WHITE PAPERS／ホワイトペーパー

Migrating a VB6 application in 10 easy steps／10 ステップで簡単に行う VB6 アプリケーション移行

- Each migration from VB6 is a story of its own, with its unique assortment of challenges and obstacles. At the same time, there all migration successful stories have many traits in common. Typically all of them can be subdivided in a few distinct steps, starting with the initial assessment, proceeding towards the zero-compilation-error and zero-runtime-error stages, and ending with the refinement and optimization stages. This whitepaper analyzes the entire process and shows which feature of VB Migration Partner can be most useful during each of these steps.

VB6 からの移行は案件毎に状況が異なり、それぞれの課題と障害の組合せが存在します。と同時に、成功事例には多くの共通点があります。通常全ての案件は、評価工程に始まり、コンパイル・エラーとルーティン・エラーを撲滅する工程、微調整と最適化の工程等、2、3 の異なるステップに分けられます。このホワイトペーパーは、VB Migration Partner の工程全体を分析し、それぞれの工程毎に、VB Migration Partner のどの機能が最も役立つかを示しています。

Comparing VB Migration Partner with Upgrade Wizard／VB Migration Partner とアップグレード・ウィザードの比較

- Why should you adopt VB Migration Partner when Visual Studio comes with a free VB6 conversion tool from Microsoft? Well, there isn't such a thing as a free lunch or a free conversion software. To prove our point, we did a very simple thing: we ran the Upgrade Wizard and VB Migration Partner on the same set of VB6 open source projects, which represent a good mix of the challenges you are going to face when porting your code to VB.NET, including database access, data binding, graphics, Windows API calls, and more. The results are stunning: even before adding a single migration pragma, our tool beats the one in Visual Studio 5 to 1.

Microsoft 社の Visual Studio は無料の VB6 変換ツールが備わっているのに、なぜ VB Migration Partner を導入する必要があるのでしょうか。それは、つまり無料の変換ツールで十分満足いくものは存在しないということです。弊社の主張を証明する為に、とても単純な試みをしてみました。アップグレード・ウィザードと VB Migration Partner を使って同じ VB6 のオープン・ソース・プロジェクトを変換してみたのです。そこには、データベース・アクセスや、データ・バインディング、グラフィッ

クス、Windows API コール等々を含む、コードを VB.NET へ移行する時に直面する豊富な事例が含まれていました。結果は、驚くべきものでした。変換の為のプラグマを 1 つも加えずに、我々のツールは、Visual Studio において 5 対 1 で勝利したのです。

Migration tools: Feature Comparison Table／移行ツール同士の機能比較テーブル

- OK, you took the decision of migrating your VB6 code to .NET using an automated tool. Your next step is selecting the right product for your needs. To help you in the decision process, we built a detailed feature table that compares the strengths and weaknesses of the three more popular VB6 conversion products on the market: Microsoft Upgrade Wizard (the free one), Artinsoft VB Upgrade Companion, and Code Architects VB Migration Partner. Rather than being a dry list of features, however, this document explains why (and if) a given characteristic is useful during the migration process, so that you can make an informed choice.

仮に、自動化されたツールを使って VB6 のコードを.NET へ移行すると決めたとしましょう。すると次には、その決定した方法に見合った製品を選択しなければなりません。決定のプロセスにおいてお役に立てるよう、詳細な機能比較表をご用意しました。この表を使えば、市場で知名度の高い 3 種類の VB6 移行ツールの長所と短所を比較することが出来ます。その移行ツールとは、Microsoft 社のアップグレード・ウィザード(無料)と、Artinsoft 社の VB アップグレード・コンパニオンと、Code Architects 社の VB Migration Partner です。この 3 商品の機能比較表は、無味乾燥な単なる機能の一覧ではありません。これを見れば、その特性が移行に役立つかどうか、又その理由がわかるので、十分な情報に基づいた選択をすることが出来ます。

17 Reasons for Using a Support Library in Migration Scenarios／マイグレーション時にサポートライブラリを使うべき17の理由

- VB Migration Partner can achieve its high success rate thanks to two factors: (1) its superior code analysis and generation engine, and (2) a powerful support library. A few developers don't realize that all VB6 conversion tools use a support library, even if their vendors prefer not to emphasize this detail. However, VB Migration Partner's library goes far beyond that, and offers many advantages that might not be apparent at first, such as embedded debug and trace features, performance optimizations, interoperability with VB6 apps. This white paper explains these and other advantages and provides a few insights about the future of code migration with VB Migration Partner.

VB Migration Partner の高い変換成功率の、(1)優れたコード分析プログラムと生成エンジン、(2)効果的なサポートライブラリです。ベンダーがそのことを強調したくないとしても、全ての VB6 変換ツールがサポートライブラリを使用していることに気づいている技術者は少数です。しかし、VB Migration Partner のライブラリは大変優れており、使い始めはあまり気づかないかもしれませんが、多くの利点があります。例えば、組込み済みのデバッグ &トレース機能や、パフォーマンス最適化機能、VB6 アプリケーションとの相互運用性などが挙げられます。このホワイトペーパーでは、これら以外の利点についての説明と、VB Migration Partner を使った移行の将来性についての考察を行います。

Reach full functional equivalence with Trace-Match methodology／Trace-Match メソドロジーで機能性の等価を保証する

- The easiest way to ascertain that the original VB6 project and the converted .NET application are equivalent is producing a set of trace files while executing the same sequence of actions on them. The Trace-Match methodology lets you add tracing statements in an easy and error-free manner, and allows you to run a battery of unattended test cases to prove that your migration project is successful.

元々の VB6 のプロジェクトと変換された.NET アプリケーションが同等であることを確認する一番簡単な方法は、同じ組合せの動作をしている間に、トレースファイルのセットを作成することです。Trace-Match メソドロジーを使うと、トレーシングステートメントを、エラーを気にすることなく簡単に追加出来ます。またテストケースを自動的に実行して、移行がうまくいったことを証明することができます。

A smart approach to 3rd-party ActiveX control conversion／サードパーティ製 ActiveX コントロールへの賢明なアプローチ

- Unlike other VB6 conversion tools, VB Migration Partner offers a flexible approach to ActiveX control conversion and migration, based on wrapper classes. This document illustrates the many advantages of wrapper classes compared to less flexible approaches that use code transformation and basic mapping techniques to achieve functional equivalence. It also explains how wrapper classes fit perfectly in large migration projects, decrease overall migration costs, avoid bottlenecks in the migration process, and allow you to split the burden among different developers team, inside or outside your company.

他社の VB6 変換ツールとは異なり、VB Migration Partner によって、ActiveX コント

ロールを移行するための柔軟性の高いアプローチが、ラッパークラスに基づいて実現します。この文書では、弊社のラッパークラスの多くの利点について解説し、機能を等しくする為のコードの置き換えや基本的なマッピングテクニックを使う柔軟性の低いアプローチと比較してみます。又同時に、大規模な移行対象プロジェクトにおいてラッパークラスがどのように、完璧に適合しているか、全体のコスト削減に貢献しているか、移行工程のボトルネックを回避しているか、異なる技術者チームの間で、社内と社外で負荷を分散させるかについて説明します。

Support library and code maintainability／サポートライブラリとコードの保守性

- In addition to providing 100% functional equivalence with the original VB6 code, VB Migration Partner's extensive support library is the key for generating code that is readable and easily maintainable. This whitepaper busts a few false myths about support libraries by comparing the concise source code that VB Migration Partner generates with the code produced by other tools that don't rely on an equally extensive library.

元々の VB6 コードと全く同等の機能を提供するだけでなく、VB Migration Partner の包括的なサポートライブラリは、読み易く保守しやすいコードを作成する鍵となります。VB Migration Partner が生成する正確なコードと、それほど包括的ではないライブラリしか持たない他社のツールによって作成されたコードを比較することにより、このホワイトペーパーは、サポートライブラリに関するいくつかの通説を打破します。

Estimate migration costs for VB6 conversion software／VB6 移行ソフトの費用見積もり

- All real-world migrations show that the largest cost is the manpower needed to prepare the VB6 code for the migration and to modify the resulting VB.NET code to ensure full functional equivalence with the original application. For this reason it is essential that the conversion software you select simplifies the job of your developers and reduces the need for these manual adjustments. This whitepaper analyzes the factors that can affect the duration and cost of a complex migration project.

実際の移行では全ての場合、移行用の VB6 コードを準備して、元々のアプリケーションと同等機能を持つように VB.NET コードの修正を行う為の人件費が、最大のコストです。この為、選択した移行ソフトを使って、技術者の作業を簡素化し手作業での調整を少なくすることが最も重要な事です。このホワイトペーパーでは、複

雑な移行プロジェクトの期間と費用に影響する要因が分析されています。

Tips for smooth migration of calls to Windows API methods／Windows API メソッドのコールを円滑に移行する為のヒント

- VB Migration Partner offers tons of improvements over Upgrade Wizard when migrating VB6 code that heavily on Windows API calls. You can take a few additional steps, however, that can make the migration even easier. Learn how by reading this article.

VB Migration Partner は、Windows API コールに大きく依存する VB6 コードを移行する時、Upgrade Wizard を遥かに超える数の改善策を提供します。更に 2, 3 の段階を踏むことによって、移行がより容易になります。詳しくはこちらの記事をご参照下さい。

ActiveX controls and wrapper classes／ActiveX コントロールとラッパークラス

- VB Migration Partner supports the migration of 3rd-party ActiveX controls by means of wrapper classes. This whitepaper describes all the steps that are necessary to create this wrapper class using the AxWrapperGen utility, how to modify the generated source code to account for special cases, and how to eventually edit the wrapper class so that it wraps a .NET control instead of the original ActiveX control. A detailed troubleshooting section will help you solve all the common problems you can meet during the process.

