil then
  MsgBox "You need to install QD3D or Quesa!"
  Return
End if

Dim v as Vector3D
v=New Vector3D
v.X=10
v.Y=10
v.Z=10

//shift the light source for a better look
Me.FloodDirection.Z = -Me.FloodDirection.Z
Me.Camera.Position=v
```

**See Also** [Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

## True Keyword

Used to set a [Boolean](#) object to the value of 'True' or test whether an existing [Boolean](#) expression is equal to **True**. A comparison of two values that are equal returns **True**.

**Syntax** **expression=True**

Expression is any valid [Boolean](#) expression.

**See Also** [False](#) keyword, [And](#), [Not](#), [Or](#) operators.

## Try Block

Used to handle [RuntimeException](#) errors prior to the [Exception](#) Block.

**Syntax** **Try**

```
//REALbasic statements
Catch [ErrorParameter] [As ErrorType]
//exception handlers
[Finally]
```

## Try Block

//code that executes even if runtime exceptions were raised

End [Try]

| Part           | Description   |
|----------------|---|
| ErrorParameter | Optional, used to determine the type of runtime exception.  |
| ErrorType      | Optional, if used, it must be used with ErrorParameter. Used to 'catch' a particular type of runtime error. If used, the Try block will handle only that type of runtime error. |

### Notes

The **Try** block is similar to the [Exception](#) block. Exceptions that occur within the **Try** block are caught by the [Catch](#) statement. It has the same syntax as the [Exception](#) statement. See the [RuntimeException](#) statement or the Runtime Errors theme for descriptions of each type of runtime error.

Unlike the [Exception](#) block, **Try** blocks can be nested. If a **Try** block does not handle an exception or re-raises it, it can be handled by the next outermost **Try** block, or by the [Exception](#) block itself if there are no containing **Try** blocks.

A **Try** block can contain its own [Finally](#) statement. The code in the [Finally](#) block will execute regardless of whether or not an exception was raised within the **Try** block.

### Example

This example uses the [Catch](#) statement in a window's Open event handler to handle out of memory exceptions when trying to draw an imported gif image. The variable myPicture is a global property of type [Picture](#). The picture "Logo" has been added to the Project Editor.

```
Sub Open
Try
myPicture=New Picture(Logo.width,Logo.height,32)
myPicture.Graphics.DrawPicture Logo,0,0

Catch err as OutOfMemoryException
MsgBox "Insufficient memory to draw the picture!"
End Try
```

The following example handles an attempt to access a nonexistent value in a [Dictionary](#).

```
Try
someValue =myDict.Value("doesn't exist")
Catch err as KeyNotFoundException
MsgBox "The requested key does not exist."
End Try
```

### See Also

[Exception](#) block; [Catch](#), [Finally](#), [Function](#), [Raise](#), [Sub](#) statements; [RuntimeException](#) class.

## Type Mismatch Error

Occurs when you try to pass a parameter of an incorrect data type to a method or function or use an incorrect data type in an assignment statement. The second line of the error message tells you the data type that REALbasic was expecting and the data type that it received.

**Examples** Trying to assign a [string](#) to an [Integer](#) variable:

```
Dim n as Integer  
n="Anthony"
```

The second line of the error message says, “Expected Integer but got String.”

Trying to assign a value to an array instead of an element of the array.

```
Dim truth(3) as Boolean  
truth=True //must have a subscript
```

The second line of the error message says, “Expected Boolean() but got Boolean.”

## TypeMismatchException Error

Occurs when you try to assign a value to an object that is an incorrect data type.

**Super Class** [RuntimeException](#)

**Notes** A **TypeMismatchException** error occurs when you try to assign a value of an incorrect data type to an object. This error can occur only if the compiler cannot determine the type of the value at compile time — for example, when using [variants](#). Ordinarily, the compiler catches incorrect typing when you try to compile the application.

**Example** The following code assigns a picture that has been added to the Project Editor to the [variant](#), v, and then tries to assign v to an [Integer](#). The [Exception](#) block displays a

message box, allowing the developer to track down the problem. Exception handling prevents the built application from quitting when the error occurs.

```
Dim v as Variant
Dim i as Integer
v=ProductLogo //a picture
i=v
Exception err
If err IsA TypeMismatchException then
  MsgBox "The variable i must be assigned a number!"
end if
```

### See Also

[RuntimeException](#) class; [Function](#), [Raise](#), [Sub](#) statements; [Nil](#) keyword; [Exception](#), [Try](#) blocks.

## Ubound Function

Returns the index of the last element in an array.

### Syntax

**result=Ubound(array[,dimension])**

| Part      | Type                    | Description  |
|-----------|-------------------------|--|
| result    | <a href="#">Integer</a> | The index of the last element in the array specified. If the passed array has no elements, <i>result</i> is set to -1.   |
| array     | Array object            | The array whose last element number you want.  |
| dimension | <a href="#">Integer</a> | Optional: Relevant only for multi-dimensional arrays. Used to specify the dimension for which you want the last element. The first dimension is numbered 1. If passed -1, it will return the number of dimensions in the array. If passed a non-existent dimension, it will cause an <a href="#">OutOfBoundsException</a> error. |

### Notes

The **Ubound** function can be used to determine the last element of an array, but it can also be used to determine the size of an array. It may appear at first that the last element number and the size of the array are one in the same but in fact they are not. All arrays have a zero element. In some cases element zero is used and in other cases it is not. You will need to keep this in mind when using the **Ubound** function to determine the number of values you have in the array. If the array is zero-based, then element zero is used to store a value and you will have to add one to the value returned by the **Ubound** function to make up for it.

**Examples** This example replaces each occurrence of X in an array with Y.

```
Dim i As Integer
For i=0 to Ubound(Names)
If Names(i)="X" Then
    Names(i)="Y"
End If
Next
```

The following example returns -1 because the newly-declared array has no elements:

```
Dim i() as Integer
Dim j as Integer
j=Ubound(i)
```

**See Also** [Dim statement](#); [Array](#), [Join](#), [Split](#) functions; [Append](#), [IndexOf](#), [Insert](#), [Pop](#), [Redim](#), [Remove](#), [Shuffle](#), [Sort](#), [Sortwith](#) methods; [ParamArray](#) keyword.

## UByte Data Type

An 8-bit (1 byte) unsigned [Integer](#). Its value can range from 0 to 255. It can be used only with the [Declare](#) statement and cannot be used to declare REALbasic variables, properties, or methods.

**See Also** [Declare statement](#); [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UShort](#), [WindowPtr](#), [WString](#) data types.

## UDPSocket Class

Used for UDP (User Datagram Protocol) communications.

**Super Class** [SocketCore](#) class.

### Properties

| Name                    | Type                    | Description  |
|-------------------------|-------------------------|--|
| <b>BroadcastAddress</b> | <a href="#">String</a>  | The machine's broadcast address. You can specify this address when you call the Write method and the data will be broadcast across the network, but you will not receive the data you sent back. |
| <b>PacketsAvailable</b> | <a href="#">Integer</a> | The number of packets available in the internal receive buffer.  |

| Name              | Type                    | Description   |
|-------------------|-------------------------|---|
| PacketsLeftToSend | <a href="#">Integer</a> | Number of packets left in the queue to send. This enables you to create a synchronous socket without needing to subclass the <b>UDPSocket</b> .   |
| RouterHops        | <a href="#">Integer</a> | Specifies how many router hops the data sent out can make. This is also known as the Time To Live (or TTL).   |
| SendToSelf        | <a href="#">Boolean</a> | Specifies whether the data you send out will be sent to yourself as well. This is also known as loopback. This property applies only to multicast sends. Setting the SendToSelf property to <a href="#">True</a> may not work on all versions of Windows. MSDN states that the SendToSelf property on NT 4 cannot be turned off. A multicasting socket will always receive the data it sends out. |

## Methods

| Name                | Parameters  | Description  |
|---------------------|---|--|
| JoinMulticast Group | Group as <a href="#">String</a>                                     | Attempts to join the specified multicast group. Returns a <a href="#">Boolean</a> . If the socket successfully joined, it returns <a href="#">True</a> . You can join as many multicast groups as you'd like. The <i>Group</i> parameter specifies a ClassD IP address in the range: 224.0.0.0 to 239.255.255.255.   |
| LeaveMulticastGroup | Group as <a href="#">String</a>                                     | Leaves the specified multicast group.  |
| Read                | [Encoding as <a href="#">TextEncoding</a> ]                         | Returns a <a href="#">Datagram</a> from the internal receive buffer. The address property of the <a href="#">Datagram</a> is the remote address from which the data was sent. The optional <i>Encoding</i> parameter enables you to specify the text encoding of the data to be returned. Use the <a href="#">Encodings</a> object to specify a text encoding. |
| Write               | Data as <a href="#">DataGram</a>                                    | Writes the data to the specified address.  |
| Write               | Address as <a href="#">String</a><br>Data as <a href="#">String</a> | Constructs a <a href="#">Datagram</a> and sends the data to the specified address. If the address is part of a multicast group, all members of the group will get the data. If the address is the broadcast address, everyone on the network will get the data. If the address is an IP address, the data is unicast to that address only.                     |

## Notes

You do not need to add a **UDPSocket** control to a window in order to use it. Since it is not a subclass of [Control](#), you can instantiate it via code.

The User Datagram Protocol, or UDP, is the basis for most high speed, highly distributed network traffic. It is a connectionless protocol that has very low overhead,

but is not as secure as TCP. Since there is no connection, you do not need to take nearly as many steps to prepare when you wish to use a **UDPSocket**.

In order to use a **UDPSocket**, it must be bound to a specific port on your machine. This is done using the `Connect` method of the [SocketCore](#) class. Once the bind has occurred, the **UDPSocket** is ready for use. It will immediately begin accepting any data that it sees on the port it has bound to. It also allows you to send data out, as well as set UDP socket options (which will be described later).

In order to differentiate between which machine is sending you what data, a **UDPSocket** uses a data structure known as a [Datagram](#). A [Datagram](#) consists of two parts, the IP address of the remote machine that sent you the data, and the payload — or the actual data itself. When you attempt to send data out, you must specify information in the form of a [Datagram](#). This information is usually the remote address of the machine you want to receive your packet, the port it should be sent to, and the data you wish to send the remote machine.

UDP sockets can operate in various modes, which are all very similar, but have vastly different uses. The mode that most resembles a TCP communication is called “unicasting.” This occurs when the IP address you specify when you write data out is that of a single machine. An example would be sending data to [www.realsoftware.com](http://www.realsoftware.com), or some other network address. It is a [Datagram](#) that has one intended receiver.

The second mode of operation is called “broadcasting.” As the name implies, this is a broadcasted write. It is akin to yelling into a megaphone. Everyone gets the message, whether they want to or not. If the machine happens to be listening on the specific port you specified, then it will receive the data on that port. This type of send is very network intensive. As you can imagine, broadcasting can amount to huge amount of network traffic. The good news is, when you broadcast data out, it does not leave your subnet. Basically, a broadcast send will not leave your network to travel out into the world. When you want to broadcast data, instead of sending the data to an IP address of a remote machine, you specify the broadcast address for your machine. This address changes from machine to machine, so REALbasic provides a property of the **UDPSocket** class which tells you the correct broadcast address.

The third mode of operation for UDP sockets is “multicasting.” It is a combination of unicasting and broadcasting that is very powerful and practical. Multicasting is a lot like a chat room: you enter the chatroom, and are able to hold conversations with everyone else in the chatroom. When you want to enter the chatroom, you call `JoinMulticastGroup`, and you only need to specify the group you wish to join. The group parameter is a special kind of IP address, called a *Class D IP*. They range from 224.0.0.0 to 239.255.255.255. Think of the IP address as the name of the chatroom. If you want to start chatting with two other people, all three of you need to call `JoinMulticastGroup` with the same Class D IP address specified as the group. When you wish to leave the chatroom, you only need to call `LeaveMulticastGroup`, and again, specify the group you wish to leave. You can join as many multicast groups as you like, you are not limited to just one at a time. When you wish to send data to the multicast

group, you only need to specify the multicast group's IP address. All people who have joined the same group as you will receive the message.

Multicasting has some extra features that make it an even more powerful utility for network applications. If you wish to receive the data you sent out with a multicast send, you can set the `SendToSelf` property (also known as loopback) of the **UDPSocket**. If it is set to [True](#), then when you do a send (to a multicast group) you will get that data back. You can also set the number of router hops a multicast datagram will take (also known as the Time to Live). When your [Datagram](#) gets sent out, it runs through a series of routers on the way to its destinations. Every time the [Datagram](#) hits a router, its `RouterHops` property gets decremented. When that number reaches zero, the [Datagram](#) is destroyed. This means you can control who gets your datagrams with a lot more precision. There are some "best guesses" as to what the value of `RouterHops` should be.

| Value | Description    |
|-------|----------------|
| 0     | Same host      |
| 1     | Same subnet    |
| 32    | Same site      |
| 64    | Same region    |
| 128   | Same continent |
| 255   | Unrestricted   |

Note that if your [Datagram](#) runs through a router that does not support multicasting, it is killed immediately.

Due to the connectionless functionality of UDP, it does not make any guarantees that your data will reach its destination. You can work around this by creating your own protocol on top of the UDP protocol which acknowledges receives.

The UDPSocket operates in asynchronous mode, but the `Connect` method works synchronously. If the connect fails, you will get an error event immediately, and the `IsConnected` property will be set immediately.

**See Also** [Datagram](#), [EasyUDPSocket](#), [SocketCore](#) classes.

---

## UInt16 Data Type

A **UInt16** is an intrinsic data type in REALbasic. It is an unsigned integer that uses two bytes of storage. It has a range of 0 to 65535. The default value of an **UInt16** is 0.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>                            | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to 2^64-1                     |

**See Also**

[Boolean](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt32](#), [UInt64](#) data types; [±](#), [\\*](#), [/](#), ≤, ≤=, ≡, ≥=, ≥, ≤>, [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

## UInt32 Data Type

A **UInt32** is an intrinsic data type in REALbasic. It is an unsigned integer that uses four bytes of storage. It has a range of 0 to 4,294,967,295. The default value of an **UInt32** is 0.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>                            | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to 2^64-1                     |

**See Also**

[Boolean](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt64](#) data types; [±](#), [\\*](#), [/](#), ≤, ≤=, ≡, ≥=, ≥, ≤>, [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

## UInt64 Data Type

A **UInt64** is an intrinsic data type in REALbasic. It is an unsigned integer that uses eight bytes of storage. It has a range of 0 to  $2^{64}-1$ . The default value of an **UInt64** is 0.

### Notes

If either operand of an integer arithmetic expression is a 64-bit value, the result will be a 64-bit value. If both operands are 32 bits or smaller, the result will be a 32-bit value. If either operand of an arithmetic expression is signed, the result will be signed; the result will only be unsigned if both operands are unsigned.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>                            | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | $-2^{63}$ to $2^{63}-1$         |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to $2^{64}-1$                 |

### See Also

[Boolean](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt8](#), [UInt16](#), [UInt32](#) data types; [=](#), [±](#), [\\*](#), [/](#), [≤](#), [≤=](#), [≡](#), [≥=](#), [≥](#), [≤>](#), [\](#), [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

---

## UInt8 Data Type

A **UInt8** is an intrinsic data type in REALbasic. It is an unsigned integer that uses one byte of storage. It has a range of 0 to 255. The default value of an **UInt8** is 0.

REALbasic offers both signed and unsigned integer data types that use one, two, four, or eight bytes of memory. The following table summarizes these data types.

| Data Type  | Number of Bytes | Range                           |
|--|-----------------|---------------------------------|
| <a href="#">Int8</a> or <a href="#">Byte</a>     | 1               | -128 to 127                     |
| <a href="#">Int16</a>                            | 2               | -32,768 to 32,767               |
| <a href="#">Int32</a> or <a href="#">Integer</a> | 4               | -2,147,483,648 to 2,147,483,647 |
| <a href="#">Int64</a>                            | 8               | -2^63 to 2^63-1                 |
| <a href="#">UInt8</a>                            | 1               | 0 to 255                        |
| <a href="#">UInt16</a>                           | 2               | 0 to 65535                      |
| <a href="#">UInt32</a>                           | 4               | 0 to 4,294,967,295              |
| <a href="#">UInt64</a>                           | 8               | 0 to 2^64-1                     |

**See Also**

[Boolean](#), [Color](#), [Double](#), [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Integer](#), [Single](#), [String](#), [UInt16](#), [UInt32](#), [UInt64](#) data types; [±](#), [\\*](#), [/](#), ≤, ≤=, ≡, ≥=, ≥, ≤>, [\](#), [IsNumeric](#), [Mod](#), [Str](#), [Val](#), [Vartype](#), functions; [Dim](#) statement.

## Unknown Pragma Name Error

You passed a name to the [#Pragma](#) keyword that is not a valid pragma. You can only use the pragma directives that are part of the REALbasic language.

**Example**

The following string is not valid.

```
#pragma myGreatPragma
```

**See Also**

[#Pragma](#) keyword.

## UnsupportedFormatException Error

Occurs when you use a [String](#) expression that cannot be evaluated.

**Super Class**

[RuntimeException](#)

**Notes**

REALbasic allows you to use a string expression to set the column widths via the [ListColumn](#) class or the ColumnWidths property of the [ListBox](#). You can use the percent sign or the “\*” symbol, as described in those sections. If you use a character that is not permitted, an **UnsupportedFormatException** will occur.

## UpDownArrows Control

---

**Example** The following specification causes an **UnsupportedFormatException** runtime error:

```
ListBox1.ColumnWidths= "50,50i"
```

**See Also** [ListBox](#), [ListColumn](#), [RuntimeException](#) classes; [Exception](#), [Try](#) blocks.

## UpDownArrows Control

Adds a pair of vertical arrows to a window. You can, for example, use an UpDownArrows control to increment or decrement the value of another object.

**Super Class** [RectControl](#)

Because this is a [RectControl](#), see the [RectControl](#) for other properties and events that are common to all [RectControl](#) objects.

### Properties

| Name        | Type                    | Description   |
|-------------|-------------------------|---|
| AcceptFocus | <a href="#">Boolean</a> | If <a href="#">True</a> , the UpDownArrows control is included in the Tab order and can get the focus. When the control has the focus, it has a selection rectangle around it; pressing the Up arrow will fire the Up event and pressing the Down arrow will fire the Down event. The default value is <a href="#">False</a> . An UpDownArrows control can get the focus only on Windows. |

### Events

| Name      | Parameters | Description  |
|-----------|------------|--|
| Down      |            | The user has clicked the Down arrow.   |
| GotFocus  |            | The UpDownArrows control has received the focus and can respond to the Up and Down arrow keys. |
| LostFocus |            | The UpDownArrows control has lost the focus.   |
| Up        |            | The user has clicked the Up arrow.   |

**See Also** [RectControl](#) class.

## Uppercase Function

Converts all characters in a [string](#) to uppercase characters.

**Syntax** **result=Uppercase(value)**

**OR*****result=stringVariable.Uppercase***

| Part           | Type                   | Description  |
|----------------|------------------------|--|
| result         | <a href="#">String</a> | A copy of the original string with all characters converted to their uppercase equivalent. |
| value          | <a href="#">String</a> | The original string.   |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> .  |

**Notes**

Returns the string with all alphabetic characters in uppercase.

**Examples**

The example below converts the value passed to uppercase.

```
Dim s As String
s=Uppercase("tHe Quick fOX") //returns "THE QUICK FOX"
s=Uppercase("the 5 lazy dogs") //returns "THE 5 LAZY DOGS"
```

The following example uses the second syntax.

```
Dim s as String = "the 5 lazy dogs"
MsgBox s.uppercase //displays "THE 5 LAZY DOGS"
```

**See Also**

[Lowercase](#), [Titlecase](#) functions.

## UserCancelled Function

Used to determine if the user has pressed Escape (on Windows or Linux) or Command-period (on Macintosh) or to cancel the execution of code.

**Syntax*****result=UserCancelled***

| Part   | Type                    | Description  |
|--------|-------------------------|--|
| result | <a href="#">Boolean</a> | <i>Result</i> is <a href="#">True</a> if the user has pressed Escape (on Windows or Linux) or Command-period (on Macintosh) or and <a href="#">False</a> if the user did not cancel. |

**Notes**

The **UserCancelled** function will continue to return [True](#) until the event handler that was executing when the user pressed Escape or Command-period is finished.

If you use the DoEvents method of the [Application](#) class to yield time back to REALbasic within a loop, then the **UserCancelled** function does not work.

If you need the equivalent functionality when you use DoEvents, then you should check the Async key state of the keyboard using the methods of the [Keyboard](#) object to see if

the proper combination has been hit. However, these two concepts are typically exclusive. You use DoEvents when you want to update the user interface while executing a loop. This gives the user a way stop the loop by clicking a button or some other interface item. You use **UserCancelled** when you're in a tight loop with no user interface element to click on.

### Examples

This example uses the **UserCancelled** function to exit a [For](#) loop if the user presses Command-period/Escape.

```
Dim i As Integer
For i = 0 to 10000
    ProgressBar1.Value = i
    If UserCancelled Then
        Exit
    End If
    Next
```

### See Also

[Exit](#), [Do...Loop](#), [For...Next](#), [While...Wend](#) statements; [Application](#) class.

---

## UShort Data Type

A 16-bit (2 byte) signed [Integer](#). Its value can range from 0 to 65535. This data type can be used only within [Declare](#) statements and cannot be used to declare REALbasic variables, properties, or methods.

### See Also

[Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [WindowPtr](#), [WString](#) data types.

---

## UVList Class

Used to store texture coordinates, often called UV coordinates. Used in conjunction with the [Material](#) class for textures an [RB3DSpace](#).

**Super Class** [Object](#)

### Properties

| Name | Type                   | Description   |
|------|------------------------|---|
| U    | <a href="#">Double</a> | The U texture coordinate. Parameter is i as <a href="#">Integer</a> . |
| V    | <a href="#">Double</a> | The V texture coordinate. Parameter is i as <a href="#">Integer</a> . |

## Methods

| Name     | Parameters   | Description   |
|----------|--|---|
| AddToAll | dU as <a href="#">Double</a> ,<br>dV as <a href="#">Double</a>   | Adds the passed texture coordinates to all items in the list.           |
| Copy     | otherUVList as <a href="#">UVList</a>  | Copies the passed <a href="#">UVList</a> to the current list.           |
| GetUV    | index as <a href="#">Integer</a> ,<br><a href="#">ByRef</a> U as <a href="#">Double</a> ,<br><a href="#">ByRef</a> V as <a href="#">Double</a> | Gets the texture coordinates, U and V, for the passed item in the list. |
| SetUV    | index as <a href="#">Integer</a> ,<br>U as <a href="#">Double</a> ,<br>V as <a href="#">Double</a>   | Sets the texture coordinates, U and V, for the passed item in the list. |

## See Also

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [Vector3D](#), [VectorList](#) classes; [Rb3DSpace](#) control.

# Val Function

Returns the numeric form of a [string](#).

## Syntax

*result=Val(string)*

OR

*result=stringVariable.Val*

| Part           | Type                   | Description                                   |
|----------------|------------------------|---|
| result         | <a href="#">Double</a> | The numeric equivalent of the string passed.  |
| string         | <a href="#">String</a> | Any valid <a href="#">string</a> expression.  |
| stringVariable | <a href="#">String</a> | Any variable of type <a href="#">String</a> . |

## Notes

The **Val** function stops reading the [string](#) at the first character it doesn't recognize as part of a number. All other characters are automatically stripped.

It does recognize prefixes [&o](#) (octal), [&b](#) (binary), and [&h](#) (hexadecimal). However, spaces are not allowed in front of the ampersand. That is, “[&h](#)FF” returns 0, but “[&h](#) FF” returns 255.

The [CDbl](#) function is the same as the **Val** function but is used when you need to pass a [string](#) that uses a character other than the period (.) as the decimal separator. It uses the decimal character specified by the operating system. For example, on Windows XP, it is set in the Regional and Language Options Control Panel. **Val** should generally be used to convert internal data, but not data entered by the user. Val is not international-savvy, but [CDbl](#) is.

## Variant Data Type

---

The [CStr](#) function is the same as the [Str](#) function but is used when you need to pass a number that uses a character other than the period (.) as the decimal separator. It uses the decimal character specified by the operating system.

**Val** returns zero if string contains no numbers.

### Examples

These examples use the **Val** function to return the numbers contained in a [string](#).

```
Dim n As Integer
```

```
n = Val("12345") //returns 12345  
n = Val(" 12345") //returns 12345  
n = Val("123 45") //returns 123  
n = Val("&hFFF") //returns 4095  
n = Val("&b1111") //returns 15
```

```
Dim s as String
```

```
s="12345"  
n=s.Val //returns 12345
```

### See Also

, [CDbl](#), [CStr](#), [Str](#) functions; [&b](#), [&h](#), [&o](#) literals.

## Variant Data Type

---

A **Variant** is a special data type that can contain any type of data. Use the [VarType](#) function or the **Variant**'s Type method to determine the data type of a variant.

Super Class [Object](#)

### Properties

| Name         | Type                    | Description   |
|--------------|-------------------------|---|
| BooleanValue | <a href="#">Boolean</a> | Returns the value of the variant as a <a href="#">Boolean</a> .   |
| ColorValue   | <a href="#">Color</a>   | Returns the value of the variant as a <a href="#">Color</a> .   |
| DateValue    | <a href="#">Date</a>    | Returns the value of the variant as a <a href="#">Date</a> . When retrieving the string value of a date, the string is returned in SQL date format<br>YYYY-MM-DD HH:MM. |
| DoubleValue  | <a href="#">Double</a>  | Returns the value of the variant as a <a href="#">Double</a> .  |
| Int32Value   | <a href="#">Int32</a>   | Returns the value of the variant as an <a href="#">Int32</a> .  |
| Int64Value   | <a href="#">Int64</a>   | Returns the value of the variant as an <a href="#">Int64</a> .  |
| IntegerValue | <a href="#">Integer</a> | Returns the value of the variant as an <a href="#">Integer</a> .  |
| ObjectValue  | <a href="#">Object</a>  | Returns the value of the variant as an <a href="#">Object</a> .   |
| SingleValue  | <a href="#">Single</a>  | Returns the value of the variant as a <a href="#">Single</a> .  |

| Name        | Type                   | Description  |
|-------------|------------------------|--|
| StringValue | <a href="#">String</a> | Returns the value of the variant as a <a href="#">String</a> . If the variant holds a <a href="#">Date</a> , it is converted to a SQLDateTime format. See the <a href="#">Date</a> class for details on this format. |
| UInt32Value | <a href="#">UInt32</a> | Returns the value of the variant as an <a href="#">UInt32</a> .  |
| UInt64Value | <a href="#">UInt64</a> | Returns the value of the variant as an <a href="#">UInt64</a> .  |

## Methods

| Name      | Parameters              | Description  |
|-----------|-------------------------|--|
| Equals    | v as Variant            | Returns a <a href="#">Boolean</a> . Returns <a href="#">True</a> if the current variant is of the same data type and value as the variant passed as a parameter.   |
| Hash      | <a href="#">Integer</a> | Returns a hash value for the variant, i.e., one that will always be the same for any variant with the same value. The Hash value is guaranteed to be unique for integers and colors. For objects, it's guaranteed to be unique among all objects currently in existence (though once an object dies, a future object may use the same hash value). |
| IsNull    |                         | Returns a <a href="#">Boolean</a> . <a href="#">True</a> if the variant is Null.   |
| IsNumeric |                         | Returns a <a href="#">Boolean</a> . <a href="#">True</a> if the variant is a numeric data type. If the variant is a string expression for a number, such as "8" instead of 8, then IsNumeric returns <a href="#">True</a> .  |
| Type      |                         | Returns an <a href="#">Integer</a> , indicating the data type of v. If the variant is not <a href="#">Nil</a> , it provides the same functionality as the <a href="#">VarType</a> global function. See the table of data types for the <a href="#">VarType</a> function.   |

## Notes

The default value of a **Variant** is 0. When you assign a variant to a [string](#), numeric, or [date](#) variable, REALbasic converts the value of the variant to the variable's data type. For example,

```
Dim v as Variant
Dim n as Integer
Dim s as String
v=25
s=v // s is "25"
v="25"
n=v // n is 25
```

If there is any ambiguity concerning the type conversion of a variant, you should use one of the properties of the **Variant** class to force the REALbasic compiler to convert the variant to the desired type. In this example, the Product number is supposed to be

the string concatenation of the Model number and the Part Number. To do this, use the `StringValue` property to tell REALbasic to do what you want:

```
Dim model,partNumber,product as Variant  
Model=100  
Partnumber=546  
Product=Model+Partnumber //product=646  
Product=Model.StringValue+Partnumber.StringValue //product=100546
```

Or, you could just type the variables as `String` and do string concatenation directly. The `Type` method returns the type of the variant, provided it is not `Nil`. For example, the following code generates an [NilObjectException](#) error.

```
Dim v as Variant  
Dim i as Integer  
i=v.type
```

On the other hand, the following example returns 0.

```
Dim v as Variant  
Dim i as Integer  
i=VarType(v)
```

Variants used in an expression convert themselves to the type of the other operand, no matter what kind of data they contain, instead of deferring the type-conversion until runtime. This only affects expressions involving a variant operand and an operand of some other type, not expressions in which both operands are variants.

Consider the following example:

```
Dim v as Variant = 0.5
```

The comparison:

```
If v > 0 then
```

returns `False` since `v` is being treated as an `Integer`. If you instead use:

```
If v > 0.0 then..
```

the comparison returns `True`.

The comparison

```
If v.DoubleValue > 0
```

also returns `True`.

**See Also** [IsNumeric](#), [VarType](#) functions; [DatabaseField](#), [Date](#), [Dictionary](#) classes; [Boolean](#), [Color](#), [Double](#), [Integer](#), [Single](#), [String](#) data types; [Dim](#) statement.

## VarType Function

Used to determine the data type of a variable.

**Syntax** **result=VarType(value)**

| Part   | Type                    | Description                       |
|--------|-------------------------|-----------------------------------|
| result | <a href="#">Integer</a> | Indicates the data type of value. |
| value  | <a href="#">Variant</a> | Value to be typed.                |

**Notes** Result takes on the following values:

| Result | Description   |
|--------|---|
| 0      | <a href="#">Nil</a>                                   |
| 2      | Integer types of 32 bits or less, signed or unsigned. |
| 3      | <a href="#">Int64</a> or <a href="#">UInt64</a>       |
| 5      | <a href="#">Double</a> or <a href="#">Single</a>      |
| 7      | <a href="#">Date</a>                                  |
| 8      | <a href="#">String</a>                                |
| 9      | <a href="#">Object</a>                                |
| 11     | <a href="#">Boolean</a>                               |
| 16     | <a href="#">Color</a>                                 |

Please note that some earlier versions of REALbasic returned 13 for [Object](#).

**Example** The following line of code returns 8 (data type of [String](#)):

```
EditField1.text=Str\(VarType\("Herman"\)\)
```

**See Also** [Variant](#) class, [Boolean](#), [Byte](#), [Color](#), [Double](#), [Int16](#), [Int32](#), [Int64](#), [Int8](#), [Integer](#), [Single](#), [String](#), [UInt16](#), [UInt32](#), [UInt64](#), [UInt8](#) data types; [IsNumeric](#) function.

## Vector3D Class

Used to represent absolute or relative position of a vector in an [RB3DSpace](#) three-dimensional space.

**Super Class** [Object](#)

## Vector3D Class

---

### Constructor Syntaxes

| Syntax                                | Parameters  | Description  |
|---------------------------------------|---|--|
| <a href="#">New Vector3D(x, y, z)</a> | x as <a href="#">Double</a><br>y as <a href="#">Double</a><br>z as <a href="#">Double</a> | Creates a <b>Vector3D</b> and initializes x, y, and z to the values passed.                  |
| <a href="#">New Vector3D(vec)</a>     | vec as <a href="#">Vector3D</a>   | Creates a <b>Vector3D</b> and initializes the new vector to the same value as the one given. |

### Properties

| Name | Type                   | Description                          |
|------|------------------------|--------------------------------------|
| X    | <a href="#">Double</a> | x coordinate of a point in 3D space. |
| Y    | <a href="#">Double</a> | y-coordinate of a point in 3D space. |
| Z    | <a href="#">Double</a> | z-coordinate of a point in 3D space. |

### Methods

| Name           | Parameters                      | Description   |
|----------------|---------------------------------|---|
| Add            | vec as <a href="#">Vector3D</a> | Adds the vector vec to the current vector, storing the result in this object.   |
| Copy           | vec as <a href="#">Vector3D</a> | Makes this <b>Vector3D</b> match the passed vector.   |
| Cross          | vec as <a href="#">Vector3D</a> | Returns the cross-product of two vectors. v1.Cross(v2) gives you a new vector which is perpendicular to the plane defined by v1 and v2. Returns a <b>Vector3D</b> .                 |
| Dot            | vec as <a href="#">Vector3D</a> | Returns the dot-product of two vectors. v1.Dot(v2) is proportional to the cosine of the angle between v1 and v2. Returns a <a href="#">Double</a> .                                 |
| Length         |                                 | Returns the length of the vector in 3D space. Returns a <a href="#">Double</a> .  |
| LenSquar<br>ed |                                 | Returns the squared length of the vector in 3D space. Returns a <a href="#">Double</a> .  |
| Minus          | vec as <a href="#">Vector3D</a> | Returns the difference of this vector and the vector vec.   |
| MultiplyBy     | d as <a href="#">Double</a>     | Multiplies this vector by the scalar d, storing the result in this object. This scales the vector; another way to scale a vector is to assign to its Length or LenSquared property. |
| Normalize      |                                 | Scales the vector so that its square length (and length) is 1.  |
| Plus           | vec as <a href="#">Vector3D</a> | Returns a <b>Vector3D</b> . Returns the sum of this vector and the passed vector.   |
| Subtract       | vec as <a href="#">Vector3D</a> | Subtracts the passed vector from this one, storing the result in this object.   |

| Name  | Parameters                  | Description   |
|-------|-----------------------------|---|
| Times | d as <a href="#">Double</a> | Returns a <b>Vector3D</b> . Returns the product of this vector and the scalar, d. |

**Notes**

The **Vector3D** class is used to represent absolute or relative position in [Rb3DSpace](#). It is simply the X, Y, and Z coordinates of a point in 3D space. Depending on how you intend to use it, they can either mean the absolute position of something in 3D space, or a change in position. You can get its length (i.e., its distance from the origin), or its squared length (which is cheaper to compute). Normalize() scales the vector so that its length becomes 1. The Cross and Dot methods allow you to perform two kinds of vector multiplication.

**Comparison of Vector Operations**

The **Vector3D** class has a complete set of methods for performing vector arithmetic, eliminating the need work with the individual elements of the vectors. You may either save the result of the operation in the calling vector or save the result in a new vector. The following table compares the six methods.

| Operation   | Save result in place | Save result in a new vector |
|-------------|----------------------|-----------------------------|
| Addition    | a.Add(b)             | c = a.Plus(b)               |
| Subtraction | a.Subtract(b)        | c = a.Minus(b)              |
| Scaling     | a.MultiplyBy(f)      | c = a.Times(f)              |

**See Also**

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), [VectorList](#) classes; [RB3DSpace](#) control.

---

## VectorList Class

Used to construct, modify, and animate a [Trimesh](#).

**Super Class** [Object](#)**Properties**

| Name | Type                   | Description  |
|------|------------------------|--|
| X    | <a href="#">Double</a> | Parameter is Index as <a href="#">Integer</a> . Accesses the X value of the item specified by Index. |
| Y    | <a href="#">Double</a> | Parameter is Index as <a href="#">Integer</a> . Accesses the Y value of the item specified by Index. |
| Z    | <a href="#">Double</a> | Parameter is Index as <a href="#">Integer</a> . Accesses the Z value of the item specified by Index. |

### Methods

| Name            | Parameters   | Description   |
|-----------------|--|---|
| AddList         | other as <b>VectorList</b>   | Adds the passed <b>VectorList</b> to the current <b>VectorList</b> .                                  |
| AddToAll        | delta as <b>Vector3D</b>   | Adds the passed <b>Vector3D</b> to all vectors in the list.   |
| Copy            | other as <b>VectorList</b>   | Copies the passed <b>VectorList</b> to the current <b>VectorList</b> .                                |
| GetVector       | i as <b>Integer</b>  | Returns a copy of X(i), Y(i), and Z(i) for the item specified by the index, i, as a <b>Vector3D</b> . |
| MultiplyAllBy   | factor as <b>Double</b>  | Multiplies all vectors in the list by the passed factor.  |
| NormalizeAll    |  | Normalizes all vectors in the list.   |
| SetToDifference | otherList1 as <b>VectorList</b> , otherList2 as <b>VectorList</b>                  | Sets the current <b>VectorList</b> to the difference between otherList1 and otherList2.               |
| SetToSum        | otherList1 as <b>VectorList</b> , otherList2 as <b>VectorList</b>                  | Sets the current <b>VectorList</b> to the sum of the passed VectorLists.                              |
| SetVector       | i as <b>Integer</b> , vector as <b>Vector3D</b>                                    | Copies the passed vector into X(i), Y(i), and Z(i).   |
| SetXYZ          | i as <b>Integer</b> , X as <b>Double</b> , Y as <b>Double</b> , Z as <b>Double</b> | Sets the X, Y, and Z properties for the vector specified by the parameter i.                          |

### Notes

This is a helper class that maintains a weak reference to the [Trimesh](#) that it came from. When you modify a **VectorList**, it modifies the [Trimesh](#) it came from. Note that it does not have a Count property; you get the count from the [Trimesh's](#) VertexCount property. This makes sense because are several lists that all have to be the same size. **VectorList** lets you individually access the X, Y, and Z of any element in the list, but it also contains several methods for doing common operations on the whole list at once. These can be used for very efficient animation.

### See Also

[Bounds3D](#), [ColorList](#), [Element3D](#), [Group3D](#), [Light3D](#), [Material](#), [Object3D](#), [Quaternion](#), [TriangleList](#), [Trimesh](#), [UVList](#), [Vector3D](#), classes; [Rb3DSpace](#) control.

---

## VirtualVolume Class

Enables you to create and maintain a hierarchy of “virtual” files within one physical file. Basic reading and writing text and binary streams are supported.

**Super Class** [Object](#)

## Properties

| Name        | Type                            | Description  |
|-------------|---------------------------------|--|
| <b>Root</b> | <a href="#">FolderItem</a><br>m | Reference to the root directory of the virtual volume. |

## Notes

The [FolderItem](#) class has two methods and one property for working with virtual volumes. Use the `CreateAsVirtualVolume` and `OpenAsVirtualVolume` methods to create and open virtual volumes. The `VirtualVolume` property of the [FolderItem](#) class returns the **VirtualVolume** if the [FolderItem](#) is in a virtual volume.

Once you've created a **VirtualVolume**, you can get its Root and navigate to it using the same [FolderItem](#) methods that you would for real files. Virtual files are cross-platform compatible, and support basic reading and writing as text or binary files. The `SaveAsPicture` and `SaveStyledEditField` methods of the [FolderItem](#) class do not work for virtual volumes. Virtual volumes do not support resource forks.

Virtual volumes flush their data to disk whenever a virtual volume is closed. This helps ensure that your data is safely on disk in the event of a crash or loss of power.

Filenames can be up to 223 bytes long. Paths are not supported, but directories are (e.g., create a virtual directory with the `CreateAsFolder` method of the [FolderItem](#) class, and navigate to it with the `Child` or `Item` methods of this class). The virtual file system supports four-byte type codes (accessed via `f.MacType`), but does not support creator codes.

## Example

The following example creates a **VirtualVolume**, gets its Root property, and writes a text file to the virtual volume.

```

Dim vv As VirtualVolume
Dim realFile As FolderItem
Dim virtFile As FolderItem
Dim outp As TextOutputStream

realFile = GetFolderItem("").Child("VV")
vv = realFile.CreateVirtualVolume
If vv = Nil then
  MsgBox "Unable to create virtual volume"
else
  virtFile = vv.Root.Child("Virtual File.txt")
  outp = virtFile.CreateTextFile
  outp.Write "Hello world!"
end if

```

## Volume Function

---

To access an existing virtual volume, use the `OpenAsVirtualVolume` method instead of the `CreateVirtualVolume` method of the [FolderItem](#) class.

**See Also** [FolderItem](#) class.

## Volume Function

Returns a [FolderItem](#) that represents a mounted volume.

**Syntax** `result=Volume(VolumeNumber)`

| Part         | Type                       | Description  |
|--------------|----------------------------|--|
| result       | <a href="#">FolderItem</a> | The mounted volume whose number was passed.                            |
| VolumeNumber | <a href="#">Integer</a>    | The number of the volume you require a <a href="#">FolderItem</a> for. |

**Notes** The **Volume** function returns a [FolderItem](#) that represents the mounted volume whose number was passed. Volume zero is the boot volume. This function can be used in conjunction with the [VolumeCount](#) function to loop through the mounted volumes.

**Examples** This example places the names of all mounted volumes into a [ListBox](#) control:

```
Dim i,n as Integer  
n= VolumeCount-1  
For i=0 to n  
    ListBox1.AddRow Volume(i).Name  
Next
```

The following example returns a [FolderItem](#) for the “Documents” folder on the user’s boot volume.

```
Dim f as FolderItem  
f=Volume(0).Child("Documents")
```

**See Also** [VolumeCount](#) function; [FolderItem](#) class.

## VolumeCount Function

Returns the number of mounted volumes.

---

|                 |  |                                |
|-----------------|--|--------------------------------|
| <b>Syntax</b>   | <b>result=VolumeCount</b>  |                                |
| <hr/>           |  |                                |
| <b>Part</b>     | <b>Type</b>  | <b>Description</b>             |
| result          | <a href="#">Integer</a>  | The number of mounted volumes. |
| <b>Notes</b>    | The <b>VolumeCount</b> function returns the number of mounted volumes. |                                |
| <b>Examples</b> | See the <a href="#">Volume</a> function for an example.                |                                |
| <b>See Also</b> | <a href="#">Volume</a> function.                                       |                                |

---

## While...Wend Statement

Repeatedly executes a series of statements while a specified condition is [True](#).

|               |   |
|---------------|---|
| <b>Syntax</b> | <b>While</b> <i>condition</i><br><br>[ <i>Statements</i> ]<br><br>[ <a href="#">Continue</a> ]<br><br>[ <a href="#">Exit</a> ]<br><br>[ <i>Statements</i> ] |
| <b>Wend</b>   |   |

| Part                     | Description   |
|--------------------------|---|
| While                    | Begins the loop.  |
| Condition                | Any valid <a href="#">Boolean</a> expression. When this expression evaluates to <a href="#">True</a> , the loop will continue to execute.   |
| Statements               | Statements to be executed repeatedly (until condition evaluates to <a href="#">False</a> ).   |
| <a href="#">Continue</a> | If a <a href="#">Continue</a> statement is present, execution will skip over the remaining statements in the loop and resume with a new iteration of the loop. Optional arguments of the <a href="#">Continue</a> statement allow you to specify which loop will iterate next, in cases of multiple nested loops. |
| <a href="#">Exit</a>     | If an <a href="#">Exit</a> statement is present, execution of the loop is terminated and resumes with the next line following the loop.   |
| Wend                     | Ends the loop. Condition is evaluated to determine if the loop should exit.   |

|              |   |
|--------------|---|
| <b>Notes</b> | If Condition is <a href="#">True</a> , all statements are executed until the <b>Wend</b> statement is reached. If Condition is still <a href="#">True</a> , the process is repeated. If Condition is <a href="#">False</a> , then execution continues with the statement following the <b>Wend</b> statement. |
|--------------|---|

**While...Wend** statements can be nested to any level. Each **Wend** statement goes with the previous **While** statement. It is permissible to place [Dim](#) statements inside loops, including While loops.

When a loop runs, it takes over the interface, preventing the user from interacting with menus and controls. If the loop is very lengthy, you can move the code for the loop to a separate [Thread](#), allowing it to execute in the background.

**Example** This example uses the **While...Wend** statement to increment a variable.

