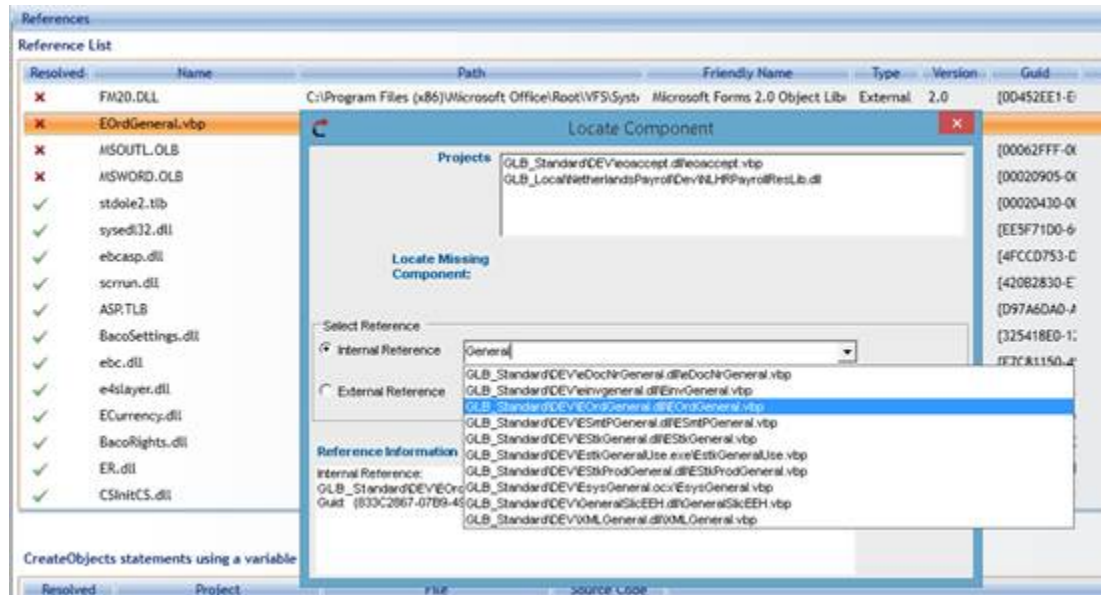


VBUC 8.2 – Release Notes

UI Enhancements

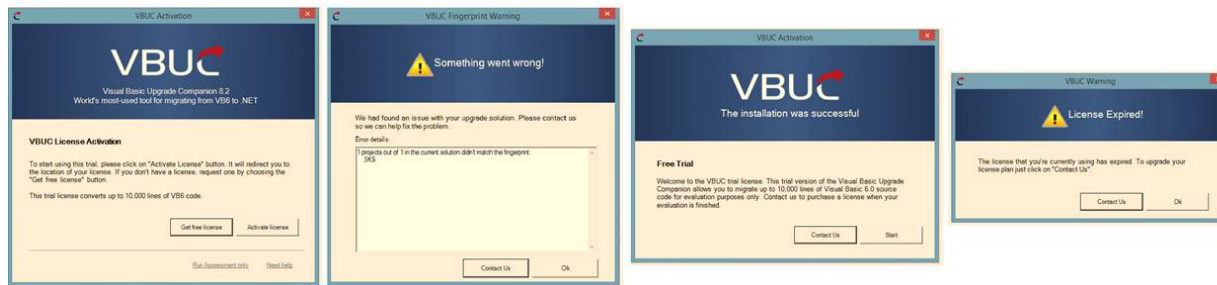
- **New filters make setting up a VBUC solution easier.** When setting up the solution for the first time, some things might need tuning; among them are the references.
 - **Searching for internal references.** When the VBUC creates a migration solution, it might not be able to find hints that a reference must be set as internal. With VBUC 8.2, a search among all the Visual Basic project files (*.vbp) can be made to easily find the corresponding VB project for a given reference.



- **Searching for Unresolved CreateObjects.** *CreateObjects* are ways to dynamically instantiate classes in VB6, without having explicit references to the binaries where those classes live. In the new release of the VBUC, a search in all these statements can be made, in order to group some of them to easily set the binary in multiple VB projects.