VB Migration Partner は、ラッパークラスによってサードパー第三者の ActiveX コントロールの移行をサポートします。このホワイトペーパーには、AxWrapperGen ユーティリティーを使用して、このラッパークラスを作成する為の必要事項が全て示されています。具体的には、ソースコードを特殊な事例に合わせて変更する仕方や、ラッパークラスを最終的に編集することによって、元々の ActiveX コントロールの代わりに.NET コントロールをラップする方法などが掲載されています。トラブルシューティングの項目の詳細な説明は、変換中に起こる共通の問題全般の解決糸口となります。

WHITE PAPERS／ホワイトペーパー

Migrating a VB6 application in 10 easy steps／10 ステップで簡単に行う VB6 アプリケーション移行

VB Migration Partner is a revolutionary software that has changed the way VB6 applications can be ported to VB.NET. Much of the manual labor that is necessary with other migration tools can be avoided (or significantly reduced) if you adopt VB Migration Partner and take advantage of its pragmas and its innovative convert-test-fix methodology.

VB Migration Partner は VB6 アプリケーションを VB.NET アプリケーションに移行する方法を変えた革命的なソフトウェアです。VB Migration Partner とプラグマおよび convert-test-fix メソドロジーを採用すれば、他のマイグレーションツールで必要な多くの手作業を回避（または著しく減少）することが可能です。

This document outlines all the steps you take when migrating a VB6 application using our tool. In spite of its highly optimistic title, not all these steps are equally easy. On the other hand, not all of them are necessary in every project.

このドキュメントでは弊社のツールを使用して VB6 アプリケーションをマイグレーションする際のすべての工程の要点を述べます。非常に楽観的なタイトルにも関わらず、これらのステップすべてが容易であるというわけではありません。一方、すべてのプロジェクトにおいてすべてのステップが必要であるというわけでもありません。

Here's the summary of what you'll read:

1. Run VB6 Bulk Analyzer and get free migration assessment
   VB6 バルクアナライザーを実行して移行アセスメントを無償で入手する
2. Download, install, and register VB Migration Partner
   VB Migration Partner をダウンロードして、インストールして、登録する
3. Prepare the VB6 code for migration
   移行に備えて VB6 コードの下準備を行う
4. Run VB Migration Partner to convert the VB6 application
   VB Migration Partner を実行して VB6 アプリケーションを変換する
5. Generate wrappers for 3-rd party ActiveX controls
   サードパーティ製 ActiveX コントロール用のラッパーを生成する
6. Reach the zero-compilation-errors stage
   コンパイルエラーゼロステージに到達する
7. Reach the zero-runtime-errors stage
   ランタイムエラーゼロステージに到達する
8. Achieve functional equivalence with original VB6 code
   元の VB6 コードに対して機能的等価を達成する
9. Optimize the converted VB.NET code
   変換された VB.NET コードを最適化する
10. Extend the VB.NET application
    VB.NET アプリケーションを機能拡張する

1. Run VB6 Bulk Analyzer and get free migration assessment／VB6 バルクアナライザーを実行して移行アセスメントを無償で入手する

The first step in any migration process consists in running our VB6 Bulk Analyzer tool on the VB6 project(s) you want to migrate. (You can download this tool from this page.)
どのような移行工程であっても最初のステップは弊社の VB6 Bulk Analyzer を移行したい VB6 プロジェクトに対して実行することです。(このツールはこちらからダウンロードできます。)

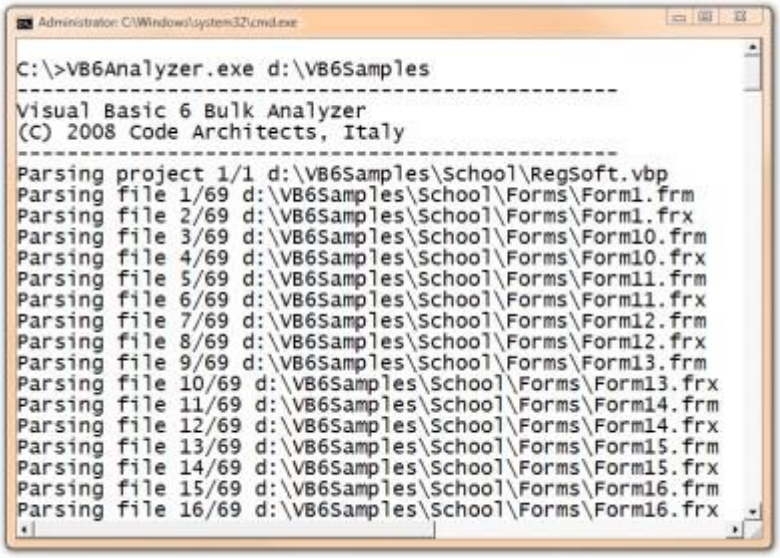
注記:
上記のリンクはベンダーの英語のウェブサイトにリンクしています。英語ウェブサイトからでも日本語版の VB6 Bulk Analyzer はダウンロードできますが、日本語による説明もお読みになりたい場合は弊社(インフォーテック)の無料診断サイトまでお越しください。

VB6 Bulk Analyzer can quickly analyze one or more directory trees, therefore it is quite capable to handle all the projects that are part of your application. It generates a simple textual report, that contains the most important information about your projects, including code metrics, the list of type libraries and ActiveX controls it uses, and – above all – the VB6 features that might require some extra efforts during the migration.

VB6 Bulk Analyzer は一つ以上のディレクトリツリーを高速で解析できるため、アプリーケーションを構成する全プロジェクトを取り扱うことが可能です。このツールはシンプルテキスト形式のレポートを生成します。レポートには、コード量、使用されているタイプライブラリと ActiveX のリスト、とりわけ、移行の際に別途工数が必要となる VB6 の特徴を含んだプロジェクトの最も重要な情報が含まれています。

VB6 Bulk Analyzer is similar to Microsoft VB6 Upgrade Assessment Tool, except it is tailored for migrations based on Code Architects VB Migration Partner rather than the less powerful Upgrade Wizard that comes with Visual Studio.

VB6 Bulk Analyzer は Visual Studio に付属している Upgrade Wizard よりパワフルな Code Architects 社の VB Migration Partner を使用する移行のために作成されていることを除けば、Microsoft VB6 Upgrade Assessment Tool に似ています。



VB6 Bulk Analyzer is a command-line tool that must be run from a prompt window. You need to move to the root folder of the directory tree that contains your source code, type **VB6Analyzer**, and press the Enter key. The tool supports a few additional options, but you rarely need them. (You can list these option by adding the **/help** option on the command line.)

VB6 Bulk Analyzer はコマンドプロンプトで実行するコマンドラインツールです。使用するには、ソースコードを含むディレクトリツリーのルートディレクトリに移動して、**VB6Analyzer** と入力して Enter キーを押す必要があります。このツールはいくつかのオプションをサポートしていますが、ほとんど必要ないでしょう。(コマンドラインで**/help** オプションを付けて実行すれば、オプションのリストを表示させることができます。)

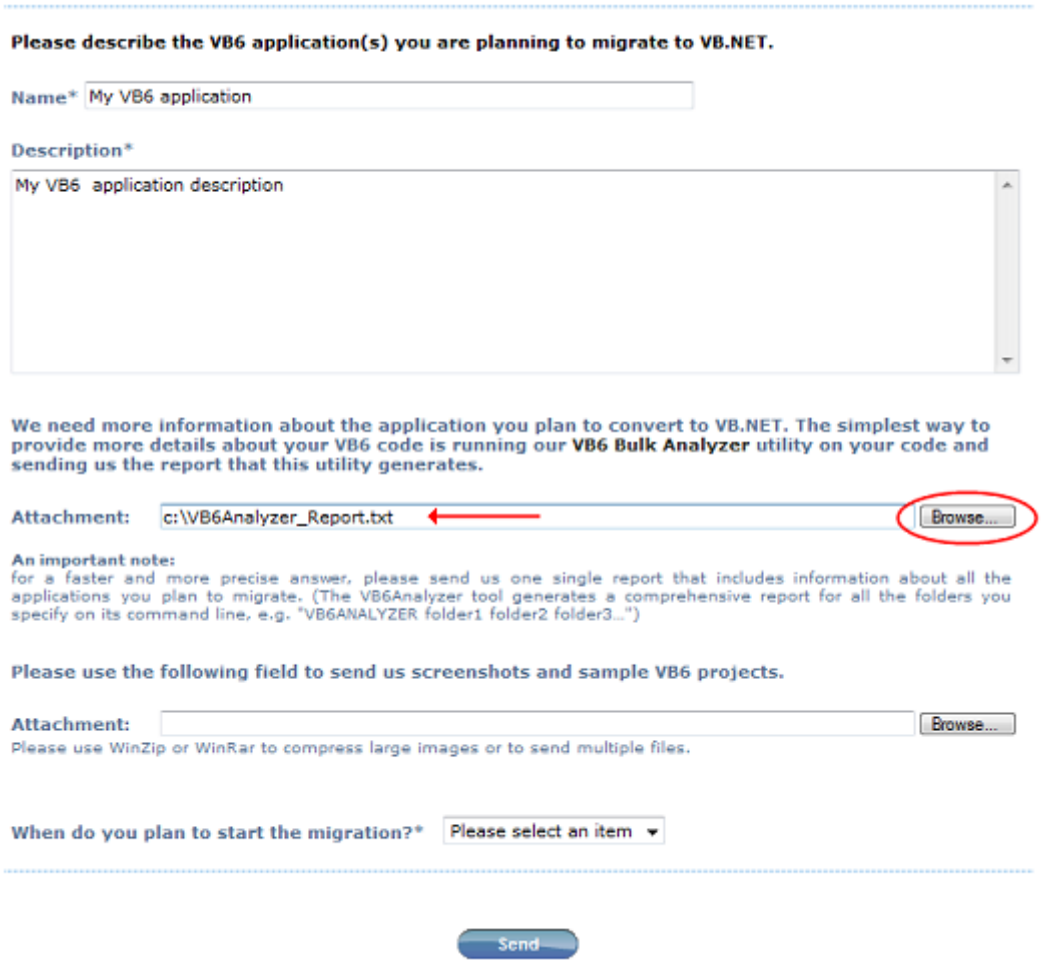
It is essential that you specify all source code folders on the command line, to create a consolidated report. The report is named VBAnalyzer_Report.txt and is created in the current directory.