```
Dim x As Integer  
While x<100  
    x=x+1  
Wend
```

**See Also** [Continue](#), [Do...Loop](#), [Exit](#), [For...Next](#) statements; [Application](#), [Thread](#) classes.

---

## Window Class

Any window.

**Super Class** [Object](#)

### Properties

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| Backcolor   | <a href="#">Color</a>   | The background color of the window. You need to set HasBackColor to <a href="#">True</a> in order to apply the Backcolor to the window.  |
| Backdrop    | <a href="#">Picture</a> | A picture object that will be drawn in the window when the window opens.   |
| BalloonHelp | <a href="#">String</a>  | Balloon help text for the window. For Mac OS "classic" only.   |
| CloseButton | <a href="#">Boolean</a> | If <a href="#">True</a> , the window will include a Close button in its Title Bar. Since this is the name of a property, you cannot use "CloseButton" as the name of a button or any other type of object.   |
| Composite   | <a href="#">Boolean</a> | (Mac OS X 10.2 and above only). If <a href="#">True</a> , It allows controls to be used on top of other controls or a background picture without a rectangle surrounding the control. Does nothing for the following window types: Modal, Movable Modal, PlainBox, and ShadowedBox. Drawer and metal windows always have compositing on. See the Notes section for more information on the Composite property. |

| Name                  | Type                        | Description   |
|-----------------------|-----------------------------|---|
| <b>Control</b>        | Array of Controls           | Zero-based array of the controls in the window.   |
| <b>ControlCount</b>   | <a href="#">Integer</a>     | The number of controls in the window.   |
| <b>DockItem</b>       |                             | (Mac OS X only). Enables you to access the <a href="#">DockItem</a> class to manipulate the dock item associated with the window. The DockItem class has two methods, UpdateNow and ResetIcon, and one property, Graphics. Use the methods of the <a href="#">Graphics</a> class to modify the appearance of the icon. (The <a href="#">Graphics</a> property may be <a href="#">Nil</a> ; it is non- <a href="#">Nil</a> only when an OS X window has been minimized to the dock.) The ResetIcon method resets the icon to its original state (default appearance). Since a Mac OS X icon is intended to be scaled automatically, you should design it as a 128x128 pixel icon. Call the UpdateNow method to redraw the icon. You can also use the ClearRect property of the <a href="#">Graphics</a> class to start over from a blank icon. Anything you can do with a <a href="#">Graphics</a> object you are able to do with the Dock's Graphics object (like drawing in a picture, or using a Shape 2D). The Dockitem property of the <a href="#">Application</a> class enables you to control the <a href="#">DockItem</a> for the whole application. |
| <b>FloaterProcess</b> | <a href="#">String</a>      | The Mac creator code for the application the window should display in front of. If this is empty, the window will float in front of all applications. The window type must be Global Floating Window for this to work.  |
| <b>Focus</b>          | <a href="#">RectControl</a> | Gets or sets the <a href="#">RectControl</a> in the window that has the focus. If no control has the focus, this property is <a href="#">Nil</a> .  |
| <b>Frame</b>          | <a href="#">Integer</a>     | The window type.<br>0-Document window<br>1-Movable Modal window<br>2-Modal Dialog box<br>3-Floating window<br>4-Plain box<br>5-Shadowed box<br>6-Rounded window<br>7-Global floating window<br>8-Sheet window (displayed as such on Mac OS X only)<br>9-Metal window (displayed as such on Mac OS X 10.2 and above only)<br>10-Drawer window (displayed as such on Mac OS X 10.2 and above only). See the section " <a href="#">Window Types</a> " on page 62 of the User's Guide for illustrations and discussion  |

| Name           | Type                     | Description  |
|----------------|--------------------------|--|
| FullScreen     | <a href="#">Boolean</a>  | When set to <a href="#">True</a> , the window resizes itself to cover the entire screen. The values of the Left, Top, Width, and Height properties are adjusted to reflect the window's new position. They cannot be changed as long as FullScreen is <a href="#">True</a> . If MenuBarVisible is set to <a href="#">True</a> , the top of the window will be partially covered by the menu bar in Mac OS. In the case of multiple monitors, the window resizes to fit the monitor which contains the greatest portion of the window. FullScreen defaults to <a href="#">False</a> . |
| Graphics       | <a href="#">Graphics</a> | Allows access to all the <a href="#">Graphics</a> methods for drawing into the window. Whatever objects are drawn using the <a href="#">Graphics</a> methods are drawn after the Backdrop image is drawn, in a separate layer. This allows you to overlay objects on top of the Backdrop image.  |
| Handle         | <a href="#">Integer</a>  | Returns a handle to the window.  |
| HasBackColor   | <a href="#">Boolean</a>  | If <a href="#">True</a> , the background color of the window is set to the value of the Backcolor property. This property must be set in order for the value of the BackColor property to be applied to the window.  |
| Height         | <a href="#">Integer</a>  | The height of the content region of the window.  |
| Left           | <a href="#">Integer</a>  | The distance (in pixels) between the left edge of the screen and the left edge of the content area of the window. The left frame of the window is taken into account.  |
| LiveResize     | <a href="#">Boolean</a>  | If <a href="#">True</a> , the window is redrawn 'live' as it is resized by dragging. Setting LiveResize to <a href="#">True</a> is recommended if you are using Drawer windows. Works on Mac OS X only.  |
| MacProcID      | <a href="#">Integer</a>  | Apple Window definition ID. Can be used on Macintosh to define window types that are not supported by the Frame property. Window types defined by MacProcID take effect only on the Macintosh platform. On Windows and Linux, the Window Type is controlled only by the Frame property. See the Notes section.   |
| MaxHeight      | <a href="#">Integer</a>  | The maximum height to which the window can be resized.   |
| MaximizeButton | <a href="#">Boolean</a>  | If <a href="#">True</a> , the window will include a Maximize button in its Tile bar.   |
| MaxWidth       | <a href="#">Integer</a>  | The maximum width to which the window can be resized.  |

| Name           | Type                        | Description   |
|----------------|-----------------------------|---|
| MenuBar        | <a href="#">MenuItem</a>    | <p>The menubar that is associated with the window. This is the menubar that is added to the project by default or is assigned using the <a href="#">Application</a> class's <code>MenuBar</code> property.</p> <p>When the window is active, this menubar is displayed. If no menubar is assigned to the window, the application's global menubar is used. To hide the menubar, set <code>MenuBar</code> to <a href="#">Nil</a>.</p> <p>On Windows and Linux, a window can have a menubar just below its Title bar. It uses the menubar assigned to the window or the application's menubar if no menubar is assigned to the window. If it is a Windows MDI application, the MDI window uses the Application's menubar. Individual document windows cannot have a menubar, but this is due to a limitation built into Windows. However, floating windows can have a menubar, and those windows will display the menubar you specify for them.</p> |
| MenuBarVisible | <a href="#">Boolean</a>     | <p>When set to <a href="#">False</a>, the menu bar is hidden whenever the window is the frontmost non-floating window. It actually is designed to hide all system-wide UI. On Macintosh, it also hides the Dock and any global floating windows. On Windows, it hides the Taskbar and Start button. If you only want to hide the menubar, set the <code>MenuBar</code> property to <a href="#">Nil</a>.</p> <p>Set <code>MenuBarVisible</code> to <a href="#">False</a> and <code>FullScreen</code> to <a href="#">True</a> to allow the window to 'take over' the entire screen. The menu bar will be redisplayed if another window (whose <code>MenuBarVisible</code> property is set to <a href="#">True</a>) becomes frontmost, or when the application is put into the background.</p> <p><code>MenuBarVisible</code> defaults to <a href="#">True</a>.</p>  |
| MinHeight      | <a href="#">Integer</a>     | The minimum height to which the window can be resized.  |
| MinimizeButton | <a href="#">Boolean</a>     | If <a href="#">True</a> , the window will include a Minimize button in its Tile bar. The default is <a href="#">True</a> . Windows only.  |
| MinWidth       | <a href="#">Integer</a>     | The minimum width to which the window can be resized.   |
| MouseCursor    | <a href="#">MouseCursor</a> | The cursor to be displayed while the mouse is within the window and the <a href="#">Application</a> class's <code>MouseCursor</code> property is <a href="#">Nil</a> . If the <a href="#">Application</a> class <code>MouseCursor</code> is not <a href="#">Nil</a> , non- <a href="#">Nil</a> <code>MouseCursors</code> belonging to any Windows or <a href="#">Controls</a> are ignored. If the Window's <code>MouseCursor</code> property is not <a href="#">Nil</a> , the <code>MouseCursor</code> properties of any <a href="#">Controls</a> are ignored. You can use the cursors in the <a href="#">.Cursors</a> object to set the mousecursor for the window.  |

## Window Class

---

| Name             | Type                    | Description  |
|------------------|-------------------------|--|
| <b>MouseX</b>    | <a href="#">Integer</a> | The X coordinate of the mouse (pixels). Measured from the top-left corner of the window.   |
| <b>MouseY</b>    | <a href="#">Integer</a> | The Y coordinate of the mouse (pixels). Measured from the top-left corner of the window.   |
| <b>Placement</b> | <a href="#">Integer</a> | <p>The location where the window will be placed when it opens.</p> <p>0 - Default<br/>1 - Parent Window<br/>2 - Main Screen<br/>3 - Parent Window Screen<br/>4 - Stagger</p> <p>For Drawer windows, the values have the following meaning:</p> <p>0 - Top/Left<br/>1 - Center<br/>2 - Bottom/Right</p> |
| <b>Resizable</b> | <a href="#">Boolean</a> | If <a href="#">True</a> , the window is resizable.   |
| <b>Title</b>     | <a href="#">String</a>  | The text in the window's Title bar.  |
| <b>Top</b>       | <a href="#">Integer</a> | The distance (in pixels) between the top edge of the screen and the top edge of the content region of the window. The window's title bar is taken into account.  |
| <b>Visible</b>   | <a href="#">Boolean</a> | If <a href="#">True</a> , the window will be visible when it is opened.  |
| <b>Width</b>     | <a href="#">Integer</a> | The width of the content region of the window. The left and right window frames are taken into account.  |

## Methods

| Name              | Parameters                         | Description  |
|-------------------|------------------------------------|--|
| AcceptFileDrop    | FileType as <a href="#">String</a> | Permits documents of type FileType to be dropped on the window. <i>FileType</i> must be a file type you specified via the <a href="#">FileType</a> class or the File Type Sets Editor. It works correctly even if you have multiple file types defined with the same File type code, and you don't specifically call AcceptFileDrop for the first such type. |
| AcceptPictureDrop |                                    | Permits pictures to be dropped on the window.  |
| AcceptRawDataDrop | Type as <a href="#">String</a>     | Permits data (of the type specified) to be dropped on the window. Type is a four-character resource code, e.g., 'snd ' or 'TEXT'   |
| AcceptTextDrop    |                                    | Permits text to be dropped on the window.  |

| Name            | Parameters   | Description   |
|-----------------|--|---|
| Close           |  | Closes the window. Once closed, a window cannot be refreshed or redrawn.  |
| DrawInto        | g as <a href="#">Graphics</a><br>x as <a href="#">Integer</a><br>y as <a href="#">Integer</a>  | Draws the contents of the window into the specified <a href="#">Graphics</a> context.   |
| Hide            |  | Makes the window invisible.   |
| Maximize        |  | Maximizes the window.   |
| Minimize        |  | Minimizes the window.   |
| Refresh         | [EraseBackground as <a href="#">Boolean</a> ]  | Repaints the entire contents of the window. If the optional parameter <i>EraseBackground</i> is <a href="#">True</a> , the background will be erased prior to redrawing the window. The default is <a href="#">True</a> .     |
| RefreshRect     | X As <a href="#">Integer</a> ,<br>Y As <a href="#">Integer</a> ,<br>Width As <a href="#">Integer</a> , Height As <a href="#">Integer</a><br>[,EraseBackground as <a href="#">Boolean</a> ] | Repaints only the region of the window specified. If the optional parameter <i>EraseBackground</i> is <a href="#">True</a> , the background will be erased prior to redrawing the area. The default is <a href="#">True</a> . |
| Restore         |  | Restores a minimized window its previous size.  |
| SetFocus        |  | Removes the focus from the control that currently has the focus, leaving no control with the focus. Use the SetFocus method of the <a href="#">RectControl</a> class to set the focus to a particular control in the window.  |
| Show            |  | Forces the window to immediately become visible rather than wait until the application is idle.   |
| ShowModal       |  | Displays the window and causes the current method to stop executing until the window closes or becomes invisible.   |
| ShowModalWithin | parentWindow as <a href="#">Window</a>   | Displays the window within the <i>parentWindow</i> . Used to display Sheet and Drawer windows within a specific parent (Mac OS X).  |

## Window Class

---

| Name       | Parameters                                  | Description  |
|------------|---|--|
| ShowWithin | parentWindow as Window [,facing as Integer] | Displays the window within the <i>parentWindow</i> . Used to display Sheet and Drawer windows within a specific parent. For a Drawer window, the optional <i>facing</i> parameter indicates which side the drawer pops out of:<br>-1 = default (determined by system -- usually left),<br>0 = top,<br>1 = bottom,<br>2 = right,<br>3 = left (same ordering as tab panels). If you omit <i>facing</i> , the default value will be used. |
| UpdateNow  |   | Flushes the back buffer on Mac OS X windows. Because Mac OS X is double-buffered, drawing is accumulated before being shown on screen. You may want to flush the buffer manually after updating a progress bar, drawing the frame of an animation on a canvas, or other things of a similar nature. UpdateNow has no effect on Windows, Linux, and Mac OS "classic".   |

## Events

| Name        | Parameters             | Description   |
|-------------|------------------------|---|
| Activate    |                        | The window is being activated.  |
| CancelClose | appQuitting as Boolean | The window is about to be closed by the user clicking the close button or by calling the <a href="#">Quit</a> method. When the application is quitting, CancelClose and Close are called as a pair for each open window.<br><a href="#">Return True</a> to prevent the window (and in the case of the <a href="#">Quit</a> method, other open windows) from closing. The parameter <i>appQuitting</i> is <a href="#">True</a> when the window is being closed because the whole application is quitting and <a href="#">False</a> when only the window is being closed. |

| Name                 | Parameters   | Description  |
|----------------------|--|--|
| Close                |  | The window is about to close. When the application is quitting, CancelClose and Close are called as a pair for each open window.<br>Prior to version 6.0, every window's CancelClose event was called and if none of them canceled, every Close event was called. This change brings applications made with REALbasic more in line with other applications from other vendors.   |
| ConstructContextMenu | Base as <a href="#">MenuItem</a> ,<br>x as <a href="#">Integer</a> ,<br>y as <a href="#">Integer</a> | Fires whenever it is appropriate to display a ContextualMenu for the control. This is the recommended way to handle contextual menus because this event figures out whether the user has requested the contextual menu, regardless of how he did it. Returns a <a href="#">Boolean</a> . Base is analogous to the menu bar for the contextual menu. Any items you add to Base will be shown as menu items. If you return <a href="#">False</a> , the event is passed up the parent hierarchy. If you return <a href="#">True</a> , the contextual menu is displayed. The parameters x and y are the mouse locations. If the event was fired because of a non-mouse event, then x and y are both set to -1. See the example of a contextual menu in the examples for the <a href="#">RectControl</a> class. |
| ContextMenuAction    | HitItem as <a href="#">MenuItem</a>  | Fires when a contextual menuitem HitItem was selected but the Action event and the MenuHandler for the menuitem did not handle the menu selection. This event gives you a chance to handle the menu selection by inspecting the menuitem's Text or Tag properties to see which item was selected. Use this in conjunction with ConstructContextMenu if you have not specified the Action event or the Menu Handler for the items on the contextual menu.   |
| Deactivate           |  | The window is being deactivated.   |
| DragEnter            | obj as <a href="#">DragItem</a>  | Fired when the <a href="#">DragItem</a> enters the <a href="#">Window</a> . Returns a <a href="#">Boolean</a> . Return <a href="#">True</a> from this event to prevent the drop from occurring.  |
| DragExit             | obj as <a href="#">DragItem</a>  | Fired when the <a href="#">DragItem</a> exits the <a href="#">Window</a> .   |

| Name            | Parameters  | Description   |
|-----------------|---|---|
| DragOver        | x as <a href="#">Integer</a> , y as <a href="#">Integer</a> , obj as <a href="#">DragItem</a> | Fired when the <a href="#">DragItem</a> is over the <b>Window</b> . The coordinates x and y are relative to the <b>Window</b> . Returns a <a href="#">Boolean</a> . Return <a href="#">True</a> from this event to prevent the drop from occurring. |
| DropObject      | Obj as <a href="#">DragItem</a>   | The item represented by Obj has been dropped on the window.   |
| EnableMenuItems |   | The user has clicked in the menu bar or pressed a keyboard shortcut assigned to one of the menu items. Use this event handler to determine whether conditions are right to enable the menu items.   |
| KeyDown         | Key as <a href="#">String</a>   | The Key passed has been pressed and not handled by an object in the window.   |
| Maximize        |   | The window has been maximized.  |
| Minimize        |   | The window has been minimized.  |
| MouseDown       | X as <a href="#">Integer</a> , Y as <a href="#">Integer</a>                                   | The mouse button has been pressed inside the window at the x,y local coordinates passed. <a href="#">Return True</a> if you are going to handle the mouseDown.  |
| MouseDrag       | X as <a href="#">Integer</a> , Y as <a href="#">Integer</a>                                   | The mouse button was pressed inside the window and moved (dragged) at the location local to the window passed in to x,y. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.                                   |
| MouseEnter      |   | The mouse has entered the window.   |
| MouseExit       |   | The mouse has left the window.  |
| MouseMove       | X as <a href="#">Integer</a> , Y as <a href="#">Integer</a>                                   | The mouse has moved within the window to the x,y local coordinates passed.  |
| MouseUp         | X as <a href="#">Integer</a> , Y as <a href="#">Integer</a>                                   | The mouse button has been released inside the window at the x,y local coordinates passed. This event will not occur unless you return <a href="#">True</a> in the MouseDown event.  |
| Moved           |   | The window has been moved.  |
| Open            |   | The window is about to open.  |
| Paint           | g as <a href="#">Graphics</a>   | The window needs to be redrawn. The parameter passed gives you access to the <a href="#">Graphics</a> class methods for drawing.  |
| Resized         |   | The window has been resized.  |

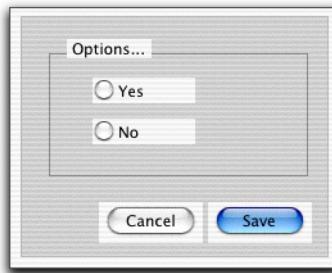
| Name     | Parameters | Description   |
|----------|------------|---|
| Resizing |            | The user is in the process of resizing the window. Resizing executes regardless of whether the LiveResize property is <a href="#">True</a> in Carbon. Note: The Top, Left, Width and Height properties have already been adjusted to account for sizing at this point if LiveResize is on, but the controls haven't drawn themselves. You could, for example, use a <a href="#">Boolean</a> property to indicate whether a <a href="#">Canvas</a> control should redraw in its Paint event. |
| Restore  |            | The window is being restored from its minimized state to its state prior to being minimized.  |

**Notes**

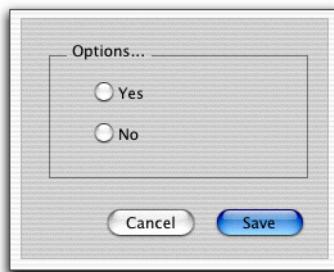
Window constructors are called after the window and its controls are created, but before the Open events are called.

**The Composite Property**

The Composite property solves a display problem that occurs under Mac OS X. In certain cases, a control that is placed in a Metal or Drawer window or on top of another control shows its rectangular border as a white background unless the window's Composite property is turned on. The problem doesn't come up if the controls are intrinsically rectangular. For Drawer and Metal window types, the Composite property is turned on automatically. The problem can also occur in regular Document windows. For example, some controls that are placed on top of a [Placard](#) control in a Document window show this problem.



In this example, [PushButtons](#), [RadioButtons](#), and a [GroupBox](#) are inside a [Placard](#) control and the window's Composite property is off. Turning on the Composite property removes the unsightly white rectangles around each control.



### Custom Window Types

If the MacProcID property is set to a non-zero value (which can be done in the IDE only), the window will be drawn according to this value. When you do so, the behaviour of the window will correspond to the value of the Frame property, but the appearance is controlled by the Window Definition provided by MacProcID. This feature allows you to define window types that are not defined directly by REALbasic (i.e., via the Frame property).

For a discussion of MacProcID values, see the Apple developer documentation “Window Definition IDs” at  
<http://developer.apple.com/techpubs/macos8/HumanInterfaceToolbox/WindowManager/WindowMgr8Ref/WindowMgrRef.b.html>.

The following table summarizes the possible values of MacProcID. Values of 1024 to 1087 require the Appearance Manager (Mac OS 8 or above); other values are pre-Appearance Manager. On Windows, the value of the Frame property controls the window type regardless of the value of MacProcID.

| Value | Description  |
|-------|--|
| 1     | Modal dialog box   |
| 2     | Modeless dialog box  |
| 3     | Modeless dialog box with shadow  |
| 4     | Movable window with no size box  |
| 5     | Movable dialog box with title bar  |
| 8     | Movable window with zoom box and size box  |
| 12    | Movable window with zoom box and no size box                                       |
| 16    | Round corner window (adding 2, 4, or 6 to 16 changes the curvature of the corners) |
| 1024  | Movable window with no size box or zoom box  |
| 1025  | Movable window with size box   |
| 1026  | Movable window with vertical zoom box and no size box                              |

| Value | Description  |
|-------|--|
| 1027  | Movable window with vertical zoom box and size box                     |
| 1028  | Movable window with horizontal zoom box and no size box                |
| 1029  | Movable window with horizontal zoom box and size box                   |
| 1030  | Movable window with full zoom box and no size box                      |
| 1031  | Movable window with full zoom box and size box                         |
| 1040  | Modeless dialog box  |
| 1041  | Modeless dialog box with shadow  |
| 1042  | Modal dialog box   |
| 1043  | Movable modal dialog box   |
| 1044  | Alert box  |
| 1045  | Movable alert box  |
| 1046  | Movable modal dialog box with size box                                 |
| 1057  | Floating window with no size box or zoom box                           |
| 1059  | Floating window with size box  |
| 1061  | Floating window with vertical zoom box                                 |
| 1063  | Floating window with vertical zoom box and size box                    |
| 1065  | Floating window with horizontal zoom box                               |
| 1067  | Floating window with horizontal zoom box and size box                  |
| 1069  | Floating window with full zoom box                                     |
| 1071  | Floating window with full zoom box and size box                        |
| 1073  | Floating window with side title bar                                    |
| 1075  | Floating window with side title bar and size box                       |
| 1077  | Floating window with side title bar and vertical zoom box              |
| 1079  | Floating window with side title bar, vertical zoom box, and size box   |
| 1081  | Floating window with side title bar and horizontal zoom box            |
| 1083  | Floating window with side title bar, horizontal zoom box, and size box |
| 1085  | Floating window with side title bar and full zoom box                  |
| 1087  | Floating window with side title bar, full zoom box, and size box       |
| 1985  | Floating window with no size box or zoom box                           |
| 1987  | Floating window with size box  |
| 1989  | Floating window with zoom box  |
| 1991  | Floating window with zoom box and size box                             |
| 1993  | Floating window with side title bar and no size or zoom boxes          |
| 1995  | Floating window with side title bar and size box                       |
| 1997  | Floating window with side title bar and zoom box                       |
| 1999  | Floating window with side title bar, size box, and zoom box.           |

## Window Class

---

### Custom Cursors

The MouseCursor property controls the appearance of the pointer when it is over the window, provided the [MouseCursor](#) property of the Application class is [Nil](#). If you also want the pointer to change to another shape when it is over a control within the window, you must either assign the new value to the Window's MouseCursor property or temporarily set the window's MouseCursor property to [Nil](#) and the control's [MouseCursor](#) property to the desired cursor. See the section on the [MouseCursor](#) class for an example.

You can assign a MouseCursor using the library of cursors in the [Cursors](#) object.

### Examples

This example sets the background color of the window to grey, provided the HasBackColor property is set to [True](#).

```
backcolor=RGB\(80,80,80\)
```

This example increases the width of the window by 20 pixels.

```
width=width+20
```

This example changes the title of the window.

```
title="Document 1"
```

### The Resizing Event Handler

This example resizes three EditFields on the form in the Resizing event. The three EditFields are aligned horizontally. As the user stretches or shrinks the window, the EditFields' widths change proportionally.

```
Dim availableSpace as Integer  
Dim field1size, field2size, field3size as Integer  
  
//subtract 40 pixels for the space between the  
//three fields and on left and right side of the window  
availableSpace=me.width-40  
  
//calculate the size of each field based on a percentage  
Field1size=availableSpace*.6 //60 percent  
Field2size=availableSpace*.3 //30 percent  
Field3size=availableSpace*.1 //10 percent  
  
//Set the field widths  
Editfield1.width=field1size  
Editfield2.width=field2size  
Editfield3.width=field3size  
  
//reposition the fields based on the new sizes  
Editfield2.left=editfield1.left+editfield1.width+10  
Editfield3.left=editfield2.left+editfield2.width+10
```

**Drawinto Example**

This example draws the contents of a window into a [Graphics](#) object. For example, if you create a new project, add a second window, and add a [Canvas](#) control to the second window then in the Paint event of the first window, put:

```
Self.drawinto(dialog1.canvas1.graphics,0,0)
```

The contents of the first window will be draw into the canvas control on the second window.

**Accessing Controls and Properties in Other Windows**

When changing a window property from another window, there are two possible approaches you can take. If you have only one instance of the window you need to reference, you can use the window's object name as a reference. For example, if you had a window called window1 that had an [EditField](#) called EditField1 and you wanted to assign the value "Fred" to the text property of that EditField, you would using the following syntax:

```
window1.EditField1.text="Fred"
```

If you have multiple instances of the same window class open at the same time, then a reference to the target window must be included as in this example where a new window is opened and its window title changed. "anotherWindow" is a window class in the Project Editor.

```
Dim w As anotherWindow
w=New anotherWindow
w.title="Your Results"
```

**Handling Drag and Drop**

For an explanation of handling drag and drop, see the [Control](#) class and the [DragItem](#) class.

**See Also**

[Window](#) function; [MessageDialog](#) class.

## Window Function

Returns a reference to an open window.

**Syntax**

**result=Window(WindowNumber)**

| Part         | Type                    | Description  |
|--------------|-------------------------|--|
| result       | <a href="#">Window</a>  | A reference to the window whose number was passed. |
| WindowNumber | <a href="#">Integer</a> | The number of the window you want a reference to.  |

### Notes

The **Window** function returns a reference to the window number passed. The window list contains all the windows that have been created. Window zero is the frontmost window. Floating windows are always in front of document windows. For example, to get the frontmost document window when you also have Floating windows, you must also check each window's Frame property. In evaluating the order of windows in the list from front to back, keep in mind that a window may not have its Visible property set to **True**. If this may be the case, check the Visible property of a window while identifying the frontmost visible window. If you don't, the code may identify the frontmost window as a window that is not Visible.

This function can be used in conjunction with the [WindowCount](#) function to loop through the open windows.

### Example

This example places the titles of all open windows into a [ListBox](#):

```
Dim i as Integer
Dim WindowTitle as String
If WindowCount > 1 then
  For i=0 to WindowCount-1
    WindowTitle=Window(i).Title
    ListBox1.AddRow windowTitle
  Next
  Return Nil
End If
```

This example gets the frontmost Document window. It considers only Document windows (Frame=0) takes into account the possibility that a window might not be visible:

```
Dim i,wc as Integer
' Compute the total number of windows for iteration.
wc = WindowCount - 1
  ' Go through the window list.
For i = 0 to wc
  ' Is this a visible document window?
  If (Window(i).Frame = 0) and Window(i).visible then
    ' This is the fist visible document window.
    Return Window(i)
  end if
Next
```

### See Also

[WindowCount](#) function; [Window](#) class.

## WindowCount Function

Returns the number of open windows.

### Syntax

**result=WindowCount**

| Part   | Type                    | Description                 |
|--------|-------------------------|-----------------------------|
| result | <a href="#">Integer</a> | The number of open windows. |

### Notes

The **WindowCount** function returns the number of open windows. This includes any windows that are invisible.

### Examples

See the [Window](#) function for an example.

### See Also

[Window](#) function; [Window](#) class.

---

## WindowPtr Data Type

A 4-byte integer which is a synonym for the Handle property of a [Window](#). You can pass a [Window](#) object to this data type and it will be treated as a pointer to the [Window](#).

This data type can be used only with the [Declare](#) statement and it cannot be used to declare REALbasic variables, properties, and methods.

### See Also

[Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WString](#) data types.

---

## WordApplication Class

Used to automate Microsoft Word.

### Super Class

[OLEObject](#)

### Notes

The language that you use to automate Microsoft Office applications is documented by Microsoft and numerous third-party books on Visual Basic for Applications (VBA). Microsoft Office applications provide online help for VBA. To access the online help, choose Macros from the Tools Menu of your MS Office application, and then choose Visual Basic Editor from the Macros submenu. When the Visual Basic editor appears, choose Microsoft Visual Basic Help from the Help menu. The help is contextual in the

## WordApplication Class

---

sense that it provides information on automating the Office application from which you launched the Visual Basic editor.

If VBA Help does not appear, you will need to install the help files. On Windows Office 2003, Office prompts you to install the VBA help files when you first request VBA help. You don't need the master CD. On Macintosh, Office v.X does not install the VBA help files as part of the full install. Quit out of Office and locate your master CD. Open the "Value Pack" folder and double-click the Value Pack installer. In the Value Pack installer dialog, scroll down to the Programmability topic, select it, and click Continue. The installer will then add the VBA help files and examples to your Office installation. When the install finishes, the VBA help files will be available to the Visual Basic editor within all your Office X applications.

Microsoft has additional information on VBA at <http://msdn.microsoft.com/vbasic/> and have published their own language references on VBA. One of several third-party books on VBA is "VB & VBA in a Nutshell: The Language" by Paul Lomax (ISBN: 1-56592-358-8).

### Example

This example creates a Word document from the text entered into and [EditField](#) on the form, EditField1. The code is in a [PushButton's](#) Action event handler. It also sets the font and font size of the text.

```
Dim word as WordApplication
Dim zdoc as WordDocument
Dim style as WordStyle
Dim v as Variant

word = New WordApplication
doc = word.documents.add
doc.range.text = Editfield1.text

v = doc.paragraphs.item(1).style
If v.objectvalue IsA wordstyle then
    style = wordstyle(v.objectvalue)
    styletext.text = style.namelocal
    stylefontname.text = style.font.name
    stylefontsize.text = Str(style.font.size)
end if

Exception err as OLEException
Msgbox err.message
```

This example does a search and replace operation on the text in EditField1, using the text in the [EditFields](#), FindText and ReplaceText. The Word document is updated after the search and replace.

```

Dim word as WordApplication
Dim doc as WordDocument
Dim s as String
Dim r as Boolean

Const wdFindContinue = 1
Const wdReplaceAll = 2

word = New WordApplication
doc = word.activedocument

s = ReplaceAll(Editfield1.text, FindText.text, ReplaceText.text)

#if TargetWin32 then
    doc.range.text = s
#else
    r = word.selection.find.execute(FindText.text, False, False, False, False, 0, True, _
        wdFindContinue, False, ReplaceText.text, wdReplaceAll)
#endif

Editfield1.text = doc.range.text

Exception err as OLEException
Msgbox err.message

```

**See Also** [ExcelApplication](#), [Office](#), [OLEObject](#), [OLEException](#), [PowerPointApplication](#) classes.

## Writable Class Interface

Provides “specs” for methods which can be used with REALbasic classes to write data to disk or ports.

### Methods

| Name       | Parameters     | Description  |
|------------|----------------|--|
| Flush      |                | Flushes the contents of the internal buffers to disk, clearing the buffers.                      |
| Write      | text as String | Writes the passed Text to the disk or port.  |
| WriteError |                | Returns a Boolean. Returns <a href="#">True</a> if there were any errors in the write operation. |

## WString Data Type

---

**Notes** The [BinaryStream](#), [IPCSocket](#), [Serial](#), [StdErr](#) and [StdOut](#) (console applications only), [TCPSocket](#), and [TextOutputStream](#) provide the **Writable** class interface. If you implement this class interface in your application, you must provide the methods with the parameters as shown here.

For more information about class interfaces and how to implement them, see the section “Class Interfaces” in the *User’s Guide*.

**See Also** [BinaryStream](#), [IPCSocket](#), [Serial](#), [StdErr](#), [StdOut](#), [TCPSocket](#), [TextOutputStream](#) classes.

---

## WString Data Type

A null-terminated UTF-16 string. This is typically used on NT-based versions of Windows. You can pass a regular REALbasic [String](#) to it and it will be converted to UTF-16 and terminated automatically.

WString can be used only with the [Declare](#) statement and cannot be used to declare REALbasic variables, properties, and methods.

**See Also** [Declare](#) statement; [Byte](#), [CFStringRef](#), [CString](#), [OSType](#), [PString](#), [Ptr](#), [Short](#), [UByte](#), [UShort](#), [WindowPtr](#) data types.

---

## XmlAttribute Class

Represents an XML attribute node.

**Super Class** [XMLNode](#)

### Properties

| Name         | Type                       | Description   |
|--------------|----------------------------|---|
| OwnerElement | <a href="#">XMLElement</a> | A reference to the <a href="#">XMLElement</a> that this attribute is part of. |

**Notes** To retrieve attributes by index, use [XMLElement](#)’s GetAttributeNode method.

**See Also** [XMLElement](#), [XMLDocument](#), [XMLNode](#) classes.

## XMLAttributeList Class

A helper class that is used in the [XMLReader's](#) StartElement event and the [XMLXsltHandler's](#) StartElement event. A basic structure for a collection of attributes. The attributes can be retrieved by index or name.

**Super Class** [Object](#)

### Properties

| Name  | Type                    | Description                           |
|-------|-------------------------|---------------------------------------|
| Count | <a href="#">Integer</a> | The number of attributes in the list. |

### Methods

| Name  | Parameters                       | Description  |
|-------|----------------------------------|--|
| Key   | Index as <a href="#">Integer</a> | Retrieves an attribute by index position. Returns the attribute as a String. <i>Index</i> is zero-based.       |
| Value | Index as <a href="#">Integer</a> | Retrieves an attribute value by index position. Returns the attribute as a String. <i>Index</i> is zero-based. |
| Value | Key as <a href="#">String</a>    | Retrieves an attribute value.  |

**See Also** [XmlAttribute](#), [XMLDocument](#), [XMLNode](#) classes.

---

## XMLCDATASection Class

**XMLCDATASection** enables you to use the [IsA](#) operator to determine whether an [XMLNode](#) is an **XMLCDATASection**.

**Super Class** [XMLNode](#)

**Properties** None

**Methods** None

**See Also** [XmlAttribute](#), [XMLComment](#), [XMLDocument](#), [XMLElement](#), [XMLNode](#) classes.

## XMLComment Class

**XMLComment** enables you to use the [IsA](#) operator to determine whether an [XMLNode](#) is an **XMLComment**.

**Super Class** [XMLNode](#)

**Properties** None

**Methods** None

**See Also** [IsA](#) operator; [XMLNode](#) class.

---

## XMLContentModel Class

Represents a parsed DTD element declaration. Expat defines this structure and **XmlContentModel** is a direct representation of it. It is passed to the ElementDecl event of [XMLReader](#). This will be useful to anyone building DTD validation routines.

**Super Class** [Object](#)

**Properties**

| Name        | Type                    | Description  |
|-------------|-------------------------|--|
| Name        | <a href="#">String</a>  | The name of the item.                                  |
| NumChildren | <a href="#">Integer</a> | The number of children.                                |
| Quant       | <a href="#">Integer</a> | XML Content Quant. See the constants for valid quants. |
| Type        | <a href="#">Integer</a> | XML content type. See the constants for valid types.   |

**Methods**

| Name  | Parameters                       | Description                         |
|-------|----------------------------------|-------------------------------------|
| Child | Index as <a href="#">Integer</a> | Returns an <b>XMLContentModel</b> . |

**Constants** Here are the constants that can be compared to the value of the Type property:

| Value | Constant                         |
|-------|----------------------------------|
| 1     | <a href="#">XML_CTYPE_EMPTY</a>  |
| 2     | <a href="#">XML_CTYPE_ANY</a>    |
| 3     | <a href="#">XML_CTYPE_MIXED</a>  |
| 4     | <a href="#">XML_CTYPE_NAME</a>   |
| 5     | <a href="#">XML_CTYPE_CHOICE</a> |

| Value | Constant      |
|-------|---------------|
| 6     | XML_CTYPE_SEQ |

Here are the constants that can be compared to the value of the Quant property:

| Value | Constant        |
|-------|-----------------|
| 0     | XML_CQUANT_NONE |
| 1     | XML_CQUANT_OPT  |
| 2     | XML_CQUANT REP  |
| 3     | XML_CQUANT_PLUS |

## Notes

If Type is XML\_CTYPE\_EMPTY or XML\_CTYPE\_ANY then quant will be XML\_CQUANT\_NONE and the other fields will be zero or NULL. If Type is XML\_CTYPE\_MIXED, then quant will be None or REP and numChildren will contain the number of elements that may be mixed in and children point to an array of XMLContent cells that will be all of XML\_CTYPE\_NAME type with no quantification.

If Type is XML\_CTYPE\_NAME, then the name points to the name and the numChildren field will be zero and children will be NULL. The quant fields indicate any quantifiers placed on the name.

CHOICE and SEQ will have name NULL, the number of children in numChildren and children will point, recursively, to an array of XMLContent cells.

The EMPTY, ANY, and MIXED types will occur only at the top level.

## See Also

[XMLReader](#) class.

---

# XMLDocument Class

Represents an XML document. It is used for both parsing existing XML data into a DOM document structure or creating a new XML document from scratch.

**Super Class** [XMLNode](#)

## Constructor

| Name        | Parameters                      | Description                                     |
|-------------|---------------------------------|---|
| XMLDocument |                                 | Creates an empty document.                      |
| XMLDocument | XML as <a href="#">String</a>   | Parses the passed XML string into the document. |
| XMLDocument | f as <a href="#">FolderItem</a> | Parses the passed XML file into the document.   |

## Properties

| Name               | Type                       | Description   |
|--------------------|----------------------------|---|
| DocumentElement    | <a href="#">XMLElement</a> | Refers to the top-level element of the document.  |
| PreserveWhiteSpace | <a href="#">Boolean</a>    | Setting to <a href="#">False</a> strips out returns and tabs between elements. Setting to <a href="#">True</a> leaves them in place. The default is <a href="#">False</a> . |
| ToString           | <a href="#">String</a>     | Contains a string representation of the document.   |

## Methods

| Name                        | Parameters  | Description   |
|-----------------------------|---|---|
| CreateAttribute             | Name as <a href="#">String</a>  | Creates an attribute node as an <a href="#">XmlAttribute</a> .  |
| CreateAttribute             | URI as <a href="#">String</a> , Name as <a href="#">String</a>              | Creates an attribute nodes as an <a href="#">XmlAttribute</a> with a namespace declaration.   |
| CreateCDATASection          | Data as <a href="#">String</a>  | Creates a CDATA section with <i>Data</i> and returns it as an <a href="#">XMLCDATASection</a> .   |
| CreateComment               | Data as <a href="#">String</a>  | Creates an <a href="#">XMLComment</a> with <i>Data</i> and returns it as an <a href="#">XMLComment</a> .  |
| CreateElement               | TagName as <a href="#">String</a>   | Creates an element as an <a href="#">XMLElement</a> . Returns an <a href="#">XMLElement</a> .   |
| CreateElement               | URI as <a href="#">String</a> , Tagname as <a href="#">String</a>           | Creates an element with a namespace declaration as an <a href="#">XMLElement</a> . Returns an <a href="#">XMLElement</a> .  |
| CreateProcessingInstruction | Target as <a href="#">String</a> , Data as <a href="#">String</a>           | Creates an <a href="#">XMLProcessingInstruction</a> with the passed <i>target</i> keyword and <i>data</i> .   |
| CreateTextNode              | Data as <a href="#">String</a>  | Creates a text node with <i>Data</i> and returns it as an <a href="#">XMLTextNode</a> .   |
| ImportNode                  | foreignNode as <a href="#">XmlNode</a> , [deep as <a href="#">Boolean</a> ] | Copies a node from another XMLDocument into the current document. If the optional parameter <i>deep</i> is <a href="#">True</a> , ImportNode will import all the child nodes of <i>foreignNode</i> . After you import it, you must then place it somewhere. |
| LoadXML                     | Doc as <a href="#">String</a>   | Parses the passed XML string into the document.   |
| LoadXML                     | f as <a href="#">FolderItem</a>   | Parses the passed XML file into the document.   |

| Name      | Parameters   | Description  |
|-----------|--|--|
| Transform | xsl as <a href="#">String</a><br>or<br>xsl as <a href="#">XMLStyleSheet</a><br>[saxHandler as <a href="#">XMLSxItHandler</a> ] | Creates a new XML document that is the result of applying an SXLT stylesheet. Returns a <a href="#">String</a> . Optionally register an event handler <i>saxHandler</i> to receive SAX style events on the output XML during the transformation. |

**Example**

This example creates a very simple XML document.

```
Dim xml as  XmlDocument
Dim root, person, firstname, lastname as  XMLNode

xml =  New XmlDocument
root = xml.AppendChild(xml.CreateElement("root"))
person = root.AppendChild(xml.CreateElement("person"))
firstname = person.AppendChild(xml.CreateElement("firstname"))
firstname.AppendChild(xml.CreateTextNode("John"))
lastname = person.AppendChild(xml.CreateElement("lastname"))
lastname.AppendChild(xml.CreateTextNode("Doe"))
//create_output is a multi-line EditField
create_output.text = xml.ToString
```

This example adds “ID” and “active” attributes to an element.

```
Dim xml as  XmlDocument
Dim root, node, attNode as  XMLNode

xml =  New XmlDocument
root = xml.AppendChild(xml.CreateElement("root"))
node = root.AppendChild(xml.CreateElement("person"))

attNode = xml.CreateAttribute("id")
attNode.value = "55"

node.SetAttribute("active", "true")

create_output.text = xml.ToString
```

This example parses XML and displays the number of “Person” nodes and the document text in an [EditField](#).