Unresolved CreateObjects			
Resolved	Project	File	Source Code
✗	GLB_BuildingBlocks\SED008_SE5331\CSEDIImportOrders.exe\CSEDIImport	GLB_BuildingBlocks\SED008_SE5331\CSEDIImportOrders.e	Set oEnvTax = CreateObject("ElmvGeneral.LinkItemHandler")
✗	GLB_Standard\DEV\EPrtPEVHdl.dll\EPrtPEVHdl.vbp	GLB_Standard\DEV\EPrtPEVHdl.dll\clsChangeSO.cls	Set oPrintSO = CreateObject("ElmvGeneral.PrintSOCreateInv")
✗	GLB_Standard\DEV\VMGlobe.dll\VMGlobe.vbp	GLB_Standard\DEV\VMGlobe.dll\Order.cls	If m_QueryInvoice Is Nothing Then Set m_QueryInvoice = CreateObject("ElmvGeneral.QueryInvoice")

- Licensing process improved.** The VBUC 8.2 automatically downloads and installs a trial license for first-time users. Also, all licensing windows have been redesigned to better inform the user about the specific situation and present helpful messages.



- Upgrade Options documentation:** We've improved all the Upgrade Options documentation. Every Upgrade Option Configuration has a description and better code samples in Visual Basic 6, C# and VB.NET.

Upgrade Options

The following table provides options to customize the generated code.

The screenshot displays the Visual Basic Upgrade Companion 8.2 (Beta) application. The 'Upgrade Options' window is open, showing a list of features on the left and a detailed description for the 'Scripting' option. The 'Scripting' option is selected, and its description explains that it maps the 'Scripting Dictionary' class to a .NET 'OrderedDictionary'. The interface also shows source code examples for both Visual Basic 6 and C#.

Scripting

The VBUC includes the following options to convert:

Conversion Options:

- To COM Interop
- To .NET classes

Description

This feature maps the "Scripting Dictionary" class to a .NET "OrderedDictionary". A dictionary is an object that stores key/item pairs. Each unique key which is usually an integer or a string. For full information about the vb6 class you can visit <https://docs.microsoft.com/en-us/office/user-interface-help/dictionary-object>

Remarks:

- This option converts the "Scripting Dictionary" class to a .NET "OrderedDictionary". Members are also mapped to OrderedDictionary members.

General Description:

An OrderedDictionary is a dictionary (hash table) that preserves the order (as VB6 does) in which the keys are inserted. A regular dictionary: order.

Visual Basic 6 Source Code Examples

```
Public Sub Foo(dict As Scripting.Dictionary)
    Call dict.Add("A", 1) : Call dict.Add("B", 2) : Call dict.Add("C", 3)
    If dict.Contains("C") Then
        MsgBox dict.Item("C")
    End If
    For Each value In dict.Keys
        MsgBox value & " prints A B C"
    Next
    For Each value In dict.Items
        MsgBox value & " prints 1 2 3"
    Next
End Sub
```

C# Source Code Examples

```
public static void Foo(OrderedDictionary dict)
{
    dict.Add("A", 1); dict.Add("B", 2); dict.Add("C", 3);
    if (dict.Contains("C"))
    {
        MessageBox.Show(Convert.ToString(dict["C"]));
    }
    foreach (object value in dict.Keys)
    {
        MessageBox.Show(Convert.ToString(value)); //prints A B C
    }
    foreach (object value in dict.Values)
    {
        MessageBox.Show(Convert.ToString(value)); //prints 1 2 3
    }
}
```

VB.NET Source Code Examples

```
Public Sub Foo(vb6 dict As OrderedDictionary)
    dict.Add("A", 1) : dict.Add("B", 2) : dict.Add("C", 3)
    If dict.Contains("C") Then
        MessageBox.Show(Convert.ToString(dict.Item("C")))
    End If
    For Each value As Object In dict.Keys
        MessageBox.Show(value, My.Application.Info.Title) & " prints A B C"
    Next value
    For Each value As Object In dict.Values
        MessageBox.Show(Convert.ToString(value), My.Application.Info.Title) & " prints 1 2 3"
    Next value
End Sub
```

Code Quality Enhancements

The quality of the generated code has been significantly improved. This is due to bug fixes and newly-implemented features.

Next are just a couple of these enhancements.

- **GoSub to local function.** *GoSub...Return* statements branch to and return from a subroutine within a procedure. The VBUC now supports this VB6 feature via local functions in .NET.

VB6 Code

```
Public Sub Foo(p As Integer)
    Dim value As String

    If p = 1 Then
        GoSub Lbl1
    Else
        GoSub Lbl2
    End If

    MsgBox value
    Exit Sub

Lbl1:
    value = "First value"
    Return

Lbl2:
    value = "Second value"
    Return
End Sub
```

.NET Code

```
public void Foo(int p)
{
    string value = "";
    if (p == 1)
    {
        Lbl1();
    }
    else
    {
        Lbl2();
    }

    MessageBox.Show(value);
    return;
}
```

```

void Lbl1()
{
    value = "First value";
}

void Lbl2()
{
    value = "Second value";
}
}

```

- **Improved criteria for declaring indexers.** The default property of a class does not have to be called *Item* and take an *integer* as parameter to be transformed into an indexer in .NET. If it has the attribute shown below, it can potentially be migrated as an indexer.

