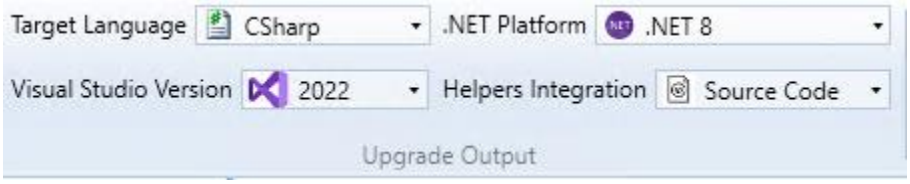
1. .NET 8 Support

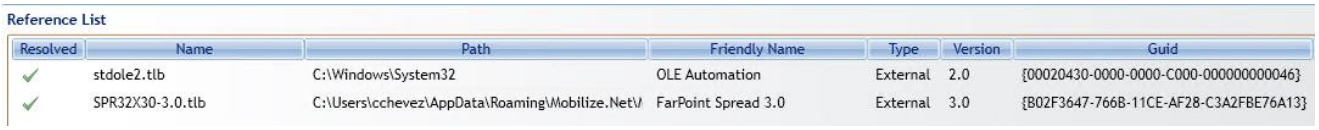
With the recent release done by Microsoft, we are proud to announce the support for .NET 8 platform. This option will allow the user to generate code using this version as target, the option is available for C# and VB.NET languages.



Note: .NET 8 platform is available only when selecting Visual Studio 2022 as target.

2. Missing References Feature

To have the best code quality, the VBUC tool needs to have all the references solved, however, this could be hard to achieve especially when the VB6 code was moved to a new machine, some libraries may be hard to install in the newest OS or the provider could no longer exist.
In order to reduce the complexity when using the VBUC, a new feature was developed, with this improvement the VBUC will try to solve all the missing references in the machine where it is installed in order to let the user upgrade the code to .NET even when some references are not present in the environment. A .tlb file will be copied to the machine with all the information that the VBUC needs to generate the code in the better way.


Reference List

Resolved	Name	Path	Friendly Name	Type	Version	Guid
✓	stdole2.tlb	C:\Windows\System32	OLE Automation	External	2.0	{00020430-0000-0000-C000-000000000046}
✓	SPR32X30-3.0.tlb	C:\Users\cchevez\AppData\Roaming\Mobilize.Net\V	FarPoint Spread 3.0	External	3.0	{B02F3647-766B-11CE-AF28-C3A2FBE76A13}

Notes: The server with the references will be populated by demand, and not all the references may be available to be downloaded.
This feature is currently only available on 64-bit OS.

Disclaimer: The reference will allow the VBUC to upgrade the code, however, some compilation errors may be shown if the license or reference for the target component does not exist in the machine where the .NET code is generated.

3. C# Features Implementation

Three different features were implemented in order to improve the code generation when selecting C#.

3.1.Discards

Discards make the intent of your code clear. A discard indicates that our code never uses the variable and, they enhance its readability and maintainability.

Note: The discards will be automatically generated when selecting Visual Studio 2017 or superior as target.

- **Current VB6 Code:**

```
'Discards
Dim a As Integer
Dim b As Integer
Dim c As Integer

a = 5
b = 6
c = 10
d = a + b + c

Dim str As String
str = ReturnString
Dim myBool As Boolean

d = UsingDiscardParameteres(10, "Test", myBool)

Dim ctl As Collection
Set ctl = New Collection
e = DiscardInSwitch(ctl)
```

- **Previous C# Code:**

```
//Discards

int a = 5;
int b = 6;
int c = 10;
int d = a + b + c;

string str = ReturnString();
bool myBool = false;

d = UsingDiscardParameteres(10, "Test", myBool);

OrderedDictionary ctl = new OrderedDictionary(System.StringComparer.OrdinalIgnoreCase);
string e = DiscardInSwitch(ctl);
```

- **Current C# Code:**

```
//Discards

int a = 5;
int b = 6;
int c = 10;
_ = a + b + c;

_ = ReturnString();
bool myBool = false;

_ = UsingDiscardParameteres(10, "Test", myBool);

OrderedDictionary ctl = new OrderedDictionary(System.StringComparer.OrdinalIgnoreCase);
_ = DiscardInSwitch(ctl);
```

3.2. Interpolated Strings

Before C# 10 it was possible to concatenate strings, the string interpolation functionality existed but did not allow interpolated string constants. this is now possible with C# 10 for readability reasons.
Notes: When targeting .NET Framework 4.6 or superior, this feature will be applied for strings in the code distinct from constants.
When targeting .NET 6 or superior, this feature will be applied to constants also.