```
Dim xdoc as XmlDocument
Dim node as XMLNode
Dim i, count as Integer
Dim out as String

// create a new document
xdoc = New XmlDocument

xdoc.PreserveWhitespace = False

// load and parse the xml in the EditField, parse_InputText
xdoc.LoadXml(parse_inputText.Text)

// lets first do some simple, static traversing of the xml
// this doesn't do any error checking to make sure stuff exists
// so if the document changes, it will break.

count = xdoc.DocumentElement.childCount
out = "People: " + Str(count) //number of Person nodes
out = out + EndOfLine

// childNodes are 0 based.
For i = 0 to count - 1

    node = xdoc.DocumentElement.Child(i)

    // this gets the textnode of the <firstname> node
    out = out + node.FirstChild.FirstChild.Value

    // this gets the textnode of the <lastname> node
    out = out + " " + node.LastChild.FirstChild.Value
    out = out + EndOfLine

    next

//display results in the EditField parse_output
parse_output.text = out
```

### See Also

[XmlAttribute](#), [XMLComment](#), [XMLCDATASection](#), [XMLODException](#),  
[XMLElement](#), [XMLEception](#), [XMLNode](#), [XMLNodeList](#),  
[XMLProcessingInstruction](#), [XMLTextNode](#) classes.

# XMLError

This exception may occur during the creation of a DOM document and something goes wrong. For example, it will occur if multiple elements are appended to the top level of a document.

**Super Class** [RuntimeError](#)

**Properties** None. Use the Message and ErrorNumber properties of the [RuntimeError](#) class.

**Constants** Compare the ErrorNumber to these class constants rather than the values to make your code more readable.

| Value | Constant                    |
|-------|-----------------------------|
| 0     | OK                          |
| 1     | INDEX_SIZE_ERR              |
| 2     | DOMSTRING_SIZE_ERR          |
| 3     | HIERARCHY_REQUEST_ERR       |
| 4     | WRONG_DOCUMENT_ERR          |
| 5     | INVALID_CHARACTER_ERR       |
| 6     | NO_DATA_ALLOWED_ERR         |
| 7     | NO_MODIFICATION_ALLOWED_ERR |
| 8     | NOT_FOUND_ERR               |
| 9     | NOT_SUPPORTED_ERR           |
| 10    | INUSE_ATTRIBUTE_ERR         |
| 11    | INVALID_STATE_ERR           |
| 12    | SYNTAX_ERR                  |
| 13    | INVALID_MODIFICATION_ERR    |
| 14    | NAMESPACE_ERR               |
| 15    | INVALID_ACCESS_ERR          |
| 16    | INVALID_NODE_TYPE           |
| 17    | QUERY_PARSE_ERR             |
| 18    | QUERY_EXECUTION_ERR         |
| 19    | NOT_OK                      |

# XMLElement Class

Used to represent an XML element.

**Super Class** [XMLNode](#)

## Properties

| Name           | Type                    | Description                                     |
|----------------|-------------------------|---|
| AttributeCount | <a href="#">Integer</a> | The number of attributes this element contains. |

## Methods

| Name             | Parameters  | Description   |
|------------------|---|---|
| GetAttribute     | Name as <a href="#">String</a><br>or<br>URI as <a href="#">String</a> ,<br>Name as <a href="#">String</a>   | Gets the value of the attribute specified by <i>Name</i> . Returns a <a href="#">String</a> .   |
| GetAttributeNode | Name as <a href="#">String</a><br>or<br>Index as <a href="#">Integer</a><br>or<br>URI as <a href="#">String</a> ,<br>Name as <a href="#">String</a>                               | Gets an XML attribute node of the attribute specified by <i>Name</i> , by <i>Index</i> position, or by URI and Name. Returns an <a href="#">XmlAttribute</a> . <i>Index</i> is zero-based.  |
| RemoveAttribute  | Name as <a href="#">String</a>  | Removes the attribute specified by <i>Name</i> or by node reference. Optionally returns as an <a href="#">XmlAttribute</a> the removed node   |
| RemoveAttribute  | attributeNode as <a href="#">XmlAttribute</a>   | Removes the attribute specified by the node reference. Optionally returns as an <a href="#">XmlAttribute</a> the removed node.  |
| SetAttribute     | Name as <a href="#">String</a> ,<br>Value as <a href="#">String</a><br>or<br>URI as <a href="#">String</a> ,<br>Name as <a href="#">String</a><br>Value as <a href="#">String</a> | Sets an attribute of the current element. The second syntax sets an attribute and declares a namespace.   |
| SetAttributeNode | AttributeNode as <a href="#">XmlAttribute</a> ,<br>[ns as <a href="#">Boolean</a> ]   | Sets an attribute node. Optionally returns a reference to a node as an <a href="#">XmlAttribute</a> that had the same name and was replaced. The optional parameter <i>ns</i> determines whether to use and declare any namespace data found in the passed <a href="#">XmlAttribute</a> . |

**See Also** [XMLDocument](#), [XMLNode](#) classes.

## XMLError

This exception occurs for a variety of reasons, such as errors in parsing XML. See the *Message* and *ErrorNumber* properties of the [RuntimeException](#) class for information about the error.

**Super Class** [RuntimeException](#)

## Properties

| Name | Type                   | Description  |
|------|------------------------|--|
| Line | <a href="#">String</a> | If it is a parsing error, the line number at which parsing failed. |
| Node | <a href="#">String</a> | If it is a parsing error, the node at which the parsing failed     |

See Also    [RuntimeException](#) class

## XMLNode Class

**XMLNode** is the base class for most of the XML Document Object Model (DOM) classes and implements many of the common properties and functions of a DOM node. This class won't usually be instantiated directly. Most objects will be created from methods of the [XMLDocument](#) class.

Note that not every property is utilized on every node type. For example, the Value property is used on an [XmlAttribute](#) and [XMLTextNode](#), but not an [XMLElement](#).

The REALbasic XML parser is based on Expat, which is described at <http://expat.sourceforge.net>.

Super Class [Object](#)

## Properties

| Name            | Type                        | Description   |
|-----------------|-----------------------------|---|
| ChildCount      | <a href="#">Integer</a>     | The number of child nodes contained by this node.   |
| FirstChild      | <a href="#">XMLNode</a>     | The first child of this node.   |
| LastChild       | <a href="#">XMLNode</a>     | The last child of this node.  |
| LastError       | <a href="#">Integer</a>     | Stores an error code if an error occurred.  |
| LocalName       | <a href="#">String</a>      | The name of the node without the prefix.  |
| Name            | <a href="#">String</a>      | The name of this node.  |
| NamespaceURI    | <a href="#">String</a>      | The namespaceURI of this node.  |
| NextSibling     | <a href="#">XMLNode</a>     | The next node.  |
| OwnerDocument   | <a href="#">XMLDocument</a> | The XMLDocument that contains this node.  |
| Parent          | <a href="#">XMLNode</a>     | The parent of this node.  |
| Prefix          | <a href="#">String</a>      | The namespace prefix of this node.  |
| PreviousSibling | <a href="#">XMLNode</a>     | The preceding node.   |
| ToString        | <a href="#">String</a>      | A string representation of this node.   |
| Type            | <a href="#">Integer</a>     | <a href="#">Integer</a> constant denoting the type, such as Element, Attribute, Textnode, and so forth. Use the constants of the <a href="#">XMLNodeType</a> object to get or set values. |

| Name  | Type                   | Description   |
|-------|------------------------|---|
| Value | <a href="#">String</a> | Used in some nodes to set or get the value, such as <code>TextNode</code> . Not supported for all node types. |

## Methods

| Name         | Parameters  | Description  |
|--------------|---|--|
| AppendChild  | NewChild as <a href="#">XMLNode</a>                                       | Adds a child after the last child. It optionally returns a reference to the new child as an <a href="#">XMLNode</a> .  |
| Child        | Index as <a href="#">Integer</a>  | Returns the child <a href="#">XMLNode</a> at the position denoted by <i>Index</i> . <i>Index</i> is zero-based.  |
| Clone        | Deep as <a href="#">Boolean</a>   | Duplicates the current node. The Deep parameter indicates whether to also duplicate all the child nodes. It returns an <a href="#">XMLNode</a> .   |
| Compare      | NodeToCompare as <a href="#">XMLNode</a>                                  | Compares two nodes. The return value works like <a href="#">StrComp</a> . It returns an <a href="#">Integer</a> :<br>If <code>string1 &lt; string2</code> it returns -1<br>If <code>string1 = string2</code> it returns 0<br>If <code>string1 &gt; string2</code> it returns 1 |
| Insert       | NewChild as <a href="#">XMLNode</a> , RefChild as <a href="#">XMLNode</a> | Inserts <i>NewChild</i> before the position of <i>RefChild</i> . It optionally returns a reference to the inserted node as an <a href="#">XMLNode</a> .  |
| RemoveChild  | OldChild as <a href="#">XMLNode</a>                                       | Removes <i>OldChild</i> .  |
| ReplaceChild | NewChild as <a href="#">XMLNode</a> , OldChild as <a href="#">XMLNode</a> | Replaces <i>oldChild</i> with <i>New child</i> . Optionally returns a reference to the new child as an <a href="#">XMLNode</a> .   |
| XQL          | Query as <a href="#">String</a> [,Map as <a href="#">String</a> ]         | Performs an XPath query and returns an <a href="#">XMLNodeList</a> of the resulting nodes. If the query has namespace references in it, you must provide declarations for them in the form of a string array, i.e.<br>("ns1", "http://foo", "ns2", "http://bar").              |

## Notes

The REALbasic XML library supports both parsing and creating of XML documents, DOM Level 2 support, XPath, XSLT, and full namespace support, including validation during creation. REALbasic's XML parser is based on the Expat library; additional information about Expat can be found by searching the internet for "Expat."

## See Also

[XMLDocument](#), [XMLElement](#), [XMLNodeList](#) classes.

# XMLNodeList Class

Stores the results of an XQL query done by the XQL method of the [XMLNode](#) class.

**Super Class** [Object](#)

## Properties

| Name      | Type                    | Description                            |
|-----------|-------------------------|--|
| LastError | <a href="#">Integer</a> | The last error code that was received. |
| Length    | <a href="#">Integer</a> | The number of nodes in the list.       |

## Methods

| Name     | Parameters                              | Description  |
|----------|---|--|
| Item     | Index as <a href="#">Integer</a>        | Retrieves the specified XML node from the list and returns it as an <a href="#">XMLNode</a> . The <i>Index</i> parameter is zero-based.                    |
| ToString | [WrapperTag as <a href="#">String</a> ] | Returns a <a href="#">String</a> representation of the nodes. The optional parameter <i>WrapperTag</i> puts a tag around the NodeList to make it parsable. |

**See Also** [XMLNode](#) class.

---

# XMLNodeType Object

Provides access to the Type constants for the Type property of [XMLNode](#).

**Constants** The **XMLNodeType** object contains constants for the Type property. They are shown in the following table.

| Value | Constant                    |
|-------|-----------------------------|
| 1     | Element_Node                |
| 2     | Attribute_Node              |
| 3     | Text_Node                   |
| 4     | CData_Section_Node          |
| 5     | Entity_Reference_Node       |
| 6     | Entity_Node                 |
| 7     | Processing_Instruction_Node |
| 8     | Comment_Node                |
| 9     | Document_Node               |
| 10    | Document_Type_Node          |
| 11    | Document_Fragment_Node      |

## XMLProcessingInstruction Class

---

| Value | Constant      |
|-------|---------------|
| 12    | Notation_Node |
| 13    | Other_Node    |

**See Also** [XMLNode](#) class.

---

## XMLProcessingInstruction Class

Represents an XML processing instruction. This class enables you to use the [IsA](#) operator to see whether an XMLNode is an XML processing instruction.

**Super Class** [XMLNode](#)

**Properties** None

**Methods** None

**See Also** [XMLNode](#) class; [IsA](#) operator.

---

## XMLReader Class

This is a wrapper class for the latest version of Expat (XML Parser Toolkit). Expat is a library for parsing XML documents. Create a subclass of **XMLReader** to get access to the events. Most of the events, methods and properties are straight mappings of the Expat features.

**Super Class** [Object](#)

**Constructor**

| Name      | Parameters   | Description  |
|-----------|--|--|
| XMLReader |  | Creates an XMLReader object.   |
| XMLReader | [XMLEncoding as <a href="#">String</a> ]<br>[,namespaceSeparator as <a href="#">String</a> ] | Creates an XMLReader object. The XMLEncoding parameter sets the encoding for the parser. The default encoding is UTF-8. The namespaceSeparator parameter forces Expat to expand namespace definitions. |

## Properties

| Name                    | Type                    | Description   |
|-------------------------|-------------------------|---|
| Base                    | <a href="#">String</a>  | Sets the base to be used for resolving relative URLs in system identifiers in declarations. |
| CurrentByteCount        | <a href="#">Integer</a> | The number of bytes in the current event.   |
| CurrentByteIndex        | <a href="#">Integer</a> | The byte position where the error occurred.   |
| CurrentColumnNumber     | <a href="#">Integer</a> | The column number the parser is currently on.   |
| CurrentLineNumber       | <a href="#">Integer</a> | The line number the parser is currently on.   |
| ErrorCode               | <a href="#">Integer</a> | The last error code.  |
| SetDefaultHandler       | <a href="#">Boolean</a> | Sets Expat to send unknown data to the default event. Entities are not parsed.              |
| SetDefaultHandlerExpand | <a href="#">Boolean</a> | Sets Expat to send unknown data to the Default event and expands entities.                  |

## Events

| Name             | Parameters  | Description  |
|------------------|---|--|
| AttListDecl      | elname as <a href="#">String</a> ,<br>atname as <a href="#">String</a><br>att_type as <a href="#">String</a> ,<br>dflt as <a href="#">String</a><br>isrequired as <a href="#">Boolean</a> | Fires at a DTD attlist declaration.  |
| Characters       | s as <a href="#">String</a>   | Fires at character data, s.  |
| Comment          | data as <a href="#">String</a>  | Fires at an XML comment, data.   |
| Default          | s as <a href="#">String</a>   | Fires for any characters in the XML document for which there is no applicable handler. |
| ElementDecl      | name as <a href="#">String</a><br>Content as<br><a href="#">XMLContentModel</a>   | Fires at a DTD element declaration.  |
| EndCDATA         |   | Fires at the end of CDATA.   |
| EndDoctypeDecl   |   | Fires at the end of a DocType declaration  |
| EndDocument      |   | Fires at the end of the document.  |
| EndElement       | Name as <a href="#">String</a>  | Fires at the end of the element, Name.   |
| EndPrefixMapping | prefix as <a href="#">String</a>  | Fires at the end of a namespace declaration.   |

| Name                  | Parameters  | Description  |
|-----------------------|---|--|
| EntityDecl            | entityName as <a href="#">String</a> ,<br>is_parameter_entity as <a href="#">Boolean</a> ,<br>Value as <a href="#">String</a> ,<br>Base as <a href="#">String</a> ,<br>SystemID as <a href="#">String</a> ,<br>publicid as <a href="#">String</a> ,<br>notationName as <a href="#">String</a> | Fires at an entity declaration. The is_parameter_entity will be True if the entity is a parameter entity. For internal entities(<!ENTITY foo "bar">), value will be populated and systemID, publicID, and notationName will be empty strings. For external entities, value will be an empty string and systemID will be populated. The publicID argument will be empty unless a public identifier was provided. The notationName argument will have a populated value only for unparsed entity declarations. |
| ExternalEntityRef     | Context as <a href="#">String</a> ,<br>Base as <a href="#">String</a> ,<br>SystemID as <a href="#">String</a> ,<br>PublicID as <a href="#">String</a>   | Fires at an external entity reference. Returns a <a href="#">Boolean</a> . <a href="#">Return False</a> to stop parsing and return True to report that handling of the external entity succeeded.  |
| NotationDecl          | NotationName as <a href="#">String</a> ,<br>Base as <a href="#">String</a> ,<br>SystemID as <a href="#">String</a> ,<br>PublicID as <a href="#">String</a>  | Fires at a notation.   |
| NotStandalone         |   | This is called if the document is not standalone, that is, it has an external subset or a reference to a parameter entity, but does not have Standalone=Yes. Returns a <a href="#">Boolean</a> . <a href="#">Return False</a> to cause parsing to stop.  |
| ProcessingInstruction | Target as <a href="#">String</a> ,<br>Data as <a href="#">String</a>  | Fires at the processing instruction, Target and Data.  |
| SkippedEntity         | EntityName as <a href="#">String</a> ,<br>is_parameter_entity as <a href="#">Boolean</a>  | Fires when an entity is reached, but entities are not currently being expanded.  |
| StartCDATA            |   | Fires at the start of CDATA.   |
| StartDoctypeDecl      | DoctypeName as <a href="#">String</a> ,<br>SystemID as <a href="#">String</a> ,<br>PublicID as <a href="#">String</a> ,<br>has_internal_subset as <a href="#">Boolean</a>   | Fires at the start of a DocType declaration.   |
| StartDocument         |   | Fires at the start of a document.  |
| StartElement          | Name as <a href="#">String</a> ,<br>AttributeList as <a href="#">XMLAttributeList</a>   | Fires at the start of the XML element, Name and AttributeList.   |
| StartPrefixMapping    | Prefix as <a href="#">String</a> ,<br>uri as <a href="#">String</a>   | Fires at the beginning of the namespace declaration, Prefix and URI.   |

| Name    | Parameters   | Description                       |
|---------|--|-----------------------------------|
| XMLDecl | version as <a href="#">String</a> ,<br>XMLEncoding as<br><a href="#">String</a> ,<br>Standalone as <a href="#">Boolean</a> | Fires on an XML declaration node. |

## Methods

| Name                       | Parameters   | Description   |
|----------------------------|--|---|
| CreateExternalEntityReader | XMLEncoding as<br><a href="#">String</a>   |   |
| DefaultCurrent             |  | If this is called from within an event, it will force the current data to route to the Default event.   |
| FreeExternalEntityReader   |  |   |
| Parse                      | s as <a href="#">String</a><br>[,isFinal as<br><a href="#">Boolean</a> ]           | Parses the data in the passed string, s. The optional parameter <i>isFinal</i> tells Expat what to do if the end of the data is reached and the XML is not well-formed, for example, if it stops in the middle of an element. Setting <i>isFinal</i> to <a href="#">False</a> will tell Expat not to throw an error because more data will be coming in another call. |
| Parse                      | file as<br><a href="#">FolderItem</a><br>[,isFinal as<br><a href="#">Boolean</a> ] | Parses the data in the passed document, file. <i>isFinal</i> tells Expat what to do if the end of the data is reached and the XML is not well-formed, as described above.   |
| Reset                      | [XMLEncoding<br>as <a href="#">String</a> ]  | Resets the parser. The optional parameter <i>XMLEncoding</i> specifies the Encoding of the data to be parsed.<br>Returns a <a href="#">Boolean</a>  |

## Constants

| Value | Constant                      |
|-------|-------------------------------|
| 0     | XML_ERROR_NONE                |
| 1     | XML_ERROR_NO_MEMORY           |
| 2     | XML_ERROR_SYNTAX              |
| 3     | XML_ERROR_NO_ELEMENTS         |
| 4     | XML_ERROR_INVALID_TOKEN       |
| 5     | XML_ERROR_UNCLOSED_TOKEN      |
| 6     | XML_ERROR_PARTIAL_CHAR        |
| 7     | XML_ERROR_TAG_MISMATCH        |
| 8     | XML_ERROR_DUPLICATE_ATTRIBUTE |

## XMLReaderException Class

---

| Value | Constant                                   |
|-------|--|
| 9     | XML_ERROR_JUNK_AFTER_DOC_ELEMENT           |
| 10    | XML_ERROR_PARAM_ENTITY_REF                 |
| 11    | XML_ERROR_UNDEFINED_ENTITY                 |
| 12    | XML_ERROR_RECURSIVE_ENTITY_REF             |
| 13    | XML_ERROR_ASYNC_ENTITY                     |
| 14    | XML_ERROR_BAD_CHAR_REF                     |
| 15    | XML_ERROR_BINARY_ENTITY_REF                |
| 16    | XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF    |
| 17    | XML_ERROR_MISPLACED_XML_PI                 |
| 18    | XML_ERROR_UNKNOWN_ENCODING                 |
| 19    | XML_ERROR_INCORRECT_ENCODING               |
| 20    | XML_ERROR_UNCLOSED_CDATA_SECTION           |
| 21    | XML_ERROR_EXTERNAL_ENTITY_HANDLING         |
| 22    | XML_ERROR_NOT_STANDALONE                   |
| 23    | XML_ERROR_UNEXPECTED_STATE                 |
| 24    | XML_ERROR_ENTITY_DECLARED_IN_PE            |
| 25    | XML_ERROR_FEATUREQUIRES_XML_DTD            |
| 26    | XML_ERROR_CANT_CHANGE_FEATURE_ONCE_PARSING |

**See Also** [XMLReaderException](#) class.

---

## XMLReaderException Class

May occur when something goes wrong during the parsing of an XML file with [XMLReader](#). Check the ErrorCode and Message properties of the [RuntimeException](#) class as well as the properties of this class for information about the error.

**Super Class** [RuntimeException](#)

### Properties

| Name                | Type                    | Description                                  |
|---------------------|-------------------------|--|
| CurrentByteCount    | <a href="#">Integer</a> | The number of bytes in the current event.    |
| CurrentByteIndex    | <a href="#">Integer</a> | The byte position where the error occurred.  |
| CurrentColumnNumber | <a href="#">Integer</a> | The column number where the error occurred.  |
| CurrentLineNumber   | <a href="#">Integer</a> | The line number at which the error occurred. |

**See Also** [RuntimeException](#), [XMLReader](#) classes.

## XMLStyleSheet Class

This is used for XML documents that represent XSLT stylesheets that will be applied to another XML document using the Transform method of the [XMLDocument](#) class.

**Super Class** [XMLDocument](#)

### Constructors

| Name          | Parameters                      | Description                               |
|---------------|---------------------------------|---|
| XMLStyleSheet |                                 | Creates an empty XSL document.            |
| XMLStyleSheet | XML as <a href="#">String</a>   | Parses an XML string into an XSL document |
| XMLStyleSheet | f as <a href="#">FolderItem</a> | Parses an XML file into the XSL document. |

### Methods

| Name    | Parameters   | Description  |
|---------|--|--|
| LoadXML | XML as <a href="#">String</a><br>or<br>f as <a href="#">FolderItem</a> | Parses an XML string or an XML file into the XSL document. |

---

## XMLTextNode Class

Represents an XML Text node.

**Super Class** [XMLNode](#)

**Properties** None

**Methods** None

**Notes** Use the **XMLTextNode** class with the [IsA](#) operator to test whether an [XMLNode](#) is an **XMLTextNode**.

**See Also** [XMLComment](#), [XMLNode](#) classes; [IsA](#) operator.

---

## XMLXslHandler Class

Used to handle SAX (Simple API for XML) style events on the XML output of an XSLT transformation.

## You Can Only Inherit From a Class Error

---

### Events

| Name                  | Parameters  | Description  |
|-----------------------|---|--|
| Characters            | Contents as <a href="#">String</a>  | Fires at character data.                           |
| Comment               | Contents as <a href="#">String</a>  | Fires at an XML comment.                           |
| EndDocument           |   | Fires at the end of the document.                  |
| EndElement            | Name as <a href="#">String</a>  | Fires at the end of an element.                    |
| EndNamespace          | Prefix as <a href="#">String</a>  | Fires at the end of a namespace declaration.       |
| ProcessingInstruction | Target as <a href="#">String</a> ,<br>Contents as <a href="#">String</a>              | Fires at an XML processing instruction.            |
| StartDocument         |   | Fires at the beginning of the document.            |
| StartElementName      | Name as <a href="#">String</a> ,<br>AttributeList as <a href="#">XmlAttributeList</a> | Fires at the beginning of an element.              |
| StartNamespace        | Prefix as <a href="#">String</a> ,<br>URI as <a href="#">String</a>                   | Fires at the beginning of a namespace declaration. |

### Notes

To use **XMLXslHandler**, create an instance of this class and put code in the events. Then pass an instance of your subclass as the second parameter of the `Transform` method of the [XMLDocument](#) class.

### See Also

[XMLDocument](#) class.

---

## You Can Only Inherit From a Class Error

Occurs only in [RBScript](#). It occurs if you set a class's Super class to an interface instead of a real class.

---

## You Can't Assign one Array to Another Array Error

You tried to assign one array to another array. To create an array that has the same elements of another array, you need to assign each value of one array to the same index value of the other array. Use a [For](#) statement and loop through all the values of the array index.

**Example**

```
Dim a(5) as Integer  
Dim b(5) as Integer  
a=b
```

**See Also**

[Dim](#) statement.

## You Can't Pass an Array to an External Function Error

You'll get this on the [Declare](#) statement if you define one of its parameters as an array.

**See Also**

[Declare](#) statement.

## You Can't Pass an Expression as a Parameter that is Defined as ByRef Error

In a function call, you tried to pass an expression rather than a variable or property to a parameter that was declared as [ByRef](#) in the called method.

**Example**

The following method takes one parameter that was declared as ByRef:

```
Sub Squarit(ByRef c as Double)  
c=c*c
```

The calling function is:

```
Dim a,b as Double  
a=5  
b=10  
Squarit(a*b) //cannot use an expression here
```

**See Also**

[ByRef](#) operator.

## You Can't Use Super Because this Class has no Super Class Error

You can't refer to the Super Class of a class that has no Super Class.

## You Can't Use the New Operator with this Class Error

You tried to create an instance from an abstract class. You can only use the [New](#) operator to create a new instance of a control already in a window, a window added to the project with the Project ▶ Add ▶ Window command or the Add Window button, or an existing menu item. The existing object acts as a template for the new instance.

You also cannot create an instance of a class that is used only as the base class for other classes, such as [RectControl](#) or [SocketCore](#).

### Example

Don't do this:

```
Dim w as Window  
w=New Window
```

Instead, use the [New](#) operator with an instance of the Window class created with the Project ▶ Add ▶ Window command or the Add Window button in the Project Editor.

```
Dim w as FindWindow  
w=New FindWindow
```

### See Also

[New](#) operator.

## You Can't Use This Variable or Property Name Because it is a Reserved Word Error

Reserved words are not available as names of user-defined objects. If you used a reserved word as a variable name in a [Dim](#) statement or as a Property Name, this message will appear.

### Examples

Using a reserved word, such as a data type (Integer, Double, Boolean, etc.) as the name of a property in the Add Property declaration area.

## You Cannot Implement a Nonexistent Event Error

Occurs only in [RBScript](#). You attempted to use an event that is not defined in REALbasic.

See Also    [RBScript](#) control.

---

## You Cannot Return a Value Because this Method has No Defined Return Type Error

You can't use the [Return](#) statement with a variable, expression, or value to be returned in a method unless you define a data type for the value being returned in the function declaration.

**Example**    No return data type defined in the method declaration:

```
Sub myMethod(a as Integer, b as Integer)
    Return a*b
```

To return a value, declare the data type of the returned value. In that case, the declaration statement in this example would be:

```
Function myMethod(a as Integer, b as Integer) as Integer
```

See Also    , [Function](#), [Return](#), [Sub](#) statements.

---

## You Cannot Use the New Operator with a Class Interface Error

You used the [New](#) operator with a class interface. You create new class interfaces using the Project ▶ Add ▶ Class Interface menu command or the Add Class Interface button in the Project Editor.

**Example**    The class interface Interface1 has already been added to the project.

```
Dim c as Interface1
c=New Interface1
```

---

## You Have Used an Operator that is not Compatible with the Data Types Specified Error

You tried to perform an operation on data types that don't support the operation. For example, multiplying strings, dividing [Booleans](#), etc.

## Examples

```
Dim c as Integer  
Dim d,e as String  
Dim g,h as Boolean  
d="Ted"  
e="Alice"  
c=d*e //error  
c=g/h //error
```

## You Must Indicate the Data Type of the Value to be Returned Error

You declared an external function using the Function keyword but did not specify the return type. This happens because the data type of the value returned was left off or because the programmer intended to declare a sub but used function instead.

**See Also** [Function](#), [Sub](#) statements.

## You Must Use the Value Returned by this Function Error

You called a function but did not assign the result to a variable or property.

**Note** If you want to call the function but ignore the result, use the [Call](#) statement.

**Examples** Trying to assign the result to an object rather than a property of the object:

```
CheckBox1=True
```

Ignoring the result returned by a built-in function:

```
Atan2(1,0) //not assigned to anything  
RGB(100,100,100) //incorrect
```

**See Also** [Call](#), [Function](#) statements.

# REAL SQL Database Language Reference

---

A REALbasic database front-end uses the Structured Query Language (SQL) to communicate with its data sources. The plug-in for your data source receives a SQL statement from REALbasic, translates the statement into a form that the data source understands (if necessary), and sends it to the data source. Both the standard and professional versions of REALbasic include a single-user SQL-based data source. The other data sources are fully supported only in the professional version of REALbasic.

The REALSELdatabase data source is supported by the subset of SQL described in this chapter. The plug-in for this data source is shipped as a integral part of the REALbasic application so there is no need to install a separate file in your plugins directory.

Except for 4D Server, the other supported data sources are native SQL back-ends. When you are working with any native SQL back-end, its REALbasic plug-in passes your SQL to the data source. Therefore any valid SQL for that data source will work in a REALbasic front-end for that source. Please refer to the documentation for your selected data source for further information on supported SQL and specifics on the data types supported by that data source. In the case of REALSELdatabase, it is based on SQLite; it is described at <http://www.sqlite.org>.

If you are unfamiliar with SQL, you will need to learn its basics before implementing your REALbasic front-end. This manual does not attempt to teach you SQL; rather, it describes the subset of SQL that is currently supported for the REALSEL database engine. Please consult one of the many good SQL references, such as *SQL for Dummies* by Allen G. Taylor (ISBN: 0-7645-0105-4), *The Practical SQL Handbook*, by Bowman, Emerson, and Darnovsky (ISBN: 0-2014-4787-8). You can also find several good

---

online SQL references by searching for “SQL command syntax” or for a specific SQL command using <http://www.google.com> or a comparable search engine.

## SQL in REALbasic

This section provides an overview of SQL for the REAL SQL Database. Here are the supported SQL statements.

| Statement                         | Description  |
|-----------------------------------|--|
| <a href="#">ALTER TABLE</a>       | Renames a table or adds a column to an existing table.   |
| <a href="#">ATTACH DATABASE</a>   | Attaches another database to the main database.  |
| <a href="#">BEGIN TRANSACTION</a> | Manually begins a transaction.   |
| <a href="#">comment</a>           | Denotes text as nonexecutable comments, treated as whitespace by the SQL parser.   |
| <a href="#">COMMIT</a>            | Commits changes (inserts, deletes, and updates) to the data and changes to the database schema (CREATE TABLE, CREATE INDEX, and so forth). |
| <a href="#">CREATE INDEX</a>      | Creates an index in a table.   |
| <a href="#">CREATE TABLE</a>      | Creates a table and specifies the fields and their attributes.   |
| <a href="#">CREATE TRIGGER</a>    | Adds database operations that are performed automatically when an event occurs.  |
| <a href="#">CREATE VIEW</a>       | Creates a prepackaged SELECT statement that can be used in the FROM clause of another SELECT statement in place of a table name.           |
| <a href="#">DELETE</a>            | Deletes a specified group of records.  |
| <a href="#">DETACH DATABASE</a>   | Detaches a database that was attached with an <a href="#">ATTACH DATABASE</a> statement.   |
| <a href="#">DROP INDEX</a>        | Removes the specified index from a table.  |
| <a href="#">DROP TABLE</a>        | Removes a table from the database.   |
| <a href="#">DROP TRIGGER</a>      | Removes a trigger that was added with a <a href="#">CREATE TRIGGER</a> statement.  |
| <a href="#">DROP VIEW</a>         | Removes a view that was added with a <a href="#">CREATE VIEW</a> statement.  |
| <a href="#">EXPLAIN</a>           | Non-standard modifier that requests the virtual machine instructions that would have been used to execute a command.                       |
| <a href="#">INSERT</a>            | Inserts a record into the specified table.   |
| <a href="#">ON CONFLICT</a>       | Non-standard clause that specifies what algorithm to use to resolve constraint conflicts.  |
| <a href="#">PRAGMA</a>            | Directive to modify the operation of the library or to query the library for internal (non-table) data.                                    |
| <a href="#">REINDEX</a>           | Deletes and then recreates indexes.  |
| <a href="#">REPLACE</a>           | An alias for the Insert or Replace variant of the <a href="#">INSERT</a> statement.  |

| <b>Statement</b>         | <b>Description</b>   |
|--------------------------|--|
| <a href="#">ROLLBACK</a> | Cancels changes (inserts, deletes, updates) to the data and changes to the database schema (CREATE TABLE, CREATE INDEX, and so forth).   |
| <a href="#">SELECT</a>   | Returns a group of rows, known in REALbasic as a recordset. In the language of SQL, this is also called a <i>cursor</i> . You can specify the columns (fields), the table(s), search conditions, grouping, and sort columns. |
| <a href="#">UPDATE</a>   | Updates existing records in a table.   |
| <a href="#">VACUUM</a>   | Cleans up empty space left in the database from other operations.  |

## Character Functions

The following functions compute values from the character or bit string passed to it.

| <b>Function</b>              | <b>Example</b>  | <b>Description</b>   |
|------------------------------|---|--|
| <a href="#">BIT_LENGTH</a>   | BIT_LENGTH('zz') returns 32   | Takes either a character string or a bit string and returns the number of bits.<br>BIT_LENGTH (string)   |
| <a href="#">CHAR_LENGTH</a>  | CHAR_LENGTH ('480') returns 3.  | Returns the number of characters in the string passed.<br>CHAR_LENGTH (string)   |
| <a href="#">LOWER</a>        | LOWER('GREENHORNET') returns "greenhornet".   | Takes a character string and returns the string in all lowercase.<br>LOWER (string)  |
| <a href="#">OCTET_LENGTH</a> | OCTET_LENGTH ('FF') returns 2.  | Returns the number of bytes in a character string or bit string expression.<br>OCTET_LENGTH (string)   |
| <a href="#">POSITION</a>     | POSITION ('G' IN 'Fred Gorman') returns 6.  | Searches for the specified target string within the source string and returns as an integer the character position where the target string starts. If it doesn't find the target string, it returns zero.<br>POSITION (targetString IN sourceString)   |
| <a href="#">SUBSTRING</a>    | SUBSTRING ('Samantha' FROM 1 FOR 3) returns 'Sam'.  | Returns the specified substring of characters from the sting passed.<br>SUBSTRING (string FROM start [FOR length])<br>Start is the starting position; if the optional FOR clause is omitted, SUBSTRING returns the characters from start to the end of the string. If FOR is included, SUBSTRING returns the length characters beginning at start. |
| <a href="#">TRIM</a>         | TRIM BOTH from ' The Matrix ' returns 'The Matrix'. Same as TRIM BOTH '' from ' The Matrix '. | Trims leading and/or trailing characters from the string passed. The default character to trim is the blank.<br>TRIM (LEADING .TRAILING BOTH [trimstring] FROM targetString)   |

## SQL Functions

| Function              | Example   | Description   |
|-----------------------|---|---|
| <a href="#">UPPER</a> | UPPER('Grand Blanc')<br>returns "GRAND<br>BLANC". | Takes a character string and returns the string in all uppercase.<br>UPPER (string) |

The following date, time, and ID functions are available:

| Function                          | Example               | Description  |
|-----------------------------------|-----------------------|--|
| <a href="#">CURRENT_DATE</a>      | CURRENT_DATE          | Returns the current date in SQL Date format, YYYY-MM-DD, as a SQL Date data type.<br>CURRENT DATE  |
| <a href="#">CURRENT_TIME</a>      | CURRENT_TIME          | Returns the current time in SQL Time, HH:MM:SS, as a SQL Time data type.<br>CURRENT TIME.  |
| <a href="#">CURRENT_TIMESTAMP</a> | CURRENT_TIMESTAMP     | Returns the current date-time in SQL TimeStamp format, YYYY-MM-DD HH:MM:SS, as a SQL TimeStamp data type.  |
| <a href="#">LAST_ROWID</a>        | LAST_ROWID ('Movies') | Returns the value of the last _rowID for the table passed. This value is useful for certain types of relational joins.<br>LAST_ROWID (tablename) |

## Aggregate Functions

The following functions are aggregate functions, sometimes known as set functions. They apply to sets of rows in a table and return the results of an arithmetic calculation. You determine the number of rows on which the calculation is based using a standard WHERE clause and include the Set function in a SELECT statement. You can use several Set functions in a SELECT statement and each calculates its value on the rows specified by the WHERE clause. However, you cannot use the DISTINCT keyword more than once. The SELECT statement returns a RecordSet. If the WHERE clause is omitted, the function is computed on all the rows in the table.

If you use the GROUP BY clause with a aggregate functions, the result will contain one row per group, as defined by the GROUP BY clause. The aggregate functions compute the summary statistics per group. With no GROUP BY clause, one row will be returned, with summary statistics for the all the records that were queried.

| Function            | Description  |
|---------------------|--|
| <a href="#">AVG</a> | Returns the numeric average of an expression. The expression must evaluate to a numeric data type. The expression can include one or more columns. The DISTINCT keyword can be used to limit the computation to distinct (unique) occurrences of each value.<br>Example: Finds the average of the Price column in the Products table, for Database products only.<br>SELECT AVG (Price) from Products WHERE Product='Database' |

| Function              | Description   |
|-----------------------|---|
| <a href="#">COUNT</a> | Returns the total number of rows accessed by the expression. The expression may include one or more columns. The DISTINCT keyword can be used to limit the evaluation to distinct (unique) occurrences of each value.<br>Example: Counts the number of Customer rows for the Zip Code '48070' only.<br>SELECT COUNT (*) from Customers WHERE Zip='48070'  |
| <a href="#">MAX</a>   | Returns the maximum value of the specified field or fields. The Max function operates on numeric, date, time, and character data types. The expression may include one or more columns.<br>Example: Finds the largest population in the Cities table.<br>SELECT MAX (Population) from Cities  |
| <a href="#">MIN</a>   | Returns the minimum value of the specified field or fields. The Min function operates on numeric, date, time, and character data types. The expression may include one or more columns.<br>Example: Finds the minimum Price in the Products table.<br>SELECT MIN (Price) from Products  |
| <a href="#">SUM</a>   | Returns the total of the specified fields. That is, the expression must evaluate to a numeric data type. The expression may include one or more columns. The DISTINCT keyword can be used to limit the evaluation to distinct (unique) occurrences of each value.<br>Example: Finds the total population from the States table for States in the NE region only.<br>SELECT SUM (Population) from States WHERE Region='NE' |

**Calling SQL Commands** You use the SELECT statement by passing it as a string to the Database class's SQLSelect method. It returns a RecordSet object that contains the records that were requested. You use the Database class's SQLExecute statement to execute all the other SQL statements. None of the other SQL statements return data.

The examples in this chapter assume that you've created an instance of the database class. The SQL example is passed as a string parameter to either the SQLExecute or SQLSelect methods.

The Syntax section gives the syntax of the SQL string, omitting the surrounding REALbasic code that's needed to pass the SQL to the data source.

## SQL Command Reference

This section is a language reference for the subset of SQL that is supported by the REALSQLdatabase data source.

## ALTER TABLE Statement

Renames a table or adds a new column to an existing table. Removing a column is not supported by REALSQLdatabase.

---

## Syntax

**ALTER TABLE *tablename* ADD [Column] *fieldname* *fieldtype***

**OR**

**ALTER TABLE *tablename* RENAME TO [*NewTableName*]**

| Part      | Description  |
|-----------|--|
| TableName | Name of the existing table.  |
| FieldName | Name of the field to add.  |
| FieldType | Data type. The REAL database supports the following data types: integer, varchar, double, boolean, date, time, timestamp, binary (blob), and string. If you are using another vendor's data source, see their documentation on supported data types. |

The keyword “Column” is assumed and can be omitted.

## Notes

Use the RENAME TO syntax to rename the table specified by

*DatabaseName.TableName* to *NewTableName*. The command cannot be used to move a table between attached databases. It only renames a table in the specified database. Any triggers or indexes in the table remain after the renaming. However, any view definitions or statements executed by triggers that refer to the table being renamed are not modified to refer to the new tablename. The triggers or view definitions must be dropped and recreated.

The ADD syntax adds a new column to the specified table. The new column is added to the end of the list of existing columns. The column definition may take any of the forms that are allowed in the [CREATE TABLE](#) statement, with the following restrictions:

- The new column may not have a Primary Key or Unique attribute,
- The column may not have a default value of CURRENT\_TIME, CURRENT\_DATE, or CURRENT\_TIMESTAMP.
- If NOT NULL is specified, then the column must have a default value other than Null.

---

**Example** The following example adds the column “VacationDays” to the table “Employees”.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile=Getfolderitem("myCompany")
db.DatabaseFile=dbFile
if db.connect then
    db.SQLExecute ("ALTER TABLE Employees ADD COLUMN VacationDays Integer")
    if db.error then
        MsgBox db.errormessage
    else
        db.SQLExecute("Commit") //no error, save change
    end if
else
    MsgBox "Connection to database failed."
end if
```

**See Also** [CREATE TABLE, DROP TABLE](#)

---

## ATTACH DATABASE

Adds a preexisting database file to the current database connection. Attached databases are removed with the DETACH DATABASE statement.

**Syntax** **ATTACH [DATABASE] *database-filename* AS *database-name***

**Notes** The names “main” and “temp” refer to the main database and the database used for temporary tables, respectively.

Tables in an attached database can be referred to with the syntax *database-name.table-name*. If an attached table doesn’t have a duplicate table name in the main database, it doesn’t require the *databasename* prefix. You cannot create a new table with the same name as a table in an attached database, but you can attach a database which contains tables whose names are duplicates of tables in the main database.

You can read from and write to an attached database and you can modify the schema of the attached database.

**See Also** [DETACH DATABASE](#)

---

# BEGIN TRANSACTION

Begins a transaction manually. The transaction usually persists until the COMMIT or ROLLBACK statement is encountered.

## Syntax

```
BEGIN [DEFERRED | IMMEDIATE | EXCLUSIVE] [TRANSACTION [NAME]]  
OR  
END [TRANSACTION[NAME]]  
OR  
COMMIT [TRANSACTION[NAME]]  
OR  
ROLLBACK [TRANSACTION[NAME]]
```

## Notes

No changes can be made to a database except within a transaction. Any statement that changes the database will automatically start a transaction if one is not already in effect. Automatically started transactions are committed at the conclusion of the command unless the ROLLBACK command is issued.

Transactions can be deferred, immediate, or exclusive. Deferred means that no locks are acquired on the database until the database is first accessed. With a deferred transaction, the BEGIN statement does nothing, as the locks are not obtained until the first read or write operation. The first read operation creates a shared lock and the first write operation creates a reserved lock.

Because the acquisition of locks is deferred until they are needed, it is possible that another thread or process could create a separate transaction and write to the database after the BEGIN on the current thread has executed. If the transaction is immediate, then reserved locks are acquired on all databases as soon as the BEGIN command is executed, without waiting for the database to be used. After a BEGIN IMMEDIATE, you are guaranteed that no other thread or process will be able to write to the database or do a BEGIN IMMEDIATE or BEGIN EXCLUSIVE. Other processes can continue to read from the database, however. An exclusive transaction causes EXCLUSIVE locks to be acquired on all databases. After a BEGIN EXCLUSIVE, you are guaranteed that no other thread or process will be able to read or write the database until the transaction is complete.

The default behavior is a Deferred transaction.

The Commit command does not actually do the Commit until all pending SQL commands finish. That is, if two or more SELECT statements are in the middle of processing when a Commit is issued, the commit will not actually take place until all of the SELECT statements finish.

## See Also

[COMMIT](#), [ROLLBACK](#), [SELECT](#).

