Visual Basic Upgrade Companion Version 6.5 Release Notes

Application Fingerprint

A VBUC license allows you to migrate a single application to .NET. You can, however, modify that application in order to improve your migration outcome. VBUC 6.5 implements a fingerprint of the application which it compares to the fingerprint generated at the time of purchase for the application you wish to migrate. You can modify your application and rerun the migration to compare results. The VBUC will verify that the fingerprint of the modified code matches the original fingerprint in the license. Normal code modifications should not cause issues; however if you modify the code to the extent that the fingerprint no longer matches the VBUC will not migrate your code. In this case please contact support.

Enhanced Compilation

A large number of improvements have been made in this release to reduce post-migration compile errors. Using several million lines of real world code, we've experienced an average reduction of 84% compared to VBUC version 6.4 (which itself already made big improvements in compilation over 6.3).

Migration Performance

This version has been tweaked to better handle large application migrations, improving net migration time by 2-5X. Smaller apps may not see a noticeable jump in the time it takes to migrate compared to prior versions due to startup overhead.

References Resolution

The VBUC is now able to locate project and class references that prior to this version would require manual resolution. Some of these references in very large projects can be difficult to understand and resolve manually but Version 6.5 can now solve many of these automatically.

Shared Files

Projects with shared files can now be migrated with more readability in the resulting code. In prior versions of VBUC, shared files were migrated using a namespace as in:

```
X=MySolutionShares.Module1.MyEnum.field1
```

This made for code that was difficult to read, but is no longer necessary in Version 6.5.

Data Access Helper Classes

Improvements were made to `FieldHelper`, `FieldsHelper`, and `RecordsetHelper` classes and subclasses to improve data access. Additionally, the classes were refactored to be more objected-oriented, with improvements in abstraction and encapsulation. As a result, better quality output code is being generated with a higher degree of automation. Some specifics:

- Fields can now be accessed through a key that cannot be distinguished as integer or string
- The following members are now supported in ADODB, ADOR, and DAO:
  - Recordset.Fields
  - TableDef.CreateField

  - Fields.Append
  - Fields.Delete
  - Fields.Count
  - Fields.Item (access by position, key, or ambiguous)
  - Field
  - Field properties and methods.
- Different ways of creating and access fields can now be combined, including using field metadata (not all metadata is supported).

Improved VB.NET Code Generation

Major improvements were made to the quality of projects targeting VB.NET, resulting in improved output code and reduced compilation errors.

Managed Type Libraries

Version 6.5 adds improved support for recognizing and converting managed TLB libraries that were created in .NET and consumed by VB6 code. Now expressions that include members from managed TLB libraries will migrate much better: VBUC will recognize that the external COM library is something that actually exists in .NET. For example, using COM interop, some properties are converted to get_/set_ methods:

Using Com Interop: `get_MyProp() | set_MyProp(object)`

If detected as managed TLB: `MyProp | object  = MyProp`

Collections Keys

VB6 collections can be accessed via a key, which can be an integer (basically identical to an array index) or a string. Previous versions of the VBUC did not handle cases where the key type could not be identified at compile time, requiring extensive manual effort post-migration. Version 6.5 now handles both cases automatically.

For example, consider the case:

```
Dim c as new Collection()
c.Add(“Key1”, “Data1”)
msgbox c(1)
msgbox c(“Key1”)
Variant v = GetSomeKeyOrIndex()
msgbox c(v)
```

The VBUC will now handle the last line in the snippet above, along with other cases of generic collections being accessed by using a variant parameter.

Menu Item Control Arrays and Separators

Previous versions of VBUC could not convert control arrays containing combinations of menu items and separators. Now the VBUC can correctly analyze the contents of these kinds of control arrays and create .NET classes for both menu items and separators. This reduces the amount of manual effort to migrate these kinds of cases. Example:

```
ToolStripItem[] menuArray = new ToolStripItem[n];
…
menuArray[0] = myToolStripMenuItem;
menuArray[1] = myToolStripSeparator;
…
((ToolStripMenuItem)menuArray[0]).DropDownItems.GetEnumerator()…
```

Platform Support

VBUC 6.5 now fully supports Windows 10 and Visual Studio 2015.

Bug Fixes

Over 130 bugs were fixed in this release.