Visual Basic Upgrade Companion Version 8.1

What’s included

Included in this download are the following:

1. VBUC 8.1 installer
2. Sample code:
    a. A simple controls demo in VB6 you can migrate with the VBUC
    b. A before and after migration sample of a large ERP-style application (Salmon King Seafood). Includes a version migrated to native web app with WebMAP.

Release Notes

This release of the VBUC contains performance improvements, bug fixes, and improvements to language mapping from VB6 to C#. Some details on specific improvements follow:

1. Collection type inference for inconsistent usages

Original VB6 code:

```
Dim x As New Collection

    Call x.Add("1", "Value1")
    Call x.Add("2", "Value2")

Dim nodeKey As String
Dim myTxt As String

    nodeKey = "Value2"
    myTxt = Left(nodeKey, x(nodeKey))

    MsgBox myTxt
```

Migrated C# code:

```
OrderedDictionary x = new OrderedDictionary(System.StringComparer.OrdinalIgnoreCase);

x.Add("Value1", "1");
x.Add("Value2", 2);
string nodeKey = "Value2";

string myTxt = nodeKey.Substring(0,
Math.Min(ReflectionHelper.GetPrimitiveValue<int>(x[nodeKey]), nodeKey.Length));

MessageBox.Show(myTxt, Application.ProductName);
```

2. Intrinsic alignment support improvement for intrinsic controls

Original VB6 code:

```
Select Case Label7.Alignment
      Case VBRUN.AlignmentConstants.vbCenter:
          MsgBox "Alignment is Center"
      Case VBRUN.AlignmentConstants.vbLeftJustify:
          MsgBox "Alignment is Left Justify"
      Case VBRUN.AlignmentConstants.vbRightJustify:
          MsgBox "Alignment is Right Justify"
      Case Else:
          MsgBox "Unexpected Alignment value"
  End Select

  Select Case Me.Check3.Alignment
      Case AlignmentConstants.vbLeftJustify
          Me.Check3.Alignment = AlignmentConstants.vbRightJustify
          Me.Option4.Alignment = AlignmentConstants.vbRightJustify
      Case AlignmentConstants.vbRightJustify
          Me.Check3.Alignment = AlignmentConstants.vbLeftJustify
          Me.Option4.Alignment = AlignmentConstants.vbLeftJustify
  End Select
```

Migrated C# code:

```
switch(UpgradeHelpers.Gui.TextAlignmentHelper.ToHorizontal(Label7.TextAlign))
{
    case HorizontalAlignment.Center :
        MessageBox.Show("Alignment is Center", Application.ProductName);
        break;
    case HorizontalAlignment.Left :
        MessageBox.Show("Alignment is Left Justify", Application.ProductName);
        break;
    case HorizontalAlignment.Right :
        MessageBox.Show("Alignment is Right Justify", Application.ProductName);
        break;
    default:
        MessageBox.Show("Unexpected Alignment value", Application.ProductName);
    break;
}

switch(UpgradeHelpers.Gui.TextAlignmentHelper.ToHorizontal(this.Check3.CheckAlign))
{
    case HorizontalAlignment.Left :
        this.Check3.CheckAlign = ContentAlignment.MiddleRight;
        this.Option4.CheckAlign = ContentAlignment.MiddleRight;
        break;
    case HorizontalAlignment.Right :
        this.Check3.CheckAlign = ContentAlignment.MiddleLeft;
        this.Option4.CheckAlign = ContentAlignment.MiddleLeft;
        break;
}
```

3. Array declaration and initialization improvements

Original VB6 code:

```
Dim fixedstring() As String * 5
ReDim fixedstring(1 To 10)
```

Migrated C# code:

```
string[] fixedstring = ArraysHelper.InitializeArray<string[]>(new int[]{10}, new object[]{5});
```

4. Improvement for the definition and initialization of elements which have a constructor with parameters

Original VB6 code:

```
Dim rs As New ADODB.Recordset
```

Migrated C# code:

```
ADORecordSetHelper rs = new ADORecordSetHelper("");
```

5. Optional parameters default value improvements

Original VB6 code:

```
Private Sub foo(Optional ByVal x)
    MsgBox IsMissing(x)
    If Not IsMissing(x) Then
        MsgBox x + "Hello World"
    End If
    Mark x
    x = "!!!"
End Sub
Private Sub goo(Optional x As String)
    MsgBox IsMissing(x)
    If Not IsMissing(x) Then
        MsgBox x + "Hello World"
    End If
    Mark x
End Sub
Public Sub Mark(x As String)
    MsgBox x
End Sub
```

Migrated C# code:

```
private void foo(string x = "")
{
    //UPGRADE_WARNING: (2065) Boolean method Information.IsMissing has a new behavior. More Information: https://www.mobilize.net/vbtonet/ewis/ewi2065
    MessageBox.Show((x == null).ToString(), Application.ProductName);
    //UPGRADE_WARNING: (2065) Boolean method Information.IsMissing has a new behavior. More Information: https://www.mobilize.net/vbtonet/ewis/ewi2065
    if (x != null)
    {
        MessageBox.Show(x + "Hello World", Application.ProductName);
    }
    Mark(x);
    x = "!!!";
}
private void goo(string x = "")
{
    MessageBox.Show((false).ToString(), Application.ProductName);
    if (true)
    {
        MessageBox.Show(x + "Hello World", Application.ProductName);
    }
    Mark(x);
}
public void Mark(string x)
{
    MessageBox.Show(x, Application.ProductName);
}
```