---

## comment Keyword

Signifies a comment within SQL.

|               |   |
|---------------|---|
| <b>Syntax</b> | -- single line comment<br>OR<br>/* multiple line comment [*/] |
|---------------|---|

|              |   |
|--------------|---|
| <b>Notes</b> | Comments within SQL are treated as whitespace by REALSQLdatabase. They can begin anywhere whitespace can be found. Using the first syntax, the comment can extend only to the end of the line. With the second syntax the comment can extend onto multiple lines. If the terminating delimiter is omitted, the rest of the SQL is treated as a comment. |
|--------------|---|

---

## COMMIT Statement

Commits (saves) changes to a database.

|               |               |
|---------------|---------------|
| <b>Syntax</b> | <b>Commit</b> |
|---------------|---------------|

|              |  |
|--------------|--|
| <b>Notes</b> | The REALSQLdatabase supports transactions. A transaction is either a set of changes to the database schema (i.e., tables, fields, and field types) or a set of changes to the contents of the database—record additions, deletions, or modifications. A transaction begins automatically by the first such change and is saved to the database by calling Commit. You can instead abort the transaction by calling ROLLBACK. This restores the database to its state when the transaction was started.<br><br>You can also commit changes to the database by calling the Database class's Commit method. |
|--------------|--|

---

## Example

The following example uses Commit to save a change to a table. Some examples in this reference use the Commit method of the Database class to commit changes.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile=Getfolderitem("myCompany")
db.DatabaseFile=dbFile
if db.connect then
    db.SQLExecute ("ALTER TABLE Employees ADD COLUMN VacationDays Integer")
    if db.error then
        MsgBox db.errormessage
    else
        db.SQLExecute("Commit") //no error, save change
    end if
else
    MsgBox "Connection to database failed."
end if
```

## See Also

[ROLLBACK](#)

---

# CREATE INDEX Statement

Creates an index in a table.

## Syntax

**CREATE [UNIQUE] INDEX [databaseName.]indexname ON tablename (fieldlist)  
[ON CONFLICT conflict-algorithm]**

| Argument  | Description   |
|-----------|---|
| IndexName | The name of the new index.  |
| TableName | The name of the table that contains the fields being indexed.   |
| Fieldlist | The names of one or more fields on which to build the index. <b>FieldList</b> consists of one or more fields/data types, separated by commas. A Field-Name can be of the form: FieldName [COLLATE collation-name] [ASC DESC] The DESC specification is currently ignored. |

## Notes

The optional COLLATE clause following each column name defines a collating sequence used for text entries in that column. The default collating sequence is the collating sequence defined for that column in the CREATE TABLE statement. If no collating sequence is defined, the built-in BINARY collating sequence is used.

If the UNIQUE keyword appears between CREATE and INDEX then duplicate index entries are not allowed. Any attempt to insert a duplicate entry will result in an error.

The optional CONFLICT clause allows you to specify an alternative default constraint conflict resolution algorithm for the index. This only makes sense if the UNIQUE

---

keyword is used since that is the only supported constraint. The default algorithm is ABORT. If a COPY, INSERT, or UPDATE statement specifies a particular conflict resolution algorithm, that algorithm is used in place of the default algorithm specified here. See the section titled ON CONFLICT for additional information.

You index columns that you use for your most frequent searches and sorts, as well as fields used for relational joins. An index makes a search much faster because the database doesn't have to scan the table sequentially to find the record you are looking for.

**Example**

The following example creates an index called "NameIndex" in the table "Employees" using the "Name" field.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile = GetFolderItem("myCompany")
db.DatabaseFile=dbFile
If db.Connect then
    db.SQLExecute ("Create Index NameIndex on Employees (Name)")
    If db.error then
        MsgBox db.errormessage
    Else
        db.SQLExecute ("Commit")
    end if
else
    MsgBox "Database connection failed."
End if
```

**See Also**

[DROP INDEX](#)

---

# CREATE TABLE Statement

Adds a table to the database.

|        |   |
|--------|---|
| Syntax | <b>CREATE [TEMP   TEMPORARY] TABLE <i>tablename</i> (<i>column-def [,column-def [,constraint]}</i>)</b> |
|        | <b>OR</b>   |
|        | <b>CREATE [TEMP TEMPORARY] TABLE [<i>databasename.</i>]<i>tablename</i> AS select-statement</b>         |

| Part              | Description  |
|-------------------|--|
| TableName         | Name of the new table.   |
| Column-def        | Name [Type] [[CONSTRAINT name] Column-constraint].   |
| Type              | TypeName<br>TypeName (number)<br>TypeName (number, number)   |
| Column-Constraint | Optional. The following constraints may be used:<br>NOT NULL [conflict clause]  <br>PRIMARY KEY [sort order] [conflict-clause] [AUTOINCREMENT]  <br>UNIQUE [conflict-clause]  <br>CHECK (expr) [conflict-clause]  <br>DEFAULT value  <br>COLLATE collation-name<br>If Not Null is used, the database will require a value for that field in every row, i.e., the field may not be missing.<br>CHECK constraints are currently ignored. |
| Constraint        | PRIMARY KEY (column-list) [conflict-clause]  <br>UNIQUE (column-list) [conflict-clause]  <br>CHECK (expr) [conflict-clause]<br>CHECK constraints are currently ignored.  |
| Conflict-clause   | ON CONFLICT conflict-algorithm   |

## Notes

Although there are a lot of options for special cases, the CREATE TABLE statement is usually the CREATE TABLE keyword, followed by the list of columns, their data types, and one or more optional constraints. There is no limit on the amount of data in a row.

Each column definition is the name of the column followed by the data type affinity for that column. It is optionally followed by one or more column constraints. The data type affinity does not restrict the type of data may be inserted into that column. It only determines how values whose data type is ambiguous may be converted. The actual storage type is a property of the value, not the column in which it is inserted.

For more information on the concepts of data type affinity and manifest typing, see the section “” on page 826.

---

If the TEMP or TEMPORARY keyword is issued, then the table is visible only to the process that opened the database and it is automatically deleted when the database is closed. Any indexes created on a temporary table are also temporary. Temporary tables and indexes are stored in a separate file distinct from the main database file.

The UNIQUE constraint causes an index to be created on the specified columns. This index must contain unique keys. The COLLATE clause specifies what text collating function to use when comparing text entries for the column. The built-in BINARY collating function is used by default.

The DEFAULT constraint specifies a default value to use when doing an INSERT. The value may be NULL, a string constant, or a number. The default value may also be one of the keywords CURRENT\_TIME, CURRENT\_DATE or

CURRENT\_TIMESTAMP. If the value is NULL, a string constant, or number, it is literally inserted into the column whenever an INSERT statement that does not specify a value for the column is executed. If the default value is CURRENT\_TIME, CURRENT\_DATE or CURRENT\_TIMESTAMP, then the current UTC date and/or time is inserted into the columns. For CURRENT\_TIME, the format is HH:MM:SS. For CURRENT\_DATE, YYYY-MM-DD. The format for CURRENT\_TIMESTAMP is YYYY-MM-DD HH:MM:SS.

Specifying a PRIMARY KEY normally creates a UNIQUE index on the corresponding columns. However, if primary key is on a single column that has the data affinity INTEGER, then that column is used internally as the actual key of the B-Tree for the table. This means that the column may only hold unique integer values. This is the one case in which the REALSQLdatabase enforces strong data typing for the column. An integer column with the UNIQUE PRIMARY KEY. constraints must only contain unique integer values.

If a table does not have an INTEGER PRIMARY KEY column, then the B-Tree key will be a automatically generated integer. The B-Tree key for a row can always be accessed using one of the special names "ROWID", "OID", or "\_ROWID\_". This is true regardless of whether or not there is an INTEGER PRIMARY KEY. An INTEGER PRIMARY KEY column man also include the keyword AUTOINCREMENT. The AUTOINCREMENT keyword modified the way that B-Tree keys are automatically generated. Additional detail on automatic B-Tree key generation is available separately.

The optional conflict-clause following each constraint allows the specification of an alternative default constraint conflict resolution algorithm for that constraint. The default is abort ABORT. Different constraints within the same table may have different default conflict resolution algorithms. If an COPY, INSERT, or UPDATE command specifies a different conflict resolution algorithm, then that algorithm is used in place of the default algorithm specified in the CREATE TABLE statement. See the section titled ON CONFLICT for additional information.

CHECK constraints are ignored in the current implementation. Support for CHECK constraints may be added in the future. As of version 2.3.0, NOT NULL, PRIMARY KEY, and UNIQUE constraints all work.

---

There are no arbitrary limits on the number of columns or on the number of constraints in a table. The total amount of data in a single row is limited to about 1 megabytes in version 2.8. In version 3.0 there is no arbitrary limit on the amount of data in a row.

The CREATE TABLE AS form defines the table to be the result set of a query. The names of the table columns are the names of the columns in the result.

The Create Table statement creates a table structure on the current data source. You use the Database class's SQLExecute method to pass a Create Table statement to REALbasic.

## Example

The following code opens an existing database, "myDVDs", and adds a table with three fields. The underscore keyword is used to break the string passed to SQLExecute into two lines in the Code Editor. It is treated by REALbasic's compiler as one logical line.

```
Dim dbFile as FolderItem
Dim rs as New Recordset
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile = GetFolderItem("myDVDs")
db.DatabaseFile=dbFile
If db.Connect() then //check if the database was opened successfully
    db.SQLExecute ("Create Table Movies (Title Varchar,Year Integer, " _ 
        +"Director Varchar)" //"\_" used to break this into two lines
    If db.Error then
        MsgBox db.ErrorMessage
    else
        db.Commit
        MsgBox "Table created successfully."
    end if
else //Connect method failed.
    Beep
    MsgBox "The database couldn't be opened."
end if
```

## See Also

[ALTER TABLE](#), [DROP TABLE](#).

---

# CREATE TRIGGER Statement

Adds triggers to the database schema. Triggers are database actions that are performed automatically when a specified database event occurs.

## Syntax

**CREATE [TEMP | TEMPORARY] TRIGGER trigger-name [BEFORE | AFTER] data-base-event ON [databaseName.]tablename trigger-action**  
**OR**

---

**CREATE [TEMP | TEMPORARY] TRIGGER trigger-name INSTEAD OF database-event ON [databaseName.]view-name trigger-action**

| Part           | Description   |
|----------------|---|
| database-event | DELETE  <br>INSERT  <br>UPDATE  <br>UPDATE OF column-list   |
| trigger-action | [FOR EACH ROW   FOR EACH STATEMENT] [WHEN expression]<br>BEGIN<br>trigger-step;[trigger-step;]<br>END |
| trigger-step   | update-statement  <br>insert-statement  <br>delete-statement  <br>select-statement                    |

## Notes

Use the **CREATE TRIGGER** statement to add triggers to the database schema. Triggers are database operations (the trigger-action) that are automatically performed when a specified database event (the database-event) occurs.

You can fire a trigger whenever a DELETE, INSERT or UPDATE of a particular table occurs or whenever an UPDATE of one or more specified columns of a table are updated.

REALSQLdatabase supports only FOR EACH ROW triggers, not FOR EACH STATEMENT triggers. Hence explicitly specifying FOR EACH ROW is optional. FOR EACH ROW implies that the SQL statements specified as trigger-steps may be executed (depending on the WHEN clause) for each database row being inserted, updated, or deleted by the statement causing the trigger to fire.

Both the WHEN clause and the trigger-steps may access elements of the row being inserted, deleted or updated using references of the form *NEW.column-name* and *OLD.column-name*, where *column-name* is the name of a column from the table that the trigger is associated with. OLD and NEW references may only be used in triggers on trigger-events for which they are relevant, as follows:

| Keyword | Description                      |
|---------|----------------------------------|
| INSERT  | NEW references are valid         |
| UPDATE  | NEW and OLD references are valid |
| DELETE  | OLD references are valid         |

If a WHEN clause is used, the SQL statements specified as trigger-steps are only executed for rows for which the WHEN clause is True. If no WHEN clause is supplied, the SQL statements are executed for all rows.

The specified trigger-time determines when the trigger-steps will be executed relative to the insertion, modification or removal of the associated row.

---

An ON CONFLICT clause may be specified as part of an UPDATE or INSERT trigger-step. If an ON CONFLICT clause is specified as part of the statement causing the trigger to fire, then this conflict handling policy is used instead.

Triggers are automatically dropped when the table that they are associated with is dropped.

Triggers may be created on views as well as ordinary tables by specifying INSTEAD OF in the CREATE TRIGGER statement. If one or more ON INSERT, ON DELETE or ON UPDATE triggers are defined on a view, then it is not an error to execute an INSERT, DELETE or UPDATE statement on the view, respectively. Thereafter, executing an INSERT, DELETE, or UPDATE on the view causes the associated triggers to fire. The real tables underlying the view are not modified (except possibly explicitly, by a trigger program).

## Example

Assuming that customer records are stored in the Customers table, and that order records are stored in the Orders table, the following trigger ensures that all associated orders are redirected when a customer changes his or her address:

```
CREATE TRIGGER update_cust_address UPDATE OF address ON Customers
BEGIN
    UPDATE orders SET address = new.address WHERE customer_name = old.name;
END;
```

With this trigger installed, executing the statement:

```
UPDATE customers SET address = '1 Main St.' WHERE name = 'Jack Jones';
```

causes the following to be automatically executed:

```
UPDATE orders SET address = '1 Main St.' WHERE customer_name = 'Jack Jones';
```

Triggers may behave oddly when created on tables with INTEGER PRIMARY KEY fields. If a BEFORE trigger program modifies the INTEGER PRIMARY KEY field of a row that will be subsequently updated by the statement that causes the trigger to fire, then the update may not occur. The workaround is to declare the table with a PRIMARY KEY column instead of an INTEGER PRIMARY KEY column.

A special SQL function RAISE() may be used within a trigger-program, with the following syntax

```
RAISE (ABORT, error-message) |
RAISE (FAIL, error-message) |
RAISE (ROLLBACK, error-message) |
RAISE (IGNORE)
```

When one of the first three forms is called during trigger-program execution, the specified ON CONFLICT processing is performed (either ABORT, FAIL or

---

ROLLBACK) and the current query terminates. An error code of SQLITE\_CONSTRAINT is returned to the user, along with the specified error message.

When RAISEIGNORE) is called, the remainder of the current trigger program, the statement that caused the trigger program to execute and any subsequent trigger programs that would have been executed are abandoned. No database changes are rolled back. If the statement that caused the trigger program to execute is itself part of a trigger program, then that trigger program resumes execution at the beginning of the next step.

Triggers are removed using the DROP TRIGGER statement. Non-temporary triggers cannot be added on a table in an attached database.

**See Also** [DROP TRIGGER](#)

---

## CREATE VIEW Statement

Assigns a name to a prepackaged SELECT statement. Once the view is created, it can be used in the FROM clause of another SELECT statement in place of a table name.

**Syntax** **CREATE [TEMP | TEMPORARY] VIEW [databasename.]viewname AS select-statement**

**Notes** If the TEMP or TEMPORARY keyword is used, the table that is created is only visible to the process that opened the database and is automatically deleted when the database is closed.

If a database-name is specified, then the view is created in the named database. It is an error to specify both a databasename and the TEMP keyword, unless the database-name is Temp. If no database name is specified, and the TEMP keyword is not present, the table is created in the main database.

You cannot COPY, DELETE, INSERT or UPDATE a view. Views are read-only. However, in many cases you can use a TRIGGER on the view to accomplish the same thing. Views are removed with the DROP VIEW command. Non-temporary views cannot be created on tables in an attached database.

**See Also** [DROP VIEW](#), [SELECT](#).

---

## DELETE Statement

Deletes the records specified in its WHERE clause.

---

**Syntax**

**DELETE FROM [database-name.]tablename WHERE searchcondition**

| Part            | Description  |
|-----------------|--|
| database-name   | Optional. Name of the database containing the table from which you are deleting records. |
| tablename       | Name of the table from which you are removing records.                                   |
| searchcondition | Boolean expression that identifies the row or rows to be updated.                        |

**Notes**

The DELETE command removes the records that satisfy the searchcondition. If no WHERE clause is included, all the rows in the specified table are removed.

**Example**

The following example deletes the record in the “Movies” table that has the title “Star Wars”.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile=New FolderItem("myMovies")
db.databaseFile=dbFile
If db.error then
    MsgBox db.errormessage
else
    db.SQLExecute ("Delete From Movies Where Title='Star Wars'")
    If db.error then
        MsgBox db.ErrorMessage
    else
        db.Commit //save deletion
        MsgBox "Record deleted successfully."
    end if
end if
```

**See Also**

[INSERT](#), [UPDATE](#).

---

## DETACH DATABASE Statement

Detaches an attached database that was previously attached using the ATTACH DATABASE statement.

**Syntax**

**DETACH [DATABASE] database-name**

| Part          | Description                              |
|---------------|--|
| database-name | The name of the database to be detached. |

---

**Notes** It is possible to attach the same database multiple times using different names. Detaching one connection will leave the others intact.

**See Also** [ATTACH DATABASE](#) statement.

---

## DROP INDEX Statement

Removes an index from a database.

**Syntax** **DROP INDEX** *[databasename.]indexname*

| Argument     | Description   |
|--------------|---|
| databasename | The name of database that contains the index to be dropped. |
| indexname    | The name of the index to be dropped.                        |

**Notes** DROP INDEX removes an index that was added via the CREATE INDEX statement. The index is completely removed and can only be restored by recreating it with another CREATE INDEX statement.

DROP INDEX does not reduce the size of the database file. Empty space is retained for later INSERTs. To remove free space, use the [VACUUM](#) statement.

**Example** The following example drops the index called “NameIndex” in the table “Employees” using the “Name” field.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile = GetFolderItem("myCompany")
db.DatabaseFile=dbFile
If db.Connect then
    db.SQLExecute ("Drop Index Employees.NameIndex")
    If db.error then
        MsgBox db.errormessage
    Else
        db.SQLExecute ("Commit")
    end if
else
    MsgBox "Database connection failed."
End if
```

**See Also** [CREATE INDEX](#)

---

## DROP TABLE Statement

Removes a table from the database.

### Syntax

#### **DROP TABLE [databasename.]tablename**

| Argument     | Description  |
|--------------|--|
| databasename | Optional. Database containing the table to be dropped. |
| tablename    | Name of table to be dropped from the database.         |

### Notes

DROP TABLE removes a table that was added via the CREATE TABLE statement. It is completely removed from the database file and cannot be recovered. All indexes are also removed.

The DROP TABLE statement does not reduce the size of the database file. Free space is retained for later INSERTs. To remove the free space in the database, use the VACUUM statement.

### Example

The following example drops the table named "Employees" from the database.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile=Getfolderitem("myCompany")
db.DatabaseFile=dbFile
If db.connect then
    db.SQLExecute ("Drop Table Employees")
    If db.error then
        MsgBox db.errormessage
    Else
        db.Commit //save changes to the schema
        MsgBox "Table dropped successfully."
    end if
Else
    MsgBox "Connection to database failed."
End if
```

### See Also

[CREATE TABLE](#), [ALTER TABLE](#).

---

## DROP TRIGGER Statement

Drops a trigger that was created with the CREATE TRIGGER statement.

---

| Syntax        | DROP TRIGGER [database-name.]trigger-name                 |
|---------------|---|
| Part          | Description   |
| database-name | Optional. Name of the database that contains the trigger. |
| trigger-name  | Name of the trigger to be dropped.                        |

**Notes** The specified trigger is deleted from the database schema. Triggers are also automatically dropped if the associated table is dropped.

**See Also** [CREATE TRIGGER](#) statement.

---

## DROP VIEW Statement

Drops a view that was created with the CREATE VIEW statement.

**Syntax** **DROP VIEW** *view-name*

| Part      | Description                     |
|-----------|---------------------------------|
| view-name | Name of the view to be dropped. |

**Notes** The specified view is removed from the database schema, but this does not affect the data in the tables. Non-temporary views in attached databases cannot be dropped.

**See Also** [CREATE VIEW](#) statement.

---

## EXPLAIN Modifier

Non-standard command modifier used to request information about a SQL statement.

**Syntax** **EXPLAIN** *SQL-statement*

| Part          | Description   |
|---------------|---|
| SQL-statement | The SQL statement about which information is being requested. |

**Notes** If the EXPLAIN keyword appears before any SQL statement then REALSQLdatabase will report back the sequence of virtual machine instructions it would have used to execute the command if the EXPLAIN keyword had not been present.

---

# INSERT Statement

Inserts a record into a table.

## Syntax

**INSERT [OR conflict-algorithm] INTO [database-name.]tablename [(column-list)] VALUES (value-list)**  
**OR**  
**INSERT [OR conflict-algorithm] INTO [database-name.]tablename [(column-list)] select-statement**

| Part               | Description  |
|--------------------|--|
| conflict-algorithm | Optional. Alternate constraint conflict resolution algorithm.  |
| database-name      | Optional. Name of the database containing the table to be inserted.  |
| tablename          | Name of the table into which the data will be inserted.  |
| column-list        | Optional. Names of the columns in tablename that will contain the values being inserted. The number of columns must match the number of values being inserted (first format) or the number of columns returned by the SELECT statement.<br>If the field list is omitted, values for all fields should be provided in order or default or NULL values will be inserted. If you supply the field list, then the list of values match the specified fields, i.e., value1 is the value for field1, value2 is for field2, and so forth. |
| value-list         | Values for each field in the row being added. Columns of the table that do not appear in the column list are filled with the default value for the column or Null if there is no default value.  |
| select-statement   | SELECT statement that returns the values that will be inserted. It must return the same number of columns as are specified by the column-list. A new INSERT will be done for every row returned by the SELECT statement. The SELECT statement can be simple or compound. If an ORDER BY clause is used, it is ignored.   |

## Notes

By default, auto-commit is off in REALSQLdatabase, so you need to call COMMIT to commit your inserts.

---

## Example

The following example inserts four records into the “Employees” table. The underscore character is used to break up long logical lines into separate lines as displayed in the Code Editor.

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile=New FolderItem("myCompany")
db.databaseFile=dbFile
If db.error then
    MsgBox db.errormessage
else
    db.SQLExecute ("Insert into Employees
(Name,Department,Salary,VacationDays)_
+ "Values ('Seymore','Accounting',5000,0)")
    db.SQLExecute ("Insert into Employees
(Name,Department,Salary,VacationDays)_
+ "Values ('Throckmorton','Marketing',4400,0)")
    db.SQLExecute ("Insert into Employees
(Name,Department,Salary,VacationDays)_
+ "Values ('Dilbert','Engineering',6400,10)")
    db.SQLExecute ("Insert into Employees
(Name,Department,Salary,VacationDays)_
+ "Values ('Winnie','Sales',3500,0)")
If db.error then
    MsgBox db.errormessage
Else
    db.Commit //save new records to database
    MsgBox "Records added successfully."
End if
rs=db.sqlSelect ("Select * from Employees") //select all Employee records
If db.error then
    MsgBox db.errormessage
Else
    //proceed with database operations here
End if
End if
```

---

## See Also

[DELETE](#), [COMMIT](#), [ROLLBACK](#).

---

## ON CONFLICT Clause

A non-standard SQL command that specifies an alternate algorithm that is used to resolve constraint conflicts. The OR clause is a synonym for ON CONFLICT.

---

**Syntax****ON CONFLICT *conflict-algorithm***

| Part               | Description   |
|--------------------|---|
| conflict-algorithm | Algorithm for resolving constraint conflicts. The choices are:<br>ROLLBACK<br>ABORT<br>FAIL<br>IGNORE<br>REPLACE<br>The default is ABORT. |

**Notes**

When used in the COPY, INSERT, and UPDATE commands it appears as the keyword OR instead of ON CONFLICT to make the syntax seem more natural. The algorithm specified in the OR clause of a COPY, INSERT, or UPDATE overrides any algorithm specified in a CREATE TABLE or CREATE INDEX. The five alternate algorithms have the following meanings.

| Algorithm | Description   |
|-----------|---|
| ABORT     | The command backs out any prior changes it might have made and aborts with a return code of SQLITE_CONSTRAINT. But no ROLLBACK is executed, so changes from prior commands within the same transaction are preserved. This is the default behavior.   |
| FAIL      | The command aborts with a return code SQLITE_CONSTRAINT. Any changes to the database that the command made prior to encountering the constraint violation are preserved and are not backed out. For example, if an UPDATE statement encountered a constraint violation on the 100th row that it attempts to update, then the first 99 row changes are preserved but changes to rows 100 and beyond never occur.   |
| IGNORE    | The one row that contains the constraint violation is not inserted or changed. However, the command continues executing normally. Other rows before and after the row that contained the constraint violation continue to be inserted or updated normally. No error is returned.  |
| REPLACE   | When a UNIQUE constraint violation occurs, the pre-existing rows that are causing the constraint violation are removed prior to inserting or updating the current row. Thus the insert or update always occurs. The command continues executing normally. No error is returned. If a NOT NULL constraint violation occurs, the NULL value is replaced by the default value for that column. If the column has no default value, then the ABORT algorithm is used. |
| ROLLBACK  | An immediate ROLLBACK occurs, thus ending the current transaction, and the command aborts with a return code of SQLITE_CONSTRAINT. If no transaction is active (other than the implied transaction that is created on every command) then this algorithm is the same as ABORT.  |

# PRAGMA Statement

Used to modify the operation of the REALSQLdatabase library or query the library.

**Syntax**    **PRAGMA name [=value]**

**OR**

**PRAGMA function(arg)**

| Part     | Description   |
|----------|---|
| name     | Name of the pragma being called.  |
| value    | Value to be assigned to the pragma. Pragmas that take an integer value also take pragma constants. The constants "On", "True", and "Yes" are equivalent to 1. The constants "Off", "False", and "No" are equivalent to 0. The strings are not case-sensitive and do not require quotes. |
| function | The name of a function.   |
| arg      | The value of the argument passed to the function.   |

Pragmas to modify library operation:

| Name          | Syntax                                   | Description   |
|---------------|--|---|
| auto-vacuum   | auto_vacuum<br>auto_vacuum=0 1           | Queries or sets the auto-vacuum flag in the database. When the auto_vacuum flag is set, the database file shrinks when a transaction that deletes data is committed. It is only possible to modify the value of the auto-vacuum flag before any tables have been created in the database. No error message is returned if an attempt to set the auto-vacuum flag is made after a table has been created.  |
| cache-size    | cache_size<br>cache_size=number of pages | Queries or changes the maximum number of database disk pages that REALSQLdatabase will hold in memory at once. Each page uses about 1.5K of memory. The default cache size is 2000. If you are doing UPDATEs or DELETEs that change many rows, you may find that increasing this value may improve performance.   |
| count-changes | count_changes<br>count_changes=0 1       | Queries or changes the count-changes flag. Normally, when the count-changes flag is not set, INSERT, UPDATE and DELETE statements return no data. When count-changes is set, each of these commands returns a single row of data consisting of one integer value - the number of rows inserted, modified or deleted by the command. The returned change count does not include any insertions, modifications or deletions performed by triggers |

| Name                   | Syntax   | Description  |
|------------------------|--|--|
| default-cache-size     | default_cache_size<br>default_cache_size = number of pages | Queries or changes the maximum number of database disk pages that REALSQLdatabase will hold in memory at once. Each page uses 1K on disk and about 1.5K in memory. This pragma works like the cache_size pragma with the additional feature that it changes the cache size persistently. With this pragma, you can set the cache size once and that setting is retained and reused every time you reopen the database.   |
| empty_result_callbacks | empty_result_callbacks<br>empty_result_callbacks=0 1       | Queries or changes the empty-result-callbacks flag. Normally, when the empty-result-callbacks flag is cleared, the callback function supplied to the sqlite3_exec() call is not invoked for commands that return zero rows of data. When empty-result-callbacks is set in this situation, the callback function is invoked exactly once, with the third parameter set to 0 (NULL). This is to enable programs that use the sqlite3_exec() API to retrieve column-names even when a query returns no data.  |
| encoding               | encoding<br>encoding=value                                 | Gets or sets the encoding. The default encoding is UTF-8. Using the second form, you can set the value to UTF-8, UTF-16, UTF-16le (little endian), UTF-16be (big endian). The second form is useful only if the main database has not already been created. When you use it, it specifies the encoding that will be used if the database is created in the same session. Attached databases use the same encoding as the main database.  |
| full-column-names      | full_column_names<br>full_column_names=0 1                 | Queries or changes the full-column-names flag. This flag affects the way REALSQLdatabase names columns returned by SELECT statements when the expression for the column is a table.column name or the "*" wildcard. Normally, such result columns are named <table-name/alias><column-name> if the SELECT statement joins two or more tables together, or simply <column-name> if the SELECT statement queries a single table. When the full-column-names flag is set, such columns are always named <table-name/alias> .<column-name> regardless of whether or not a join is performed. |
| page-size              | page_size<br>page_size=bytes                               | Queries or sets the page-size of the database. This can be set only before the database is created. The page size must be a power of 2 greater than or equal to 512 and less than or equal to 8192.  |

| Name               | Syntax   | Description   |
|--------------------|--|---|
| short-column-names | short_column_names<br>short_column_names=0<br>11 | Queries or sets the short-column-names flag. This flag affects the way REALSQLdatabase names columns of data returned by SELECT statements when the expression for the column is a table-column name or the "*" wildcard. Normally, such result columns are named <table-name/alias>.<column-name> if the SELECT statement joins two or more tables together, or simply <column-name> if the SELECT statement queries a single table. When the short-column-names flag is set, such columns are always named <column-name> regardless of whether or not a join is performed.  |
| synchronous        | synchronous<br>synchronous=value                 | Queries or sets the synchronous flag. Value can be assigned OFF (0), NORMAL (1), ore FULL (2). When synchronous is FULL (2), the REALSQLdatabase database engine will pause at critical moments to make sure that data has actually been written to the disk surface before continuing. This ensures that if the operating system crashes or if there is a power failure, the database will be uncorrupted after rebooting. FULL synchronous is very safe, but it is also slow. When synchronous is NORMAL (1, the default), the REALSQLdatabase database engine will pause at the most critical moments, but less often than in FULL mode. There is a very small chance that a power failure at just the wrong time could corrupt the database in NORMAL mode. But in practice, you are more likely to suffer a catastrophic disk failure or some other unrecoverable hardware fault. With synchronous OFF (0), REALSQLdatabase continues without pausing as soon as it has handed data off to the operating system. If the application crashes, the data will be safe, but the database might become corrupted if the operating system crashes or the computer loses power before that data has been written to the disk. On the other hand, some operations are as much as 50 or more times faster with synchronous OFF. |

| Name                 | Syntax   | Description  |
|----------------------|--|--|
| temp-store           | temp_store<br>temp_store=value                             | Queries or sets the temp-store parameter. Value can be DEFAULT (0), FILE (1), or MEMORY (2). When temp_store is DEFAULT (0), the compile-time C preprocessor macro TEMP_STORE is used to determine where temporary tables and indices are stored. When temp_store is MEMORY (2) temporary tables and indices are kept in memory. When temp_store is FILE (1) temporary tables and indices are stored in a file. The temp_store_directory pragma can be used to specify the directory containing this file. FILE is specified. When the temp_store setting is changed, all existing temporary tables, indices, triggers, and views are immediately deleted. |
| temp-store-directory | temp_store_directory<br>temp_store_directory=directoryName | Queries or sets the temp-store-directory. This is the directory where files used for storing temporary tables and indexes are located. The new setting lasts only for the current connection and resets to its default value for each new connection opened. When this directory is changed, all existing temporary tables, indexes, triggers, and viewers are immediately deleted. If you set this pragma, it should be done just after the database is opened.   |

Pragmas to query the database schema:

| Name             | Syntax                       | Description  |
|------------------|------------------------------|--|
| database-list    | database_list                | Invoke the callback function once with information about the database. Arguments include the index and the name the database was attached with. The first row will be for the main database. The second row will be for the database used to store temporary tables. |
| foreign-key-list | foreign_key_list(table-name) | For each foreign key that references a column in the argument table, invoke the callback function with information about that foreign key. The callback function will be invoked once for each column in each foreign key.   |
| index-info       | index_info(index-name)       | For each column that the named index references, invoke the callback function once with information about that column, including the column name, and the column number.   |
| index-list       | index_list(table-name)       | For each index on the named table, invoke the callback function once with information about that index. Arguments include the index name and a flag to indicate whether or not the index must be unique.   |

| Name       | Syntax                 | Description  |
|------------|------------------------|--|
| table-info | table_info(table-name) | For each column in the named table, invoke the callback function once with information about that column, including the column name, data type, whether or not the column can be NULL, and the default value for the column. |

Pragmas to query or modify version values:

| Name           | Syntax                                   | Description   |
|----------------|--|---|
| schema-version | schema_version<br>schema_version=integer | Gets or sets the value of the schema-version. It is a 32-bit signed integer stored in the database header. This is usually manipulated only by REALSQLdatabase. It is incremented whenever the database schema is modified. Interfering with this process is not recommended. |
| user-version   | user_version<br>user_version=integer     | Gets or sets the use-version. It is also a 32-bit signed integer stored in the database header. This is not used internally, so it may be used for any purpose.   |

Pragmas to debug the library:

| Name            | Syntax               | Description  |
|-----------------|----------------------|--|
| integrity-check | integrity_check      | Does an integrity check of the entire database. It looks for out-of-order records, missing pages, malformed records, and corrupt indexes. If any problems are found, then a string is returned which is a description of all problems. If everything is in order, "ok" is returned.  |
| parser-trace    | parser_trace=0   1   | Turns tracing of the SQL parser on or off. This is used for debugging. The constants ON and OFF can be used instead of 0 and 1.  |
| vdbe-trace      | vdbe_trace= 0   1    | Turns tracing of the virtual database on or off. This is used for debugging. The constants ON and OFF can be used instead of 0 and 1.  |
| vdbe-listing    | vdbe_listing= -0   1 | Turns listing of the virtual machine programs on or off. When listing is on, the entire content of a program is printed just prior to beginning execution. This is like automatically executing an <a href="#">EXPLAIN</a> prior to each statement. The statement executes normally after the listing is printed. This is used for debugging. The constants ON and OFF can be used instead of 0 and 1. |

---

# REINDEX Statement

Deletes and recreates indexes from scratch.

**Syntax**    **REINDEX *collation-name***

OR

**REINDEX [*database-name*.]table/index-name**

| Part                    | Description  |
|-------------------------|--|
| <i>collation-name</i>   | Name of a collation sequence.                      |
| <i>database-name</i>    | The database in which the reindexing will be done. |
| <i>table/index-name</i> | Name of the table or index to be recreated.        |

**Notes**

The REINDEX command is used primarily when the definition of a collation sequence had changed and the current index is no longer in sync.

In the first form, all indexes in all attached databases that use the named collation sequence are recreated. In the second form, if [*database-name*.]table/index-name identifies a table, then all indices associated with the table are rebuilt. If an index is identified, then only this specific index is deleted and recreated.

If no database-name is specified and there exists both a table or index and a collation sequence of the specified name, then indices associated with the collation sequence only are reconstructed. This ambiguity may be dispelled by always specifying a database-name when reindexing a specific table or index.

**See Also**    [CREATE INDEX](#)

---

# REPLACE Statement

An alias for the INSERT OR REPLACE version of the [INSERT](#) command. This alias is provided for compatibility with [mySQL](#).

---

**Syntax**

**REPLACE INTO [database-name.]table-name [(column-list)] VALUES value-list**  
OR  
**REPLACE INTO [database-name.]table-name [(column-list)] select-statement**

| Part             | Description  |
|------------------|--|
| database-name    | Optional. Name of the database containing the table to be inserted.  |
| table-name       | Name of the table into which the data will be inserted.  |
| column-list      | Optional. Names of the columns in tablename that will contain the values being inserted. The number of columns must match the number of values being inserted (first format) or the number of columns returned by the SELECT statement.<br>If the field list is omitted, values for all fields should be provided in order or default or NULL values will be inserted. If you supply the field list, then the list of values match the specified fields, i.e., value1 is the value for field1, value2 is for field2, and so forth. |
| value-list       | Values for each field in the row being added. Columns of the table that do not appear in the column list are filled with the default value for the column or Null if there is no default value.  |
| select-statement | SELECT statement that returns the values that will be inserted. It must return the same number of columns as are specified by the column-list. A new INSERT will be done for every row returned by the SELECT statement. The SELECT statement can be simple or compound. If an ORDER BY clause is used, it is ignored.   |

**See Also** [INSERT Statement](#).

---

## ROLLBACK Statement

Restores the database to its original state prior to the start of a transaction.

**Syntax** **ROLLBACK**

**Notes** The REALSQLdatabase supports transactions. A transaction is either a set of changes to the database schema (i.e., tables, fields, and field types) or a set of changes to the contents of the database—record additions, deletions, or modifications performed by the INSERT, DELETE, or UPDATE statements. A transaction begins automatically with the first such change and is saved to the database by calling Commit. You can instead abort the transaction by calling ROLLBACK. This restores the database to the state it was when the transaction was started.

You can also rollback changes to the database by calling the Database class's Rollback method.

**See Also** [COMMIT](#).

# SELECT Statement

Use the SQL Select statement to query the database. You pass a valid SQL Select statement as a string to the SQLSelect method of the Database class. It selects and optionally groups and sorts records in one or more tables.

## Syntax

```
result=SELECT [ALL | DISTINCT] result FROM table-list  
WHERE expr  
GROUP BY expr-list  
HAVING expr  
[compound-op select]  
ORDER BY sort-expr-list  
[LIMIT integer [(OFFSET [, integer])]
```

| Part           | Description  |
|----------------|--|
| ALLDISTINCT    | Optional keyword. Distinct retrieves only unique values, i.e., it retrieves only unique rows. For example 'DISTINCT Cities' retrieves only one record per city even if there are several records per city. The ALL keyword retrieves all records, even if the values are not unique. 'ALL' is assumed unless DISTINCT is used.                     |
| result         | One or more result-columns that specify the columns to be returned. The resulting group of records is returned as a REALbasic RecordSet object. The SQL SELECT statement must be passed to the Database class's SQLSelect method rather than SQLExecute.   |
| result-column  | Specifies a column of data to be returned of the form,<br>* I tablename.* I expr [{AS] string}<br>If result-column is equal to "*" then all columns of all tables are returned. If the expression is the name of a table followed by "./*" then all of the columns in that table are returned. Otherwise it is the list of columns to be returned. |
| table          | table-name [AS alias] I (select) [AS alias] The syntax for a table allows you to specify an alias for the tablename that is used later in the select statement, i.e., "ACTORS AS a"  |
| table-list     | table [join-op table join-args]<br>List of tables separated by commas. If a tablename has spaces in it, it must be surrounded by brackets, e.g., [Product Groups]  |
| join-op        | The permissible expressions for the join operator are:<br>, I [NATURAL] [LEFT I RIGHT] I FULL] [OUTER I INNER I CROSS] JOIN  |
| join-args      | [ON expr] [USING (id-list)]  |
| sort-expr-list | expr [sort-order] [,expr[sort-order]]  |
| sort-order     | [COLLATE collation-name] [ASC I DESC]  |
| compound_op    | UNION I UNION ALL I INTERSECT I EXCEPT   |

| <b>Notes</b>                                      | Entire books have been written about the SQL SELECT statement, especially on the topic of relational database operations.   |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
|---|---|-------------------------|--------------------|--|-------------|--|-----------|---|----------|---|-------------------------|--------------|------|---|-------------------|--------------|------|---|-----------|------------|------|---|-----------|-----------------|------|--------------|-------------|--------------|---|---------------|---|
| <b>Wildcard Searches</b>                          | The REAL database supports the two SQL wildcard characters, percent, '%', and the underscore character, '_'. The percent wildcard stands for any number of characters and the underscore stands for any single character. Placing the % at the beginning or the end of the search string does 'ends with' and 'begins with' searches, respectively. Using the % on both sides of the search string specifies a 'contains' search. For example,  |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
|   | <table border="1"> <thead> <tr> <th><b>SELECT statement</b></th><th><b>Search Type</b></th></tr> </thead> <tbody> <tr> <td>SELECT * from authors where au_lname LIKE "Jon%"</td><td>Begins with</td></tr> <tr> <td>SELECT * from authors where au_lname LIKE "%son"</td><td>Ends with</td></tr> <tr> <td>SELECT * from authors where au_lname LIKE "%man%"</td><td>Contains</td></tr> </tbody> </table>   | <b>SELECT statement</b> | <b>Search Type</b> | SELECT * from authors where au_lname LIKE "Jon%" | Begins with | SELECT * from authors where au_lname LIKE "%son" | Ends with | SELECT * from authors where au_lname LIKE "%man%" | Contains |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| <b>SELECT statement</b>                           | <b>Search Type</b>  |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| SELECT * from authors where au_lname LIKE "Jon%"  | Begins with   |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| SELECT * from authors where au_lname LIKE "%son"  | Ends with   |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| SELECT * from authors where au_lname LIKE "%man%" | Contains  |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
|   | <p>The search criterion is case-sensitive.</p> <p>The underscore character is the wildcard character for any single character. For example,</p> <pre>SELECT * from authors where au_lname LIKE "B_r"</pre> <p>Returns "Bar", "Bor", etc.</p>  |                         |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| <b>SELECT Examples</b>                            | <p>You do relational operations (e.g., "joins") by specifying the tables to be joined in the SELECT statement's <b>tablelist</b> and indicating in the WHERE clause how the tables are to be joined, i.e., rows in the 'many' table whose foreign key matches the primary key in the 'one' table. Please refer to a SQL reference book for detailed information on relational operations.</p> <p>A join retrieves data from two or more tables. A join usually uses a WHERE clause that specifies the rows to be included in the recordset. The WHERE clause usually refers to primary and/or foreign key fields in the tables being joined. The most common type of join retrieves one or more rows in one table that are logically linked to each row in a different table.</p> <p>Consider the following simple tables, Movies and Actors</p> <p><b>Movies</b></p> <table border="1"> <thead> <tr> <th><b>RowID</b></th><th><b>Title</b></th><th><b>Director</b></th><th><b>Year</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>Star Wars</td><td>George Lucas</td><td>1977</td></tr> <tr> <td>2</td><td>Raiders of the Lost Ark</td><td>George Lucas</td><td>1981</td></tr> <tr> <td>3</td><td>American Graffiti</td><td>George Lucas</td><td>1973</td></tr> <tr> <td>4</td><td>Apollo 13</td><td>Ron Howard</td><td>1995</td></tr> <tr> <td>5</td><td>Cast Away</td><td>Robert Zemeckis</td><td>2000</td></tr> </tbody> </table> <p><b>Actors</b></p> <table border="1"> <thead> <tr> <th><b>RowID</b></th><th><b>Name</b></th><th><b>Movie</b></th></tr> </thead> <tbody> <tr> <td>1</td><td>Harrison Ford</td><td>1</td></tr> </tbody> </table> | <b>RowID</b>            | <b>Title</b>       | <b>Director</b>                                  | <b>Year</b> | 1  | Star Wars | George Lucas                                      | 1977     | 2 | Raiders of the Lost Ark | George Lucas | 1981 | 3 | American Graffiti | George Lucas | 1973 | 4 | Apollo 13 | Ron Howard | 1995 | 5 | Cast Away | Robert Zemeckis | 2000 | <b>RowID</b> | <b>Name</b> | <b>Movie</b> | 1 | Harrison Ford | 1 |
| <b>RowID</b>                                      | <b>Title</b>  | <b>Director</b>         | <b>Year</b>        |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| 1   | Star Wars   | George Lucas            | 1977               |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| 2   | Raiders of the Lost Ark   | George Lucas            | 1981               |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| 3   | American Graffiti   | George Lucas            | 1973               |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| 4   | Apollo 13   | Ron Howard              | 1995               |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| 5   | Cast Away   | Robert Zemeckis         | 2000               |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| <b>RowID</b>                                      | <b>Name</b>   | <b>Movie</b>            |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |
| 1   | Harrison Ford   | 1                       |                    |  |             |  |           |   |          |   |                         |              |      |   |                   |              |      |   |           |            |      |   |           |                 |      |              |             |              |   |               |   |

---

## Actors

| RowID | Name             | Movie |
|-------|------------------|-------|
| 2     | Mark Hamill      | 1     |
| 3     | Carrie Fisher    | 1     |
| 4     | Harrison Ford    | 2     |
| 5     | Ron Howard       | 3     |
| 6     | Richard Dreyfuss | 3     |
| 7     | Tom Hanks        | 4     |
| 8     | Tom Hanks        | 5     |

The third column in the Actors table identifies a movie in which the actor appeared.