整理されたレポートを作成するためには、コマンドラインですべてのソースコードのディレクトリを指定することは必須となります。このレポートは VBAnalyzer_Report.txt という名前でカレントディレクトリに作成されます。

The contents of the VBAnalyzer_Report.txt should be enough self-explanatory, but you don't need to interpret it yourself. Instead, go to our Contact Us page, fill a few fields with your name and email, select **"Question (technical)"** from the Reason combobox, and use the special Attachment field to upload the report file.

VBAnalyzer_Report.txt の内容は十分説明不要と言えるでしょうし、翻訳する必要もありません。むしろ、弊社のお問い合わせページに来て、いくつかの項目に名前やメールアドレスを記入し、理由に**「(技術的な)質問」**を選択し、レポートファイルをアップロード用の Attachment 欄を記入してください。

注記:
上記の説明はコードアーキテクツ社に問合せる場合のものです。日本国内においては弊社(インフォーテック)までお問い合わせください。お問い合わせ方法については弊社(インフォーテック)の無料診断サイトをご覧ください。



We typically analyze the report as soon as we receive it, of course if our office is open when you send it. (Please account for time zone difference… we live at G.M.T. +1.) We typically reply in a few hours, even though complex reports might take longer.

レポートをお送りいただいた時間が弊社の営業時間内であれば、レポートを受け取り次第、形式的な分析を行います。(時差を考慮願います。弊社時間は GTM+1 です。)複雑なレポートにより時間がかかるとしても、弊社(インフォーテック)は通常数時間で回答を送付いたします。

注記:
イタリアとの時差(8 時間)を考慮しても翌営業日には回答いたします。

In return for your report you will receive a very detailed textual assessment of your VB6 application, with tips on how to leverage VB Migration Partner for a smooth migration experience. If a given issue has multiple solutions, the assessment document explains the pros and cons of each option. For example, the assessment often includes recommendations on the best way to migrate your ActiveX controls and your database-intensive code, among the other things.

ユーザーのレポートの回答として、ユーザーは VB Migration Partner を使用して順調なマイグレーションを行うためのヒントとともに VB6 アプリケーションの詳細なアセスメントを受け取ります。与えられた課題が複数の解答を持つ場合、このアセスメントは各解答の賛否について説明します。例えば、アセスメントはとりわけ ActiveX コントロールとデータベースに依存するコードを移行する最良の方法を推奨することが多くあります。

The mail from Code Architects usually contains also information about the VB Migration Partner license that is most appropriate for your needs. For more information about our license terms, read the FAQ section and download the End User License Agreement (EULA).

弊社からのメールはたいていユーザーの必要に対してもっともふさわしい VB Migration Partner のライセンスについての情報も含みます。弊社のライセンス期間についてのより多くの情報については、FAQ をお読みいただくか、End User License Agreement (EULA)をダウンロードしてください。

You can purchase the product or just apply for a temporary Trial License. VB Migration Partner Trial Edition is identical to the Full edition except for four details:

ユーザーは製品をお買い上げいただくか、期間限定のトライアルライセンスをご利用いただけます。VB Migration Partner のトライアル版は 4 つの点を除いて正規版と同等の機能を持ちます。

- it expires in 7 days
  試用期間は 7 日間です。
- migrations require an active Internet connection
  常時インターネット接続が必要です。
- a few pragmas have been disabled
  いくつかのプラグマは使用できません。
- the VB.NET code that it generates can't be expanded with new forms and classes
  生成された VB.NET ソースコードに新しいフォームやクラスを追加することはできません。

Notice that users of the Trial Edition enjoy the same degree of support that regular users do. Providing tech support is an expensive activity and this explains why the Trial Edition expires in a week and its validity can't be extended.

注意していただきたい点は、トライアル版のユーザーは正規版ユーザーと同等のサポートを受けることができますが、1 週間でトライアル版が使用できなくなります。延長ができない理由は技術サポートが高価であるためです。

注記:
現在、弊社(インフォーテック)では試用期間を 30 日まで延長しております。また、お客様のご要望に合わせてさらに試用期間を延長することも可能です。さらに、作業環境からインターネットへの常時アクセスができないお客様にはオフライン版もご提供いたしております。

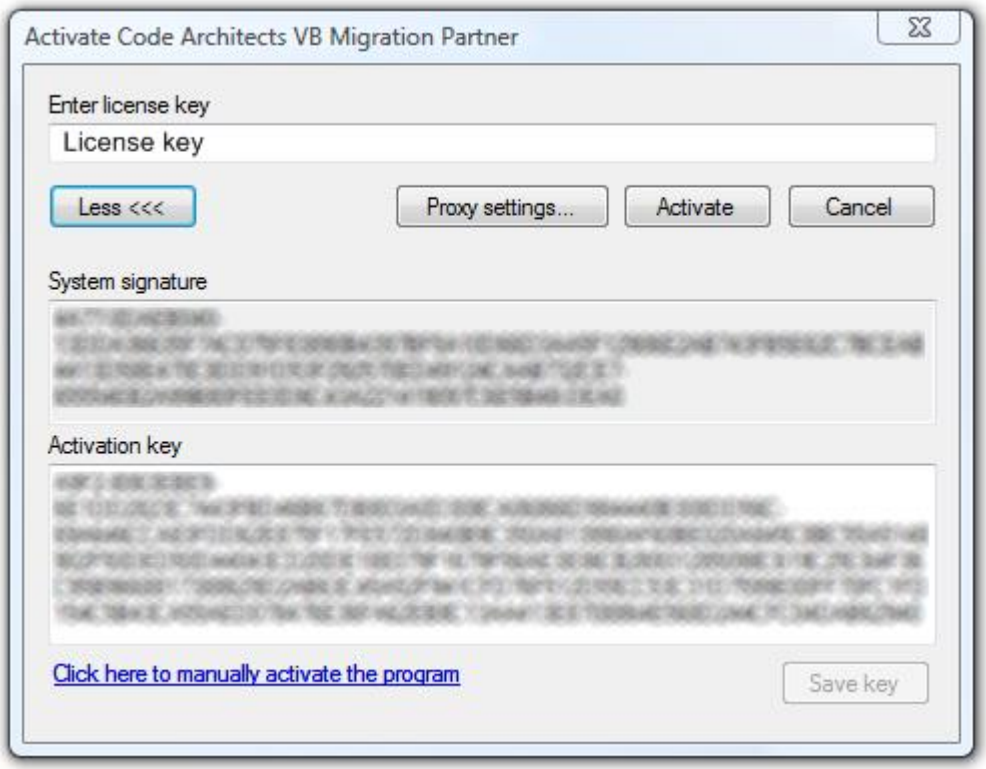
2. Download, install, and register VB Migration Partner／VB Migration Partner をダウンロードして、インストールして、登録する

This is surely the simplest step in the entire process.

このステップはすべての工程で最も単純です。

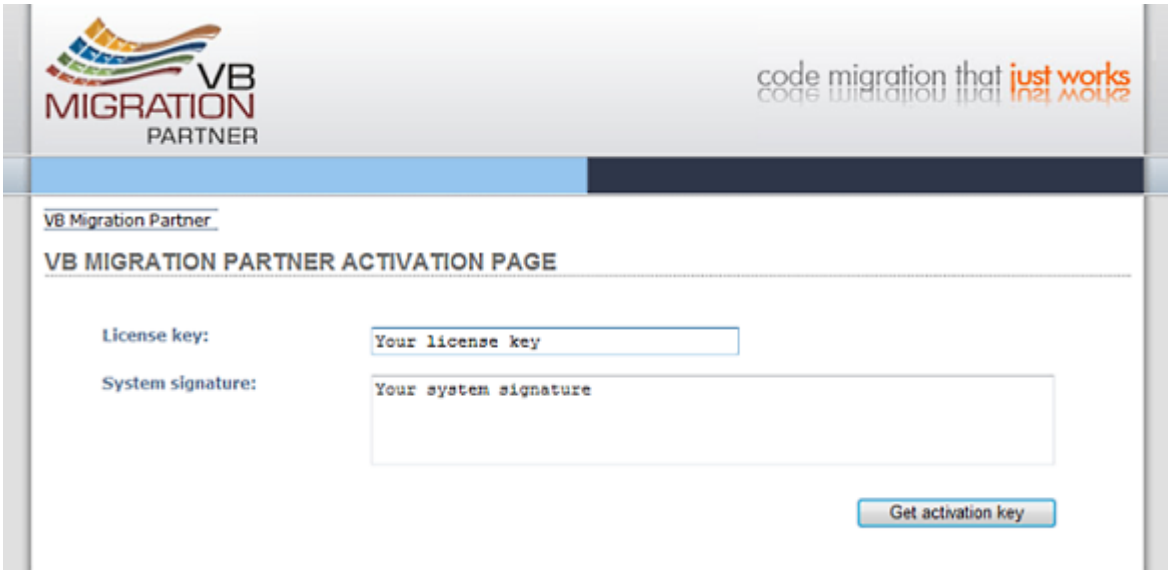
Regardless of whether you purchased the commercial edition of VB Migration Partner or are just testing the Trial Edition, you will receive a download URL. Follow the link, download the file, unzip it, and run the setup procedure.

ユーザーが VB Migration Partner の正規版を購入するかトライアル版を試用するかに関わらず、ユーザーはダウンロード用の URL を受け取ります。そのリンクにしたがって、ファイルをダウンロードし、圧縮ファイルを解凍して、セットアッププログラムを実行してください。



The first time you run VB Migration Partner you'll have to register your copy. This procedure is fully automated and runs as soon as you attempt your first migration. In some rare cases – for example, if your firewall is very picky about which packets can run over the network – the automated procedure fails and you'll have to perform a manual registration.