- **Current VB6 Code:**

```
Const bookPrice = 24.99
Const BookName = "The Lord of The Rings: The Two Towers"

Const VideoGame = "Mortal Kombat "
Const ONE = 1
Const TWO = 2
Const TEN = "X"
Const Version1 = VideoGame & ONE
Const Version2 = VideoGame & TWO
Const Version10 = VideoGame & TEN

Dim str02 As String
Dim fullName As String
```

```
Sub main()
    'Interpolated Strings
    str02 = "Book '" & BookName & "' price is " & bookPrice

    Name = "Peter"
    lastName = "Jackson"
    fullName = Name + " " + lastName
```

- **Previous C# Code:**

```
const double bookPrice = 24.99d;
const string BookName = "The Lord of The Rings: The Two Towers";

const string VideoGame = "Mortal Kombat ";
const int ONE = 1;
const int TWO = 2;
const string TEN = "X";
static readonly string Version1 = VideoGame + ONE.ToString();
static readonly string Version2 = VideoGame + TWO.ToString();
const string Version10 = VideoGame + TEN;

static string str02 = "";
static string fullName = "";
```

```
0 references
public static void Main()
{
    //Interpolated Strings
    str02 = "Book '" + BookName + "' price is " + bookPrice.ToString();

    string Name = "Peter";
    string lastName = "Jackson";
    fullName = Name + " " + lastName;
```

- **Current C# Code Targeting .NET Framework 4.8:**

```
const double bookPrice = 24.99d;
const string BookName = "The Lord of The Rings: The Two Towers";

const string VideoGame = "Mortal Kombat ";
const int ONE = 1;
const int TWO = 2;
const string TEN = "X";
static readonly string Version1 = VideoGame + ONE.ToString();
static readonly string Version2 = VideoGame + TWO.ToString();
const string Version10 = VideoGame + TEN;

static string str02 = "";
static string fullName = "";

0 references
public static void Main()
{
    //Interpolated Strings
    str02 = $"Book '{BookName}' price is {bookPrice.ToString()}";

    string Name = "Peter";
    string lastName = "Jackson";
    fullName = $"{Name} {lastName}";
```

- **Current C# Code Targeting .NET 8:**

```
const double bookPrice = 24.99d;
const string BookName = "The Lord of The Rings: The Two Towers";

const string VideoGame = "Mortal Kombat ";
const int ONE = 1;
const int TWO = 2;
const string TEN = "X";
static readonly string Version1 = $"{VideoGame}{ONE.ToString()}";
static readonly string Version2 = $"{VideoGame}{TWO.ToString()}";
const string Version10 = VideoGame + TEN;

static string str02 = "";
static string fullName = "";

0 references
public static void Main()
{
    //Interpolated Strings
    str02 = $"Book '{BookName}' price is {bookPrice.ToString()}";

    string Name = "Peter";
    string lastName = "Jackson";
    fullName = $"{Name} {lastName}";
```

3.3. Improved Pattern Matching

C# 9 allows you to use relational pattern which enables the use of <, >, <= and >= in patterns, however, this improvement will be applied to some patterns only.

Note: This improvement will be applied automatically when targeting .NET 5 or superior.

- **Current VB6 Code:**

```
Public Function GetTax1(p As Product) As Integer
    Select Case p.CategoryID
        Case 1
            GetTax1 = 0
        Case Is < 5
            GetTax1 = 5
        Case Is > 20
            GetTax1 = 15
        Case Else
            GetTax1 = 10
    End Select
End Function
```

```
Public Function GetTax2(p As Product) As Integer
    Select Case p.CategoryID
        Case 0 Or 1
            GetTax2 = 0
        Case 2 To 5
            GetTax2 = 5
        Case Is > 20
            GetTax2 = 15
        Case Else
            GetTax2 = 10
    End Select
End Function
```

- **Previous C# Code:**

```
internal static int GetTax1(Product p)
{
    int switchVar = p.CategoryID;
    if (switchVar == 1)
    {
        return 0;
    }
    else if (switchVar < 5)
    {
        return 5;
    }
    else if (switchVar > 20)
    {
        return 15;
    }
    else
    {
        return 10;
    }
}
internal static int GetTax2(Product p)
{
    int switchVar = p.CategoryID;
    if (switchVar == (0 | 1))
    {
        return 0;
    }
    else if (switchVar >= 2 && switchVar <= 5)
    {
        return 5;
    }
    else if (switchVar > 20)
    {
        return 15;
    }
    else
    {
        return 10;
    }
}
```

- **Current C# Code:**

```
internal static int GetTax1(Product p)
{
    switch(p.CategoryID)
    {
        case 1 :
            return 0;
        case < 5 :
            return 5;
        case > 20 :
            return 15;
        default:
            return 10;
    }
}
internal static int GetTax2(Product p)
{
    switch(p.CategoryID)
    {
        case 0 | 1 :
            return 0;
        case >= 2 and <= 5 :
            return 5;
        case > 20 :
            return 15;
        default:
            return 10;
    }
}
```

4. Other Improvements include but are not limited to:

- Databases mappings
- COM Interop improvements
- Microsoft controls mappings
- Grids mappings
- GoTo/GoSub feature improvements
- Helpers improvements to reduce SQL vulnerabilities