With this setup, the following SELECT statements can be used to query the database.

### Simple Selection

These statements select all records and all columns in a table:

```
Select * from Movies  
Select * from Actors
```

Each statement returns the tables shown above.

The following statement uses the Distinct lists all the actors only once:

```
Select distinct Name from Actors
```

It returns the following RecordSet:

| Name             |
|------------------|
| Harrison Ford    |
| Mark Hamill      |
| Carrie Fisher    |
| Ron Howard       |
| Richard Dreyfuss |
| Tom Hanks        |

### Relational Operations

Here are some examples of queries that use the WHERE and GROUP clauses to extract data from both tables.

Find all actors in 'Star Wars':

Each table is denoted by a letter and the WHERE clause specifies the records to be retrieved. It specifies that the Title column must equal 'Star Wars' and the rows in the Actors table must have the value of the Movie field equal to the RowID column in

---

Movies that has the ‘Star Wars’ record. This is RowID=1. That means the matching rows in Actors are the first three rows.

```
Select a.name from Actors A, Movies M where a.movie=m._rowid and m.title='Star Wars'
```

The Select statement specifies that only the Actors.Name column be retrieved, so the result is:

| Name          |
|---------------|
| Harrison Ford |
| Mark Hamill   |
| Carrie Fisher |

Find all movies directed by Lucas and report the actor’s name, the movie’s name, and the year:

This example differs from the previous only in that the columns to be returned are from both the Actors and Movies tables:

```
Select A.Name,M.Title,M.Year from Actors A, Movies M where A.movie=M._rowid and M.Director='George Lucas'
```

The WHERE clause works the same way as the previous example. It selects actors whose Movie column matches the Movie table’s RowID for Lucas-directed flicks. In this example, the column list is from both tables. The result is this:

| Name             | Movie                   | Year |
|------------------|-------------------------|------|
| Ron Howard       | American Graffiti       | 1973 |
| Richard Dreyfuss | American Graffiti       | 1973 |
| Harrison Ford    | Raiders of the Lost Ark | 1981 |
| Harrison Ford    | Star Wars               | 1977 |
| Mark Hamill      | Star Wars               | 1977 |
| Carrie Fisher    | Star Wars               | 1977 |

Find all actors who have done a movie with Harrison Ford:

```
Select A.name from Actors A, Actors B where A.movie = B.movie and B.name='Harrison Ford' and A.name<>'Harrison Ford'
```

The result is:

| Name          |
|---------------|
| Mark Hamill   |
| Carrie Fisher |

## Wildcard Search

Find anyone who is both an actor and a director:

```
Select A.Name from Actors A, Movies M where A.Name = M.Director
```

The result is:

| Name       |
|------------|
| Ron Howard |

## Aggregate Functions

Find any movies with the word “War” in the title (ignoring case):

```
Select title, year from Movies where Upper(title) like '%WAR%'
```

The result is:

| Title     | Year |
|-----------|------|
| Star Wars | 1977 |

Find out how many actors we have for each movie (by title):

```
Select A.Title, Count(B.Name) from Movies A, Actors B where B.Movie = A._rowID group by B.Movie
```

The result is:

| Title                   | Count(Name) |
|-------------------------|-------------|
| Star Wars               | 3           |
| Raiders of the Lost Ark | 1           |
| American Graffiti       | 2           |
| Apollo 13               | 1           |
| Cast Away               | 1           |

Find how when each director made his first and last movie (in this database).

```
Select Director, Min(Year), Max(Year) from Movies Group by Director
```

The result is:

| Director        | Min(Year) | Max(Year) |
|-----------------|-----------|-----------|
| George Lucas    | 1973      | 1981      |
| Robert Zemeckis | 2000      | 2000      |
| Ron Howard      | 1995      | 1995      |

Find how many movies each actor has done, but display only those who have done more than one.

```
Select Name, Count(*) from Actors group by Name having Count(*) > 1
```

---

The result is

| Name          | Count |
|---------------|-------|
| Harrison Ford | 2     |
| Tom Hanks     | 2     |

## UPDATE Statement

Modifies existing data in a table.

### Syntax

```
UPDATE [OR conflict-algorithm] [database-name.table-name]  
SET assignment [,assignment]  
[WHERE expr]
```

| Argument                  | Description  |
|---------------------------|--|
| <i>conflict-algorithm</i> | Optional conflict algorithm used to resolve constraint issues. See <a href="#">ON CONFLICT</a> for the list of algorithms. |
| <i>database-name</i>      | Optional name of the database containing the table to be updated.  |
| <i>table-name</i>         | Name of the table to be updated.   |
| <i>assignment</i>         | Specifies a column to the left of an equals sign and an arbitrary expression to the right.                                 |
| <i>expr</i>               | Expression that restricts the rows in <i>table-name</i> to be updated.   |

### Notes

Use the UPDATE statement to change the value of specified columns in selected rows of a table. Each assignment in an UPDATE statement specifies a column name to the left of the equals sign and an arbitrary expression to the right. The expressions may use the values of other columns. All expressions are evaluated before any assignments are made. You can include the optional WHERE clause to restrict which rows are updated.

The optional OR (a.k.a. [ON CONFLICT](#)) clause allows the specification of an alternative constraint conflict resolution algorithm to use during this command. Call the Commit command or the Database class's Commit method to save your updates or call Rollback to reset the database to its state prior to the update.

---

**Example** The following example changes the value of the Year field to 1976 for the movie with the title “Rabid.”

```
Dim dbFile as FolderItem
Dim db as REALSQLdatabase
db=New REALSQLdatabase
dbFile=New FolderItem("myMovies")
db.databaseFile=dbFile
If db.error then
    MsgBox db.errormessage
else
    db.sqlexecute ("UPDATE Movies SET Year=1976 WHERE title='Rabid'")
    If db.error then
        MsgBox db.errormessage
    Else
        db.Commit
        MsgBox "Records updated successfully."
    End if
End if
```

---

## VACUUM Statement

Cleans up an index or table

### Syntax

#### **VACUUM (*index or table-name*)**

| Part       | Description   |
|------------|---|
| index      | An index previously created via <a href="#">CREATE INDEX</a> . Currently ignored. |
| table-name | The name of an existing table. Currently ignored.                                 |

### Notes

VACUUM cleans up a database by removing empty space.

When an object (table, index, or trigger) is dropped from the database, it leaves behind empty space. This makes the database file larger than it needs to be, but the presence of empty space can speed up inserts. In time, inserts and deletes can leave the database file structure fragmented, which slows down disk access to the database contents. The VACUUM command cleans the main database by copying its contents to a temporary database file and reloading the original database file from the copy. This eliminates free pages, aligns table data to be contiguous, and otherwise cleans up the database file structure. It is not possible to perform this process on an attached database file.

This command will fail if there is an active transaction. This command has no effect on an in-memory database.

---

# SQL Functions

---

## ABS Function

Returns the absolute value of the passed value.

### Syntax

*result=ABS(x)*

| Part   | Type    | Description  |
|--------|---------|--|
| result | Numeric | Absolute value of x.                               |
| x      | Numeric | The number whose absolute value is to be returned. |

### Example

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
rs=db.SQLSelect ("ABS(-16)") //returns 16
```

---

## BIT\_LENGTH Function

Returns the number of bits in a character or bit string.

### Syntax

*result= BIT\_LENGTH(String)*

| Part   | Type    | Description                              |
|--------|---------|--|
| result | Integer | Number of bits in String.                |
| String | String  | Character or bit string to be evaluated. |

### Example

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
rs=db.SQLSelect ("SELECT BIT_LENGTH('zz')") //returns 16  
rs=db.SQLSelect ("SELECT BIT_LENGTH('cat')") /returns 24
```

---

## Coalesce Function

Returns a copy of the first non-Null argument. If all arguments are Null, then Null is returned. There must be at least two arguments. This is identical to IsNull except it can evaluate more than two arguments.

---

**Syntax** *result=IfNull(X,Y,...)*

| Part   | Type | Description                           |
|--------|------|---------------------------------------|
| result | Any  | Absolute value of x.                  |
| x      | Any  | The first parameter to be evaluated.. |
| y      | Any  | The second parameter to be evaluated. |

**See Also** [IfNull](#) function.

---

## CHAR\_LENGTH Function

Returns the number of characters in a string.

**Syntax** *result=CHAR\_LENGTH(String)*

| Part   | Type    | Description                       |
|--------|---------|-----------------------------------|
| result | Integer | Number of bits in String.         |
| String | String  | Character string to be evaluated. |

### Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT CHAR_LENGTH('Madison Ave.')") returns 12.
rs=db.SQLSelect ("SELECT CHAR_LENGTH('480')") //returns 3.
```

**See Also** [BIT\\_LENGTH](#), [OCTET\\_LENGTH](#).

---

## IfNull Function

Returns a copy of the first non-Null argument. If both arguments are Null then Null is returned.

**Syntax** *result=IfNull(X,Y)*

| Part   | Type | Description                           |
|--------|------|---------------------------------------|
| result | Any  | Absolute value of x.                  |
| x      | Any  | The first parameter to be evaluated.  |
| y      | Any  | The second parameter to be evaluated. |

**See Also** [Coalesce](#) function.

---

## Last\_Insert\_rowid Function

Returns the RowID of the last row insert from this connection to the database.

### Syntax

**result=Last\_Insert\_rowID**

| Part   | Type    | Description                                   |
|--------|---------|---|
| result | Integer | The RowID of the last insert to the database. |

---

## Length Function

Returns the length of the string passed.

### Syntax

**result=Length(str)**

| Part   | Type    | Description                       |
|--------|---------|-----------------------------------|
| result | Integer | The length of the string passed.  |
| str    | String  | Character string to be evaluated. |

---

## Like Function

Implements the “X LIKE Y” syntax of SQL.

### Syntax

**result = LIKE(X, Y [,Z])**

| Part   | Type    | Description                            |
|--------|---------|--|
| result | Boolean | The result of the X LIKE Y comparison. |
| X      | String  | string to be evaluated.                |
| Y      | String  | The pattern to which X is compared.    |
| Z      | String  | Optional Escape clause. See Notes.     |

### Notes

The LIKE operator does a pattern matching comparison. The Y parameter contains the pattern, the X parameter contains the string to match against the pattern. A percent symbol % in the pattern matches any sequence of zero or more characters in the string. An underscore \_ in the pattern matches any single character in the string. Any other character matches itself or it's lower/upper case equivalent (i.e. case-insensitive matching). REALSQLdatabase only understands upper/lower case for 7-bit Latin characters. This means that the LIKE operator is case sensitive for 8-bit iso8859 characters or UTF-8 characters. For example, the expression 'a' LIKE 'A' is TRUE but 'æ' LIKE 'Æ' is FALSE.).

---

If the optional ESCAPE clause is present, then the expression following the ESCAPE keyword must evaluate to a string consisting of a single character. This character may be used in the LIKE pattern to include literal percent or underscore characters. The escape character followed by a percent symbol, underscore or itself matches a literal percent symbol, underscore or escape character in the string, respectively.

---

## LOWER Function

Returns the string passed in all lowercase text.

### Syntax

**result=LOWER(String)**

| Part   | Type   | Description                         |
|--------|--------|-------------------------------------|
| result | String | The passed string in all lowercase. |
| String | String | Character string to be evaluated.   |

### Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT LOWER('GREENHORNET')")
//returns "greenhornet".
```

### See Also

[UPPER.](#)

---

## NullIf Function

Returns the first parameter if they are different, otherwise it returns Null.

### Syntax

**result= NullIf(X,Y)**

| Part   | Type | Description                           |
|--------|------|---------------------------------------|
| result | Any  | X or Y if they are different or Null. |
| X      | Any  | The first item to be compared.        |
| Y      | Any  | The second item to be compared.       |

---

## OCTET\_LENGTH Function

Returns the number of bytes in a character string or bit string expression.

---

## Syntax

**result=OCTET\_LENGTH (*string*)**

| Part   | Type    | Description                        |
|--------|---------|------------------------------------|
| result | Integer | Number of bytes in <b>String</b> . |
| String | String  | Character string to be evaluated.  |

## Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet

rs=db.SQLSelect ("SELECT OCTET_LENGTH ('F')") //returns 1.
rs=db.SQLSelect ("SELECT OCTET_LENGTH ('FF')") //returns 2.
```

## See Also

[BIT\\_LENGTH](#), [CHAR\\_LENGTH](#).

---

## POSITION Function

Searches for the specified target string within the source string and returns the character position where the target string starts.

## Syntax

**result=POSITION (*targetString IN sourceString*)**

| Part         | Type    | Description  |
|--------------|---------|--|
| result       | Integer | Character position where targetString first occurs in sourceString. If targetString is not found, result is set to zero. |
| targetString | String  | Character string to search for.  |
| sourceString | String  | Character string to be evaluated   |

## Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet

rs=db.SQLSelect ("SELECT POSITION ('G' IN 'Fred Gorman')") //returns 6.
```

## See Also

[SUBSTRING](#), [TRIM](#).

---

## Quote Function

Quotes the value passed to it for inclusion in another SQL statement.

---

**Syntax****result= Quote (X)**

| Part   | Type   | Description   |
|--------|--------|---|
| result | String | Value of X suitable for inclusion into another SQL statement. |
| X      | Any    | The value being quoted.                                       |

---

## Random Function

Returns a random integer.

**Syntax****result = Random**

| Part   | Type    | Description                                       |
|--------|---------|---|
| result | Integer | Random number between -2147483648 and +2147483647 |

---

## Round Function

Rounds off the passed number.

**Syntax****result = Round(X[, Y])**

| Part   | Type    | Description  |
|--------|---------|--|
| result | Number  | X rounded to the number of decimal places specified by Y.  |
| X      | Number  | The number to be rounded.  |
| Y      | Integer | The number of digits to the right of the decimal point that X is to be rounded to. If Y is omitted, it is set to zero. |

---

## SUBSTR Function

Returns the specified substring of characters from the string passed.

**Syntax****result=SUBSTR(targetString FROM start [FOR length])**

| Part         | Type    | Description  |
|--------------|---------|--|
| result       | String  | The string of characters extracted from targetString by position.                |
| targetString | String  | Character string from which to extract a substring.                              |
| Start        | Integer | Starting character position in targetString from which to extract the substring. |

|               |         |  |
|---------------|---------|--|
| <i>Length</i> | Integer | Optional parameter indicating the number of characters following Start to extract. If Length is omitted, all the characters from Start to the end of targetString will be extracted. |
|---------------|---------|--|

## Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet

rs=db.SQLSelect ("SELECT SUBSTRING('Samantha' FROM 1 For 3)")
//returns Sam'.
```

**See Also** [POSITION](#), [TRIM](#).

## TRIM Function

Trims leading and/or trailing characters from the string passed.

### Syntax

**result=TRIM (LEADING|TRAILING|BOTH [trimstring] FROM targetString)**

| Part                | Type   | Description   |
|---------------------|--------|---|
| result              | String | The string of characters after trimming targetString by removing leading and/or trailing instances of trimString.   |
| trimString          | String | Optional. Character to remove from start or end of <a href="#">targetString</a> . If omitted, the space is assumed. |
| <i>targetString</i> | String | String from which to trim characters.   |

### Notes

TRIM requires that you specify that you want to trim Leading or Trailing instances of the trimString (or both) and allows you to specify the string to be trimmed. The From keyword is required if either the trimstring or the Leading/Trailing/Both options are specified.

### Examples

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet

rs=db.SQLSelect(TRIM (BOTH from ' The Matrix ')) //returns 'The Matrix'.
rs=db.SQLSelect(TRIM (Leading '' from ' Matrix')) //areturns 'Matrix'.
```

**See Also** [POSITION](#), [SUBSTRING](#).

---

## TYPEOF Function

Returns the data type of the passed expression.

### Syntax

**result = TYPEOF (X)**

| Part   | Type   | Description   |
|--------|--------|---|
| result | String | The data type of the expression passed. The valid values are: Null, Integer, Real, Text, and Blob |
| X      | Any    | Value whose data type is to be determined.  |

---

## UPPER Function

Returns the string passed in all uppercase text.

### Syntax

**result=UPPER (String)**

| Part   | Type   | Description                         |
|--------|--------|-------------------------------------|
| result | String | The passed string in all uppercase. |
| String | String | Character string to be evaluated.   |

### Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet
.
rs=db.SQLSelect ("SELECT UPPER('Grand Blanc')")
//returns "GRAND BLANC".
```

### See Also

[LOWER](#).

---

## CURRENT\_DATE Function

Returns the current date in the SQL Date format, YYYY-MM-DD.

### Syntax

**result=CURRENT\_DATE**

| Part   | Type    | Description                                |
|--------|---------|--|
| result | SQLDate | The current date in the YYYY-MM-DD format. |

---

## Example

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
  
rs=db.SQLSelect ("SELECT CURRENT_DATE") //returns '2003-10-07'
```

---

## CURRENT\_TIME Function

Returns the current time in the SQL Time format, HH:MM:SS.

### Syntax

*result=CURRENT\_TIME*

| Part   | Type    | Description                                       |
|--------|---------|---|
| result | SQLTime | The current time in the HH:MM:SS SQL Time format. |

## Example

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
  
rs=db.SQLSelect ("SELECT CURRENT_TIME") //returns '13:51:30', for example
```

### See Also

[CURRENT\\_TIMESTAMP](#).

---

## CURRENT\_TIMESTAMP Function

Returns the current date-time in the SQLTimestamp format, YYYY-MM-DD HH:MM:SS.

### Syntax

*result=CURRENT\_TIMESTAMP*

| Part   | Type         | Description   |
|--------|--------------|---|
| result | SQLTimestamp | The current date/ time in the YYYY-MM-DD HH:MM:SS SQL Timestamp format. |

## Example

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
  
rs=db.SQLSelect ("SELECT CURRENT_TIMESTAMP")  
//returns '2003-10-07 13:52:57' for example
```

---

See Also [CURRENT\\_TIME](#).

---

## LAST\_ROWID Function

Returns the value of the last \_rowID for the table passed. This value is useful for certain types of relational joins. The REALSQLdatabase automatically adds a row\_ID field to every table as a unique identifier. Values for this field are assigned automatically unless values are assigned by the user.

Syntax **result=LAST\_ROWID (tablename)**

| Part      | Type    | Description   |
|-----------|---------|---|
| result    | Integer | The last value of the _rowID column that has been added to the specified table. |
| tablename | String  | The name of the table for which you want the last _rowID.                       |

### Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT LAST_ROWID ('Employees')")
```

## Aggregate Functions

---

### AVG Function

Computes the average value of an expression across rows.

Syntax **result=AVG [DISTINCT|ALL] (expression)**

| Part             | Type    | Description   |
|------------------|---------|---|
| result           | Double  | The average of expression across the rows specified by searchExpression.    |
| expression       | Numeric | Column or function of one or more columns evaluates to a numeric data type. |
| SearchExpression |         | SQL WHERE clause that selects the rows that are evaluated.                  |

If you use the DISTINCT keyword, only unique values of expression will be included in the calculation. If you use ALL, all values of expression will be included, including duplicate ones. The default is ALL.

---

## Example

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet

rs=db.SQLSelect("Select AVG (Population) FROM Cities")
rs=db.SQLSelect("Select AVG (Income) FROM Cities WHERE State='CA'")
```

### See Also

[COUNT](#), [SUM](#).

---

## COUNT Function

Computes the number of rows in an expression.

### Syntax

**result=COUNT [DISTINCT] expression FROM searchExpression**

| Part             | Type    | Description  |
|------------------|---------|--|
| result           | Integer | The total of expression across the rows specified by searchExpression. |
| expression       | Any     | Column or function of one or more columns.                             |
| SearchExpression |         | SQL WHERE clause that selects the rows that are evaluated.             |

If you use the DISTINCT keyword, only unique values of expression will be included in the calculation. If you use \* as the expression, all rows will be counted.

When **expression** is specified, COUNT returns the number of rows for the columns in **expression**.

### Example

This example returns the number of rows in the Movies table with non-missing values in the Title and Director fields. The second example counts the rows with non-missing values for Title in which the Director field equals 'Lucas'.

```
Dim db as New REALSQLdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT COUNT (DISTINCT Director) from Movies")
rs=db.SQLSelect ("Select Count (*) from Movies WHERE Director = 'Lucas'")
rs=db.SQLSelect ("Select Count (*) from Movies")
```

### See Also

[AVG](#), [SUM](#).

---

## MAX Function

Returns the maximum value of an expression.

---

**Syntax**

**result=MAX expression FROM searchExpression**

| Part                    | Type              | Description   |
|-------------------------|-------------------|---|
| result                  | Double            | The maximum value of expression across the rows specified by searchExpression.              |
| expression              | Numeric or String | Column or function of one or more columns that evaluates to a numeric or string expression. |
| <i>SearchExpression</i> |                   | SQL WHERE clause that selects the rows that are evaluated.                                  |

**Example**

This example finds the largest population in the Cities table.

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
. . .  
rs=db.SQLSelect("SELECT MAX (Population) from Cities")
```

**See Also**

[MIN](#).

---

## MIN Function

Returns the minimum value of the specified column or columns.

**Syntax**

**result=MIN expression FROM searchExpression**

| Part                    | Type              | Description   |
|-------------------------|-------------------|---|
| result                  | Double            | The maximum value of expression across the rows specified by searchExpression.              |
| expression              | Numeric or String | Column or function of one or more columns that evaluates to a numeric or string expression. |
| <i>SearchExpression</i> |                   | SQL WHERE clause that selects the rows that are evaluated.                                  |

**Example**

This example finds the minimum Price in the Products table.

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
. . .  
rs=db.SQLSelect("SELECT MIN (Price) from Products")
```

**See Also**

[MAX](#).

# SUM Function

Returns the total of the specified numeric fields.

## Syntax

**result=SUM [DISTINCT|ALL] expression FROM searchExpression**

| Part                    | Type    | Description   |
|-------------------------|---------|---|
| result                  | Double  | The maximum value of expression across the rows specified by searchExpression.    |
| expression              | Numeric | Column or function of one or more columns that evaluates to a numeric expression. |
| <i>SearchExpression</i> |         | SQL WHERE clause that selects the rows that are evaluated.                        |

If you use the DISTINCT keyword, only unique values of expression will be included in the calculation. If you use ALL, all values of expression will be included, including duplicate ones. The default is ALL.

## Notes

*Expression* must evaluate to a numeric data type. It can include one or more columns. The DISTINCT keyword can be used to limit the evaluation to distinct (unique) occurrences of each value.

## Example

This example finds the total population from the States table for States in the NE region only.

```
Dim db as New REALSQLdatabase  
Dim rs as New RecordSet  
. . .  
rs=db.SQLSelect("SELECT SUM (Population) from States WHERE Region='NE'")
```

## See Also

[AVG](#), [COUNT](#).



# REALdatabase SQL Language Reference

---

A REALbasic database front-end uses the Structured Query Language (SQL) to communicate with its data sources. The plug-in for your data source receives a SQL statement from REALbasic, translates the statement into a form that the data source understands (if necessary), and sends it to the data source.

In version 5.5 of REALbasic, the data source known as REALdatabase shipped with both the standard and professional versions of REALbasic. It uses the limited subset of SQL that is described in this chapter. Beginning with REALbasic 2005, the REALdatabase engine was replaced by the REALSEQLdatabase data source. It supports a much more extensive subset of SQL and is now the recommended choice. This appendix describes the obsolescent REALdatabase data source. It is included in REALbasic 2005 for backwards compatibility only.

The REALdatabase supports a subset of SQL/92 and SQL/99, including queries that involve self-joins, aggregate functions, and more. For the set of features that the REALdatabase engine supports, its syntax is fully SQL compliant (with a few minor extensions, like the Boolean data type). It also returns standard SQL error codes.

Every REALdatabase table has a special column called “\_rowID” which is a unique identifier for that row. This column is added and maintained automatically, and serves as a convenient join field for building relational databases.

Except for 4D Server, the other supported data sources are native SQL back-ends. When you are working with any native SQL back-end, its REALbasic plug-in passes your SQL to the data source. Therefore any valid SQL for that data source will work

---

in a REALbasic front-end for that source. Please refer to the documentation for your selected data source for further information on supported SQL and specifics on the data types supported by that data source.

If you are unfamiliar with SQL, you will need to learn its basics before implementing your REALbasic front-end. This manual does not attempt to teach you SQL; rather, it describes the subset of SQL that is currently supported for the REAL database engine and 4th Dimension. Please consult one of the many good SQL references, such as *SQL for Dummies* by Allen G. Taylor (ISBN: 0-7645-0105-4), *The Practical SQL Handbook*, by Bowman, Emerson, and Darnovsky (ISBN: 0-2014-4787-8). You can also find several good online SQL references by searching for “SQL command syntax” or for a specific SQL command using <http://www.google.com> or a comparable search engine.

## Supported SQL Syntax

A semi-formal specification of the SQL syntax supported by the new REAL database engine is given in this section. It uses the following conventions: a vertical bar (“|”) separates items in a list of alternatives; square brackets (“[“ and “]”) indicate an optional part which may be included or omitted; and curly braces (“{“ and “}”) indicate a required part which may not be omitted. Items in angle brackets (e.g. “<select-item-list>”) indicate a more complex part which may be detailed in its own specification entry. Any such item which ends in “-string” is a simple identifier (e.g., “foo”).

```
valid-sql-input:  
{ <select-statement>  
| <insert-statement>  
| <update-statement>  
| <delete-statement>  
| <create-table-statement>  
| <create-index-statement>  
| <alter-table-statement>  
| <drop-table-statement>  
| <drop-index-statement>  
| COMMIT  
| ROLLBACK  
}  
  
select-statement:  
SELECT [ ALL | DISTINCT ] <select-item-list>  
FROM <table-list>  
[ WHERE <expression> ]  
[ GROUP BY <column-ref-list> ]  
[ HAVING <expression> ]  
[ ORDER BY <expression> ]
```

---

select-item-list: one or more comma-separated items of the form:  
  <column-expression> [ [ AS ] <column-name-string> ]

table-list: one or more comma-separated items of the form:  
  <table-name-string> [ [ AS ] <range-variable-string> ]

insert-statement:

```
INSERT INTO <table-name-string> [ ( <column-name-list> ) ]
{ DEFAULT VALUES
| VALUES ( <expression-list> )
| <select-statement>
}
```

update-statement:

```
UPDATE <table-name-string> SET <assignment-list>
[ WHERE <expression> ]
```

assignment-list: one or more comma-separated items of the form:

```
<column-name-string> = { <expression> | DEFAULT | NULL }
```

delete-statement:

```
DELETE FROM <table-name-string>
[ WHERE <expression> ]
```

create-table-statement:

```
CREATE TABLE <table-name-string> ( <column-def-list> )
```

column-def-list: one or more comma-separated items of the form:

```
<column-name-string> <column-type-string>
```

create-index-statement:

```
CREATE INDEX <index-name-string> ON <table-name-string>
( <column-name-list> )
```

alter-table-statement:

```
ALTER TABLE <table-name-string>
{ ADD [ COLUMN ] <column-name-string> <column-type-string>
| DROP [ COLUMN ] <column-name-string> { RESTRICT | CASCADE }
}
```

drop-table-statement:

```
DROP TABLE <table-name-string> { RESTRICT | CASCADE }
```

drop-index-statement:

```
DROP INDEX <table-name-string>.<index-name-string>
```

expression:

```
{ <expression> OR <expression> }
```

---

```

| <expression> AND <expression>
| NOT <expression>
| <expression> { = | <> | < | <= | > | >= } <expression>
| <expression> [ NOT ] LIKE <expression> [ ESCAPE <expression> ]
| <expression> { + | - } <expression>
| <expression> { * | / } <expression>
| -<expression>
| ( <expression> )
| <function>
| { <number> | <string-literal> | TRUE | FALSE }
| DATE <date-string>
| TIME <time-string>
| TIMESTAMP <timestamp-string>
| <column-reference>
}

column-reference:
{ <column-name-string>
| <range-variable-string>.<column-name-string>
| *
| <range-variable-string>.*
}

function:
{ BIT_LENGTH ( <expression> )
| OCTET_LENGTH ( <expression> )
| CHAR_LENGTH ( <expression> )
| CURRENT_DATE
| CURRENT_TIME
| CURRENT_DATETIME
| LOWER ( <expression> )
| UPPER ( <expression> )
| POSITION ( <expression> IN <expression> )
| SUBSTRING ( <expression> FROM <expression> [ FOR <expression> ] )
| TRIM ( [ LEADING | TRAILING | BOTH ]
    [ <expression> ] [ FROM ] <expression> )
| AVG ( [ DISTINCT | ALL ] <expression> )
| MAX ( [ DISTINCT | ALL ] <expression> )
| MIN ( [ DISTINCT | ALL ] <expression> )
| SUM ( [ DISTINCT | ALL ] <expression> )
| COUNT ( [ DISTINCT | ALL ] <expression> )
| COUNT (*)
| LAST_ROWID ( 'table-name-string' )
}

```

---

## Supported Data Types

SQL for the REALdatabase supports a subset of data types supported by SQL. They are described in the following table:

| FieldType       | Numeric Code | Description  |
|-----------------|--------------|--|
| Integer         | 3            | A numeric data type with no fractional part. The maximum number of digits is implementation-specific. The REAL database supports 4-byte integers, which provide a range of $\pm 2,000,000,000$ . If you are using another data source, check the documentation of your data source.  |
| Text or VarChar | 5            | Stores alphabetic data, in which the number of characters vary from record to record, but you don't want to pad the unused characters with blanks. For example, "VARCHAR (20)" specifies a VARCHAR field with a maximum length of 20 characters. The REAL database supports the VarChar field type, not the Text type. Theoretically, a REALdatabase VarChar field can store a maximum of $2^{31}$ bytes; if you specify a maximum length in a CREATE TABLE or ALTER TABLE statement, it is ignored. |
| Double          | 7            | Stores double-precision floating-point numbers. The REAL database uses 8-byte doubles.   |
| Date            | 8            | Stores year, month, and day values of a date, in the format YYYY-MM-DD. The year value is four digits; the month and day values are two digits.  |
| Time            | 9            | Stores hour, minute, and second values of a time, in the HH:MM:SS format. The hours and minutes are two digits. The seconds values is also two digits, may include a optional fractional part, e.g., 09:55:25.248. The default length of the fractional part is zero.  |
| TimeStamp       | 10           | Stores both date and time information, in the format YYYY-MM-DD HH:MM:SS. The lengths of the components of a TimeStamp are the same as for Time and Date, except that the default length of the fractional part of the time component is six digits rather than zero. If a TimeStamp values has no fractional component, then its length is 19 digits If it has a fractional component, its length is 20 digits, plus the length of the fractional component.  |
| Boolean         | 12           | Stores the values of TRUE or FALSE.  |
| Binary          | 14           | Stores code, images, and hexadecimal data. Consult the documentation of your data source for information on the maximum size of a Binary field. The REALdatabase stores up to $2^{31}$ bytes.  |
| String          | 18           | Text up to $2^{31}$ bytes. The same as VarChar.  |

The REALdatabase supports an escape character in string literals. Use the SET ESCAPE command to choose which character to use (for example, "SET ESCAPE p" to use the backslash). Once set, that character will be removed from any string literal, and the subsequent character will be treated as part of the

---

string data. To remove the escape character, use SET ESCAPE by itself. The default is no escape character (per the SQL standard).

## SQL in REALbasic

This section provides an overview of SQL for the REALdatabase. Here are the supported SQL statements.

| Statement                    | Description  |
|------------------------------|--|
| <a href="#">ALTER TABLE</a>  | Adds or drops a column in an existing table.   |
| <a href="#">COMMIT</a>       | Commits changes (inserts, deletes, and updates) to the data and changes to the database schema (CREATE TABLE, CREATE INDEX, and so forth).   |
| <a href="#">CREATE INDEX</a> | Creates an index in a table.   |
| <a href="#">CREATE TABLE</a> | Creates a table and specifies the fields and their attributes.   |
| <a href="#">DELETE</a>       | Deletes a specified group of records.  |
| <a href="#">DROP TABLE</a>   | Removes a table from the database.   |
| <a href="#">DROP INDEX</a>   | Remove the specified index from a table.   |
| <a href="#">INSERT</a>       | Inserts a record into the specified table.   |
| <a href="#">ROLLBACK</a>     | Cancels changes (inserts, deletes, updates) to the data and changes to the database schema (CREATE TABLE, CREATE INDEX, and so forth).   |
| <a href="#">SELECT</a>       | Returns a group of rows, known in REALbasic as a recordset. In the language of SQL, this is also called a <i>cursor</i> . You can specify the columns (fields), the table(s), search conditions, grouping, and sort columns. |
| <a href="#">UPDATE</a>       | Updates existing records in a table.   |

## Character Functions

The following functions compute values from the character or bit string passed to it.

| Function                     | Example                                     | Description   |
|------------------------------|---|---|
| <a href="#">BIT_LENGTH</a>   | BIT_LENGTH('zz') returns 32                 | Takes either a character string or a bit string and returns the number of bits.<br>BIT_LENGTH ( <i>string</i> ) |
| <a href="#">CHAR_LENGTH</a>  | CHAR_LENGTH ('480') returns 3.              | Returns the number of characters in the string passed.<br>CHAR_LENGTH ( <i>string</i> )                         |
| <a href="#">LOWER</a>        | LOWER('GREENHORNET') returns "greenhornet". | Takes a character string and returns the string in all lowercase.<br>LOWER ( <i>string</i> )                    |
| <a href="#">OCTET_LENGTH</a> | OCTET_LENGTH ('FF') returns 2.              | Returns the number of bytes in a character string or bit string expression.<br>OCTET_LENGTH ( <i>string</i> )   |

| Function                  | Example   | Description   |
|---------------------------|---|---|
| <a href="#">POSITION</a>  | POSITION ('G' IN 'Fred Gorman') returns 6.  | Searches for the specified target string within the source string and returns as an integer the character position where the target string starts. If it doesn't find the target string, it returns zero.<br>POSITION ( <i>targetString</i> IN <i>sourceString</i> )  |
| <a href="#">SUBSTRING</a> | SUBSTRING ('Samantha' FROM 1 For 3) returns 'Sam'.  | Returns the specified substring of characters from the sting passed.<br>SUBSTRING ( <i>string</i> FROM <i>start</i> [FOR <i>length</i> ])<br><i>Start</i> is the starting position; if the optional FOR clause is omitted, SUBSTRING returns the characters from start to the end of the string. If FOR is included, SUBSTRING returns the <i>length</i> characters beginning at <i>start</i> . |
| <a href="#">TRIM</a>      | TRIM BOTH from ' The Matrix ' returns 'The Matrix'. Same as TRIM BOTH '' from ' The Matrix '. | Trims leading and/or trailing characters from the string passed. The default character to trim is the blank.<br>TRIM (LEADING TRAILING BOTH [ <i>trimstring</i> ] FROM <i>targetString</i> )  |
| <a href="#">UPPER</a>     | UPPER('Grand Blanc') returns "GRAND BLANC".   | Takes a character string and returns the string in all uppercase.<br>UPPER ( <i>string</i> )  |

## SQL Functions

The following date, time, and ID functions are available:

| Function                          | Example               | Description   |
|-----------------------------------|-----------------------|---|
| <a href="#">CURRENT_DATE</a>      | CURRENT_DATE          | Returns the current date in SQL Date format, YYYY-MM-DD, as a SQL Date data type.<br>CURRENT DATE   |
| <a href="#">CURRENT_TIME</a>      | CURRENT_TIME          | Returns the current time in SQL Time, HH:MM:SS, as a SQL Time data type.<br>CURRENT TIME.   |
| <a href="#">CURRENT_TIMESTAMP</a> | CURRENT_TIMESTAMP     | Returns the current date-time in SQL TimeStamp format, YYYY-MM-DD HH:MM:SS, as a SQL TimeStamp data type.   |
| <a href="#">LAST_ROWID</a>        | LAST_ROWID ('Movies') | Returns the value of the last _rowID for the table passed. This value is useful for certain types of relational joins.<br>LAST_ROWID ( <i>tablename</i> ) |

## Aggregate Functions

The following functions are aggregate functions, sometimes known as set functions. They apply to sets of rows in a table and return the results of an arithmetic

calculation. You determine the number of rows on which the calculation is based using a standard WHERE clause and include the Set function in a SELECT statement. You can use several Set functions in a SELECT statement and each calculates its value on the rows specified by the WHERE clause. However, you cannot use the DISTINCT keyword more than once. The SELECT statement returns a RecordSet. If the WHERE clause is omitted, the function is computed on all the rows in the table.

If you use the GROUP BY clause with a aggregate functions, the result will contain one row per group, as defined by the GROUP BY clause. The aggregate functions compute the summary statistics per group. With no GROUP BY clause, one row will be returned, with summary statistics for the all the records that were queried.

| Function              | Description  |
|-----------------------|--|
| <a href="#">AVG</a>   | Returns the numeric average of an expression. The expression must evaluate to a numeric data type. The expression can include one or more columns. The DISTINCT keyword can be used to limit the computation to distinct (unique) occurrences of each value.<br>Example: Finds the average of the Price column in the Products table, for Database products only.<br>SELECT AVG (Price) from Products WHERE Product='Database' |
| <a href="#">COUNT</a> | Returns the total number of rows accessed by the expression. The expression may include one or more columns. The DISTINCT keyword can be used to limit the evaluation to distinct (unique) occurrences of each value.<br>Example: Counts the number of Customer rows for the Zip Code '48070' only.<br>SELECT COUNT (*) from Customers WHERE Zip='48070'   |
| <a href="#">MAX</a>   | Returns the maximum value of the specified field or fields. The Max function operates on numeric, date, time, and character data types. The expression may include one or more columns.<br>Example: Finds the largest population in the Cities table.<br>SELECT MAX (Population) from Cities   |
| <a href="#">MIN</a>   | Returns the minimum value of the specified field or fields. The Min function operates on numeric, date, time, and character data types. The expression may include one or more columns.<br>Example: Finds the minimum Price in the Products table.<br>SELECT MIN (Price) from Products   |
| <a href="#">SUM</a>   | Returns the total of the specified fields. That is, the expression must evaluate to a numeric data type. The expression may include one or more columns. The DISTINCT keyword can be used to limit the evaluation to distinct (unique) occurrences of each value.<br>Example: Finds the total population from the States table for States in the NE region only.<br>SELECT SUM (Population) from States WHERE Region='NE'      |

## Calling SQL Commands

You use the SELECT statement by passing it as a string to the Database class's SQLSelect method. It returns a RecordSet object that contains the records that were

---

requested. You use the Database class's SQLExecute statement to execute all the other SQL statements. None of the other SQL statements return data.

The examples in this chapter assume that you've created an instance of the database class. The SQL example is passed as a string parameter to either the SQLExecute or SQLSelect methods.

The Syntax section gives the syntax of the SQL string, omitting the surrounding REALbasic code that's needed to pass the SQL to the data source.

## Return Codes

The REALdatabase returns both a result code and a message, which you can access via the ErrorCode and ErrorMessage properties of the Database class. In the ErrorMessage property, the error code appears in brackets after the error text.

| Code  | Description  |
|-------|--|
| 00000 | Success. Successful SELECT statements return this code.                        |
| 02000 | No records found. If a SELECT statement returns no rows, it returns this code. |
| 01004 | Warning: String truncated on right.  |
| 01S09 | Warning: Invalid SQL keyword.  |
| 07000 | Error: Dynamic SQL error.  |
| 42S01 | Error: Table already exists.   |
| 42S02 | Error: Table not found.  |
| 42S11 | Error: Index already exists.   |
| 42S12 | Error: Index not found.  |
| 42S21 | Error: Column already exists.  |
| 42S22 | Error: Column not found.   |
| 21S01 | Error: Insert value list does not match column list.                           |
| 22000 | Error: Data exception.   |
| 22023 | Error: Invalid parameter.  |
| 25000 | Error: Invalid transaction state.  |
| 0A000 | Error: Unsupported SQL feature.  |
| 72000 | Error: Internal SQL processor error.   |

## SQL Command Reference

This section is a language reference for the subset of SQL that is supported by the REALdatabase data source.

---

## ALTER TABLE Statement

Adds or drops a column in a table.

---

**Syntax**

**ALTER TABLE *tablename* ADD [Column] *fieldname* *fieldtype***  
**OR**  
**ALTER TABLE *tablename* DROP [Column] *fieldname* {RESTRICT|CASCADE}**

| Part             | Description  |
|------------------|--|
| <i>TableName</i> | Name of the existing table.  |
| <i>FieldName</i> | Name of the field to add.  |
| <i>FieldType</i> | Data type. The REAL database supports the following data types: integer, varchar, double, boolean, date, time, timestamp, binary (blob), and string. If you are using another vendor's data source, see their documentation on supported data types. |

The keyword “Column” is assumed and can be omitted.

**Example** The following example adds the column “VacationDays” to the table “Employees”.

```
Dim dbFile as FolderItem
Dim db as RealDatabase
db=New RealDatabase
dbFile=Getfolderitem("myCompany")
db.DatabaseFile=dbFile
if db.connect then
    db.SQLExecute ("ALTER TABLE Employees ADD COLUMN VacationDays Integer")
if db.error then
    MsgBox db.errormessage
else
    db.SQLExecute("Commit") //no error, save change
end if
else
    MsgBox "Connection to database failed."
end if
```

**See Also** [CREATE TABLE, DROP TABLE](#)

---

## COMMIT Statement

Commits (saves) changes to a database.

**Syntax** **Commit**

**Notes** The REALdatabase supports transactions. A transaction is either a set of changes to the database schema (i.e., tables, fields, and field types) or a set of changes to the contents of the database—record additions, deletions, or modifications. A transaction begins automatically by the first such change and is saved to the

---

database by calling Commit. You can instead abort the transaction by calling ROLLBACK. This restores the database to its state when the transaction was started.

You can also commit changes to the database by calling the Database class's Commit method.

## Example

The following example uses Commit to save a change to a table. Some examples in this reference use the Commit method of the Database class to commit changes.

```
Dim dbFile as FolderItem
Dim db as RealDatabase
db=New RealDatabase
dbFile=Getfolderitem("myCompany")
db.DatabaseFile=dbFile
if db.connect then
    db.SQLExecute ("ALTER TABLE Employees ADD COLUMN VacationDays
Integer")
    if db.error then
        MsgBox db.errormessage
    else
        db.SQLExecute("Commit") //no error, save change
    end if
else
    MsgBox "Connection to database failed."
end if
```

## See Also

[ROLLBACK](#)

---

# CREATE INDEX Statement

Creates an index in a table.