最初に VB Migration Partner を実行したら、ユーザーは登録を行う必要があります。この処理は完全に自動化されており、ユーザーが最初のマイグレーションを実行しようとすると実行されるようになっています。例えば、ユーザーのファイアーウォールがネットワーク上を移動するパケットに対する監視が厳しいとか、いくつかのレアケースにおいて、自動登録処理は失敗します。そのときは、手動で登録を行う必要があります。



Each time you launch it, VB Migration Partner checks whether a new version is available and asks you whether you want to download it. We recommend that you always run the most recent build that is available on our servers.

VB Migration Partner は実行する度に、最新版がダウンロード可能でないかをチェックし、ダウンロードするかどうかをユーザーに質問します。弊社はユーザーが常に弊社サーバーの最新版を実行されることを推奨しています。

Before using VB Migration Partner you should have a quick read at its manual, with a keen eye on the section related to migration pragmas. If you master pragmas you can make VB Migration Partner do whatever you want, therefore the time you devote to studying them won't be wasted.

VB Migration Partner をご使用いただく前にユーザーはマニュアルを一読すべきです。特にプラグマに関する章を熱心に読むべきです。もし、ユーザーがプラグマを習得すれば、やりたいことは何でも VB Migration Partner にさせることができます。だから、そのための学習時間は無駄にはなりません。

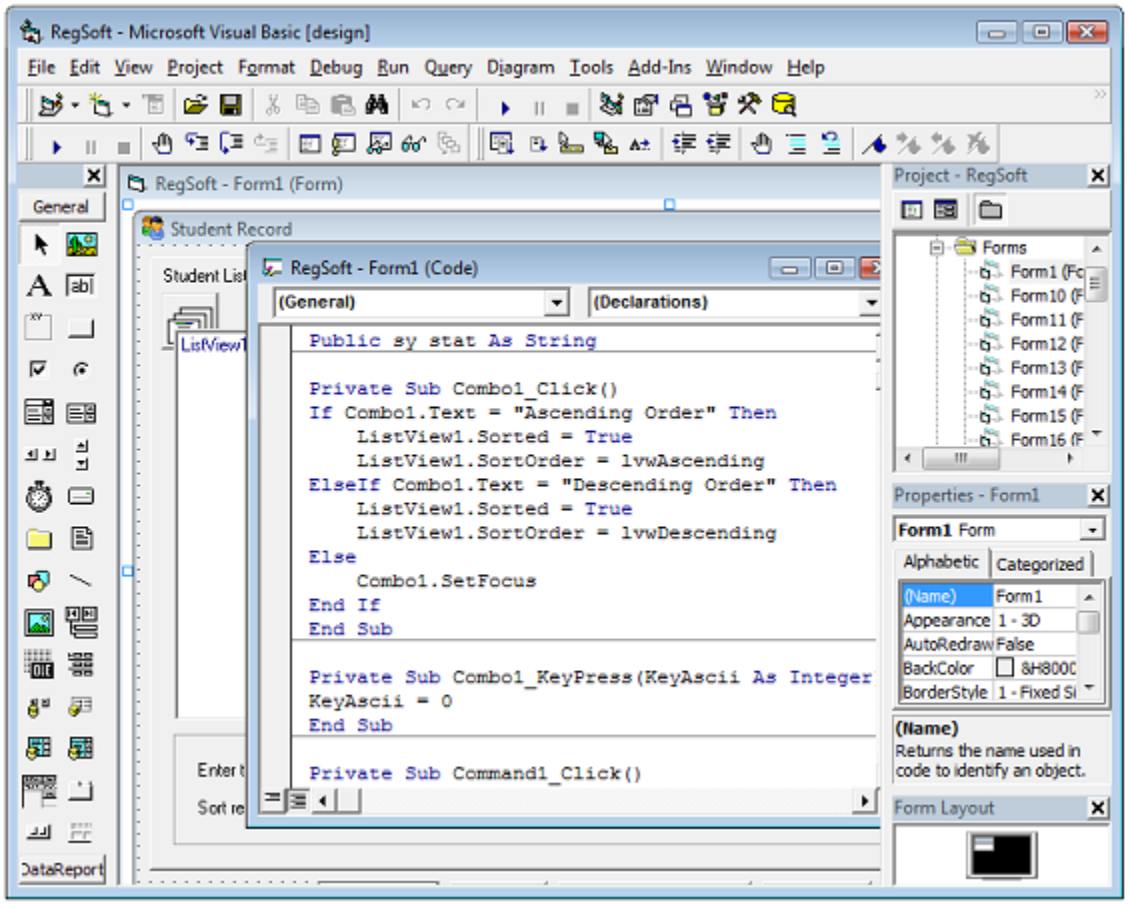
Also, we recommend that you give at least a quick look at our knowledge base. Don't read each and every article, just a quick read of their titles should be enough to ring a bell in your brain if and when you bump into a migration issue that we know how to solve.

また、弊社ではユーザーが少なくともサポート技術情報を一読することを推奨しています。すべての記事を読まなくても、記事のタイトルを素早く読むだけで、弊社が知りうる解決方法の記事を発見することができるでしょう。

3. Prepare the VB6 code for migration／移行に備えて VB6 コードの下準備をする

VB Migration Partner is capable to deal with virtually the entire set of documented VB6 features, therefore this step is seldom required, and you can often migrate your VB6 code without having to prepare it in any way. (In this respect, VB Migration Partner is ***much*** more powerful than any other VB conversion tool on the market.)

VB Migration Partner は実質的には VB6 がサポートするすべての特徴を扱うことができます。そのため、このステップはほとんど必要ありません。ユーザーはたいてい VB6 ソースコードを下準備なしでマイグレーションすることができます。(この点について、VB Migration Partner は市場に出回っている他の VB コンバージョンツールと比べて***非常に***パワフルであると言えます。)



The reasons why you might need to edit the VB6 code before attempting the migration are pointed out in the assessment document you receive from the VB Migration Partner team. Two of the most common VB6 features that require some care in this stage are outlined below.

ユーザーがマイグレーションを実行する前に VB6 ソースコードを編集する必要があるかもしれない理由は、ユーザーが VB Migration Partner team から受け取ったアセスメントに記述されています。いくらかの作業を必要とするもっとも一般的なふたつの VB6 の機能を以下に示します。

Windows API wrapping／Windows API ラッピング

VB Migration Partner supports all VB6 features related to Declare statements – for example, As Any and callback parameters are virtually always translated correctly.

VB Migration Partner は Declare ステートメントに関する VB6 の機能をサポートします。例えば、As Any と callback パラメータは実際、常に正しく変換されます。

However, some calls to Windows API or other external DLLs might use the undocumented VarPtr, StrPtr, and ObjPtr methods. These VB6 methods aren't supported under VB.NET and there is no automated way to migrate them. The report that VB6 Bulk Analyzer generates highlights whether your application uses these keywords. If it does, you should use the Find command in the VB6 IDE to locate each one of them and see whether you can remove individual occurrences.

しかし、いくつかの Windows API や他の外部 DLL の呼び出しは VarPtr や StrPtr や ObjPtr といったサポートされていないメソッドを使用しているかもしれません。これらの VB6 のメソッドは VB.NET ではサポートされません。また、それらを自動的に変換

する方法もありません。VB6 Bulk Analyzer が生成するレポートはユーザーのアプリケーションがこれらのメソッドを使用しているかどうかを強調しています。もし使用されていれば、ユーザーはそれを取り除くことができるかどうか知るために VB6 の IDE で検索コマンドを使って個々に探し出さなければなりません。

In many cases you can prepare your Declare-intensive VB6 code for migration by wrapping all calls inside methods. For a detailed discussion of this technique, please read the whitepaper "Tips for smooth migration of Windows API calls".

多くのケースでは、ユーザーは移行に備えて、メソッドに対応する呼び出しをラッピングすることによって Declare に対応した VB6 コードを準備することができます。この技法についての詳細な記述は「Tips for smooth migration of Windows API calls」という whitepaper をお読みください。

ActiveX EXE projects／ActiveX プロジェクト

The .NET Framework doesn't support anything that can be considered as equivalent to ActiveX EXEs. VB Migration Partner allows you to choose whether the project should be converted into a standard EXE project or a .NET DLL, therefore you must decide what is most appropriate in each specific case.

. NET Framework は ActiveX EXE と同じであると見なされるものはどれもサポートしません。VB Migration Partner は標準的な EXE のプロジェクトに変換するか、. NET DLL に変換するかをユーザーに選択させることができます。それによって、ユーザーはそれぞれのケースの特性に応じてもっとも適切なものを選択することができます。

