Type Conversion Functions

These functions are compiled inline, meaning the conversion code is part of the code that evaluates the expression. Execution is faster because there is no call to a procedure to accomplish the conversion. Each function coerces an expression to a specific data type.

General syntax is:

```
CDataType(expression)    example  CInt(expression)
```

The `expression` is required, may be any `String` expression or numeric expression.

Return Types
The function name determines the return type, as shown in the following table:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Return type</th>
<th>Range for expression argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBool</td>
<td>Boolean</td>
<td>Any valid <code>String</code> or numeric expression.</td>
</tr>
<tr>
<td>CByte</td>
<td>Byte</td>
<td>0 through 255; fractions are rounded.</td>
</tr>
<tr>
<td>CChar</td>
<td>Char</td>
<td>Any valid <code>String</code> expression; value can be 0 through 65535.</td>
</tr>
<tr>
<td>CDate</td>
<td>Date</td>
<td>Any valid representation of a date and time.</td>
</tr>
<tr>
<td>CDbl</td>
<td>Double</td>
<td>-1.79769313486231E+308 through -4.94065645841247E-324 for negative values; 4.94065645841247E-324 through 1.79769313486231E+308 for positive values.</td>
</tr>
<tr>
<td>CDec</td>
<td>Decimal</td>
<td>+/-79,228,162,514,264,337,593,543,950,335 for zero-scaled numbers, that is, numbers with no decimal places. For numbers with 28 decimal places, the range is +/-7.9228162514264337593543950335. The smallest possible non-zero number is 0.00000000000000000000000000000001.</td>
</tr>
<tr>
<td>CInt</td>
<td>Integer</td>
<td>-2,147,483,648 through 2,147,483,647; fractions are rounded.</td>
</tr>
<tr>
<td>CLng</td>
<td>Long</td>
<td>-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807; fractions are rounded.</td>
</tr>
<tr>
<td>CObj</td>
<td>Object</td>
<td>Any valid expression.</td>
</tr>
<tr>
<td>CShort</td>
<td>Short</td>
<td>-32,768 through 32,767; fractions are rounded.</td>
</tr>
<tr>
<td>CSng</td>
<td>Single</td>
<td>-3.402823E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.402823E+38 for positive values.</td>
</tr>
<tr>
<td>CStr</td>
<td>String</td>
<td>Returns for <code>CStr</code> depend on the <code>expression</code> argument.</td>
</tr>
</tbody>
</table>
Remarks

If the expression passed to the function is outside the range of the data type to which it is being converted, an error occurs.

In general, you can use the data type conversion functions to force the result of some operation to a particular data type rather than the default data type. For example, use **CDec** to force decimal arithmetic in cases where single-precision, double-precision, or integer arithmetic normally would occur.

When the fractional part is exactly 0.5, **CInt** and **CLng** always round it to the nearest even number. For example, 0.5 rounds to 0 and 1.5 rounds to 2. **CInt** and **CLng** differ from the **Fix** and **Int** functions, which truncate, rather than round, the fractional part of a number. Also, **Fix** and **Int** always return a value of the same type as is passed in.

Use the **IsDate** function to determine if a value can be converted to a date and time. **CDate** recognizes date literals and time literals as well as numbers that fall within the range of acceptable dates.

**CDate** recognizes date formats according to the locale setting of your system. You must provide the day, month, and year in the correct order for your locale, or the date may not be interpreted correctly. A long date format is not recognized if it contains a day-of-the-week string, such as "Wednesday".

The **Date** data type always contains both date and time information. For purposes of type conversion, Visual Basic .NET considers 1/1/1 (January 1 of the year 1) to be a neutral value for the date, and 00:00:00 (midnight) to be a neutral value for the time. If you convert a **Date** value to a string, **CStr** does not include neutral values in the resulting string. For example, if you convert #January 1, 0001 9:30:00# to a string, the result is "9:30:00 AM"; the date information is suppressed. However, the date information is still present in the original **Date** value and can be recovered with functions such as **DatePart**.

The **CType** function takes a second argument, **typename**, and coerces **expression** to **typename**, where **typename** can be any data type, structure, class, or interface.

Some Examples of the above functions:

**CInt**
This example uses the **CInt** function to convert a value to **Integer**.

Dim MyDouble As Double
Dim MyInt As Integer
MyDouble = 2345.5678
MyInt = CInt(MyDouble) ' MyInt is set to 2346.

**CByte**
This example uses the **CByte** function to convert an expression to a **Byte**.

Dim MyDouble As Double
Dim MyByte As Byte
MyDouble = 125.5678
MyByte = CByte(MyDouble) ' MyByte is set to 126.

**CDec**
This example uses the **CDec** function to convert a numeric value to **Decimal**.

Dim MyDouble As Double
Dim MyDecimal As Decimal
MyDouble = 10000000.0587
MyDecimal = CDec(MyDouble) ' Convert to Decimal.
CChar
This example uses the **CChar** function to convert a the first character of a **String** expression to a **Char** type.
Dim MyString As String
Dim MyChar As Char
MyString = "BCD" ' CChar converts only first character of string.
MyChar = CChar(MyString) ' MyChar is set to "B".
The input argument to **CChar** must be of data type **String**. You cannot use **CChar** to convert a number to a character, because **CChar** cannot accept a numeric data type. This example obtains a number representing a code point (character code) and converts it to the corresponding character. It uses **InputBox** to obtain the string of digits, **CInt** to convert the string to type **Integer**, and **ChrW** to convert the number to type **Char**.
Dim MyDigits As String ' Input string of digits to be converted.
Dim CodePoint As Integer ' Number to be represented as a character.
Dim MyChar As Char
MyDigits = InputBox("Enter code point of character:")
CodePoint = CInt(MyDigits) ' Convert entire string to Integer.
MyChar = ChrW(CodePoint) ' MyChar is set to Char value of code point.

CDbl
This example uses the **CDbl** function to convert an expression to **Double**.
Dim MyDec As Decimal
Dim MyDouble As Double
MyDec = 234.456784D ' Literal type character D makes MyDec a Decimal.
MyDouble = CDbl(MyDec * 8.2D * 0.01D) ' Convert result to a Double.

CStr
This example uses the **CStr** function to convert a numeric value to **String**.
Dim MyDouble As Double
Dim MyString As String
MyDouble = 437.324
MyString = CStr(MyDouble) ' MyString is set to "437.324".

CBool
This example uses the **CBool** function to convert expressions to **Boolean** values. If an expression evaluates to a nonzero value, **CBool** returns **True**; otherwise, it returns **False**.
Dim A, B, C As Integer
Dim Check As Boolean
A = 5
B = 5
Check = CBool(A = B) ' Check is set to True.
' ...
C = 0
Check = CBool(C) ' Check is set to False.