## Syntax

**CREATE INDEX *indexname* ON *tablename* (*fieldlist*)**

| Argument         | Description   |
|------------------|---|
| <i>IndexName</i> | The name of the new index.  |
| <i>TableName</i> | The name of the table that contains the fields being indexed.   |
| <i>Fieldlist</i> | The names of one or more fields on which to build the index. <i>FieldList</i> consists of one or more fields/data types, separated by commas. |

## Notes

You index columns that you use for your most frequent searches and sorts, as well as fields used for relational joins. An index makes a search much faster because the database doesn't have to scan the table sequentially to find the record you are looking for.

---

**Example**

The following example creates an index called “NameIndex” in the table “Employees” using the “Name” field.

```
Dim dbFile as FolderItem
Dim db as REALdatabase
db=New REALdatabase
dbFile = GetFolderItem("myCompany")
db.DatabaseFile=dbFile
If db.Connect then
    db.SQLExecute ("Create Index NameIndex on Employees (Name)")
    If db.error then
        MsgBox db.errormessage
    Else
        db.SQLExecute ("Commit")
    end if
else
    MsgBox "Database connection failed."
End if
```

**See Also**

[DROP INDEX](#)

---

## CREATE TABLE Statement

Adds a table to the database.

**Syntax**

**CREATE TABLE *tablename* (*fieldname1 fieldtype [not null],...,FieldnameN fieldtype [not null]*)**

| Part             | Description  |
|------------------|--|
| <i>TableName</i> | Name of the new table.   |
| <i>FieldName</i> | Name of a field.   |
| <i>FieldType</i> | Data type. The REAL database supports the following data types: integer, varchar, double, boolean, date, time, timestamp, binary (blob), and string. |
| <i>Not Null</i>  | Optional. If Not Null is used, the database will require a value for that field in every row, i.e., the field may not be missing.                    |

**Notes**

The Create Table statement creates a table structure on the current data source. You use the Database class's SQLExecute method to pass a Create Table statement to REALbasic. The REALdatabase data source automatically adds a Primary Key field called “\_RowID” to your specifications. This means you do not have to manually add and manage your own Primary Key field. Use the \_RowID field for relational joins and other situations that require a record identifier column.

---

## Example

The following code opens an existing database, "myDVDs", and adds a table with three fields. The underscore keyword is used to break the string passed to SQLExecute into two lines in the Code Editor. It is treated by REALbasic's compiler as one logical line.

```
Dim dbFile as FolderItem
Dim rs as New Recordset
Dim db as REALdatabase
db=New REALdatabase
dbFile = GetFolderItem("myDVDs")
db.DatabaseFile=dbFile
If db.Connect() then //check if the database was opened successfully
    db.SQLExecute ("Create Table Movies (Title Varchar,Year Integer,_
        + "Director Varchar)" //"_ used to break this into two lines
    If db.Error then
        MsgBox db.ErrorMessage
    else
        db.Commit
        MsgBox "Table created successfully."
    end if
else //Connect method failed.
    Beep
    MsgBox "The database couldn't be opened."
end if
```

## See Also

[ALTER TABLE](#), [DROP TABLE](#).

---

## DELETE Statement

Deletes the records specified in its WHERE clause.

## Syntax

**DELETE FROM *tablename* WHERE *searchcondition***

| Part                   | Description   |
|------------------------|---|
| <i>tablename</i>       | Name of the table from which you are removing records.            |
| <i>searchcondition</i> | Boolean expression that identifies the row or rows to be updated. |

---

## Example

The following example deletes the record in the “Movies” table that has the title “Star Wars”.

```
Dim dbFile as FolderItem  
Dim db as RealDatabase  
db=New RealDatabase  
dbFile=New FolderItem("myMovies")  
db.databaseFile=dbFile  
If db.error then  
    MsgBox db.errormessage  
else  
    db.SQLExecute ("Delete From Movies Where Title='Star Wars'")  
    If db.error then  
        MsgBox db.ErrorMessage  
    else  
        db.Commit //save deletion  
        MsgBox "Record deleted successfully."  
    end if  
end if
```

## See Also

[INSERT](#), [UPDATE](#).

---

## DROP INDEX Statement

Removes an index from a table.

## Syntax

**DROP INDEX *tablename.indexname***

| Argument         | Description  |
|------------------|--|
| <i>tablename</i> | The name of table that contains the index to be dropped. |
| <i>indexname</i> | The name of the index to be dropped.                     |

---

**Example**

The following example drops the index called “NameIndex” in the table “Employees” using the “Name” field.

```
Dim dbFile as FolderItem
Dim db as REALdatabase
db=New REALdatabase
dbFile = GetFolderItem("myCompany")
db.DatabaseFile=dbFile
If db.Connect then
    db.SQLExecute ("Drop Index Employees.NameIndex")
    If db.error then
        MsgBox db.errormessage
    Else
        db.SQLExecute ("Commit")
    end if
else
    MsgBox "Database connection failed."
End if
```

**See Also**

[CREATE INDEX](#)

---

## DROP TABLE Statement

Removes a table from the database.

**Syntax**

**DROP TABLE *tablename* {RESTRICT| CASCADE}**

| Argument         | Description                                    |
|------------------|--|
| <i>tablename</i> | Name of table to be dropped from the database. |

---

## Example

The following example drops the table named “Employees” from the database.

```
Dim dbFile as FolderItem
Dim db as RealDatabase
db=New RealDatabase
dbFile=Getfolderitem("myCompany")
db.DatabaseFile=dbFile
If db.connect then
    db.SQLExecute ("Drop Table Employees")
If db.error then
    MsgBox db.errormessage
Else
    db.Commit //save changes to the schema
    MsgBox "Table dropped successfully."
end if
Else
    MsgBox "Connection to database failed."
End if
```

---

## See Also

[CREATE TABLE](#), [ALTER TABLE](#).

---

# INSERT Statement

Inserts a record into a table.

---

## Syntax

**INSERT INTO *tablename* [*fieldname1*,...*fieldnameN*] VALUES  
(*value1*,...*valueN*)**

| Argument                      | Description   |
|-------------------------------|---|
| <i>tablename</i>              | Name of the existing table.   |
| <i>fieldname1</i> to <i>N</i> | Optional. Names of the fields in <i>tablename</i> . If the field list is omitted, values for all fields must be provided in order. If you supply the field list, then the list of values match the specified fields, i.e., <i>value1</i> is the value for <i>field1</i> , <i>value2</i> is for <i>field2</i> , and so forth. You can insert a value for the _rowID field as long as the value you specify is not already in use. Otherwise, the value for _rowID will be provided by the database engine. |
| <i>value1</i> to <i>N</i>     | Values for each field in the row being added. In place of a value, you can use the keyword DEFAULT to assign the default value for the column.  |

---

## Example

The following example inserts four records into the “Employees” table. The underscore character is used to break up long logical lines into separate lines as displayed in the Code Editor.

```
Dim dbFile as FolderItem
Dim db as RealDatabase
db=New RealDatabase
dbFile=New FolderItem("myCompany")
db.databaseFile=dbFile
If db.error then
    MsgBox db.errormessage
else
    db.SQLExecute ("Insert into Employees
    (Name,Department,Salary,VacationDays)_
        + "Values ('Seymore','Accounting',5000,0)")
    db.SQLExecute ("Insert into Employees
    (Name,Department,Salary,VacationDays)_
        + "Values ('Throckmorton','Marketing',4400,0)")
    db.SQLExecute ("Insert into Employees
    (Name,Department,Salary,VacationDays)_
        + "Values ('Dilbert','Engineering',6400,10)")
    db.SQLExecute ("Insert into Employees
    (Name,Department,Salary,VacationDays)_
        + "Values ('Winnie','Sales',3500,0)")
If db.error then
    MsgBox db.errormessage
Else
    db.Commit //save new records to database
    MsgBox "Records added successfully."
End if
rs=db.sqlSelect ("Select * from Employees") //select all Employee records
If db.error then
    MsgBox db.errormessage
Else
    //proceed with database operations here
End if
End if
```

---

## See Also

[DELETE](#), [COMMIT](#), [ROLLBACK](#).

---

## ROLLBACK Statement

Restores the database to its original state prior to the start of a transaction.

---

## Syntax

**ROLLBACK**

---

## Notes

The REALdatabase supports transactions. A transaction is either a set of changes to the database schema (i.e., tables, fields, and field types) or a set of changes to the contents of the database—record additions, deletions, or modifications performed by the INSERT, DELETE, or UPDATE statements. A transaction begins automatically with the first such change and is saved to the database by calling Commit. You can instead abort the transaction by calling ROLLBACK. This restores the database to the state it was when the transaction was started.

You can also rollback changes to the database by calling the Database class's Rollback method.

## See Also

[COMMIT](#).

---

# SELECT Statement

Use the SQL Select statement to query the database. You pass a valid SQL Select statement as a string to the SQLSelect method of the Database class. It selects and optionally groups and sorts records in one or more tables.

## Syntax

***result=SELECT [ALL | DISTINCT] expressionList FROM tablelist WHERE [LIKE] searchcondition GROUP BY groupingcondition HAVING filtercondition ORDER BY sortCondition***

| Part                  | Description   |
|-----------------------|---|
| <i>result</i>         | The resulting group of records is returned as a REALbasic RecordSet object. The SQL SELECT statement must be passed to the Database class's SQLSelect method rather than SQLExecute.  |
| <i>ALL DISTINCT</i>   | Optional keyword. <i>DISTINCT</i> retrieves only unique values for <i>fieldlist</i> , i.e., it retrieves only unique rows. For example 'DISTINCT Cities' retrieves only one record per city even if there are several records per city. The ALL keyword retrieves all records, even if the values for <i>fieldlist</i> are not unique. 'ALL' is assumed unless DISTINCT is used.  |
| <i>expressionList</i> | List of columns or functions of columns separated by commas. Use the asterisk (*) to retrieve all fields in <i>table</i> . If <i>tablelist</i> refers to multiple tables, use the syntax <i>tablename.fieldname</i> to refer to a field. For example, if the database has an Actors table with fields Name, Role and a Movies table with fields Director, Title, Year, then you would refer to an actor's name as Actors.Name or a movie title as Movies.Title. When doing queries on multiple tables, it accepts aliases for table names. For example, "SELECT A.NAME, B.NAME FROM ACTORS A, ACTORS B WHERE A.MOVIE = B.MOVIE". The <i>expressionList</i> can be as simple as a list of fields but can also do calculations using math operators or call the SQL or aggregate functions. |

| <b>Part</b>              | <b>Description</b>   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
|--------------------------|--|-------------------|---------------|----------|---|-----------|----|-----------|---|-----------------------|----|--------------|---|--------------------------|----|
| <i>tablelist</i>         | List of tables separated by commas. If a tablename has spaces in it, it must be surrounded by brackets, e.g., [Product Groups]   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| <i>LIKE</i>              | Optional keyword that compares two character strings for a partial match (begins with, contains, ends with) instead of an exact (full) match. When you use LIKE, you use the SQL wildcard characters '%' and '_' in the search string. The '%' stands for a string of any characters and the '_' stands for any one character. For example, "WHERE Movies LIKE 'War%'" specifies movies that start with "War", "WHERE Name '%son%" contains names that contains the string "son". See "Wildcard Searches" on page 896. The NOT operator can precede LIKE to negate the search term. For example "Title NOT LIKE 'Assistant%'".   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| <i>searchcondition</i>   | <p>Expression that specifies a subset of the rows in the table or tables. Must evaluate to a Boolean whose value is True for the desired rows. If <i>tablename</i> refers to multiple tables, use the syntax <i>tablename.fieldname</i> in <i>searchcondition</i>, i.e., Customers.Customer_ID=Invoices.Customer_ID.</p> <p><i>Searchcondition</i> may use the following comparison symbols:</p> <table> <thead> <tr> <th><b>Comparison</b></th> <th><b>Symbol</b></th> </tr> </thead> <tbody> <tr> <td>Equality</td> <td>=</td> </tr> <tr> <td>Not equal</td> <td>&lt;&gt;</td> </tr> <tr> <td>Less than</td> <td>&lt;</td> </tr> <tr> <td>Less than or equal to</td> <td>&lt;=</td> </tr> <tr> <td>Greater Than</td> <td>&gt;</td> </tr> <tr> <td>Greater Than or equal to</td> <td>&gt;=</td> </tr> </tbody> </table> <p>You use single quote marks in the <i>searchcondition</i>. You can create a compound <i>searchcondition</i> using the OR, AND, and NOT conjunctions, e.g. "Customers.Customer_ID=56 AND Invoices.Customer_ID=56".</p> | <b>Comparison</b> | <b>Symbol</b> | Equality | = | Not equal | <> | Less than | < | Less than or equal to | <= | Greater Than | > | Greater Than or equal to | >= |
| <b>Comparison</b>        | <b>Symbol</b>  |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| Equality                 | =  |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| Not equal                | <>   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| Less than                | <  |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| Less than or equal to    | <=   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| Greater Than             | >  |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| Greater Than or equal to | >=   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| <i>groupingcondition</i> | Field or fields on which you wish to group the rows in the recordset. If you use more than one field, separate the fieldnames by commas. If a GROUP BY clause is used, the rows are organized in groups defined by the fields in this clause.  |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |
| <i>filtercondition</i>   | If the optional GROUP BY clause is included in the SELECT statement, you can optionally include a HAVING condition which includes only specified values of the GROUP BY fields. The HAVING clause filters rows created by the GROUP BY specification. For example, "GROUP By Movies.Director HAVING Movies.Director='Kubrick' OR Movies.Director='Fellini' OR Movies.Director='Wood'". This statement groups the records by director but includes only three directors, Kubrick, Fellini, and Wood. The HAVING clause must compare the value of a GROUP BY column to a constant value.   |                   |               |          |   |           |    |           |   |                       |    |              |   |                          |    |

| Part          | Description  |
|---------------|--|
| sortcondition | Field or fields on which you wish to sort the individual rows in the recordset. Use ORDER BY rather than GROUP BY if the sortCondition is unlikely to define groups, such as Last Name. By default, the rows are sorted in ascending order. If you wish to use descending order, include the modifier DESC, i.e., 'ORDER BY invoices.date DESC'. |

## Notes

Entire books have been written about the SQL SELECT statement, especially on the topic of relational database operations.

## Wildcard Searches

The REAL database supports the two SQL wildcard characters, percent, '%', and the underscore character, '\_'. The percent wildcard stands for any number of characters and the underscore stands for any single character. Placing the % at the beginning or the end of the search string does 'ends with' and 'begins with' searches, respectively. Using the % on both sides of the search string specifies a 'contains' search. For example,

| SELECT statement                                  | Search Type |
|---|-------------|
| SELECT * from authors where au_lname LIKE "Jon%"  | Begins with |
| SELECT * from authors where au_lname LIKE "%son"  | Ends with   |
| SELECT * from authors where au_lname LIKE "%man%" | Contains    |

The search criterion is case-sensitive.

The underscore character is the wildcard character for any single character. For example,

SELECT \* from authors where au\_lname LIKE "B\_r"

Returns "Bar", "Bor", etc.

## SELECT Examples

You do relational operations (e.g., "joins") by specifying the tables to be joined in the SELECT statement's *tablelist* and indicating in the WHERE clause how the tables are to be joined, i.e., rows in the 'many' table whose foreign key matches the primary key in the 'one' table. Please refer to a SQL reference book for detailed information on relational operations.

A join retrieves data from two or more tables. A join usually uses a WHERE clause that specifies the rows to be included in the recordset. The WHERE clause usually refers to primary and/or foreign key fields in the tables being joined. The most common type of join retrieves one or more rows in one table that are logically linked to each row in a different table.

Consider the following simple tables, Movies and Actors

### Movies

| RowID | Title     | Director     | Year |
|-------|-----------|--------------|------|
| 1     | Star Wars | George Lucas | 1977 |

## Movies

| RowID | Title                   | Director        | Year |
|-------|-------------------------|-----------------|------|
| 2     | Raiders of the Lost Ark | George Lucas    | 1981 |
| 3     | American Graffiti       | George Lucas    | 1973 |
| 4     | Apollo 13               | Ron Howard      | 1995 |
| 5     | Cast Away               | Robert Zemeckis | 2000 |

## Actors

| RowID | Name             | Movie |
|-------|------------------|-------|
| 1     | Harrison Ford    | 1     |
| 2     | Mark Hamill      | 1     |
| 3     | Carrie Fisher    | 1     |
| 4     | Harrison Ford    | 2     |
| 5     | Ron Howard       | 3     |
| 6     | Richard Dreyfuss | 3     |
| 7     | Tom Hanks        | 4     |
| 8     | Tom Hanks        | 5     |

The third column in the Actors table identifies a movie in which the actor appeared.

With this setup, the following SELECT statements can be used to query the database.

**Simple Selection** These statements select all records and all columns in a table:

```
Select * from Movies  
Select * from Actors
```

Each statement returns the tables shown above.

The following statement uses the Distinct lists all the actors only once:

```
Select distinct Name from Actors
```

It returns the following RecordSet:

| Name             |
|------------------|
| Harrison Ford    |
| Mark Hamill      |
| Carrie Fisher    |
| Ron Howard       |
| Richard Dreyfuss |
| Tom Hanks        |

## Relational Operations

Here are some examples of queries that use the WHERE and GROUP clauses to extract data from both tables.

Find all actors in 'Star Wars':

Each table is denoted by a letter and the WHERE clause specifies the records to be retrieved. It specifies that the Title column must equal 'Star Wars' and the rows in the Actors table must have the value of the Movie field equal to the RowID column in Movies that has the 'Star Wars' record. This is RowID=1. That means the matching rows in Actors are the first three rows.

```
Select a.name from Actors A, Movies M where a.movie=m._rowid and m.title='Star Wars'
```

The Select statement specifies that only the Actors.Name column be retrieved, so the result is:

| Name          |
|---------------|
| Harrison Ford |
| Mark Hamill   |
| Carrie Fisher |

Find all movies directed by Lucas and report the actor's name, the movie's name, and the year:

This example differs from the previous only in that the columns to be returned are from both the Actors and Movies tables:

```
Select A.Name,M.Title,M.Year from Actors A, Movies M where A.movie=M._rowid and M.Director='George Lucas'
```

The WHERE clause works the same way as the previous example. It selects actors whose Movie column matches the Movie table's RowID for Lucas-directed flicks. In this example, the column list is from both tables. The result is this:

| Name             | Movie                   | Year |
|------------------|-------------------------|------|
| Ron Howard       | American Graffiti       | 1973 |
| Richard Dreyfuss | American Graffiti       | 1973 |
| Harrison Ford    | Raiders of the Lost Ark | 1981 |
| Harrison Ford    | Star Wars               | 1977 |
| Mark Hamill      | Star Wars               | 1977 |
| Carrie Fisher    | Star Wars               | 1977 |

Find all actors who have done a movie with Harrison Ford:

```
Select A.name from Actors A, Actors B where A.movie = B.movie and B.name='Harrison Ford' and A.name<>'Harrison Ford'
```

---

The result is:

| Name          |
|---------------|
| Mark Hamill   |
| Carrie Fisher |

Find anyone who is both an actor and a director:

```
Select A.Name from Actors A, Movies M where A.Name = M.Director
```

The result is:

| Name       |
|------------|
| Ron Howard |

**Wildcard Search** Find any movies with the word “War” in the title (ignoring case):

```
Select title, year from Movies where Upper(title) like '%WAR%'
```

The result is:

| Title     | Year |
|-----------|------|
| Star Wars | 1977 |

**Aggregate Functions** Find out how many actors we have for each movie (by title):

```
Select A.Title, Count(B.Name) from Movies A, Actors B where B.Movie = A._rowID group by B.Movie
```

The result is:

| Title                   | Count(Name) |
|-------------------------|-------------|
| Star Wars               | 3           |
| Raiders of the Lost Ark | 1           |
| American Graffiti       | 2           |
| Apollo 13               | 1           |
| Cast Away               | 1           |

Find how when each director made his first and last movie (in this database).

```
Select Director, Min(Year), Max(Year) from Movies Group by Director
```

The result is:

| Director        | Min(Year) | Max(Year) |
|-----------------|-----------|-----------|
| George Lucas    | 1973      | 1981      |
| Robert Zemeckis | 2000      | 2000      |

---

| Director   | Min(Year) | Max(Year) |
|------------|-----------|-----------|
| Ron Howard | 1995      | 1995      |

Find how many movies each actor has done, but display only those who have done more than one.

Select Name, Count(\*) from Actors group by Name having Count(\*) > 1

The result is

| Name          | Count |
|---------------|-------|
| Harrison Ford | 2     |
| Tom Hanks     | 2     |

---

## UPDATE Statement

Modifies existing data in a table.

### Syntax

**UPDATE *tablename* SET *fieldname1=expression1,... fieldnameN=expressionN* WHERE *searchcondition***

| Argument               | Description   |
|------------------------|---|
| <i>tablename</i>       | Name of table containing fields to be updated.  |
| <i>fieldname</i>       | Field in <i>TableName</i>   |
| <i>expression</i>      | Value to be assigned to <i>FieldName</i> . The expression can be a value or either of the keywords DEFAULT or NULL. |
| <i>searchcondition</i> | Boolean expression that identifies the row or rows to be updated.   |

### Notes

Call the Commit command or the Database class's Commit method to save your updates or call Rollback to reset the database to its state prior to the update.

You can also perform updates using the Edit and Update methods of the RecordSet class within the REALbasic language. See the section "Editing Records" on page 495 for more information

---

## Example

The following example changes the value of the Year field to 1976 for the movie with the title "Rabid."

```
Dim dbFile as FolderItem
Dim db as RealDatabase
db=New RealDatabase
dbFile=New FolderItem("myMovies")
db.databaseFile=dbFile
If db.error then
    MsgBox db.errormessage
else
    db.sqlexecute ("UPDATE Movies SET Year=1976 WHERE title='Rabid'")
    If db.error then
        MsgBox db.errormessage
    Else
        db.Commit
        MsgBox "Records updated successfully."
    End if
End if
```

---

## SQL Functions

---

### BIT\_LENGTH Function

Returns the number of bits in a character or bit string.

#### Syntax

***result= BIT\_LENGTH(String)***

| Part          | Type    | Description                              |
|---------------|---------|--|
| <i>result</i> | Integer | Number of bits in <i>String</i> .        |
| <i>String</i> | String  | Character or bit string to be evaluated. |

#### Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT BIT_LENGTH('zz')) //returns 16
rs=db.SQLSelect ("SELECTBIT_LENGTH('cat')") /returns 24
```

---

### CHAR\_LENGTH Function

Returns the number of characters in a string.

---

## Syntax

**result=CHAR\_LENGTH(*String*)**

| Part          | Type    | Description                       |
|---------------|---------|-----------------------------------|
| <i>result</i> | Integer | Number of bits in <i>String</i> . |
| <i>String</i> | String  | Character string to be evaluated. |

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT CHAR_LENGTH('Madison Ave.')") returns 12.
rs=db.SQLSelect ("SELECT CHAR_LENGTH('480')") //returns 3.
```

## See Also

[BIT\\_LENGTH](#), [OCTET\\_LENGTH](#).

---

## LOWER Function

Returns the string passed in all lowercase text.

## Syntax

**result=LOWER(*String*)**

| Part          | Type   | Description                         |
|---------------|--------|-------------------------------------|
| <i>result</i> | String | The passed string in all lowercase. |
| <i>String</i> | String | Character string to be evaluated.   |

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT LOWER('GREENHORNET')")
//returns "greenhornet".
```

## See Also

[UPPER](#).

---

## OCTET\_LENGTH Function

Returns the number of bytes in a character string or bit string expression.

## Syntax

**result=OCTET\_LENGTH (*string*)**

| Part          | Type    | Description                        |
|---------------|---------|------------------------------------|
| <i>result</i> | Integer | Number of bytes in <i>String</i> . |

---

|               |               |                                   |
|---------------|---------------|-----------------------------------|
| <i>String</i> | <i>String</i> | Character string to be evaluated. |
|---------------|---------------|-----------------------------------|

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
rs=db.SQLSelect ("SELECT OCTET_LENGTH ('F')") //returns 1.
rs=db.SQLSelect ("SELECT OCTET_LENGTH ('FF')") //returns 2.
```

**See Also** [BIT\\_LENGTH](#), [CHAR\\_LENGTH](#).

---

## POSITION Function

Searches for the specified target string within the source string and returns the character position where the target string starts.

**Syntax** ***result=POSITION (targetString IN sourceString)***

| Part                | Type    | Description   |
|---------------------|---------|---|
| <i>result</i>       | Integer | Character position where <i>targetString</i> first occurs in <i>sourceString</i> . If <i>targetString</i> is not found, <i>result</i> is set to zero. |
| <i>targetString</i> | String  | Character string to search for.   |
| <i>sourceString</i> | String  | Character string to be evaluated  |

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
rs=db.SQLSelect ("SELECT POSITION ('G' IN 'Fred Gorman')") //returns 6.
```

**See Also** [SUBSTRING](#), [TRIM](#).

---

## SUBSTRING Function

Returns the specified substring of characters from the string passed.

**Syntax** ***result=SUBSTRING (targetString FROM start [FOR length])***

| Part          | Type   | Description  |
|---------------|--------|--|
| <i>result</i> | String | The string of characters extracted from <i>targetString</i> by position. |

|                     |         |  |
|---------------------|---------|--|
| <i>targetString</i> | String  | Character string from which to extract a substring.  |
| <i>Start</i>        | Integer | Starting character position in <i>targetString</i> from which to extract the substring.  |
| <i>Length</i>       | Integer | Optional parameter indicating the number of characters following <i>Start</i> to extract. If <i>Length</i> is omitted, all the characters from <i>Start</i> to the end of <i>targetString</i> will be extracted. |

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
.
.
rs=db.SQLSelect ("SELECT SUBSTRING('Samantha' FROM 1 For 3)")
//returns Sam'.
```

## See Also

[POSITION](#), [TRIM](#).

# TRIM Function

Trims leading and/or trailing characters from the string passed.

## Syntax

**result=TRIM (LEADING|TRAILING|BOTH [trimstring] FROM targetString)**

| Part                | Type   | Description  |
|---------------------|--------|--|
| <i>result</i>       | String | The string of characters after trimming <i>targetString</i> by removing leading and/or trailing instances of <i>trimString</i> . |
| <i>trimString</i>   | String | Optional. Character to remove from start or end of <i>targetString</i> . If omitted, the space is assumed.                       |
| <i>targetString</i> | String | String from which to trim characters.  |

## Notes

TRIM requires that you specify that you want to trim Leading or Trailing instances of the trimString (or both) and allows you to specify the string to be trimmed. The From keyword is required if either the trimstring or the Leading/Trailing/Both options are specified.

## Examples

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
.
.
rs=db.SQLSelect(TRIM (BOTH from ' The Matrix ')) //returns 'The Matrix'.
rs=db.SQLSelect(TRIM (Leading ' ' from ' Matrix')) //areturns 'Matrix'.
```

---

## See Also

[POSITION](#), [SUBSTRING](#).

---

## UPPER Function

Returns the string passed in all uppercase text.

### Syntax

***result=UPPER (String)***

| Part          | Type   | Description                         |
|---------------|--------|-------------------------------------|
| <i>result</i> | String | The passed string in all uppercase. |
| <i>String</i> | String | Character string to be evaluated.   |

### Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
.
rs=db.SQLSelect ("SELECT UPPER('Grand Blanc')")
//returns "GRAND BLANC".
```

---

## See Also

[LOWER](#).

---

## CURRENT\_DATE Function

Returns the current date in the SQL Date format, YYYY-MM-DD.

### Syntax

***result=CURRENT\_DATE***

| Part          | Type    | Description                                |
|---------------|---------|--|
| <i>result</i> | SQLDate | The current date in the YYYY-MM-DD format. |

### Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
.
rs=db.SQLSelect ("SELECT CURRENT_DATE") //returns '2003-10-07'
```

---

## CURRENT\_TIME Function

Returns the current time in the SQL Time format, HH:MM:SS.

---

## Syntax

***result=CURRENT\_TIME***

| Part          | Type    | Description                                       |
|---------------|---------|---|
| <i>result</i> | SQLTime | The current time in the HH:MM:SS SQL Time format. |

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
rs=db.SQLSelect ("SELECT CURRENT_TIME") //returns '13:51:30', for example
```

## See Also

[CURRENT\\_TIMESTAMP](#).

---

## CURRENT\_TIMESTAMP Function

Returns the current date-time in the SQLTimestamp format, YYYY-MM-DD HH:MM:SS.

## Syntax

***result=CURRENT\_TIMESTAMP***

| Part          | Type         | Description   |
|---------------|--------------|---|
| <i>result</i> | SQLTimestamp | The current date/ time in the YYYY-MM-DD HH:MM:SS SQL Timestamp format. |

## Example

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
rs=db.SQLSelect ("SELECT CURRENT_TIMESTAMP")
//returns '2003-10-07 13:52:57' for example
```

## See Also

[CURRENT\\_TIME](#).

---

## LAST\_ROWID Function

Returns the value of the last \_rowID for the table passed. This value is useful for certain types of relational joins. The REALdatabase automatically adds a row\_ID field to every table as a unique identifier. Values for this field are assigned automatically unless values are assigned by the user.

---

## Syntax

**result=LAST\_ROWID (*tablename*)**

| Part             | Type    | Description   |
|------------------|---------|---|
| <i>result</i>    | Integer | The last value of the _rowID column that has been added to the specified table. |
| <i>tablename</i> | String  | The name of the table for which you want the last _rowID.                       |

## Example

```
Dim db as New REALdatabase  
Dim rs as New RecordSet  
rs=db.SQLSelect ("SELECT LAST_ROWID ('Employees')")
```

# Aggregate Functions

## AVG Function

Computes the average value of an expression across rows.

## Syntax

**result=AVG [DISTINCT|ALL] (*expression*)**

| Part                    | Type    | Description   |
|-------------------------|---------|---|
| <i>result</i>           | Double  | The average of <i>expression</i> across the rows specified by <i>searchExpression</i> . |
| <i>expression</i>       | Numeric | Column or function of one or more columns evaluates to a numeric data type.             |
| <i>SearchExpression</i> |         | SQL WHERE clause that selects the rows that are evaluated.                              |

If you use the DISTINCT keyword, only unique values of *expression* will be included in the calculation. If you use ALL, all values of *expression* will be included, including duplicate ones. The default is ALL.

## Example

```
Dim db as New REALdatabase  
Dim rs as New RecordSet  
. . .  
rs=db.SQLSelect("Select AVG (Population) FROM Cities")  
rs=db.SQLSelect("Select AVG (Income) FROM Cities WHERE State='CA'")
```

## See Also

[COUNT](#), [SUM](#).

# COUNT Function

Computes the number of rows in an expression.

## Syntax

**result=COUNT [DISTINCT] expression FROM searchExpression**

| Part                    | Type    | Description   |
|-------------------------|---------|---|
| result                  | Integer | The total of <i>expression</i> across the rows specified by <i>searchExpression</i> . |
| <i>expression</i>       | Any     | Column or function of one or more columns.  |
| <i>SearchExpression</i> |         | SQL WHERE clause that selects the rows that are evaluated.                            |

If you use the DISTINCT keyword, only unique values of *expression* will be included in the calculation. If you use \* as the expression, all rows will be counted.

When *expression* is specified, COUNT returns the number of rows for the columns in *expression*.

## Example

This example returns the number of rows in the Movies table with non-missing values in the Title and Director fields. The second example counts the rows with non-missing values for Title in which the Director field equals 'Lucas'.

```
Dim db as New REALdatabase
Dim rs as New RecordSet
rs=db.SQLSelect ("SELECT COUNT (DISTINCT Director) from Movies")
rs=db.SQLSelect ("Select Count (*) from Movies WHERE Director = 'Lucas'")
rs=db.SQLSelect ("Select Count (*) from Movies")
```

## See Also

[AVG](#), [SUM](#).

# MAX Function

Returns the maximum value of an expression.

## Syntax

**result=MAX expression FROM searchExpression**

| Part                    | Type              | Description   |
|-------------------------|-------------------|---|
| result                  | Double            | The maximum value of <i>expression</i> across the rows specified by <i>searchExpression</i> . |
| <i>expression</i>       | Numeric or String | Column or function of one or more columns that evaluates to a numeric or string expression.   |
| <i>SearchExpression</i> |                   | SQL WHERE clause that selects the rows that are evaluated.                                    |

---

**Example**

This example finds the largest population in the Cities table.

```
Dim db as New REALdatabase  
Dim rs as New RecordSet  
  
rs=db.SQLSelect("SELECT MAX (Population) from Cities")
```

**See Also**

[MIN](#).

---

## MIN Function

Returns the minimum value of the specified column or columns.

**Syntax**

***result=MIN expression FROM searchExpression***

| Part                    | Type              | Description   |
|-------------------------|-------------------|---|
| <i>result</i>           | Double            | The maximum value of <i>expression</i> across the rows specified by <i>searchExpression</i> . |
| <i>expression</i>       | Numeric or String | Column or function of one or more columns that evaluates to a numeric or string expression.   |
| <i>SearchExpression</i> |                   | SQL WHERE clause that selects the rows that are evaluated.                                    |

**Example**

This example finds the minimum Price in the Products table.

```
Dim db as New REALdatabase  
Dim rs as New RecordSet  
  
rs=db.SQLSelect("SELECT MIN (Price) from Products")
```

**See Also**

[MAX](#).

---

## SUM Function

Returns the total of the specified numeric fields.

**Syntax**

***result=SUM [DISTINCT|ALL] expression FROM searchExpression***

| Part | Type | Description |
|------|------|-------------|
|      |      |             |

|                         |         |   |
|-------------------------|---------|---|
| <i>result</i>           | Double  | The maximum value of <i>expression</i> across the rows specified by <i>searchExpression</i> . |
| <i>expression</i>       | Numeric | Column or function of one or more columns that evaluates to a numeric expression.             |
| <i>SearchExpression</i> |         | SQL WHERE clause that selects the rows that are evaluated.                                    |

If you use the DISTINCT keyword, only unique values of *expression* will be included in the calculation. If you use ALL, all values of *expression* will be included, including duplicate ones. The default is ALL.

## Notes

*Expression* must evaluate to a numeric data type. It can include one or more columns. The DISTINCT keyword can be used to limit the evaluation to distinct (unique) occurrences of each value.

## Example

This example finds the total population from the States table for States in the NE region only.