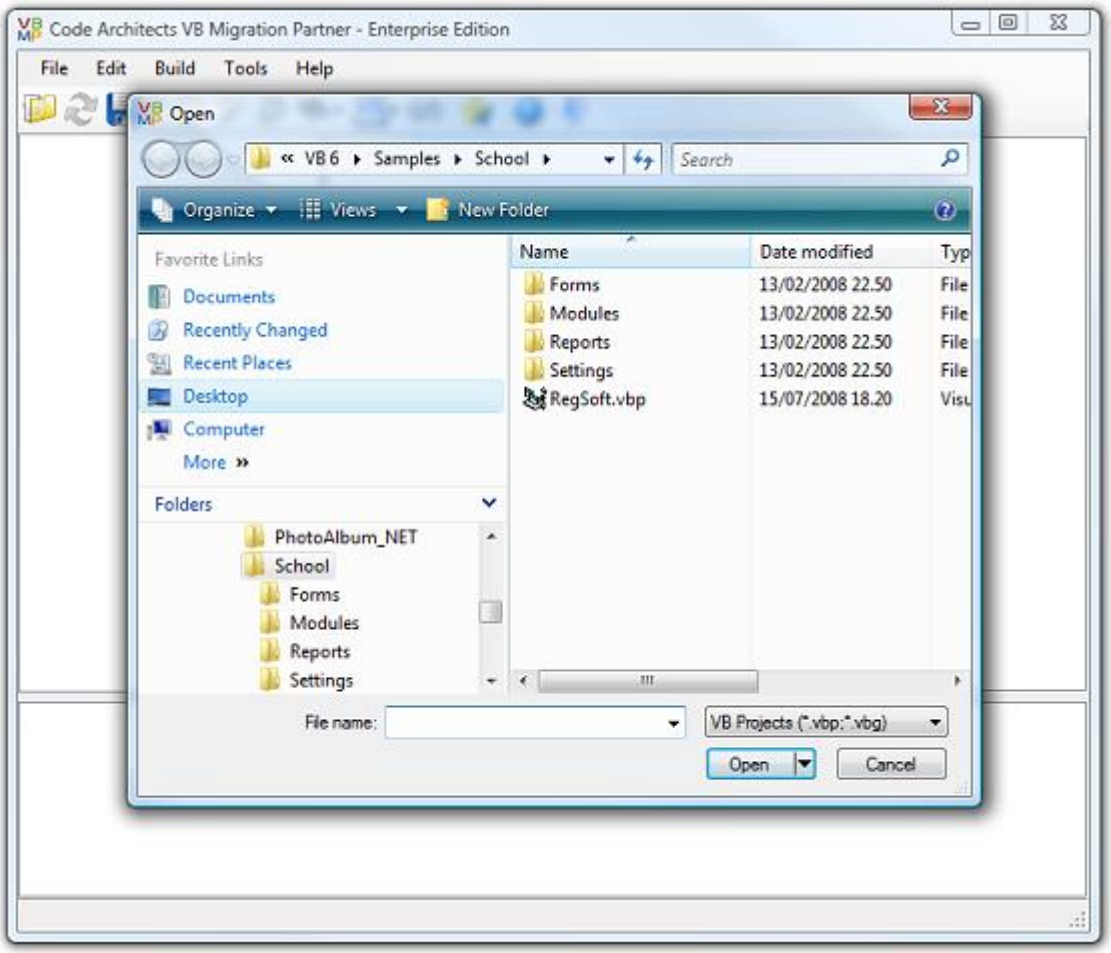
There can be several reasons why a VB6 project had to be compiled as an ActiveX EXE. One of the most common was that you needed a stand-alone executable that could also expose objects to the outside, for example an order processing application that exposes an object model for other apps (or plug-ins) to access and manipulate the data using a simplified and consistent interface. In such cases, you can solve elegantly your migration problems by splitting the ActiveX EXE project in two parts: an ActiveX DLL project that exposes the object model and a standard EXE project that shows the user interface and references the DLL.

VB6 プロジェクトを ActiveX EXE としてコンパイルしておいたのにはいくつかの理由が考えられます。もっとも一般的な理由はユーザーがオブジェクトを外部に公開することもできるスタンドアロンアプリケーションを必要としていたことが考えられます。例えば、単純で（技術的に）安定したインターフェースを使ってデータにアクセスしたり、データを操作したりする外部アプリケーション（またはプラグイン）にオブジェクトを公開する注文処理アプリケーションです。そのようなケースでは、ユーザーは ActiveX EXE を二つに分割することでマイグレーションの課題を見事に解決することができます。オブジェクトモデルを公開する ActiveX DLL プロジェクトとユーザーインターフェースを公開して、DLL を参照する標準 EXE です。

4. Run VB Migration Partner to convert the VB6 application／VB Migration Partner を実行して VB6 アプリケーションを変換する

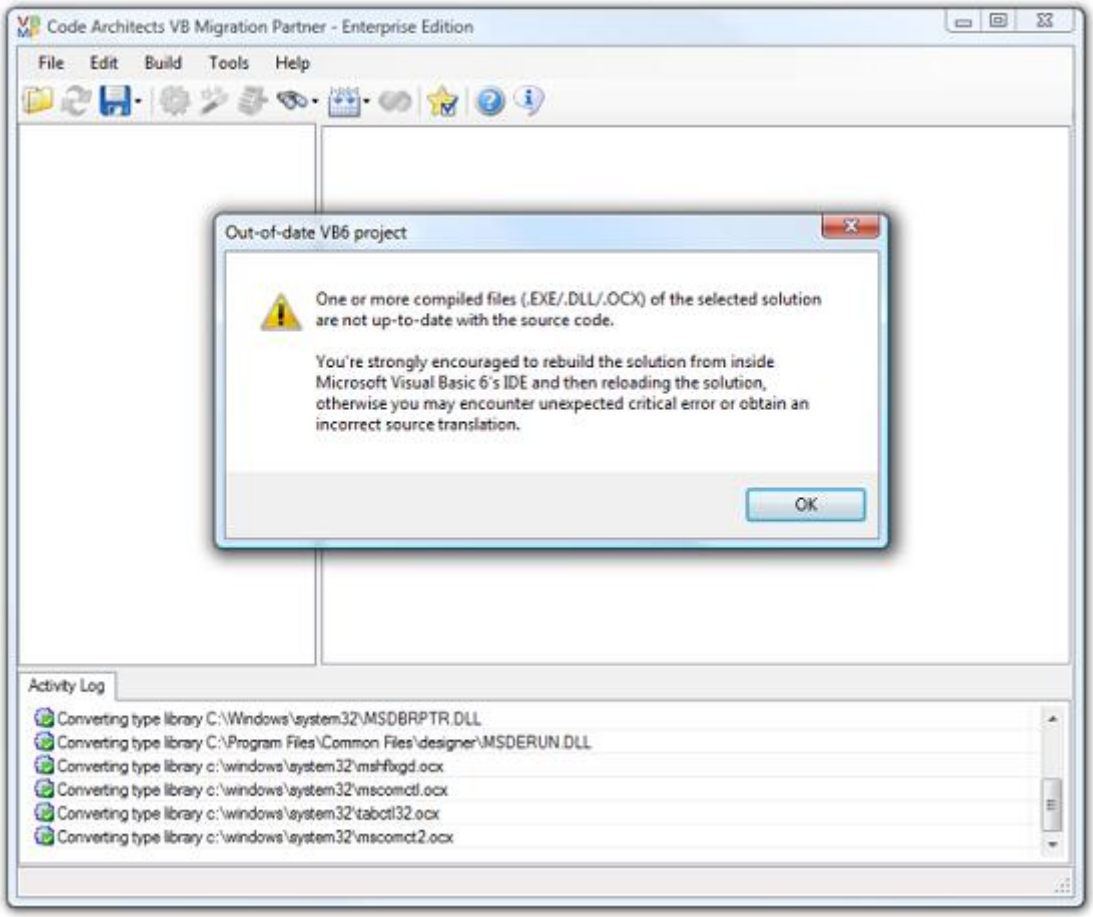
It's now time for the first run through VB Migration Partner. If you have a project group (a .vbg file) that gathers all the VB6 projects of your app, load it in VB Migration Partner. If you have separate .vbp projects, consider whether you should create a .vbg file for the only purpose of migrating them in one single operation. In general, migrating multiple projects in one step delivers better results, because VB Migration Partner can generate better code for inter-project calls.

さあ、いよいよ VB Migration Partner を実行するときです。ユーザーがアプリケーションのプロジェクトすべてを集めたプロジェクトグループ(vbg ファイル)をお持ちなら、それを VB Migration Partner に読み込んでください。もし、分割された vbp ファイルをお持ちなら、一回の操作でそれらをマイグレーションするための vbg ファイルを作成すべきかどうかを検討してください。一般的には複数のプロジェクトを 1 回でマイグレーションしたほうが良い結果が得られます。なぜなら、VB Migration Partner はプロジェクト内部の呼び出しに対して、より良いコードを生成できるからです。



When loading the project, VB Migration Partner checks that the EXE or DLL executable file is found and its creation date is earlier than all the source files in the project. If this isn't the case, the following message appears when loading the project:

プロジェクトを読み込むとき、VB Migration Partner は、EXE や DLL といった実行可能ファイルが見つかること、その作成日付がプロジェクトのすべてのソースファイルより新しいことを確認します。もし、そうでなければ、プロジェクトを読み込んだときに次のメッセージが表示されます。



VB Migration performs this test because it guarantees that the VB6 source code is syntactically correct and has compiled without errors. If you are sure that all post-compilation changes are OK and haven't introduced any syntax error, just press OK and continue. If you don't want to be bothered by this message in the future, you can disable the test by selecting the "Omit warning if VB6 executable is missing or outdated" option in the General tab of the Tools-Options dialog box.

VB Migration はがこの確認を実行する理由は、VB6 ソースコードが全体的に正しく、エラーなくコンパイルされていることを保障するためです。ユーザーがすべてのコンパイル後の変更に問題がなく、文法エラーが報告されていないことを確認できるのなら、OK を押して処理を続行してください。もし、今後このメッセージを表示させたくないのであれば、Tools メニューの Options ダイアログの General タブの「Omit warning if VB6 executable is missing or outdated」(実行可能な VB6 モジュールがないか日付が異なっている場合は警告を無視する)オプションを選択して確認処理を無効にすることができます。

Once the project is successfully loaded, you can convert it to VB.NET by either selecting the Built-Convert to VB.NET command, or clicking on the magic wand icon, or typing the F5 shortcut key. At the end of the conversion process, VB Migration Partner generates as many as six different kind of reports. Each of these reports can be used for a different purpose:

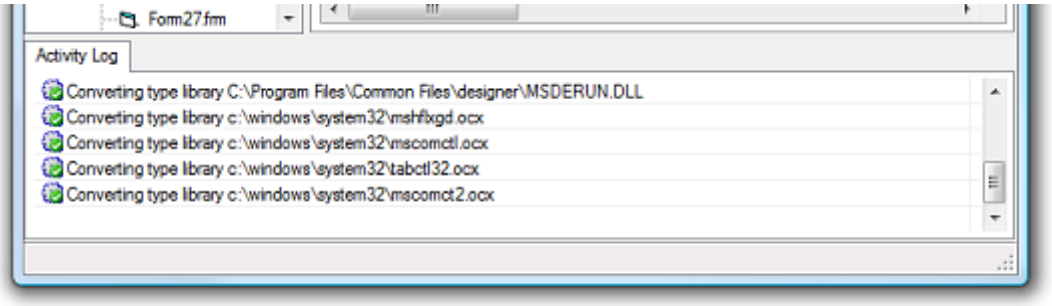
一旦、プロジェクトが正常に読み込まれたら、ユーザーは Built メニューの Convert to VB.NET コマンドを選択するか、魔法の杖のアイコンをクリックする、F5 ショートカットキーのいずれかによって VB.NET にコンバートすることができます。変換過程の最後に VB Migration Partner は 6 種類ものレポートを生成します。各レポートは異なる目的のために使用することができます。

Activity Log／実行ログ

- VB Migration Partner emits several messages in this window during the migration process. You should check these messages to ensure that all went well, that all type libraries and ActiveX controls were found, that no unrecognized syntax forms where

found, etc. If an error message is found in this window, odds are that you tried to convert a VB6 project that can't compile, you didn't installed all COM type libraries correctly, or you run VB Migration Partner on a computer different from the computer where you compiled the VB6 project.