VB6 Code

```

'The return parameter type does not have to be integer
[Hidden] Attribute Item.VB_UserMemId = 0
Property Get MyItem(anyName as string) as string
    ...
End Property

```

.NET Code

```

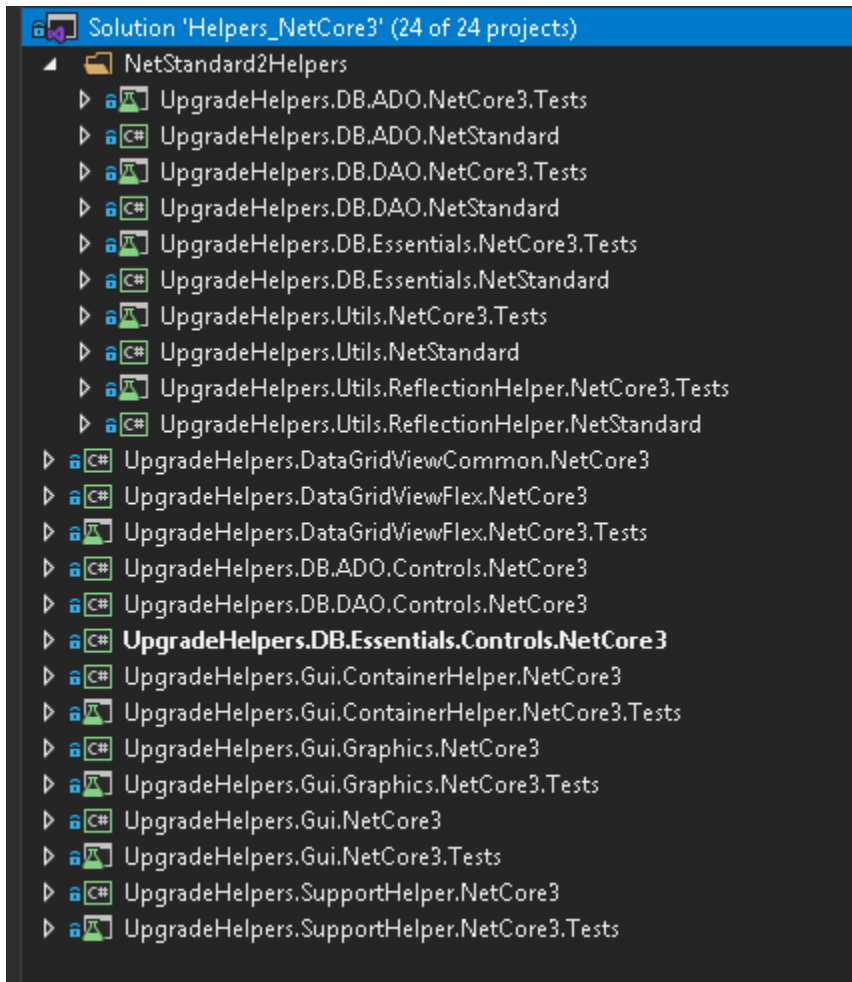
// The return parameter type does not have to be integer
public string this[string anyName]
{
    get { ... }
    set { ... }
}

```

Upgrade Helpers Improvements

The Upgrade Helpers have also been improved. Here is a summary of the changes made.

- **Existing Helpers Improvements.** The following Helpers classes have been enhanced: TrueDBGrid helper, Farpoint spread helper, Flexgrid helper, StringsHelper, Printer Helper, XArrayHelpers, Crystal Reports helper.
- **New Helpers.**
 - **IOHelper**, a utility class that provides functionality related to Input/Output operations.
 - **DBITech CtdEdit Helper**, a third-party control.
 - **Component One C1Report Helper**, a third-party control.
 - **Crystal Report Helper**, a third-party control.
- **First version of .NET Core helpers.** We've chosen a priority sub-set of helpers to have a .NET Core 3.1 version as a starting point to be .NET Core Compliant. The following list corresponds to the helpers that already have a working .NET Standard or .NET Core Version.



Other Improvements

- Improved comparison of colors with other colors, enums, and numeric values.
- ByRef to ByVal rules improvements.
- Remove unnecessary castings in basic operations.
- Performance improvements in the ASP migrator.
- Control arrays events and indexing improvements.
- New mappings for MQAX and MSMQ third party controls.
- New mappings for Ambient properties.
- Saving user-defined-events info during preprocess to improve code quality.