```
Dim db as New REALdatabase
Dim rs as New RecordSet
.
rs=db.SQLSelect("SELECT SUM (Population) from States WHERE Region='NE'"")
```

## See Also

[AVG](#), [COUNT](#).

# Alphabetical Command List

|  |    |
|--|----|
| - Operator . . . . .   | 6  |
| " " Literal . . . . .  | 6  |
| #If...#Endif Statement . . . . .   | 7  |
| #Pragma Directives . . . . .   | 8  |
| &b Literal . . . . .   | 10 |
| &c Literal . . . . .   | 10 |
| &h Literal . . . . .   | 11 |
| &o Literal . . . . .   | 11 |
| ' Operator . . . . .   | 12 |
| * Operator . . . . .   | 13 |
| + Operator . . . . .   | 13 |
| / Operator . . . . .   | 14 |
| // Operator . . . . .  | 14 |
| < Operator . . . . .   | 15 |
| <= Operator . . . . .  | 16 |
| <> Operator . . . . .  | 17 |
| = Operator . . . . .   | 18 |
| > Operator . . . . .   | 19 |
| >= Operator . . . . .  | 20 |
| \ Operator . . . . .   | 21 |
| ^ Operator . . . . .   | 22 |
| _ Keyword . . . . .  | 22 |
| A Method with a ParamArray Cannot have any Optional Parameters Error . . . . . | 23 |
| A ParamArray cannot be Passed by Reference Error . . . . .                     | 23 |
| A ParamArray cannot have a Default Value Error . . . . .                       | 23 |
| A ParamArray's Element Type may not be Another Array Error . . . . .           | 24 |
| A Parameter passed by Reference Cannot have a Default Value Error . . . . .    | 24 |
| Abs Function . . . . .   | 24 |
| Acos Function . . . . .  | 25 |
| AddressBook Class . . . . .  | 25 |
| AddressBookAddress Class . . . . .   | 27 |
| AddressBookContact Class . . . . .   | 28 |
| AddressBookData Class . . . . .  | 30 |
| AddressBookGroup Class . . . . .   | 32 |
| AddressBookRecord Class . . . . .  | 34 |
| AddressOf Operator . . . . .   | 34 |
| An Assigns Parameter cannot have a Default Value Error . . . . .               | 35 |
| An Extends Parameter cannot have a Default Value Error . . . . .               | 35 |
| An Ordinary Parameter cannot Follow a ParamArray Error . . . . .               | 36 |
| And Operator . . . . .   | 36 |
| App Function . . . . .   | 37 |
| Append Method . . . . .  | 37 |

## Alphabetical Command List

---

|  |    |
|--|----|
| AppleEvent Class . . . . .                                     | 38 |
| AppleEventDescList Class . . . . .                             | 42 |
| AppleEventObjectSpecifier Class . . . . .                      | 43 |
| AppleEventRecord Class . . . . .                               | 44 |
| AppleEventTarget Class . . . . .                               | 45 |
| AppleEventTemplate Class . . . . .                             | 46 |
| AppleMenuItem Class . . . . .                                  | 46 |
| Application Class . . . . .                                    | 47 |
| ApplicationSupportFolder Function. . . . .                     | 53 |
| ArcShape Class . . . . .                                       | 55 |
| Array Bounds must be Integers Error . . . . .                  | 57 |
| Array Function . . . . .                                       | 55 |
| Asc Function . . . . .   | 57 |
| AscB Function . . . . .  | 58 |
| Asin Function. . . . .   | 59 |
| Assigns can only be used on the last parameter Error . . . . . | 60 |
| Assigns Keyword . . . . .                                      | 60 |
| Atan Function . . . . .  | 61 |
| Atan2 Function. . . . .  | 61 |
| AutoDiscovery Class . . . . .                                  | 62 |
| Beep Method. . . . .   | 64 |
| BevelButton Control. . . . .                                   | 64 |
| Bin Function . . . . .   | 68 |
| BinaryStream Class . . . . .                                   | 69 |
| Bitwise Class . . . . .  | 73 |
| Boolean Data Type. . . . .                                     | 75 |
| Bounds3D Class. . . . .  | 76 |
| Break Keyword. . . . .   | 77 |
| ByRef Keyword. . . . .   | 77 |
| Byte Data Type. . . . .  | 79 |
| ByVal Keyword. . . . .   | 79 |
| Call Statement . . . . .                                       | 80 |
| Cannot Assign a Value to this Property Error. . . . .          | 80 |
| Cannot Get this Property's Value Error . . . . .               | 81 |
| Canvas Control . . . . .                                       | 81 |
| Catch Statement . . . . .                                      | 85 |
| CDbl Function . . . . .  | 86 |
| Ceil Function . . . . .  | 86 |
| CFStringRef Data Type. . . . .                                 | 87 |
| Checkbox Control . . . . .                                     | 87 |
| Chr Function . . . . .   | 88 |
| ChrB Function . . . . .  | 89 |
| ClearFocus Method . . . . .                                    | 90 |
| Clipboard Class. . . . .                                       | 91 |
| CLong Function . . . . .                                       | 92 |
| CMY Function . . . . .   | 93 |

|   |     |
|---|-----|
| Collection Class . . . . .                            | 94  |
| Color Data Type . . . . .                             | 95  |
| ColorList Class . . . . .                             | 97  |
| ComboBox Control. . . . .                             | 98  |
| ConsoleApplication Class . . . . .                    | 100 |
| Const Statement . . . . .                             | 102 |
| Constructor Method . . . . .                          | 103 |
| ContainerControl Class . . . . .                      | 104 |
| ContextualMenu Control . . . . .                      | 108 |
| Continue Statement . . . . .                          | 110 |
| Control Class . . . . .                               | 111 |
| ConvertEncoding Function . . . . .                    | 112 |
| Cos Function . . . . .                                | 113 |
| CountFields Function . . . . .                        | 114 |
| CountFieldsB Function. . . . .                        | 114 |
| CriticalSection Class . . . . .                       | 115 |
| CStr Function . . . . .                               | 117 |
| CString Data Type . . . . .                           | 117 |
| Cursors Object . . . . .                              | 117 |
| CurveShape Class. . . . .                             | 119 |
| DarkBevelColor Function . . . . .                     | 120 |
| DarkTingeColor Function . . . . .                     | 121 |
| DataAvailableProvider Interface Class . . . . .       | 121 |
| Database Class . . . . .                              | 121 |
| Database4DServer Class . . . . .                      | 128 |
| DatabaseField Class . . . . .                         | 129 |
| DatabaseQuery Control . . . . .                       | 131 |
| DatabaseRecord Class . . . . .                        | 132 |
| DataControl Control. . . . .                          | 134 |
| Datagram Class. . . . .                               | 139 |
| DataNotificationReceiver Interface Class . . . . .    | 140 |
| DataNotifier Interface Class . . . . .                | 140 |
| Date Class. . . . .                                   | 140 |
| DebugBuild Constant . . . . .                         | 144 |
| DebugDumpObjects Method . . . . .                     | 144 |
| Declaration: Contains a Reserved Word Error . . . . . | 145 |
| Declare Statement . . . . .                           | 145 |
| DecodeBase64 Function . . . . .                       | 152 |
| DecodeQuotedPrintable Function . . . . .              | 152 |
| DecodeURLComponent Function . . . . .                 | 153 |
| DefineEncoding Function . . . . .                     | 153 |
| DesktopFolder Function . . . . .                      | 154 |
| Destructor Method. . . . .                            | 155 |
| Destructors Can't Have Parameters Error . . . . .     | 156 |
| Dictionary Class . . . . .                            | 156 |
| Dim Statement . . . . .                               | 158 |

## Alphabetical Command List

---

|   |     |
|---|-----|
| DisclosureTriangle Control . . . . .  | 162 |
| Do...Loop Statement. . . . .  | 164 |
| DockItem Class . . . . .  | 162 |
| DocumentsFolder Function . . . . .  | 163 |
| Double Data Type . . . . .  | 166 |
| DragItem Class . . . . .  | 167 |
| EasyTCPSocket Class . . . . .   | 171 |
| EasyUDPSocket Class . . . . .   | 173 |
| EditableMovie Class . . . . .   | 176 |
| EditField Control . . . . .   | 181 |
| Element3D Class . . . . .   | 192 |
| EmailAttachment Class . . . . .   | 194 |
| EmailHeaders Class. . . . .   | 195 |
| EmailMessage Class . . . . .  | 196 |
| EnableMenuItems Method . . . . .  | 198 |
| EncodeBase64 Function . . . . .   | 198 |
| EncodeQuotedPrintable Function. . . . .                                       | 199 |
| EncodeURLComponent Function . . . . .   | 199 |
| Encoding Function . . . . .   | 200 |
| Encodings Object. . . . .   | 201 |
| End Quote Missing Error. . . . .  | 202 |
| EndOfLine Class . . . . .   | 202 |
| ExcelApplication Class . . . . .  | 203 |
| Exception Block . . . . .   | 204 |
| Exception Objects Must be of Type RuntimeException Error . . . . .            | 208 |
| Exit Statement . . . . .  | 208 |
| Exp Function . . . . .  | 209 |
| ExportPicture Function . . . . .  | 209 |
| Extends can only be used on the first parameter Error . . . . .               | 212 |
| Extends Keyword. . . . .  | 211 |
| External Functions Cannot Use Objects as Parameters Error. . . . .            | 212 |
| External Functions Cannot Use String Data Types as Parameters Error . . . . . | 213 |
| False Keyword . . . . .   | 213 |
| FigureShape Class . . . . .   | 213 |
| FileType Class. . . . .   | 215 |
| FillColor Function . . . . .  | 217 |
| Finally Block . . . . .   | 217 |
| Floor Function . . . . .  | 218 |
| FolderItem Class . . . . .  | 218 |
| FolderItemDialog Class . . . . .  | 233 |
| Font Function . . . . .   | 235 |
| FontCount Function . . . . .  | 235 |
| FontsFolder Function . . . . .  | 236 |
| For Each...Next Statement. . . . .  | 237 |
| For...Next Statement. . . . .   | 238 |
| Format Function . . . . .   | 240 |

|  |     |
|--|-----|
| FrameColor Function . . . . .                          | 242 |
| Function Statement . . . . .                           | 242 |
| FunctionNotFoundException . . . . .                    | 245 |
| GameInputDevice Class . . . . .                        | 245 |
| GameInputElement Class . . . . .                       | 246 |
| GameInputManager Class . . . . .                       | 246 |
| GetAppleEventTarget Function . . . . .                 | 249 |
| GetFolderItem Function . . . . .                       | 249 |
| GetFontTextEncoding Function . . . . .                 | 251 |
| GetIndexedObjectDescriptor Function . . . . .          | 252 |
| GetInternetTextEncoding Function . . . . .             | 252 |
| GetNamedObjectDescriptor Function . . . . .            | 253 |
| GetOpenFolderItem Function . . . . .                   | 253 |
| GetOrdinalObjectDescriptor Function . . . . .          | 255 |
| GetPropertyObjectDescriptor Function . . . . .         | 256 |
| GetQTCrossFadeEffect Function . . . . .                | 256 |
| GetQTGraphicsExporter Function . . . . .               | 257 |
| GetQTSMPTEEffect Function . . . . .                    | 257 |
| GetRangeObjectDescriptor Function . . . . .            | 262 |
| GetSaveFolderItem Function . . . . .                   | 262 |
| GetStringComparisonObjectDescriptor Function . . . . . | 263 |
| GetTemporaryFolderItem Function . . . . .              | 264 |
| GetTestObjectDescriptor Function . . . . .             | 265 |
| GetTextConverter Function . . . . .                    | 265 |
| GetTextEncoding Function . . . . .                     | 266 |
| GetTrueFolderItem Function . . . . .                   | 267 |
| GetUniqueIDObjectDescriptor Function . . . . .         | 267 |
| GOTO Statement . . . . .                               | 268 |
| GOTO Target Not Found Error . . . . .                  | 269 |
| Graphics Class . . . . .                               | 269 |
| Group2D Class . . . . .                                | 275 |
| Group3D Class . . . . .                                | 276 |
| GroupBox Control . . . . .                             | 277 |
| GuessJapaneseEncoding Function . . . . .               | 278 |
| Hex Function . . . . .                                 | 278 |
| HighlightColor Function . . . . .                      | 279 |
| HSV Function . . . . .                                 | 279 |
| HTMLViewer Control . . . . .                           | 280 |
| HTTPSecureSocket Class . . . . .                       | 283 |
| HTTPSocket Class . . . . .                             | 286 |
| If...Then...Else Statement . . . . .                   | 290 |
| IllegalCastException Runtime Error . . . . .           | 292 |
| ImageWell Control . . . . .                            | 294 |
| IndexOf Method . . . . .                               | 295 |
| Input Method . . . . .                                 | 296 |
| Insert Method . . . . .                                | 296 |

## Alphabetical Command List

---

|  |     |
|--|-----|
| InStr Function . . . . .                                   | 297 |
| InStrB Function. . . . .                                   | 298 |
| Int16 Data Type . . . . .                                  | 299 |
| Int32 Data Type . . . . .                                  | 300 |
| Int64 Data Type . . . . .                                  | 300 |
| Int8 Data Type . . . . .                                   | 301 |
| Integer Data Type . . . . .                                | 302 |
| InternetHeaders Class . . . . .                            | 302 |
| InvalidParentException Error . . . . .                     | 303 |
| IPCSocket Class . . . . .                                  | 304 |
| Is Operator . . . . .                                      | 307 |
| IsA Operator . . . . .                                     | 307 |
| IsContextualClick Function . . . . .                       | 310 |
| IsNumeric Function. . . . .                                | 311 |
| Join Function . . . . .                                    | 312 |
| Keyboard Object. . . . .                                   | 312 |
| KeyChain Class . . . . .                                   | 316 |
| KeyChainException Error . . . . .                          | 319 |
| KeyChainItem Class . . . . .                               | 321 |
| KeyNotFoundException Error . . . . .                       | 323 |
| Left Function . . . . .                                    | 323 |
| LeftB Function . . . . .                                   | 324 |
| Len Function . . . . .                                     | 325 |
| LenB Function . . . . .                                    | 325 |
| Light3D Class . . . . .                                    | 326 |
| LightBevelColor Function . . . . .                         | 327 |
| LightTingeColor Function . . . . .                         | 327 |
| Line Control . . . . .                                     | 328 |
| ListBox Control. . . . .                                   | 329 |
| ListColumn Class . . . . .                                 | 357 |
| ListSelectionNotificationReceiver Class Interface. . . . . | 359 |
| ListSelectionNotifier Class Interface. . . . .             | 359 |
| Log Function . . . . .                                     | 360 |
| Logic Operations require Boolean Values Error . . . . .    | 360 |
| Lowercase Function . . . . .                               | 360 |
| LTrim Function . . . . .                                   | 361 |
| Material Class . . . . .                                   | 361 |
| Max Function. . . . .                                      | 362 |
| MD5 Function . . . . .                                     | 363 |
| MD5Digest Class . . . . .                                  | 363 |
| MDIWindow Class . . . . .                                  | 364 |
| Me Cannot be used in a Method of a Module Error . . . . .  | 368 |
| Me Keyword . . . . .                                       | 367 |
| MemoryBlock Class. . . . .                                 | 369 |
| MenuItem Class . . . . .                                   | 372 |
| MessageDialog Class. . . . .                               | 377 |

|  |     |
|--|-----|
| MessageDialogButton Class . . . . .  | 380 |
| Microseconds Function . . . . .  | 381 |
| Mid Function . . . . .   | 382 |
| MidB Function . . . . .  | 383 |
| Min Function . . . . .   | 383 |
| Mod Operator . . . . .   | 384 |
| MouseCursor Class . . . . .  | 385 |
| Movie Class . . . . .  | 386 |
| MoviePlayer Control . . . . .  | 387 |
| MsgBox Function. . . . .   | 392 |
| Mutex Class. . . . .   | 395 |
| MySQLDatabase Class . . . . .  | 396 |
| Network Object . . . . .   | 397 |
| NetworkInterface Object . . . . .  | 398 |
| New Operator . . . . .   | 399 |
| NewAppleEvent Function . . . . .   | 402 |
| NewAppleEventTarget Function . . . . .   | 402 |
| NewMemoryBlock Function . . . . .  | 403 |
| NewPicture Function. . . . .   | 404 |
| Nil Keyword . . . . .  | 406 |
| NilObjectException Runtime Error . . . . .   | 407 |
| NoOpenTransportException Error. . . . .  | 408 |
| Not Operator. . . . .  | 409 |
| NotePlayer Control. . . . .  | 409 |
| NthField Function . . . . .  | 412 |
| NthFieldB Function. . . . .  | 413 |
| Object Class. . . . .  | 414 |
| Object2D Class . . . . .   | 414 |
| Object3D Class . . . . .   | 416 |
| Oct Function . . . . .   | 426 |
| ODBCDatabase Class. . . . .  | 424 |
| Office Class . . . . .   | 426 |
| OLEContainer Control . . . . .   | 419 |
| OLEException Error . . . . .   | 420 |
| OLEObject Class . . . . .  | 420 |
| OLEParameter Class . . . . .   | 423 |
| One of the Interfaces of this Class is not of Type Class Interface Error . . . . . | 429 |
| Only Boolean Constants can be Used with #If Error . . . . .                        | 430 |
| Only One Parameter may use the Assigns Option Error . . . . .                      | 430 |
| Only One Parameter my use the ParamArray Option Error . . . . .                    | 430 |
| Only String Constants can be used for Declaring Libraries Error . . . . .          | 431 |
| Only the First Parameter may use the Extends Option Error . . . . .                | 431 |
| Only the Last Parameter may use the Assigns Option Error . . . . .                 | 431 |
| OpenBaseDatabase Class . . . . .   | 431 |
| OpenCSVCursor Function . . . . .   | 433 |
| OpenDBFCursor Function . . . . .   | 435 |

## Alphabetical Command List

---

|  |     |
|--|-----|
| OpenDialog Class . . . . .   | 433 |
| OpenDTFDatabase Function.  | 436 |
| OpenPrinter Function . . . . .                                       | 437 |
| OpenPrinterDialog Function . . . . .                                 | 437 |
| OpenURLMovie Function . . . . .                                      | 438 |
| Operator_Add Function . . . . .                                      | 439 |
| Operator_AddRight Function . . . . .                                 | 440 |
| Operator_And Function . . . . .                                      | 440 |
| Operator_AndRight Function . . . . .                                 | 441 |
| Operator_Compare Function . . . . .                                  | 441 |
| Operator_Convert Method . . . . .                                    | 442 |
| Operator_Divide Function . . . . .                                   | 443 |
| Operator_DivideRight Function . . . . .                              | 444 |
| Operator_IntegerDivide Function . . . . .                            | 444 |
| Operator_IntegerDivideRight Function. . . . .                        | 445 |
| Operator_Lookup Function . . . . .                                   | 445 |
| Operator_Modulo Function . . . . .                                   | 446 |
| Operator_ModuloRight Function . . . . .                              | 447 |
| Operator_Multiply Function. . . . .                                  | 447 |
| Operator_MultiplyRight Function. . . . .                             | 448 |
| Operator_Negate Function . . . . .                                   | 449 |
| Operator_Not Function . . . . .                                      | 449 |
| Operator_Or Function . . . . .                                       | 449 |
| Operator_OrRight Function . . . . .                                  | 450 |
| Operator_Power Function . . . . .                                    | 450 |
| Operator_PowerRight Function. . . . .                                | 451 |
| Operator_Subtract Function. . . . .                                  | 452 |
| Operator_SubtractRight Function. . . . .                             | 452 |
| Optional Keyword . . . . .   | 453 |
| Or Operator . . . . .  | 454 |
| OracleDatabase Class . . . . .                                       | 454 |
| OSType Data Type . . . . .   | 455 |
| OutOfBoundsException Error . . . . .                                 | 455 |
| OutOfMemoryException Error . . . . .                                 | 457 |
| Oval Control . . . . .   | 457 |
| OvalShape Class . . . . .  | 458 |
| PagePanel Control . . . . .  | 459 |
| Paragraph Class . . . . .  | 460 |
| ParamArray Keyword . . . . .   | 461 |
| Parameter and Default Value Must be of the Same Type Error . . . . . | 463 |
| Parameters are not Compatible with this Function Error . . . . .     | 463 |
| ParseDate Function . . . . .   | 463 |
| Permissions Class . . . . .  | 464 |
| Picture Class . . . . .  | 466 |
| PixmapShape Class . . . . .  | 468 |
| Placard Control. . . . .   | 469 |

|   |     |
|---|-----|
| Pop Method . . . . .                            | 471 |
| POP3SecureSocket Class . . . . .                | 471 |
| POP3Socket Class. . . . .                       | 475 |
| PopupArrow Control. . . . .                     | 479 |
| PopupMenu Control. . . . .                      | 480 |
| PostgreSQLDatabase Class. . . . .               | 483 |
| Pow Function. . . . .                           | 484 |
| PowerPointApplication Class . . . . .           | 484 |
| PreferencesFolder Function . . . . .            | 487 |
| PrefsMenuItem Class. . . . .                    | 488 |
| Print Method. . . . .                           | 488 |
| PrinterSetup Class . . . . .                    | 489 |
| ProgressBar Control . . . . .                   | 492 |
| ProgressWheel Control . . . . .                 | 493 |
| PString Data Type . . . . .                     | 494 |
| Ptr Data Type. . . . .                          | 494 |
| PushButton Control . . . . .                    | 494 |
| QT3DAudio Class. . . . .                        | 496 |
| QTEffect Class . . . . .                        | 497 |
| QTEffectSequence Class . . . . .                | 498 |
| QTGraphicsExporter Class . . . . .              | 499 |
| QTSoundTrack Class . . . . .                    | 501 |
| QTTrack Class. . . . .                          | 504 |
| QTUserData Class . . . . .                      | 505 |
| QTVideoTrack Class . . . . .                    | 508 |
| Quaternion Class. . . . .                       | 510 |
| QuickTime Object . . . . .                      | 496 |
| Quit Method . . . . .                           | 513 |
| QuitMenuItem Class . . . . .                    | 514 |
| RadioButton Control. . . . .                    | 514 |
| Raise Statement . . . . .                       | 516 |
| RaiseEvent Statement . . . . .                  | 516 |
| Random Class. . . . .                           | 517 |
| Range Class. . . . .                            | 518 |
| RB3DSpace Control . . . . .                     | 519 |
| RBScript Control . . . . .                      | 527 |
| RbScriptAlreadyRunningException error . . . . . | 540 |
| RBVersion Constant . . . . .                    | 540 |
| RBVersionString Constant . . . . .              | 541 |
| Readable Class Interface. . . . .               | 541 |
| REALdatabase Class . . . . .                    | 542 |
| REALSQLdatabase Class . . . . .                 | 546 |
| RecordSet Class. . . . .                        | 550 |
| Rectangle Control . . . . .                     | 553 |
| RectControl Class. . . . .                      | 554 |
| RectShape Class . . . . .                       | 563 |

## Alphabetical Command List

---

|   |     |
|---|-----|
| Redim Method . . . . .  | 564 |
| RegEx Class . . . . .   | 567 |
| RegExException Error . . . . .                                | 574 |
| RegExMatch Class . . . . .                                    | 574 |
| RegExOptions Class . . . . .                                  | 575 |
| RegExSearchPatternException Error . . . . .                   | 576 |
| RegistryAccessErrorException . . . . .                        | 565 |
| RegistryItem Class . . . . .                                  | 565 |
| Rem Statement. . . . .  | 576 |
| Remove Method . . . . .                                       | 577 |
| Replace Function. . . . .                                     | 578 |
| ReplaceAll Function . . . . .                                 | 580 |
| ReplaceAllB Function . . . . .                                | 580 |
| ReplaceB Function . . . . .                                   | 579 |
| ReplaceLineEndings Function . . . . .                         | 581 |
| ResourceFork Class. . . . .                                   | 582 |
| Return Keyword . . . . .                                      | 586 |
| RGB Function. . . . .   | 587 |
| RGBSurface Object. . . . .                                    | 587 |
| Right Function . . . . .                                      | 589 |
| RightB Function . . . . .                                     | 590 |
| Rnd Function . . . . .  | 590 |
| Round Function . . . . .                                      | 591 |
| RoundRectangle Control . . . . .                              | 591 |
| RoundRectShape Class. . . . .                                 | 592 |
| RTrim Function. . . . .                                       | 593 |
| Runtime Object . . . . .                                      | 594 |
| RuntimeException Class . . . . .                              | 595 |
| SaveAsDialog Class. . . . .                                   | 599 |
| Screen Class. . . . .   | 600 |
| Screen Function . . . . .                                     | 602 |
| ScreenCount Function . . . . .                                | 602 |
| Scrollbar Control . . . . .                                   | 603 |
| Select Case Statement . . . . .                               | 605 |
| SelectColor Function. . . . .                                 | 607 |
| SelectFolder Function . . . . .                               | 607 |
| SelectFolderDialog Class. . . . .                             | 608 |
| Self Cannot be used in the Method of a Module Error . . . . . | 611 |
| Self Keyword. . . . .   | 610 |
| Semaphore Class . . . . .                                     | 611 |
| Separator Control . . . . .                                   | 612 |
| Serial Control. . . . .                                       | 613 |
| SerialPort Class . . . . .                                    | 619 |
| ServerSocket Class . . . . .                                  | 620 |
| ServiceApplication Class . . . . .                            | 622 |
| Shell Class. . . . .  | 623 |

|  |     |
|--|-----|
| ShellNotRunningException Error . . . . . | 626 |
| Short Data Type . . . . .                | 626 |
| ShowURL Method . . . . .                 | 627 |
| Shuffle Method . . . . .                 | 627 |
| ShutdownItemsFolder Function. . . . .    | 628 |
| Sign Function. . . . .                   | 629 |
| Sin Function . . . . .                   | 630 |
| Single Data Type . . . . .               | 630 |
| Slider Control. . . . .                  | 631 |
| SMTPSecureSocket Class . . . . .         | 632 |
| SMTPSocket Class . . . . .               | 633 |
| SOAPException Error. . . . .             | 637 |
| SOAPMethod Class. . . . .                | 637 |
| SOAPResult Class. . . . .                | 640 |
| SocketCore Class . . . . .               | 641 |
| Sort Method . . . . .                    | 645 |
| Sortwith Method. . . . .                 | 646 |
| Sound Class. . . . .                     | 647 |
| Speak Method . . . . .                   | 648 |
| SpecialFolder Object. . . . .            | 649 |
| Split Function. . . . .                  | 652 |
| SplitB Function . . . . .                | 652 |
| SpotlightException Class. . . . .        | 653 |
| SpotlightItem Class. . . . .             | 654 |
| SpotlightQuery Class. . . . .            | 654 |
| Sprite Class . . . . .                   | 657 |
| SpriteSurface Control . . . . .          | 659 |
| Sqrt Function . . . . .                  | 665 |
| SSLSocket Control . . . . .              | 665 |
| StackOverflowException Error . . . . .   | 667 |
| StandardInputStream Class . . . . .      | 668 |
| StandardOutputStream Class . . . . .     | 669 |
| StandardToolbarItem Control. . . . .     | 669 |
| StartupItemsFolder Function . . . . .    | 670 |
| Static Statement . . . . .               | 671 |
| StaticText Control . . . . .             | 674 |
| StdErr Method . . . . .                  | 676 |
| StdIn Method. . . . .                    | 676 |
| StdOut Method . . . . .                  | 677 |
| Str Function . . . . .                   | 677 |
| StrComp Function . . . . .               | 678 |
| String Data Type . . . . .               | 679 |
| StringProvider Class Interface . . . . . | 680 |
| StringShape Class . . . . .              | 681 |
| StyledText Class . . . . .               | 681 |
| StyledTextPrinter Class. . . . .         | 685 |

## Alphabetical Command List

---

|   |     |
|---|-----|
| StyleRun Class . . . . .  | 686 |
| Sub Statement . . . . .   | 689 |
| Super Cannot be used in a Module Method Error . . . . .                                       | 690 |
| Syntax Error . . . . .  | 690 |
| System Object . . . . .   | 694 |
| SystemFolder Function. . . . .  | 701 |
| TabPanel Control. . . . .   | 702 |
| Tan Function . . . . .  | 704 |
| TargetBigEndian Constant. . . . .   | 705 |
| TargetCarbon Constant . . . . .   | 705 |
| TargetHasGUI Constant . . . . .   | 706 |
| TargetLinux Constant . . . . .  | 706 |
| TargetLittleEndian Constant . . . . .   | 707 |
| TargetMachO Constant . . . . .  | 707 |
| TargetMacOS Constant . . . . .  | 708 |
| TargetMacOSClassic Constant. . . . .  | 708 |
| TargetWin32 Constant. . . . .   | 708 |
| TCPSocket Control . . . . .   | 709 |
| TemporaryFolder Function . . . . .  | 713 |
| TextColor Function. . . . .   | 713 |
| TextConverter Class . . . . .   | 714 |
| TextEncoding Class. . . . .   | 720 |
| TextInputStream Class . . . . .   | 726 |
| TextOutputStream Class . . . . .  | 729 |
| The Counter variable and the Variable Following Next Must be the Same Error . . . . .         | 731 |
| The keyword 'Then' is expected after this If statement's condition Error . . . . .            | 732 |
| The Size of an Array Must Be a Constant or a Number Error . . . . .                           | 732 |
| There is no Class with this Name Error . . . . .  | 733 |
| This #else Does Not Have a Matching #If Preceding It Error. . . . .                           | 735 |
| This #endif Does Not Have a Matching #If Preceding It Error . . . . .                         | 735 |
| This Array Has Fewer Dimensions than You Have Provided Error. . . . .                         | 733 |
| This Array Has More Dimensions than You Have Provided Error . . . . .                         | 734 |
| This Array Method Works for only One-dimensional Arrays . . . . .                             | 734 |
| This Class is Missing One or More Methods of an Interface it Implements Error . . . . .       | 735 |
| This Global Variable has the Same Name as a Class Error . . . . .                             | 736 |
| This Global Variable has the Same Name as a Global Function Error . . . . .                   | 736 |
| This Global Variable has the Same Name as a Module Error . . . . .                            | 736 |
| This is not An Array but you are using it as One Error. . . . .                               | 737 |
| This Item Conflicts with Another Item of the Same Name Error . . . . .                        | 737 |
| This Kind of Array Cannot be Sorted Error . . . . .   | 738 |
| This Local Variable or Constant has the Same Name as a Declare in this Method Error . . . . . | 739 |
| This Local Variable or Parameter has the Same Name as a Constant Error . . . . .              | 739 |
| This Local Variable or Parameter has the Same Name as a Constant Error . . . . .              | 739 |
| This Method Doesn't Return a Value Error . . . . .  | 740 |
| This Method is Protected. It can only be Called From Within its Class Error . . . . .         | 740 |
| This Method is too Long Error. . . . .  | 740 |

|  |     |
|--|-----|
| This Method or Property Does Not Exist Error . . . . .                               | 740 |
| This Method Requires Fewer Parameters than were Passed Error. . . . .                | 741 |
| This Method Requires More Parameters than were Passed Error . . . . .                | 742 |
| This Name is Already in Use Error . . . . .  | 743 |
| This Property Has the Same Name as a Class Method Error . . . . .                    | 743 |
| This Property Has the Same Name as a Method Error . . . . .                          | 743 |
| This Property Has the Same Name as an Event Error. . . . .                           | 743 |
| This Property is Protected. It can only be Used From Within its Class Error. . . . . | 743 |
| This Type Conversion is only implemented for one-dimensional Arrays Error . . . . .  | 744 |
| Thread Class . . . . .   | 744 |
| ThreadAlreadyRunningException Error. . . . .   | 747 |
| Ticks Function . . . . .   | 748 |
| Timer Object . . . . .   | 748 |
| Titlecase Function . . . . .   | 750 |
| ToolbarItem Control . . . . .  | 751 |
| TrashFolder Function . . . . .   | 753 |
| TrayItem Class . . . . .   | 754 |
| TriangleList Class. . . . .  | 756 |
| Trim Function . . . . .  | 757 |
| Trimesh Class . . . . .  | 757 |
| True Keyword . . . . .   | 759 |
| Try Block . . . . .  | 759 |
| Type Mismatch Error. . . . .   | 761 |
| TypeMismatchException Error. . . . .   | 761 |
| Ubound Function . . . . .  | 762 |
| UBYTE Data Type . . . . .  | 763 |
| UDPSocket Class . . . . .  | 763 |
| UInt16 Data Type . . . . .   | 766 |
| UInt32 Data Type . . . . .   | 767 |
| UInt64 Data Type . . . . .   | 768 |
| UInt8 Data Type . . . . .  | 768 |
| Unknown Pragma Name Error . . . . .  | 769 |
| UnsupportedFormatException Error . . . . .   | 769 |
| UpDownArrows Control. . . . .  | 770 |
| Uppercase Function . . . . .   | 770 |
| UserCancelled Function . . . . .   | 771 |
| UShort Data Type . . . . .   | 772 |
| UVList Class. . . . .  | 772 |
| Val Function . . . . .   | 773 |
| Variant Data Type . . . . .  | 774 |
| VarType Function . . . . .   | 777 |
| Vector3D Class . . . . .   | 777 |
| VectorList Class. . . . .  | 779 |
| VirtualVolume Class . . . . .  | 780 |
| Volume Function. . . . .   | 782 |
| VolumeCount Function . . . . .   | 782 |

## Alphabetical Command List

---

|   |     |
|---|-----|
| While...Wend Statement . . . . .  | 783 |
| Window Class . . . . .  | 784 |
| Window Function . . . . .   | 797 |
| WindowCount Function . . . . .  | 799 |
| WindowPtr Data Type . . . . .   | 799 |
| WordApplication Class. . . . .  | 799 |
| Writeable Class Interface . . . . .   | 801 |
| WString Data Type . . . . .   | 802 |
| XMLAttribute Class . . . . .  | 802 |
| XMLAttributeList Class. . . . .   | 803 |
| XMLCDATASection Class. . . . .  | 803 |
| XMLComment Class . . . . .  | 804 |
| XMLContentModel Class. . . . .  | 804 |
| XMLOutputClass Class . . . . .  | 805 |
| XMLDOMException Error . . . . .   | 809 |
| XMLElement Class . . . . .  | 809 |
| MLError Error . . . . .   | 810 |
| XMLNode Class. . . . .  | 811 |
| XMLNodeList Class . . . . .   | 813 |
| XMLNodeType Object . . . . .  | 813 |
| XMLProcessingInstruction Class . . . . .  | 814 |
| XMLReader Class . . . . .   | 814 |
| XMLReaderException Class . . . . .  | 818 |
| XMLStyleSheet Class . . . . .   | 819 |
| XMLTextNode Class . . . . .   | 819 |
| XMLXslHandler Class. . . . .  | 819 |
| You Can Only Inherit From a Class Error . . . . .   | 820 |
| You Can't Assign one Array to Another Array Error . . . . .                                   | 820 |
| You Can't Pass an Array to an External Function Error . . . . .                               | 821 |
| You Can't Pass an Expression as a Parameter that is Defined as ByRef Error . . . . .          | 821 |
| You Can't Use Super Because this Class has no Super Class Error . . . . .                     | 821 |
| You Can't Use the New Operator with this Class Error . . . . .                                | 822 |
| You Can't Use This Variable or Property Name Because it is a Reserved Word Error. . . . .     | 822 |
| You Cannot Implement a Nonexistent Event Error . . . . .                                      | 822 |
| You Cannot Return a Value Because this Method has No Defined Return Type Error . . . . .      | 823 |
| You Cannot Use the New Operator with a Class Interface Error . . . . .                        | 823 |
| You Have Used an Operator that is not Compatible with the Data Types Specified Error. . . . . | 823 |
| You Must Indicate the Data Type of the Value to be Returned Error. . . . .                    | 824 |
| You Must Use the Value Returned by this Function Error . . . . .                              | 824 |

# Commands by Theme

## Database

|  |     |
|--|-----|
| <b>Data Sources</b>  |     |
| Database4DServer Class . . . . .   | 128 |
| MySQLDatabase Class . . . . .  | 396 |
| ODBCDatabase Class . . . . .   | 424 |
| OpenBaseDatabase Class . . . . .   | 431 |
| OpenCSVCursor method . . . . .   | 433 |
| OpenDBFCursor (FolderItem) ▶ RecordSet . . . . .                         | 435 |
| OpenDTFDatabase (dbFile,blobFile,username,password) ▶ Database . . . . . | 436 |
| OracleDatabase Class . . . . .   | 454 |
| PostgreSQLDatabase Class . . . . .                                       | 483 |
| REALDatabase Class . . . . .   | 542 |
| REALSQLDatabase Class . . . . .  | 546 |
| <b>Front End</b>   |     |
| Database Class . . . . .   | 121 |
| DatabaseField Class . . . . .  | 129 |
| DatabaseQuery Control . . . . .  | 131 |
| DatabaseRecord Class . . . . .   | 132 |
| DataControl Control . . . . .  | 134 |
| RecordSet Class . . . . .  | 550 |

## Errors

|  |     |
|--|-----|
| <b>Debugging Tools</b>   |     |
| DebugDumpObjects(filename) . . . . .   | 144 |
| Runtime Object . . . . .   | 594 |
| <b>Error Handling</b>  |     |
| Catch Statement . . . . .  | 85  |
| Exception Block . . . . .  | 204 |
| Finally Block . . . . .  | 217 |
| Nil Keyword . . . . .  | 406 |
| Raise Statement . . . . .  | 516 |
| RuntimeException Class . . . . .   | 595 |
| StdErr ▶ StandardOutputStream . . . . .  | 676 |
| Try Block . . . . .  | 759 |
| <b>Error Messages</b>  |     |
| A Method with a ParamArray Cannot have any Optional Parameters error . . . . . | 23  |
| A ParamArray cannot have a Default Value error . . . . .                       | 23  |
| A ParamArray's Element Type may not be Another Array error . . . . .           | 24  |
| A Parameter passed by Reference Cannot have a Default Value error . . . . .    | 24  |
| An Assigns Parameter cannot have a Default Value error . . . . .               | 35  |

|   |     |
|---|-----|
| An Extends Parameter cannot have a Default Value error . . . . .                              | 35  |
| An Ordinary Parameter cannot Follow a ParamArray error . . . . .                              | 36  |
| Array Bounds must be Integers error . . . . .   | 57  |
| Assigns can only be used on the last parameter Error . . . . .                                | 60  |
| Cannot Assign a Value to this Property error . . . . .  | 80  |
| Cannot Get This Property's Value error . . . . .  | 81  |
| Declaration Contains a Reserved Word error . . . . .  | 145 |
| Destructors Can't Have Parameters error . . . . .   | 156 |
| End Quote Missing error . . . . .   | 202 |
| Exception Objects Must be of Type RuntimeException error . . . . .                            | 208 |
| Extends can only be used on the first parameter Error . . . . .                               | 212 |
| External Functions Cannot Use Objects as Parameters error . . . . .                           | 212 |
| External Functions Cannot Use String Data Types as Parameters error . . . . .                 | 213 |
| GOTO Target Not Found error . . . . .   | 269 |
| Logic Operations require Boolean Values error . . . . .                                       | 360 |
| Me Cannot be used in a Method of a Module error . . . . .                                     | 368 |
| One of the Interfaces of this Class is not of Type Class Interface . . . . .                  | 429 |
| Only Boolean Constants can be Used with #If error . . . . .                                   | 430 |
| Only One Parameter may use the Assigns Option error . . . . .                                 | 430 |
| Only One Parameter may use the ParamArray Option error . . . . .                              | 430 |
| Only String Constants Can Be Used For Declaring Libraries error . . . . .                     | 431 |
| Only the First Parameter may use the Extends Option error . . . . .                           | 431 |
| Only the Last Parameter may use the Assigns Option error . . . . .                            | 431 |
| Parameter and Default Value Must be of the Same Type error . . . . .                          | 463 |
| Parameters are not Compatible with this Function error . . . . .                              | 463 |
| Self Cannot be used in the Method of a Module error . . . . .                                 | 611 |
| Size of an Array Must Be a Constant or a Number error . . . . .                               | 732 |
| Super Cannot be used in a Module Method error . . . . .                                       | 690 |
| Syntax Error error . . . . .  | 690 |
| The Counter variable and the Variable Following Next Must be the Same error . . . . .         | 731 |
| The keyword 'Then' is expected after this if statement's condition error . . . . .            | 732 |
| There is no Class with this Name error . . . . .  | 733 |
| This #else Does Not Have a Matching #if Preceding It error . . . . .                          | 735 |
| This #endif Does Not Have a Matching #If Preceding It error . . . . .                         | 735 |
| This Array Has Fewer Dimensions than You Have Provided error . . . . .                        | 733 |
| This Array Has More Dimensions than You Have Provided error . . . . .                         | 734 |
| This Array Method Works for only One-dimensional Arrays error . . . . .                       | 734 |
| This Class is Missing One or More Methods of an Interface it Inherits error . . . . .         | 735 |
| This Global Variable has the Same Name as a Class error . . . . .                             | 736 |
| This Global Variable has the Same Name as a Global Function error . . . . .                   | 736 |
| This Global Variable has the Same Name as a Module error . . . . .                            | 736 |
| This is not an Array but you are using it as One error . . . . .                              | 737 |
| This Item Conflicts with Another Item of the Same Name error . . . . .                        | 737 |
| This Kind of Array Cannot be Sorted error . . . . .   | 738 |
| This Local Variable or Constant has the Same Name as a Declare in this Method error . . . . . | 739 |
| This Local Variable or Parameter has the Same Name as a Constant error . . . . .              | 739 |

---

|  |     |
|--|-----|
| This Method Doesn't Return a Value error .....                                       | 740 |
| This Method is Protected. It can only be Called From Within its Class error .....    | 740 |
| This Method is too Long error.....   | 740 |
| This Method or Propert Does Not Exist error.....                                     | 740 |
| This Method Requires Fewer Parameters than were Passed error.....                    | 741 |
| This Method Requires More Parameters than were Passed error.....                     | 742 |
| This Name is Already in Use error.....   | 743 |
| This Property Has the Same Name as a Class Method error .....                        | 743 |
| This Property Has the Same Name as a Method error.....                               | 743 |
| This Property Has the Same Name as an Event error.....                               | 743 |
| This Property is Protected. It can only be Used From Within its Class error.....     | 743 |
| This Type Conversion is only implemented for one-dimensional Arrays error .....      | 744 |
| Type Mismatch error .....  | 761 |
| Unknown Pragma Name error .....  | 769 |
| You Can Only Inherit From a Class error .....  | 820 |
| You Can't Assign one Array to Another Array error .....                              | 820 |
| You Can't Pass an Array to an External Function error .....                          | 821 |
| You Can't Pass an Expression as a Parameter that is Defined as ByRef error .....     | 821 |
| You Can't Use Super Because this Class has no Super Class error.....                 | 821 |
| You Can't Use the New Operator with this Class error .....                           | 822 |
| You Can't Use This Variable or Property Name Because it is a Reserved Word.....      | 822 |
| You Cannot Implement a Nonexistent Event error .....                                 | 822 |
| You Cannot Return a Value Because this Method has No Defined Return Type.....        | 823 |
| You Cannot Use the New Operator with a Class Interface error .....                   | 823 |
| You Have Used an Operator that is not Compatible with the Data Types Specified error | 823 |
| You Must Indicate the Data Type of the Value to be Returned error .....              | 824 |
| You Must Use the Value Returned by this Function error.....                          | 824 |

## Runtime Errors

|   |     |
|---|-----|
| FunctionNotFoundException Error .....       | 245 |
| IllegalCastException Error .....            | 292 |
| InvalidParentException Error.....           | 303 |
| KeyChainException Error.....                | 319 |
| KeyNotFoundException Error .....            | 323 |
| NilObjectException Error .....              | 407 |
| NoOpenTransportException Error.....         | 408 |
| OLEException Error .....                    | 420 |
| OutOfBoundsException Error .....            | 455 |
| OutOfMemoryException Error .....            | 457 |
| RbScriptAlreadyRunningException error ..... | 540 |
| RegExException Error.....                   | 574 |
| RegExSearchPatternException Error .....     | 576 |
| RegistryAccessErrorException Error .....    | 565 |
| Runtime Exception Class .....               | 595 |
| ShellNotRunningException Error .....        | 626 |
| SOAPException Error .....                   | 637 |
| StackOverFlowException Error .....          | 667 |

|   |     |
|---|-----|
| ThreadAlreadyRunningException . . . . . | 747 |
| TypeMismatchException Error . . . . .   | 761 |
| UnsupportedFormatException . . . . .    | 769 |
| XMLDOMException Error . . . . .         | 809 |
| XMLException Error . . . . .            | 810 |
| XMLReaderException Class . . . . .      | 818 |

**Files****Dialogs**

|  |     |
|--|-----|
| FolderItemDialog class . . . . .                 | 233 |
| GetOpenFolderItem(filter) ► FolderItem . . . . . | 253 |
| OpenDialog Class . . . . .                       | 433 |
| SaveAsDialog Class . . . . .                     | 599 |
| SelectFolder ► FolderItem . . . . .              | 607 |
| SelectFolderDialog class . . . . .               | 608 |

**Files**

|  |     |
|--|-----|
| CountFields(source,separator) ► Integer . . . . .  | 114 |
| CountFieldsB(source,separator) ► Integer . . . . . | 114 |
| FileType Class . . . . .                           | 215 |
| FolderItem Class . . . . .                         | 218 |
| Permissions Class . . . . .                        | 464 |

**Input/Output**

|   |     |
|---|-----|
| BinaryStream Class . . . . .  | 69  |
| ExportPicture (pic) ► Boolean . . . . .                             | 209 |
| GetFolderItem(path) ► String . . . . .                              | 249 |
| GetSaveFolderItem(filter, default file name) ► FolderItem . . . . . | 262 |
| Readable Class Interface . . . . .                                  | 541 |
| SelectFolder ► FolderItem . . . . .                                 | 607 |
| StandardInputStream Class . . . . .                                 | 668 |
| StandardOutputStream Class . . . . .                                | 669 |
| TextInputStream Class . . . . .                                     | 726 |
| TextOutputStream Class . . . . .                                    | 729 |
| VirtualVolume Class . . . . .                                       | 780 |
| Volume(volumeNumber) ► FolderItem . . . . .                         | 782 |
| VolumeCount ► Integer . . . . .                                     | 782 |
| Writeable Class Interface . . . . .                                 | 801 |

**OS**

|                             |     |
|-----------------------------|-----|
| Permissions Class . . . . . | 464 |
|-----------------------------|-----|

**SpecialFolders**

|  |     |
|--|-----|
| ApplicatonSupportFolder ► FolderItem . . . . . | 53  |
| DesktopFolder ► FolderItem . . . . .           | 154 |
| DocumentsFolder ► FolderItem . . . . .         | 163 |
| FontsFolder ► FolderItem . . . . .             | 236 |
| GetTemporaryFolderItem ► FolderItem . . . . .  | 264 |

|  |     |
|--|-----|
| GetTrueFolderItem (Name) ► FolderItem . . . . .  | 267 |
| PreferencesFolder ► FolderItem . . . . .         | 487 |
| ShutdownItemsFolder ► FolderItem . . . . .       | 628 |
| SpecialFolder(FolderName) ► FolderItem . . . . . | 649 |