—VB Migration Partner はマイグレーションの過程でこのウインドウにいくつかのメッセージを表示します。ユーザーはすべてがうまく行っていること、すべてのライブラリと ActiveX コントロールが見つかったこと、特定できない文型が見つかっていないことなどを確認するためにこれらのメッセージを確認すべきです。もし、エラーメッセージがこのウインドウで発見されたら、コンパイルできない VB6 プロジェクトを変換しようとしたか、すべての COM タイプライブラリを正しくインストールしなかったか、VB6 プロジェクトをコンパイルした PC とは異なる PC で VB Migration Partner を実行した可能性があります。



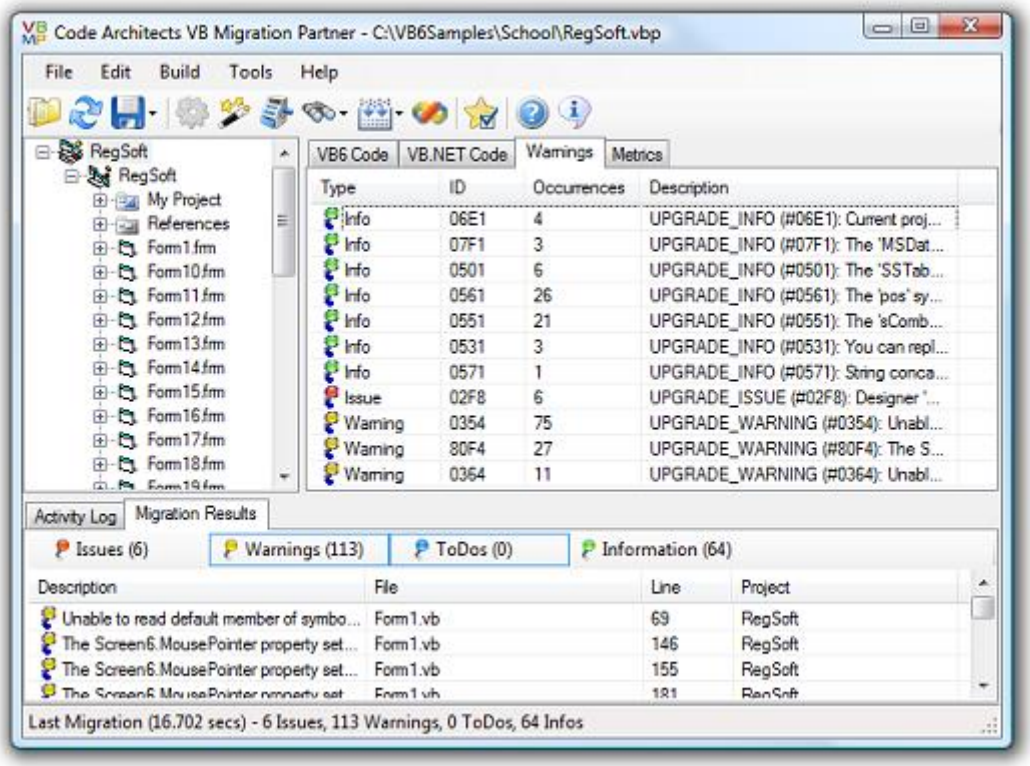
Migration Results／変換結果

– This window displays all migration warnings and errors. The presence of a warning or error doesn't necessarily indicate that the conversion has failed; it only means that you have to pay close attention to the specified portions of the VB.NET and ensure that they converted correctly.

—このウインドウはすべてのマイグレーションの警告とエラーが表示されます。警告やエラーが存在することは必ずしも変換が失敗したことを意味しません。それは単にユーザーが VB.NET の特定の箇所に注意しなければならいこと、またそれらが正しく変換されることを確認しなければならないということを意味します。

You can also get a summary of all migration results – with all issues and warnings in a given project or file – grouped by their type – by clicking on the Warnings tab in the right pane.

また、ユーザーは右ペインの Warnings タブをクリックしてマイグレーション結果の概要を見ることもできます。マイグレーション結果はプロジェクトやファイルについてのすべての問題と警告を含んでおり、タイプごとに分類されています。



Compilation Results／コンパイル結果

– You can use the Compile command in the Build menu to have VB Migration Partner run the vbc.exe compiler behind the scenes, gather its output, and display all messages in the bottom pane. If you are lucky and your VB6 code isn't too complex, you'll see just a few errors in this window. (Our statistics show that VB.NET apps generated by VB Migration Partner have one compilation error for every 1,076 executable statements on the average.)

—ユーザーは、VB Migration Partner に vbc.exe コンパイラをバックグラウンド実行させ、出力情報を収集し、下部ペインにすべてのメッセージを表示させるために、Build メニューの Compile コマンドを使うことができます。もし、ユーザーが幸運かつ VB6 ソースコードがあまり複雑でなかったなら、このウインドウにはほんの少しエラーが表示されるだけでしょう。(弊社の統計では VB Migration Partner によって生成された VB.NET アプリケーションでは平均して 1076 の実行可能行に対してコンパイルエラーは 1 件でした。)

If you are *very* lucky, this window will display no error messages and the VB.NET code is ready to compile and run. It doesn't happen often, though, but it happens.

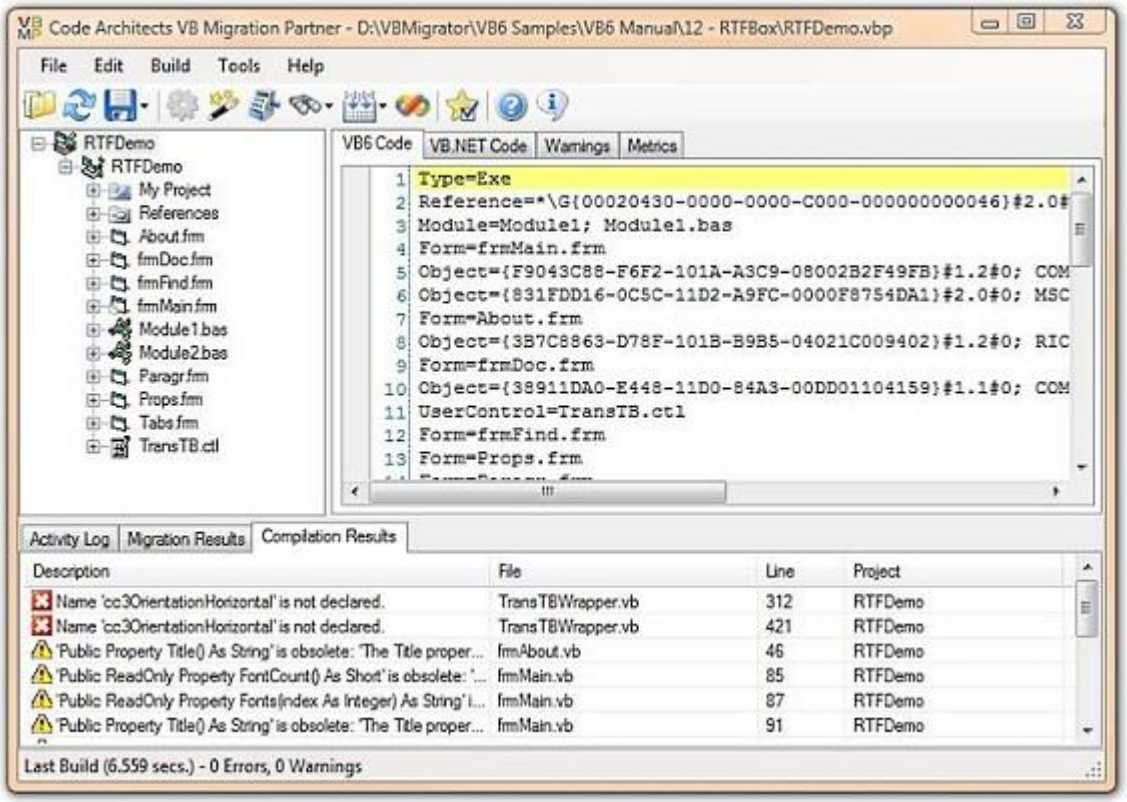
*非常に*幸運であれば、このウインドウには何もメッセージは表示されず、VB.NET ソースコードはコンパイルおよび実行の準備が整っています。よくあることではありませんが、起こりうることです。

If there are many compilation errors, don't feel discouraged. Most of these errors are likely to have the same cause. They will go away after adding a few well-chosen pragmas.

もし、多くのコンパイルエラーがあっても、がっかりしないでください。これのエラーのほとんどはおそらく同じ原因によるものです。よく選んだ2、3のプラグマを使えばそれらはなくなります。

Also, notice that the Visual Basic compiler has a limit of 102 error messages, after which the compilation is aborted, therefore you will never see more than 102 errors. This means that you still see 102 errors even after fixing some of them, because you then see errors that weren't visible before.