6. Numeric types decimal-double casting improvements

Original VB6 code:

```
If (CDec(strLOC1) >= 11 And CDec(strLOC1) <= 11.9) Or (CDec(strLOC1) >= 18 And CDec(strLOC1) <= 18.6) Then
    '
Else
    '
End If
```

Migrated C# code:

```
object strLOC1 = null;

if ((ReflectionHelper.GetPrimitiveValue<decimal>(strLOC1) >= 11 &&
ReflectionHelper.GetPrimitiveValue<decimal>(strLOC1) <= 11.9m) ||
(ReflectionHelper.GetPrimitiveValue<decimal>(strLOC1) >= 18 &&
ReflectionHelper.GetPrimitiveValue<decimal>(strLOC1) <= 18.6m))
{
    //
}
else
{
    //
}
```

7. VB6 enumeration with singular names support

Original VB6 code

```
Public Enum anEnum
    [Very Good] = 1
    [No Bad]
End Enum

Public Enum Cars
    [4F Ford] = 0
    Chevrolet = 1
    [Chrysler 300] = 2
    [205 Honda Civic] = 3
    Subaru = 4
End Enum

    Dim e As anEnum
    e = [No Bad]
    e = [Very Good]

    Dim c As Cars
    c = [205 Honda Civic]
    c = [4F Ford]
    c = Chevrolet
    c = [Chrysler 300]
    c = Subaru
```

Migrated C# code:

```
public enum anEnum
{
    Very_Good = 1,
    No_Bad
}

public enum Cars
{
    _4F_Ford = 0,
    Chevrolet = 1,
    Chrysler_300 = 2,
    _205_Honda_Civic = 3,
    Subaru = 4
}

    anEnum e = Form1.anEnum.No_Bad;
    e = Form1.anEnum.Very_Good;

    Cars c = Form1.Cars._205_Honda_Civic;
    c = Form1.Cars._4F_Ford;
    c = Cars.Chevrolet;
    c = Form1.Cars.Chrysler_300;
    c = Cars.Subaru;
```

8. Nested loops improvement

Since C# does not have a mechanism to exit a specified loop, both `Exit For` and `Exit Loop` would generate a break statement that would exit the first loop it found. This would be a problem when loops were nested, and the `Exit` instruction was meant for the outermost loop. This behavior has been fixed and the correct loop will be exited.

Original VB6 code:

```
For x = 1 To 10
    Do
        Exit For
    Loop While x < 10
Next x
```

Migrated C# code:

```
for (int x = 1; x <= 10; x++)
{
    do
    {
        goto exit_for;
    }
    while(x < 10);
}
exit_for:;
```

9. Improved detection of COM objects that should be released when method ends

Original VB6 code:

```
Dim a As VBCOMClass
Dim v As Integer
If (b) Then
  Set a = New VBCOMClass

End If
v = 2
If (b) Then
   Set a = Nothing
End If
```

Migrated C# code:

```
VBCOMLibrary.VBCOMClass a = null;
try
{

    a = null;
    if (b)
    {
        a = new VBCOMLibrary.VBCOMClass();
    }
    int v = 2;
    if (b)
    {
        a = MemoryHelper.ReleaseAndCleanObject<VBCOMLibrary.VBCOMClass>(a, null);
    }
}
finally
{
    MemoryHelper.ReleaseAndCleanObject(a);
}
```

10. Improved fixed length string support

New helper functions replicate the original VB6 functionality in .NET

Original VB6 code:

```
Private Sub Foo()
   Dim str As String * 10
   str = "ABCDE"
   MsgBox "Value: '" + str + "'"
   MsgBox "Length: " + CStr(Len(str))

   str = "0123456789ABCDE"
   MsgBox "Value: '" + str + "'"
   MsgBox "Length: " + CStr(Len(str))

   Foo str
   MsgBox "Value: '" + str + "'"
   MsgBox "Length: " + CStr(Len(str))
End Sub
```

Migrated C# code:

```
private void Foo() {
   string str = new string('\0', 10);
    str = StringsHelper.GetFixedLengthString("ABCDE", 10);
        MessageBox.Show("Value: '" + str + "'");
        MessageBox.Show("Length: " + Strings.Len(str).ToString());

    str=StringsHelper.GetFixedLengthString("0123456789ABCDE", 10);
        MessageBox.Show("Value: '" + str + "'");
        MessageBox.Show("Length: " + Strings.Len(str).ToString());

   Foo(ref str);
        str = StringsHelper.GetFixedLengthString(str, 10);
        MessageBox.Show("Value: '" + str + "'");
        MessageBox.Show("Length: " + Strings.Len(str).ToString());
}
```

11. Miscellaneous improvements and bug fixes:

- Improved struct declarations
- Improved comparisons between currency and `ADODB.Recordset.Field` types
- Improved conversion from twips to pixels
- Improved type correction from `object` or `variant` to user-defined type (still requires manual library reference fixups).