| StartupItemsFolder ► FolderItem . . . . .        | 670 |
| SystemFolder ► FolderItem . . . . .              | 701 |
| TemporaryFolder ► FolderItem . . . . .           | 713 |
| TrashFolder ► FolderItem . . . . .               | 753 |

## Graphics/Multimedia

### 3D Graphics

|                              |     |
|------------------------------|-----|
| Bounds3D . . . . .           | 76  |
| ColorList Class . . . . .    | 97  |
| Element3D Class . . . . .    | 192 |
| Group3D Class . . . . .      | 276 |
| Light3D . . . . .            | 326 |
| Material Class . . . . .     | 361 |
| Object3D Class . . . . .     | 416 |
| Quaternion Class . . . . .   | 510 |
| RB3DSpace Control . . . . .  | 519 |
| TriangleList Class . . . . . | 756 |
| Trimesh Class . . . . .      | 757 |
| UVList . . . . .             | 772 |
| Vector3D Class . . . . .     | 777 |
| VectorList Class . . . . .   | 779 |

### Graphics

|   |     |
|---|-----|
| ArcShape Class . . . . .                      | 55  |
| Canvas Control . . . . .                      | 81  |
| CMY(cyan, magenta, yellow) ► Color . . . . .  | 93  |
| Color Data Type . . . . .                     | 95  |
| CurveShape Class . . . . .                    | 119 |
| DarkBevelColor ► Color . . . . .              | 120 |
| DarkTingeColor ► Color . . . . .              | 121 |
| ExportPicture (pic) ► Boolean . . . . .       | 209 |
| FigureShape Class . . . . .                   | 213 |
| FillColor ► Color . . . . .                   | 217 |
| FrameColor ► Color . . . . .                  | 242 |
| Graphics Class . . . . .                      | 269 |
| Group2D Class . . . . .                       | 275 |
| HighlightColor ► Color . . . . .              | 279 |
| HSV(hue, saturation, value) ► Color . . . . . | 279 |
| ImageWell Control . . . . .                   | 294 |
| LightBevelColor ► Color . . . . .             | 327 |
| LightTingeColor ► Color . . . . .             | 327 |
| Movie Class . . . . .                         | 386 |

|   |     |
|---|-----|
| MoviePlayer Control .....                         | 387 |
| NewPicture(width, height, depth) ► Picture .....  | 404 |
| Object2D Class .....                              | 414 |
| OpenPrinter(pageSetup) ► Graphics .....           | 437 |
| OpenPrinterDialog(pageSetup) ► Graphics .....     | 437 |
| Oval Control .....                                | 457 |
| OvalShape Class .....                             | 458 |
| Picture Class .....                               | 466 |
| PixmapShape Class .....                           | 468 |
| RectShape Class .....                             | 563 |
| Rgb(red, green, blue) ► Color .....               | 587 |
| RGBSurface Object .....                           | 587 |
| RoundRectShape Class .....                        | 592 |
| SelectColor (ByRef color, prompt) ► Boolean ..... | 607 |
| Sprite Class .....                                | 657 |
| SpriteSurface Control .....                       | 659 |
| StringShape class .....                           | 681 |
| TextColor ► Color .....                           | 713 |
| <b>Printing</b>                                   |     |
| PrinterSetup Class .....                          | 489 |
| <b>QuickTime</b>                                  |     |
| EditableMovie Class .....                         | 176 |
| GetQTCrossFadeEffect Function ► QTEffect .....    | 256 |
| GetQTGraphicsExporter Method .....                | 257 |
| GetQTSMPTEEffect (ID) ► QTEffect .....            | 257 |
| Movie Class .....                                 | 386 |
| MoviePlayer Control .....                         | 387 |
| OpenURLMovie ► Movie .....                        | 438 |
| QT3DAudio Class .....                             | 496 |
| QTEffect Class .....                              | 497 |
| QTEffectSequence class .....                      | 498 |
| QTGraphicsExporter Class .....                    | 499 |
| QTSoundTrack Class .....                          | 501 |
| QTTrack Class .....                               | 504 |
| QTUserData Class .....                            | 505 |
| QTVideoTrack Class .....                          | 508 |
| QuickTime Object .....                            | 496 |
| <b>Sound</b>                                      |     |
| Beep Method .....                                 | 64  |
| Movie Class .....                                 | 386 |
| MoviePlayer Control .....                         | 387 |
| NotePlayer Control .....                          | 409 |
| Sound Class .....                                 | 647 |
| Speak Method .....                                | 648 |
| <b>Vector Graphics</b>                            |     |

|                                |     |
|--------------------------------|-----|
| ArcShape Class . . . . .       | 55  |
| CurveShape Class . . . . .     | 119 |
| FigureShape Class . . . . .    | 213 |
| Group2D Class . . . . .        | 275 |
| Object2D Class . . . . .       | 414 |
| OvalShape Class . . . . .      | 458 |
| PixmapShape Class . . . . .    | 468 |
| RectShape Class . . . . .      | 563 |
| RoundRectShape Class . . . . . | 592 |
| StringShape Class . . . . .    | 681 |

## Hardware

|                                   |     |
|-----------------------------------|-----|
| <b>Classes</b>                    |     |
| SerialPort . . . . .              | 619 |
| <b>Game Input</b>                 |     |
| GameInputDevice Class . . . . .   | 245 |
| GameInputElement Class . . . . .  | 246 |
| GameInputManager Class . . . . .  | 246 |
| <b>Monitors/Screens</b>           |     |
| Screen Class . . . . .            | 600 |
| Screen Function . . . . .         | 602 |
| <b>OS</b>                         |     |
| Keyboard . . . . .                | 312 |
| NetworkInterface Object . . . . . | 398 |

## Internet/Networking

|   |     |
|---|-----|
| <b>Easy Networking</b>                        |     |
| AutoDiscovery Class . . . . .                 | 62  |
| EasyTCPSocket Class . . . . .                 | 171 |
| EasyUDPSocket Class . . . . .                 | 173 |
| <b>Email</b>                                  |     |
| POP3SecureSocket Class . . . . .              | 471 |
| POP3Socket Class . . . . .                    | 475 |
| SMTPSecureSocket Class . . . . .              | 632 |
| SMTPSocket Class . . . . .                    | 633 |
| <b>Internet</b>                               |     |
| Datagram Class . . . . .                      | 139 |
| DecodeBase64(str) ► String . . . . .          | 152 |
| DecodeQuotedPrintable(str) ► String . . . . . | 152 |
| DecodeURLComponent Function . . . . .         | 153 |
| EmailAttachment Class . . . . .               | 194 |
| EmailHeaders Class . . . . .                  | 195 |
| EmailMessage Class . . . . .                  | 196 |

|   |     |
|---|-----|
| EncodeBase64(str) ► String . . . . .          | 198 |
| EncodeQuotedPrintable(str) ► String . . . . . | 199 |
| EncodeURLComponent Function . . . . .         | 199 |
| HTMLViewer Control . . . . .                  | 280 |
| HTTPSecureSocket Class . . . . .              | 283 |
| HTTPSocket Class . . . . .                    | 286 |
| InternetHeaders Class . . . . .               | 302 |
| MD5(str) ► String . . . . .                   | 363 |
| MD5Digest Class . . . . .                     | 363 |
| Network Object . . . . .                      | 397 |
| NetworkInterface Object . . . . .             | 398 |
| POP3SecureSocket Class . . . . .              | 471 |
| POP3Socket Class . . . . .                    | 475 |
| ShowURL Method . . . . .                      | 627 |
| SMTPSecureSocket Class . . . . .              | 632 |
| SMTPSocket Class . . . . .                    | 633 |
| SocketCore Class . . . . .                    | 641 |
| SSLSocket . . . . .                           | 665 |
| TCPSocket Control . . . . .                   | 709 |
| UDPSocket Contrl . . . . .                    | 763 |

### Networking

|                                   |     |
|-----------------------------------|-----|
| AutoDiscovery Class . . . . .     | 62  |
| Datagram Class . . . . .          | 139 |
| EasyTCPSocket Class . . . . .     | 171 |
| EasyUDPSocket Class . . . . .     | 173 |
| EmailAttachment Class . . . . .   | 194 |
| EmailHeaders Class . . . . .      | 195 |
| EmailMessage Class . . . . .      | 196 |
| HTTPSecureSocket Class . . . . .  | 283 |
| HTTPSocket Class . . . . .        | 286 |
| IPCSocket Class . . . . .         | 304 |
| Network Object . . . . .          | 397 |
| NetworkInterface Object . . . . . | 398 |
| POP3SecureSocket Class . . . . .  | 471 |
| POP3Socket Class . . . . .        | 475 |
| Serial Control . . . . .          | 613 |
| ServerSocket Class . . . . .      | 620 |
| ShowURL Method . . . . .          | 627 |
| SMTPSecureSocket Class . . . . .  | 632 |
| SMTPSocket Class . . . . .        | 633 |
| SocketCore Class . . . . .        | 641 |
| SSLSocket Control . . . . .       | 665 |
| System Object . . . . .           | 694 |
| TCPSocket Control . . . . .       | 709 |
| UDPSocket Control . . . . .       | 763 |

**SOAP**

|                           |     |
|---------------------------|-----|
| SOAPException Error ..... | 637 |
| SOAPMethod Class .....    | 637 |
| SOAPResult Class .....    | 640 |

**Text Encoding**

|   |     |
|---|-----|
| GetInternetTextEncoding (InternetEncoding) ► TextEncoding ..... | 252 |
|---|-----|

**Language****Arrays**

|  |     |
|--|-----|
| Append .....                                   | 37  |
| Array(ElementList) ► Array .....               | 55  |
| Dim Statement .....                            | 158 |
| IndexOf .....                                  | 295 |
| Insert .....                                   | 296 |
| Join(sourceArray, delimiter) ► String .....    | 312 |
| OutOfBoundsException Error .....               | 455 |
| OutOfMemoryException Error .....               | 457 |
| ParamArray keyword .....                       | 461 |
| Pop Method .....                               | 471 |
| Redim .....                                    | 564 |
| Remove .....                                   | 577 |
| Shuffle .....                                  | 627 |
| Sort .....                                     | 645 |
| Sortwith .....                                 | 646 |
| Split(source, delimiter) ► String array .....  | 652 |
| SplitB(source, delimiter) ► String array ..... | 652 |
| Static Statement .....                         | 671 |
| Ubound(array) ► Integer .....                  | 762 |

**Boolean**

|                    |     |
|--------------------|-----|
| And Operator ..... | 36  |
| False .....        | 213 |
| Nil .....          | 406 |
| Not Operator ..... | 409 |
| Or Operator .....  | 454 |
| True .....         | 759 |

**Class Interfaces**

|   |     |
|---|-----|
| DataAvailableProvider .....             | 121 |
| DataNotificationReceiver .....          | 140 |
| DataNotifier Class Interface .....      | 140 |
| ListSelectionNotificationReceiver ..... | 359 |
| ListSelectionNotifier .....             | 359 |
| Readable .....                          | 541 |
| StringProvider .....                    | 680 |
| Writeable Class Interface .....         | 801 |

**Classes**

|                            |     |
|----------------------------|-----|
| AddressBook                | 25  |
| AddressBookAddress         | 27  |
| AddressBookContact         | 28  |
| AddressBookData            | 30  |
| AddressBookGroup           | 32  |
| AddressBookRecord          | 34  |
| AppleEvent                 | 38  |
| AppleEventDescList         | 42  |
| AppleEventObjectSpecifier  | 43  |
| AppleEventRecord           | 44  |
| AppleEventTarget           | 45  |
| AppleEventTemplate         | 46  |
| AppleMenuItem              | 46  |
| Application                | 47  |
| ArcShape                   | 55  |
| AutoDiscovery              | 62  |
| BevelButton Control        | 64  |
| BinaryStream               | 69  |
| Bitwise                    | 73  |
| Bounds3D                   | 76  |
| Canvas Control             | 81  |
| Clipboard                  | 91  |
| Collection                 | 94  |
| ColorList                  | 97  |
| ConsoleApplication         | 100 |
| ContainerControl           | 104 |
| Control                    | 111 |
| CriticalSection            | 115 |
| CurveShape                 | 119 |
| Database                   | 121 |
| Database4DServer           | 128 |
| DatabaseField              | 129 |
| DatabaseQuery              | 131 |
| DatabaseQuery Control      | 131 |
| DatabaseRecord             | 132 |
| Datagram                   | 139 |
| Date                       | 140 |
| Dictionary                 | 156 |
| DisclosureTriangle Control | 162 |
| DockItem                   | 162 |
| DragItem                   | 167 |
| EasyTCPSocket              | 171 |
| EasyUDPSocket              | 173 |
| EditableMovie              | 176 |
| EditField Control          | 181 |

|                           |     |
|---------------------------|-----|
| Element3D .....           | 192 |
| EmailAttachment .....     | 194 |
| EmailHeaders .....        | 195 |
| EmailMessage .....        | 196 |
| EndOfLine .....           | 202 |
| ExcelApplication .....    | 203 |
| FigureShape .....         | 213 |
| FileType .....            | 215 |
| FolderItem .....          | 218 |
| FolderItemDialog .....    | 233 |
| GameInputDevice .....     | 245 |
| GameInputElement .....    | 246 |
| GameInputManager .....    | 246 |
| Graphics .....            | 269 |
| Group2D .....             | 275 |
| Group3D .....             | 276 |
| GroupBox Control .....    | 277 |
| HTTPSecureSocket .....    | 283 |
| HTTPSocket .....          | 286 |
| ImageWell Control .....   | 294 |
| InternetHeaders .....     | 302 |
| IPCSocket .....           | 304 |
| IsA Operator .....        | 307 |
| KeyChain .....            | 316 |
| KeyChainItem .....        | 321 |
| Light3D .....             | 326 |
| Line Control .....        | 328 |
| ListBox Control .....     | 329 |
| ListColumn .....          | 357 |
| Material .....            | 361 |
| MD5Digest .....           | 363 |
| MDIWindow .....           | 364 |
| MemoryBlock .....         | 369 |
| MenuItem .....            | 372 |
| MessageDialog .....       | 377 |
| MessageDialogButton ..... | 380 |
| MouseCursor .....         | 385 |
| Movie .....               | 386 |
| MoviePlayer Control ..... | 387 |
| Mutex .....               | 395 |
| MySQLDatabase .....       | 396 |
| New Operator .....        | 399 |
| NotePlayer Control .....  | 409 |
| Object .....              | 414 |
| Object2D .....            | 414 |
| Object3D .....            | 416 |

|                                       |     |
|---------------------------------------|-----|
| ODBCDatabase . . . . .                | 424 |
| Office . . . . .                      | 426 |
| OLEContainer . . . . .                | 419 |
| OLEObject . . . . .                   | 420 |
| OLEParameter . . . . .                | 423 |
| OpenBaseDatabase . . . . .            | 431 |
| OpenDialog . . . . .                  | 433 |
| OracleDatabase . . . . .              | 454 |
| Oval Control . . . . .                | 457 |
| OvalShape . . . . .                   | 458 |
| PagePanel Control . . . . .           | 459 |
| Paragraph . . . . .                   | 460 |
| Permissions . . . . .                 | 464 |
| Picture . . . . .                     | 466 |
| PixmapShape . . . . .                 | 468 |
| Placard Control . . . . .             | 469 |
| POP3SecureSocket . . . . .            | 471 |
| POP3Socket . . . . .                  | 475 |
| PopupArrow Control . . . . .          | 479 |
| PopupMenu Control . . . . .           | 480 |
| PostgreSQLDatabase . . . . .          | 483 |
| PowerPointApplication Class . . . . . | 484 |
| PrefsMenuItem . . . . .               | 488 |
| PrinterSetup . . . . .                | 489 |
| ProgressBar Control . . . . .         | 492 |
| ProgressWheel Control . . . . .       | 493 |
| PushButton Control . . . . .          | 494 |
| QT3DAudio . . . . .                   | 496 |
| QTEffect . . . . .                    | 497 |
| QTEffectSequence . . . . .            | 498 |
| QTGraphicsExporter . . . . .          | 499 |
| QTSoundTrack . . . . .                | 501 |
| QTTrack . . . . .                     | 504 |
| QTUserData . . . . .                  | 505 |
| QTVideoTrack . . . . .                | 508 |
| Quaternion . . . . .                  | 510 |
| QuitMenuItem . . . . .                | 514 |
| RadioButton Control . . . . .         | 514 |
| Random . . . . .                      | 517 |
| Range . . . . .                       | 518 |
| RBScript . . . . .                    | 527 |
| REALDatabase . . . . .                | 542 |
| REALSQLDatabase . . . . .             | 546 |
| RecordSet . . . . .                   | 550 |
| Rectangle Control . . . . .           | 553 |
| RectControl . . . . .                 | 554 |

|   |     |
|---|-----|
| RectShape . . . . .                     | 563 |
| RegEx . . . . .                         | 567 |
| RegExMatch . . . . .                    | 574 |
| RegExOptions . . . . .                  | 575 |
| RegistryItem . . . . .                  | 565 |
| ResourceFork . . . . .                  | 582 |
| RoundRectangle Control . . . . .        | 591 |
| RoundRectShape . . . . .                | 592 |
| RuntimeException . . . . .              | 595 |
| SaveAsDialog . . . . .                  | 599 |
| Screen . . . . .                        | 600 |
| Scrollbar Control . . . . .             | 603 |
| SelectFolderDialog . . . . .            | 608 |
| Semaphore . . . . .                     | 611 |
| Separator Control. . . . .              | 612 |
| Serial Control . . . . .                | 613 |
| SerialPort . . . . .                    | 619 |
| ServerSocket . . . . .                  | 620 |
| ServiceApplication . . . . .            | 622 |
| Shell. . . . .                          | 623 |
| Slider Control . . . . .                | 631 |
| SMTPSecureSocket . . . . .              | 632 |
| SMTPSocket . . . . .                    | 633 |
| SOAPMethod . . . . .                    | 637 |
| SOAPResult . . . . .                    | 640 |
| SocketCore . . . . .                    | 641 |
| Sound . . . . .                         | 647 |
| Sprite . . . . .                        | 657 |
| SpriteSurface Control. . . . .          | 659 |
| SSLocket Control. . . . .               | 665 |
| StandardInputStream. . . . .            | 668 |
| StandardOutputStream. . . . .           | 669 |
| StaticText Control. . . . .             | 674 |
| StringShape . . . . .                   | 681 |
| StyledText . . . . .                    | 681 |
| StyledTextPrinter Class. . . . .        | 685 |
| StyleRun . . . . .                      | 686 |
| TabPanel Control . . . . .              | 702 |
| TextConverter. . . . .                  | 714 |
| TextEncoding . . . . .                  | 720 |
| TextInputStream. . . . .                | 726 |
| TextOutputStream . . . . .              | 729 |
| Thread. . . . .                         | 744 |
| ThreadAlreadyRunningException . . . . . | 747 |
| Timer Object. . . . .                   | 748 |
| TrayItem . . . . .                      | 754 |

|                                |     |
|--------------------------------|-----|
| TriangleList .....             | 756 |
| Trimesh .....                  | 757 |
| UDPSocket .....                | 763 |
| UpDownArrows Control .....     | 770 |
| UVList .....                   | 772 |
| Vector3D .....                 | 777 |
| VectorList .....               | 779 |
| VirtualVolume .....            | 780 |
| Window .....                   | 784 |
| WordApplication .....          | 799 |
| XMLAttribute .....             | 802 |
| XMLAttributeList .....         | 803 |
| XMLCDATASection .....          | 803 |
| XMLComment .....               | 804 |
| XMLContentModel .....          | 804 |
| XMLDocument .....              | 805 |
| XMLODEException .....          | 809 |
| XMLElement .....               | 809 |
| MLEException .....             | 810 |
| XMLNode .....                  | 811 |
| XMLNodeList .....              | 813 |
| XMLProcessingInstruction ..... | 814 |
| XMLReader .....                | 814 |
| XMLReaderException .....       | 818 |
| XMLStyleSheet .....            | 819 |
| XMLTextNode .....              | 819 |
| XMLXsiHandler .....            | 819 |

### Code Execution

|                              |     |
|------------------------------|-----|
| #If ...#Else...#Endif .....  | 7   |
| // Operator .....            | 14  |
| _ Keyword .....              | 22  |
| ' Operator .....             | 12  |
| Assigns Keyword .....        | 60  |
| Break Keyword .....          | 77  |
| ByRef Keyword .....          | 77  |
| ByVal Keyword .....          | 79  |
| Call Statement .....         | 80  |
| Constructor Method .....     | 103 |
| Continue Statement .....     | 110 |
| Critical Section Class ..... | 115 |
| Destructor Method .....      | 155 |
| Do...Loop .....              | 164 |
| EnableMenuItem Method .....  | 198 |
| Exit Statement .....         | 208 |
| Extends Keyword .....        | 211 |
| Finally Block .....          | 217 |

---

|                                   |     |
|-----------------------------------|-----|
| For Each...Next . . . . .         | 237 |
| For...Next . . . . .              | 238 |
| Function Statement . . . . .      | 242 |
| GOTO Statement . . . . .          | 268 |
| If...Then...Else . . . . .        | 290 |
| Mutex Class. . . . .              | 395 |
| Optional Keyword . . . . .        | 453 |
| Pragma Directives. . . . .        | 8   |
| Quit Method. . . . .              | 513 |
| RaiseEvent Statement . . . . .    | 516 |
| RBScript Control . . . . .        | 527 |
| Rem Statement. . . . .            | 576 |
| Return Keyword . . . . .          | 586 |
| Select Case . . . . .             | 605 |
| Semaphore Class. . . . .          | 611 |
| Sub Statement . . . . .           | 689 |
| Thread Class . . . . .            | 744 |
| Timer Object. . . . .             | 748 |
| UserCancelled ▶ Boolean . . . . . | 771 |
| While...Wend . . . . .            | 783 |

## Console Applications

|   |     |
|---|-----|
| ConsoleApplication Class. . . . .       | 100 |
| Input ▶ String . . . . .                | 296 |
| Print Method . . . . .                  | 488 |
| ServiceApplication Class. . . . .       | 622 |
| StandardInputStream Class . . . . .     | 668 |
| StandardOutputStream Class . . . . .    | 669 |
| StdErr ▶ StandardOutputStream . . . . . | 676 |
| StdIn ▶ StandardInputStream. . . . .    | 676 |
| StdOut ▶ StandardOutputStream . . . . . | 677 |
| TargetHasGUI ▶ Boolean . . . . .        | 706 |

## Constants

|  |     |
|--|-----|
| DebugBuild ▶ Boolean . . . . .         | 144 |
| RBVersion ▶ Double . . . . .           | 540 |
| RBVersionString ▶ String . . . . .     | 541 |
| TargetBigEndian ▶ Boolean . . . . .    | 705 |
| TargetCarbon ▶ Boolean . . . . .       | 705 |
| TargetHasGUI ▶ Boolean . . . . .       | 706 |
| TargetLinux ▶ Boolean . . . . .        | 706 |
| TargetLittleEndian ▶ Boolean . . . . . | 707 |
| TargetMachO ▶ Boolean. . . . .         | 707 |
| TargetMacOS ▶ Boolean. . . . .         | 708 |
| TargetMacOSClassic ▶ Boolean . . . . . | 708 |
| TargetWin32 ▶ Boolean . . . . .        | 708 |

|  |     |
|--|-----|
| <b>Data Types</b>                            |     |
| Boolean                                      | 75  |
| Byte   | 79  |
| CFString                                     | 87  |
| Color  | 95  |
| CString                                      | 117 |
| Double                                       | 166 |
| Int16  | 299 |
| Int32  | 300 |
| Int64  | 300 |
| Int8   | 301 |
| Integer                                      | 302 |
| IsNumeric(arg) ► Boolean                     | 311 |
| OSType                                       | 455 |
| PString                                      | 494 |
| Ptr  | 494 |
| Short  | 626 |
| Single                                       | 630 |
| String                                       | 679 |
| UByte  | 763 |
| UInt16                                       | 766 |
| UInt32                                       | 767 |
| UInt64                                       | 768 |
| UInt8  | 768 |
| UShort                                       | 772 |
| Variant                                      | 774 |
| WindowPtr                                    | 799 |
| WString                                      | 802 |
| <b>Date and Time</b>                         |     |
| Date Class                                   | 140 |
| Microseconds ► Double                        | 381 |
| ParseDate (Text, ByRef ParsedDate) ► Boolean | 463 |
| Ticks ► Integer                              | 748 |
| <b>Literals</b>                              |     |
| "  | 6   |
| &b   | 10  |
| &c   | 10  |
| &h   | 11  |
| &o   | 11  |
| <b>Math</b>                                  |     |
| Abs(value) ► Double                          | 24  |
| Acos(value) ► Double                         | 25  |
| Asin(value) ► Double                         | 59  |
| Atan(value) ► Double                         | 61  |
| Atan2(y,x) ► Double                          | 61  |

|  |     |
|--|-----|
| Bin(value) ► String . . . . .                              | 68  |
| CDbl(string) ► Double . . . . .                            | 86  |
| Ceil(value) ► Double . . . . .                             | 86  |
| Clong(string) ► Int64 . . . . .                            | 92  |
| Cos(value) ► Double . . . . .                              | 113 |
| Exp(value) ► Double . . . . .                              | 209 |
| Floor(value) ► Double . . . . .                            | 218 |
| Hex(value) ► String . . . . .                              | 278 |
| Log(value) ► Double . . . . .                              | 360 |
| Max(value1, value2) ► Double . . . . .                     | 362 |
| Min(value1, value2) ► Double . . . . .                     | 383 |
| Oct(value) ► String . . . . .                              | 426 |
| Pow(value, power) ► Double . . . . .                       | 484 |
| Random Class . . . . .                                     | 517 |
| Rnd ► Double. . . . .                                      | 590 |
| Round(value) ► Double . . . . .                            | 591 |
| Sign(value) ► Integer. . . . .                             | 629 |
| Sin(value) ► Double . . . . .                              | 630 |
| Sqrt(value) ► Double . . . . .                             | 665 |
| Tan(value) ► Double . . . . .                              | 704 |
| Val(string) ► Double . . . . .                             | 773 |
| <b>Multithreading</b>                                      |     |
| CriticalSection Class . . . . .                            | 115 |
| Mutex Class. . . . .                                       | 395 |
| Semaphore Class. . . . .                                   | 611 |
| Thread Class . . . . .                                     | 744 |
| <b>Object Binding</b>                                      |     |
| DataAvailableProvider Class Interface . . . . .            | 121 |
| DataNotificationReceiver Class Interface Class. . . . .    | 140 |
| DataNotifier Class Interface. . . . .                      | 140 |
| ListSelectionNotificationReceiver Class Interface. . . . . | 359 |
| ListSelectionNotifier Class Interface . . . . .            | 359 |
| StringProvider Class Interface . . . . .                   | 680 |
| <b>Objects</b>   |     |
| App Function . . . . .                                     | 37  |
| Cursor Object . . . . .                                    | 117 |
| Keyboard . . . . .   | 312 |
| Me Function . . . . .                                      | 367 |
| New Operator. . . . .                                      | 399 |
| QuickTime. . . . .   | 496 |
| Readable Class Interface . . . . .                         | 541 |
| Self Function. . . . .                                     | 610 |
| System. . . . .  | 694 |
| Writable Class Interface . . . . .                         | 801 |

## Operators

|                             |     |
|-----------------------------|-----|
| -                           | 6   |
| *                           | 13  |
| +                           | 13  |
| /                           | 14  |
| <                           | 15  |
| <=                          | 16  |
| <>                          | 17  |
| =                           | 18  |
| >                           | 19  |
| >=                          | 20  |
| \.                          | 21  |
| ^ Operator                  | 22  |
| And                         | 36  |
| Bitwise Class               | 73  |
| Is                          | 307 |
| IsA                         | 307 |
| Mod                         | 384 |
| New                         | 399 |
| Not                         | 409 |
| Operator_Add                | 439 |
| Operator_AddRight           | 440 |
| Operator_And                | 440 |
| Operator_AndRight           | 441 |
| Operator_Compare            | 441 |
| Operator_Convert            | 442 |
| Operator_Divide             | 443 |
| Operator_DivideRight        | 444 |
| Operator_IntegerDivide      | 444 |
| Operator_IntegerDivideRight | 445 |
| Operator_Lookup             | 445 |
| Operator_Modulo             | 446 |
| Operator_ModuloRight        | 447 |
| Operator_Multiply           | 447 |
| Operator_MultiplyRight      | 448 |
| Operator_Not                | 449 |
| Operator_Or                 | 449 |
| Operator_OrRight            | 450 |
| Operator_Power              | 450 |
| Operator_PowerRight         | 451 |
| Operator_Subtract           | 452 |
| Operator_SubtractRight      | 452 |
| Or                          | 454 |

## OS

|                      |    |
|----------------------|----|
| #If...#Else...#Endif | 7  |
| AddressOf Operator   | 34 |

|   |     |
|---|-----|
| Application Class . . . . .   | 47  |
| ApplicationSupportFolder ► FolderItem . . . . .                     | 53  |
| Beep Method . . . . .   | 64  |
| Clipboard Class . . . . .   | 91  |
| DarkBevelColor ► Color . . . . .                                    | 120 |
| DarkTingeColor ► Color . . . . .                                    | 121 |
| DebugBuild ► Boolean . . . . .                                      | 144 |
| Declare Statement . . . . .   | 145 |
| DesktopFolder ► FolderItem . . . . .                                | 154 |
| DockItem class . . . . .  | 162 |
| DocumentsFolder ► FolderItem . . . . .                              | 163 |
| EndOfLine Class . . . . .   | 202 |
| FillColor ► Color . . . . .   | 217 |
| Font(index) ► String . . . . .                                      | 235 |
| FontCount ► Integer . . . . .                                       | 235 |
| FontsFolder ► FolderItem . . . . .                                  | 236 |
| FrameColor ► Color . . . . .  | 242 |
| GetOpenFolderItem(filter) ► FolderItem . . . . .                    | 253 |
| GetSaveFolderItem(filter, default file name) ► FolderItem . . . . . | 262 |
| GetTemporaryFolderItem ► FolderItem . . . . .                       | 264 |
| HighlightColor ► Color . . . . .                                    | 279 |
| Keyboard . . . . .  | 312 |
| LightBevelColor ► Color . . . . .                                   | 327 |
| LightTingeColor ► Color . . . . .                                   | 327 |
| MDIWindow Class . . . . .   | 364 |
| MemoryBlock Class . . . . .   | 369 |
| Microseconds ► Double . . . . .                                     | 381 |
| MouseCursor Class . . . . .   | 385 |
| NewMemoryBlock(size) ► MemoryBlock . . . . .                        | 403 |
| OpenPrinterDialog(pageSetup) ► Graphics . . . . .                   | 437 |
| Permissions Class . . . . .   | 464 |
| PreferencesFolder ► FolderItem . . . . .                            | 487 |
| QuickTime Object . . . . .  | 496 |
| RBVersion ► Double . . . . .  | 540 |
| RBVersionString ► String . . . . .                                  | 541 |
| ResourceFork Class . . . . .  | 582 |
| Screen Class . . . . .  | 600 |
| Screen Function . . . . .   | 602 |
| ScreenCount ► Integer . . . . .                                     | 602 |
| SelectFolder ► FolderItem . . . . .                                 | 607 |
| Serial Control . . . . .  | 613 |
| SerialPort Class . . . . .  | 619 |
| Shell Class . . . . .   | 623 |
| ShowURL Method . . . . .  | 627 |
| ShutdownItemsFolder ► FolderItem . . . . .                          | 628 |
| Speak Method . . . . .  | 648 |

## Macintosh

---

|  |     |
|--|-----|
| SpecialFolder(FolderName) ▶ FolderItem . . . . . | 649 |
| StartupItemsFolder ▶ FolderItem . . . . .        | 670 |
| System Object . . . . .                          | 694 |
| SystemFolder ▶ FolderItem . . . . .              | 701 |
| TargetBigEndian ▶ Boolean . . . . .              | 705 |
| TargetCarbon ▶ Boolean . . . . .                 | 705 |
| TargetHasGUI ▶ Boolean . . . . .                 | 706 |
| TargetLinux ▶ Boolean . . . . .                  | 706 |
| TargetLittleEndian ▶ Boolean . . . . .           | 707 |
| TargetMachO ▶ Boolean . . . . .                  | 707 |
| TargetMacOS ▶ Boolean . . . . .                  | 708 |
| TargetMacOSClassic ▶ Boolean . . . . .           | 708 |
| TargetWin32 ▶ Boolean . . . . .                  | 708 |
| TemporaryFolder ▶ FolderItem . . . . .           | 713 |
| Ticks ▶ Integer . . . . .                        | 748 |
| TrashFolder ▶ FolderItem . . . . .               | 753 |
| TrayItem Class . . . . .                         | 754 |
| UserCancelled ▶ Boolean . . . . .                | 771 |
| Volume(volumeNumber) ▶ FolderItem . . . . .      | 782 |
| VolumeCount ▶ Integer . . . . .                  | 782 |
| <b>Regular Expressions</b>                       |     |
| RegEx Class . . . . .                            | 567 |
| RegExMatch Class . . . . .                       | 574 |
| RegExOptions Class . . . . .                     | 575 |
| <b>User Interface</b>                            |     |
| StandardToolbarItem . . . . .                    | 669 |
| ToolbarItem . . . . .                            | 751 |
| <b>Variables</b>                                 |     |
| Append . . . . .                                 | 37  |
| Const statement . . . . .                        | 102 |
| Declare Statement . . . . .                      | 145 |
| Dim Statement . . . . .                          | 158 |
| Pop Method . . . . .                             | 471 |
| Static Statement . . . . .                       | 671 |
| VarType (value) ▶ Integer . . . . .              | 777 |

## Macintosh

### Address Books

|                                    |    |
|------------------------------------|----|
| AddressBook Class . . . . .        | 25 |
| AddressBookAddress Class . . . . . | 27 |
| AddressBookContact Class . . . . . | 28 |
| AddressBookData Class . . . . .    | 30 |
| AddressBookGroup Class . . . . .   | 32 |
| AddressBookRecord Class . . . . .  | 34 |

|  |     |
|--|-----|
| System Object . . . . .  | 694 |
| <b>AppleEvents</b>   |     |
| AppleEvent Class . . . . .   | 38  |
| AppleEventDescList Class . . . . .   | 42  |
| AppleEventObjectSpecifier Class . . . . .  | 43  |
| AppleEventRecord Class . . . . .   | 44  |
| AppleEventTarget Class . . . . .   | 45  |
| AppleEventTemplate Class . . . . .   | 46  |
| Application Class . . . . .  | 47  |
| GetAppleEventTarget (Prompt, ListLabel) ▶ AppleEventTarget . . . . .   | 249 |
| GetIndexedObjectDescriptor(desiredClass, Object, Index) ▶ AppleEventObjectSpecifier . . . . .                            | 252 |
| GetNamedObjectDescriptor(desiredClass, object, name) ▶ AppleEventObjectSpecifier . . . . .                               | 253 |
| GetOrdinalObjectDescriptor(desiredClass, object, ordinalKey) ▶ AppleEventObjectSpecifier . . . . .                       | 255 |
| GetPropertyObjectDescriptor(object, name) ▶ AppleEventObjectSpecifier. . . . .   | 256 |
| GetRangeObjectDescriptor(desiredClass, object, rangeStart, rangeEnd) ▶ AppleEventObjectSpecifier . . . . .               | 262 |
| GetStringComparisonObjectDescriptor (comparisionKey, comparisonForm, field, value) ▶ AppleEventObjectSpecifier . . . . . | 263 |
| GetTestObjectDescriptor(desiredClass, object, comparison) ▶ AppleEventObjectSpecifier . . . . .                          | 265 |
| GetUniqueIDObjectDescriptor(DesiredClass, Object, ID) ▶ AppleEventObjectSpecifier . . . . .                              | 267 |
| NewAppleEvent(eventClass, eventID, creatorCode) ▶ AppleEvent . . . . .   | 402 |
| NewAppleEventTarget (Name, Computer, Zone, PortType) ▶ AppleEventTarget . . . . .  | 402 |
| <b>Controls</b>  |     |
| StandardToolbarItem. . . . .   | 669 |
| ToolbarItem . . . . .  | 751 |
| <b>Keychains</b>   |     |
| KeyChain Class . . . . .   | 316 |
| KeyChainException Error . . . . .  | 319 |
| KeyChainItem Class . . . . .   | 321 |
| System Object . . . . .  | 694 |
| <b>Office Automation</b>   |     |
| ExcelApplication. . . . .  | 203 |
| Office Class . . . . .   | 426 |
| OLEObject Class . . . . .  | 420 |
| PowerPointApplication Class. . . . .   | 484 |
| WordApplication Class. . . . .   | 799 |
| <b>OLE</b>   |     |
| OLE Parameter Class. . . . .   | 423 |
| OLEException Error . . . . .   | 420 |
| <b>Resources</b>   |     |
| ResourceFork Class . . . . .   | 582 |
| <b>Spotlight</b>   |     |
| SpotlightException Class . . . . .   | 653 |
| SpotlightItem Class. . . . .   | 654 |

---

|   |     |
|---|-----|
| <b>SpotlightQuery Class</b> .....                           | 654 |
| <b>User Interface</b>                                       |     |
| <b>DockItem</b> .....                                       | 162 |
| <br>  |     |
| <b>Text</b>   |     |
| <b>Classes</b>  |     |
| <b>StandardInputStream</b> .....                            | 668 |
| <b>StandardOutputStream</b> .....                           | 669 |
| <b>Console Applications</b>                                 |     |
| <b>Input</b> ▶ <b>String</b> .....                          | 296 |
| <b>StdIn</b> ▶ <b>StandardInputStream</b> .....             | 676 |
| <b>StdOut</b> ▶ <b>StandardOutputStream</b> .....           | 677 |
| <b>Controls</b>   |     |
| <b>EditField</b> .....                                      | 181 |
| <b>StaticText</b> .....                                     | 674 |
| <b>Fonts</b>  |     |
| <b>Font(index)</b> ▶ <b>String</b> .....                    | 235 |
| <b>FontCount</b> ▶ <b>Integer</b> .....                     | 235 |
| <b>FontsFolder</b> ▶ <b>FolderItem</b> .....                | 236 |
| <b>Printing</b>   |     |
| <b>OpenPrinter(pageSetup)</b> ▶ <b>Graphics</b> .....       | 437 |
| <b>OpenPrinterDialog(pageSetup)</b> ▶ <b>Graphics</b> ..... | 437 |
| <b>PrinterSetup Class</b> .....                             | 489 |
| <b>StyledTextPrinter Class</b> .....                        | 685 |
| <b>Regular Expressions</b>                                  |     |
| <b>RegEx Class</b> .....                                    | 567 |
| <b>RegExException Error</b> .....                           | 574 |
| <b>RegExMatch Class</b> .....                               | 574 |
| <b>RegExOptions Class</b> .....                             | 575 |
| <b>RegExSearchPatternException Error</b> .....              | 576 |
| <b>Strings</b>  |     |
| <b>&lt;</b> .....   | 15  |
| <b>&lt;=</b> .....  | 16  |
| <b>&lt;&gt;</b> .....                                       | 17  |
| <b>=</b> .....  | 18  |
| <b>&gt;</b> .....   | 19  |
| <b>&gt;=</b> .....  | 20  |
| <b>Asc(string)</b> ▶ <b>Integer</b> .....                   | 57  |
| <b>AscB(string)</b> ▶ <b>Integer</b> .....                  | 58  |
| <b>CDbl(string)</b> ▶ <b>Double</b> .....                   | 86  |
| <b>Chr(value)</b> ▶ <b>String</b> .....                     | 88  |
| <b>ChrB(value)</b> ▶ <b>String</b> .....                    | 89  |
| <b>CLong(string)</b> ▶ <b>Int64</b> .....                   | 92  |

|  |     |
|--|-----|
| CountFields(source,separator) ► Integer . . . . .                  | 114 |
| CountFieldsB(source,separator) ► Integer . . . . .                 | 114 |
| CStr(value) ► String . . . . .                                     | 117 |
| EndOfLine Class . . . . .  | 202 |
| Format(number, formatSpec) ► String . . . . .                      | 240 |
| InStr([start], source, find) ► Integer . . . . .                   | 297 |
| InStrB([start], source, find) ► Integer . . . . .                  | 298 |
| Join(sourceArray, delimiter) ► String . . . . .                    | 312 |
| Left(source, count) ► String . . . . .                             | 323 |
| LeftB(source, count) ► String . . . . .                            | 324 |
| Len(string) ► Integer . . . . .                                    | 325 |
| LenB(string) ► Integer . . . . .                                   | 325 |
| Lowercase(value) ► String . . . . .                                | 360 |
| LTrim(sourceString) ► String . . . . .                             | 361 |
| Mid(source, start, [length]) ► String . . . . .                    | 382 |
| MidB(source, start, [length]) ► String . . . . .                   | 383 |
| NthField(source, separator, fieldNumber) ► String . . . . .        | 412 |
| NthFieldB(source, separator, fieldNumber) ► String . . . . .       | 413 |
| Replace(sourceString, oldString, newString) ► String . . . . .     | 578 |
| ReplaceAll(sourceString, oldString, newString) ► String . . . . .  | 580 |
| ReplaceAllB(sourceString, oldString, newString) ► String . . . . . | 580 |
| ReplaceB(sourceString, oldString, newString) ► String . . . . .    | 579 |
| ReplaceLineEndings(sourceString, LineEnding) ► String . . . . .    | 581 |
| Right(source, count) ► String . . . . .                            | 589 |
| RightB(source, count) ► String . . . . .                           | 590 |
| RTrim(sourceString) ► String . . . . .                             | 593 |
| Split(source, delimiter) ► String array . . . . .                  | 652 |
| SplitB(source, delimiter) ► String array . . . . .                 | 652 |
| Str(number) ► String . . . . .                                     | 677 |
| StrComp(string1, string2, mode) ► Integer . . . . .                | 678 |
| String data type . . . . .   | 679 |
| StringShape Class . . . . .  | 681 |
| Titlecase(value) ► String . . . . .                                | 750 |
| Trim(sourceString) ► String . . . . .                              | 757 |
| Uppercase(value) ► String . . . . .                                | 770 |
| Val(string) ► Double . . . . .                                     | 773 |
| VarType (value) ► Integer . . . . .                                | 777 |
| <b>Styled Text</b>   |     |
| Paragraph Class . . . . .  | 460 |
| Range Class . . . . .  | 518 |
| StyledText Class . . . . .   | 681 |
| StyledTextPrinter Class . . . . .                                  | 685 |
| StyleRun Class . . . . .   | 686 |
| <b>Text Encoding</b>   |     |
| ConvertEncoding(str, newEncoding) ► String . . . . .               | 112 |
| DefineEncoding(str, enc) ► String . . . . .                        | 153 |

## User Interface

---

|  |     |
|--|-----|
| Encoding(str) ► TextEncoding .....                                   | 200 |
| Encodings Object .....   | 201 |
| GetFontTextEncoding(FontName) ► TextEncoding .....                   | 251 |
| GetInternetTextEncoding (InternetEncoding) ► TextEncoding.....       | 252 |
| GetTextConverter(inputEncoding, outputEncoding) ► TextConverter..... | 265 |
| GetTextEncoding (Base, Variant, Format) ► TextEncoding .....         | 266 |
| GuessJapaneseEncoding ► TextEncoding .....                           | 278 |
| TextConverter Class .....  | 714 |
| TextEncoding Class.....  | 720 |
| <b>XML</b>   |     |
| SOAPException Error .....  | 637 |
| SOAPMethod Class.....  | 637 |
| SOAPResult Class .....   | 640 |
| XmlAttribute Class .....   | 802 |
| XmlAttributeList Class.....  | 803 |
| XMLCDATASection Class .....  | 803 |
| XMLComment Class .....   | 804 |
| XMLContentModel Class .....  | 804 |
| XMLDocument Class.....   | 805 |
| XMLODException Class.....  | 809 |
| XMLElement Class .....   | 809 |
| XMLError Class .....   | 810 |
| XMLNode Class.....   | 811 |
| XMLNodeList Class .....  | 813 |
| XMLNodeType Object .....   | 813 |
| XMLProcessingInstruction Class.....                                  | 814 |
| XMLReader Class .....  | 814 |
| XMLReaderException Class .....                                       | 818 |
| XMLStyleSheet Class.....   | 819 |
| XMLTextNode Class .....  | 819 |
| XMLXsiHandler Class .....  | 819 |

## User Interface

|                              |     |
|------------------------------|-----|
| <b>Controls</b>              |     |
| BevelButton .....            | 64  |
| Canvas.....                  | 81  |
| Checkbox .....               | 87  |
| ClearFocus Method .....      | 90  |
| ComboBox .....               | 98  |
| ContainerControl Class ..... | 104 |
| ContextualMenu .....         | 108 |
| Control Class.....           | 111 |
| DatabaseQuery.....           | 131 |
| DataControl .....            | 134 |
| DisclosureTriangle .....     | 162 |

|                                       |     |
|---------------------------------------|-----|
| EditField . . . . .                   | 181 |
| GroupBox . . . . .                    | 277 |
| HTMLViewer . . . . .                  | 280 |
| ImageWell . . . . .                   | 294 |
| IsContextualClick ▶ Boolean . . . . . | 310 |
| Line . . . . .                        | 328 |
| ListBox . . . . .                     | 329 |
| MoviePlayer . . . . .                 | 387 |
| NotePlayer . . . . .                  | 409 |
| Oval . . . . .                        | 457 |
| PagePanel . . . . .                   | 459 |
| Placard . . . . .                     | 469 |
| PopupArrow . . . . .                  | 479 |
| PopupMenu . . . . .                   | 480 |
| ProgressBar . . . . .                 | 492 |
| ProgressWheel . . . . .               | 493 |
| Pushbutton . . . . .                  | 494 |
| RadioButton . . . . .                 | 514 |
| RB3DSpace . . . . .                   | 519 |
| RBScript . . . . .                    | 527 |
| Rectangle . . . . .                   | 553 |
| RectControl . . . . .                 | 554 |
| RoundRectangle . . . . .              | 591 |
| Scrollbar . . . . .                   | 603 |
| Separator . . . . .                   | 612 |
| Serial . . . . .                      | 613 |
| Slider . . . . .                      | 631 |
| SpotlightQuery . . . . .              | 654 |
| SpriteSurface . . . . .               | 659 |
| StandardToolbarItem . . . . .         | 669 |
| StaticText . . . . .                  | 674 |
| TabPanel . . . . .                    | 702 |
| TCPSocket . . . . .                   | 709 |
| ToolbarItem . . . . .                 | 751 |
| UpDownArrows . . . . .                | 770 |
| <b>Cursors</b>                        |     |
| Cursor Object . . . . .               | 117 |
| MouseCursor Class . . . . .           | 385 |
| <b>Menus</b>                          |     |
| AppleMenuItem Class . . . . .         | 46  |
| BevelButton Control . . . . .         | 64  |
| ContextualMenuItem Control . . . . .  | 108 |
| EnableMenuItem Method . . . . .       | 198 |
| IsContextualClick ▶ Boolean . . . . . | 310 |
| MenuItem Class . . . . .              | 372 |
| PopupMenu Control . . . . .           | 480 |

## Windows

---

|   |     |
|---|-----|
| PrefsMenuItem Class .....                         | 488 |
| QuitMenuItem Class.....                           | 514 |
| <b>Monitors/Screens</b>                           |     |
| ScreenCount ► Integer .....                       | 602 |
| <b>Toolbars</b>                                   |     |
| StandardToolbarItem Control.....                  | 669 |
| ToolbarItem Control .....                         | 751 |
| <b>Windows</b>                                    |     |
| ClearFocus Method .....                           | 90  |
| MDIWindow Class .....                             | 364 |
| MessageDialog Class .....                         | 377 |
| MessageDialogButton Class.....                    | 380 |
| MsgBox(message,[buttons],[title]) ► Integer ..... | 392 |
| Window Class .....                                | 784 |
| Window Function.....                              | 797 |
| WindowCount ► Integer.....                        | 799 |
| <b>Windows</b>                                    |     |
| Office Automation                                 |     |
| ExcelApplication Class .....                      | 203 |
| Office Class .....                                | 426 |
| OLEObject Class .....                             | 420 |
| PowerPointApplication Class.....                  | 484 |
| WordApplication Class.....                        | 799 |
| OLE   |     |
| OLE Parameter Class.....                          | 423 |
| OLEContainer Class .....                          | 419 |
| OLEException .....                                | 420 |
| OLEObject Class .....                             | 420 |
| Registry Items                                    |     |
| RegistryAccessErrorException Error .....          | 565 |
| RegistryItem Class.....                           | 565 |
| User Interface                                    |     |
| MDIWindow Class .....                             | 364 |
| TrayItem Class.....                               | 754 |
| <b>Windows</b>                                    |     |
| MDIWindow Class .....                             | 364 |