もうひとつ、Visual Basic のコンパイラは 102 個までしかエラーメッセージを表示できないという制限があります。そのため、コンパイルが中断された後、ユーザーは 102 個以上のエラーを確認することができません。これはいくつかのエラーを修正した後でもまだ102個あるかもしれないということです。なぜなら、ユーザーは後になって以前は表示されていかったエラーを見つけるからです。



Code Metrics／コードメトリクス

– the Metrics tab in the right pane displays stats about the project or file selected in the tree view on the left. This information can be useful for a number of purposes. For example, you may sort projects and files on their size, so that you can start with the simplest project or you can assign each project to a different people in your team.

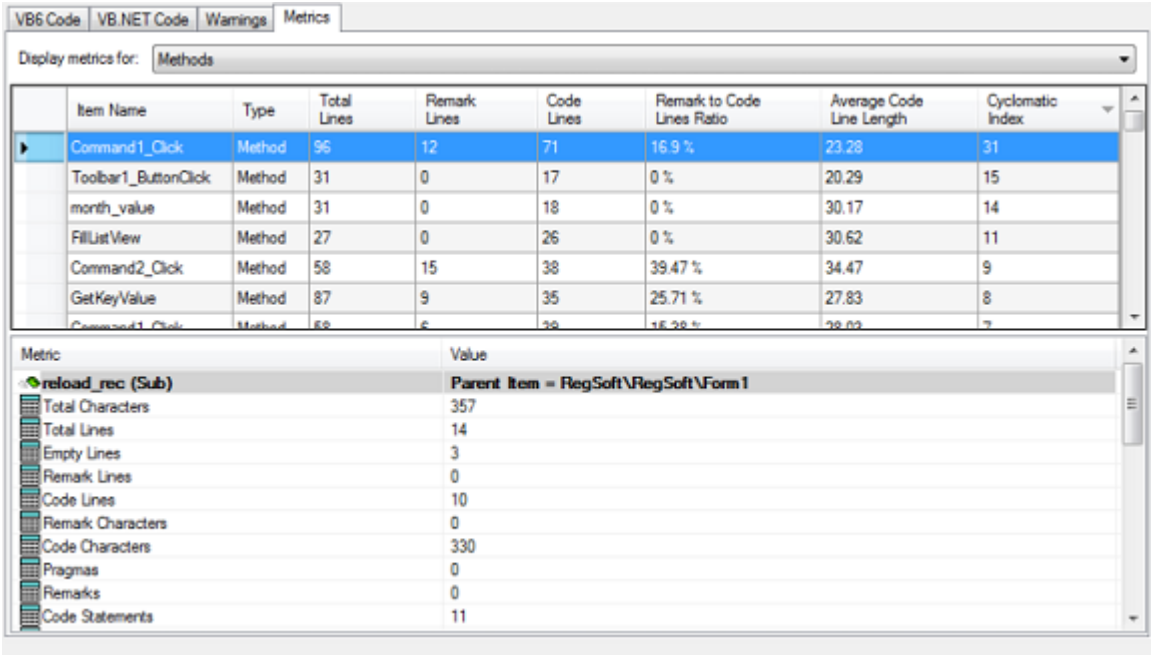
—右ペインの Metrics タブは左側のツリービューから選択されたプロジェクトやファイルの統計を表示します。この情報は多くの目的のために役立ちます。例えば、プロジェクトやファイルをサイズで並び変えれば、ユーザーはもっとも簡単なプロジェクトから開始することができますし、各プロジェクトをチームのことなるメンバーに割り当てることができます。

One of the most important values in the table is the Cyclomatic index, which measures all the possible execution paths inside a method. For example, a method that contains only an If block has a CI equal to 2; if the Else block contains a Select Case block with five Case sections, then the method's CI is equal to 6, and so forth. The CI is important in migration scenarios because it represents the approximate number of tests you should perform to ensure that the VB.NET code behaves exactly like the original VB6 code. By sorting all methods in a file on their CI you can quickly see which methods are more complex and should be give more attention during the test and QA phase.

この表のもっとも重要な値は Cyclomatic index(CI)です。これはメソッド内のすべての可能性のある実行パスを数えたものです。例えば、If ブロックをひとつだけ含んでいるメソッドは 2 という CI を持っています。もし、Else ブロックが 5 つの Case セクションを含む Select Case ブロックをひとつ含んでいれば、そのメソッドの CI は 6 となります。CI はマイグレーションシナリオにおいて重要です。なぜなら、それは VB.NET ソースコードが正確に VB6 ソースコードのように動作することを確認するために行うべきおおよそのテストケースの数を表しているからです。ファイルごとに CI 値ですべてのメソッドを並び替えることでユーザーはどのメソッドがより複雑か、どのメソッドのテストと品質保証(QA)の工程により注意を払うべきかをすべやく知ることができます。

The Metrics tab also show how many “problematic” VB6 items are in a given project or file, including Variant variables, Gosub keywords, On Error statements, auto-instancing (As New) variables, arrays with non-zero lower bounds, and so forth. High values for these counters are indicators that you might need to pay more attention to the corresponding classes and methods.

Metrics タブもまた、プロジェクトやファイルにどの程度問題のある VB6 の機能が含まれているかを表しています。Metrics タブは、バリアント変数、Gosub キーワード、On Error ステートメント、自動インスタンス化（As New)変数、下限値が 0 でない配列などを含んでいます。これらのカウンターが高い値を示している場合、ユーザーはそのクラスやメソッドに注意する必要があるかもしれません。



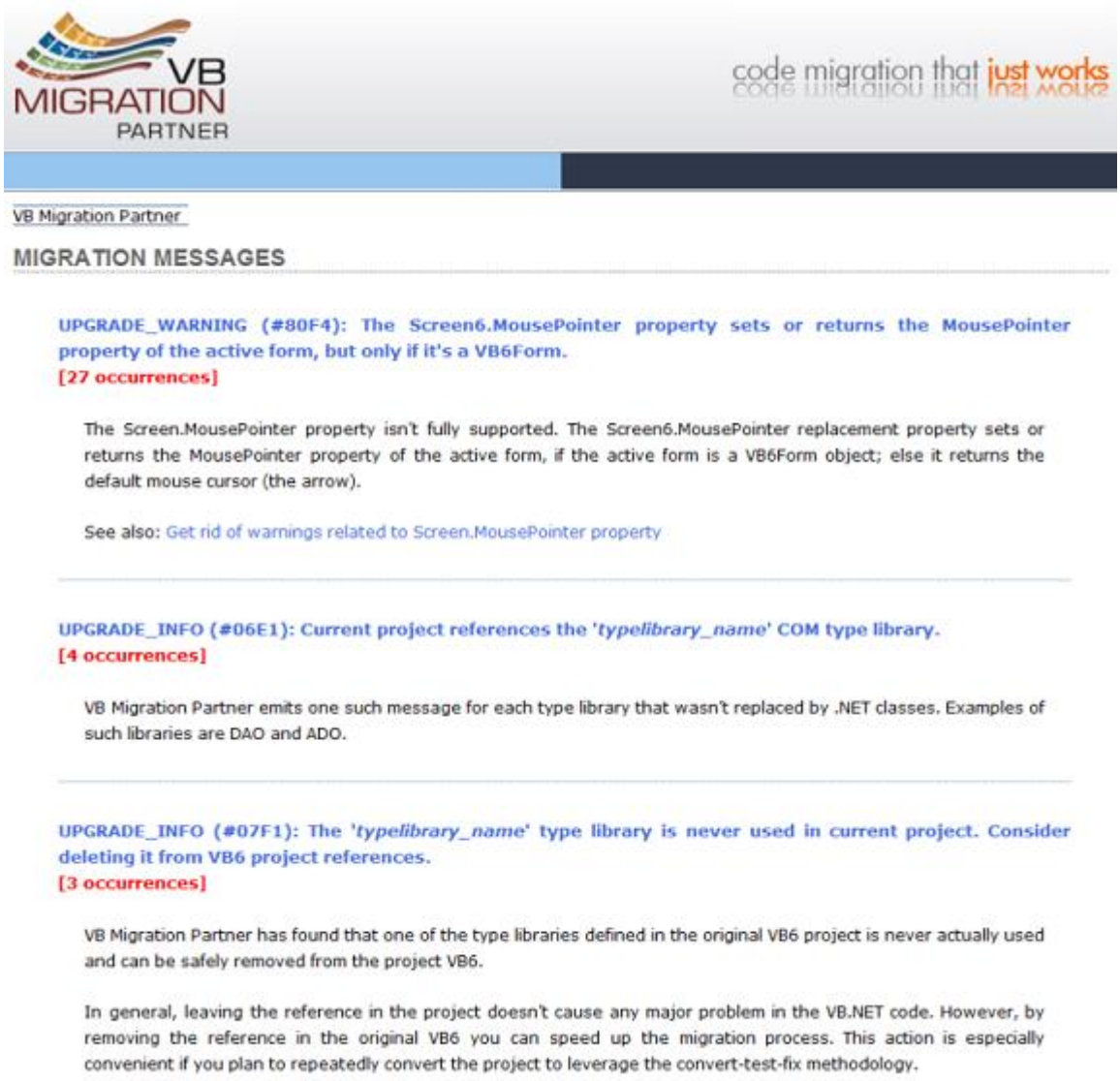
Migration message explanation／変換メッセージの説明

– If you are new to migrations from VB6, odds are that many migration errors and warnings will look a bit obscure. Or maybe you understand what a message means and still have no cue on how to fix it. In such cases, you can use the Explain Errors and Warnings command in the Tools menu to produce a detailed explanation of each migration message and a description of available workarounds.

—もし、VB6 からのマイグレーションが初めてなら、多くのマイグレーションエラーと警告はちょっと分かりにくいでしょう。そうでなければ、メッセージが何を意味するのか理解できるかもしれませんが、修正するための方法についてはまだ何も得られないかもしれません。そのようなケースでは、各マイグレーションメッセージの詳細な説明や有効な解決策の記述を読むために Tools メニューの「Explain Errors and Warnings」（エラーと警告を説明する）コマンドを使用することができます。

Notice that this report is created on Code Architects' Web server, therefore it requires a working Internet connection. Only the ID and the number of occurrences of each warning are sent to our server; no information about your application ever leaves your computer.

このレポートは弊社のウェブサーバーで作成されるので、インターネット接続が必要であることを覚えてください。各警告の ID と発生番号だけが弊社のサーバーに送信されます。ユーザーアプリケーションに関する情報は何も弊社のサーバーには残りません。



Assessment report／アセスメントレポート

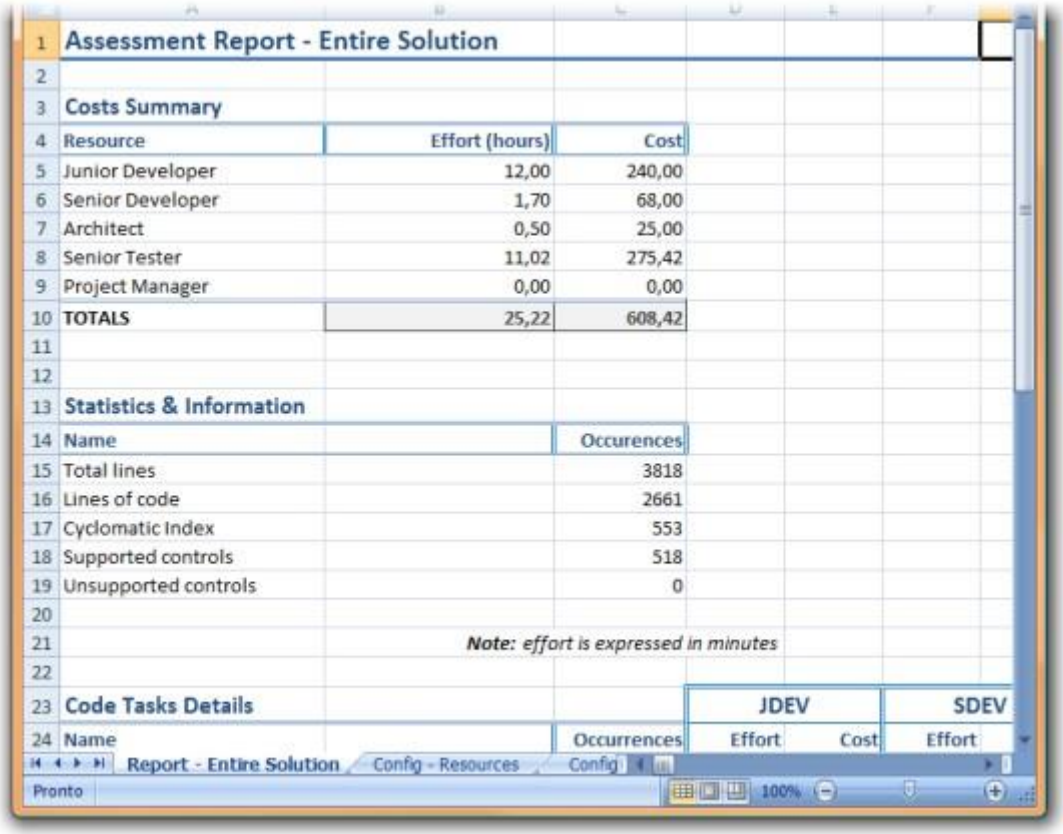
– You can generate a Microsoft Excel worksheet containing a very detailed report of all the migration errors and warnings, together with an estimation of the time required to fix them, get a fully working VB.NET application, and complete the migration process. The report also contains an estimation of costs, which are evaluated by multiplying the time required for the hourly wage of the people in your team. The report accounts for five professional roles: junior and senior developer, test, architect, and product manager.

—ユーザーはすべてのマイグレーションエラーと警告、修正に必要な予想時間、VB.NET アプリケーションが完全に動作するまでの時間、マイグレーション過程が終了するまでの時間を含んだ非常に詳細なレポートを含んだ Microsoft Excel のワークシートを生成することができます。このレポートは費用の見積もりも含んでいます。費用は必要な時間にユーザーのチームのメンバ

一の時間給を掛けることによって見積られます。このレポートは 5 つの担当を考慮しています。初級技術者、上級技術者、検査担当者、設計書、プロダクトマネージャーです。

All values in the spreadsheet can be customized and you can even provide a custom template if you wish, in the Reports tab of the Tools–Options dialog box.

すべてのワークシートの値はカスタマイズできますし、Tools メニューの Options ダイアログの Reports タブでお望みのカスタムテンプレートを使用することもできます。



1	**Assessment Report - Entire Solution**					
2						
3	**Costs Summary**					
4	Resource	Effort (hours)	Cost			
5	Junior Developer	12,00	240,00			
6	Senior Developer	1,70	68,00			
7	Architect	0,50	25,00			
8	Senior Tester	11,02	275,42			
9	Project Manager	0,00	0,00			
10	TOTALS	25,22	608,42			
11						
12						
13	**Statistics & Information**					
14	Name		Occurences			
15	Total lines		3818			
16	Lines of code		2661			
17	Cyclomatic Index		553			
18	Supported controls		518			
19	Unsupported controls		0			
20						
21			*Note: effort is expressed in minutes*			
22						
23	**Code Tasks Details**			JDEV		SDEV
24	Name		Occurrences	Effort	Cost	Effort

Report - Entire Solution | Config - Resources | Config

Pronto 100%

Before you produce your first assessment report, a word of caution is in order. ***All time/cost estimations are based on the assumption that each warning/error must be fixed singularly.*** This assumption always brings to very high numbers, and therefore to a very expensive migration.

最初のアセスメントレポートを作成する前に警告しておきます。***すべての時間/費用の見積もりは各警告/エラーが個々に修正されなければならないという仮定に基づいて見積られています。***この仮定はつねに非常に高い数値となっており、そのため、非常にマイグレーション費用が高価となっています。

For example, if the original VB6 code contains many Variant variables, odds are that the VB.NET code will contains hundreds of warnings #0354 (“Unable to read default member of symbol 'XYZ'. Consider using the GetDefaultMember6 helper method.”). According to the standard Excel template, it takes 1 minute from a junior developer and 1 minute for a tester to fix this warning, which means that even a simple VB6 project might require many hours just to get rid of this warning.

例えば、元となる VB6 ソースが多くのバリアント変数を含む場合、VB.NET ソースが非常に多くの#0354 の警告(「'XYZ'オブジェクトの初期メンバーを読み込むことができません。GetDefaultMember6 ヘルパーメソッドの使用を検討してください。」)を含むで

しょう。標準のエクセルテンプレートにしたがえば、この警告を修正するために初級技術者の作業に 1 分、検査担当者の作業に 1 分かかります。これは、単純な VB6 プロジェクトでさえ警告を取り除くために多くの時間を必要とする、ということです。

In practice, however, you often need only a project-level pragma to avoid a whole set of warnings of same type. In this specific example, the following pragma makes VB Migration Partner generate a call to GetDefaultMember6 helper method, which nicely solves the problem and prevents the generation of #0354 warnings:

しかし、実際には、たいてい、同じタイプの警告を避けるためにプロジェクトレベルのプラグマがあれば十分です。特別な例として、次にあげるプラグマは VB Migration Partner に GetDefaultMember6 ヘルパーメソッドの呼び出しを生成します。これは、#0354 の警告が発生しないように問題をうまく解決します。

```
'## project:DefaultMemberSupport True
```

The bottom line is, be aware that cost/time evaluations you read in assessment reports are often over-estimated and should be taken with a grain of salt.

結論は、アセスメントレポートに記載されている費用/時間の見積りはたいてい過剰見積りなので、まともに受け取ってはいけない、ということです。

5. Generate wrappers for 3-rd party ActiveX controls／サードパーティ製 ActiveX コントロールに対してラッパーを生成する

This step is necessary only if your project uses one or more ActiveX controls that aren't in VB6 toolbox.

このステップは VB6 のツールボックスに含まれない ActiveX コントロールを使用する場合に必要となります。

For such ActiveX controls it is necessary that you generate and compile a *wrapper class*, a process that requires the following actions:

そのような ActiveX コントロールに対して、ユーザーは *ラッパークラス*を生成しコンパイルする必要があります。その過程は次の作業を必要とします。

1. Run the AxWrapperGen utility to generate the code for the wrapper class. The utility generates a single VB.NET class library project that contains one wrapper class for each control contained in an .ocx file.
   ラッパークラスのソースを生成するために AxWrapperGen ユーティリティーを実行します。このツールは一つの ocx ファイルを含んだ各コントロールに対するひとつのラッパークラスを含んだ単純な VB.NET クラスライブラリプロジェクトを生成します。
2. Load the generated VB.NET project in Visual Studio, read the instructions at the top of each wrapper class, and edit the code accordingly.
   Visual Studio で生成された VB.NET プロジェクトを開きます。各ラッパークラスの上の説明を読んで、それに従ってソースコードを修正してください。

3. Compile the project and deploy the resulting DLL in VB Migration Partner's install folder.
   プロジェクトをコンパイルし、作成された DLL を VB Migration Partner をインストールしたディレクトリに配置してください。

Editing the wrapper class is necessary only for complex ActiveX controls, such as grids. In most other cases, the code that AxWrapperGen generates compiles correctly at the first attempt.

グリッドのような複雑な ActiveX コントロールに対してはラッパークラスを修正する必要があります。その他のほとんどのケースでは、AxWrapperGen が生成するソースコードは 1 回で正常にコンパイルされます。

Notice that these operations are optional, in the sense that you might prefer to immediately skip to next step, without having prepared any ActiveX control wrapper. In this case you'll end up with one or more red rectangles on your VB.NET forms, each one working as a placeholder for an unrecognized ActiveX control. The decision of not converting ActiveX controls with AxWrapperGen can be convenient in some cases, for example if the ActiveX control is used only a few times in the entire project and you plan to replace it with a native .NET control anyway when the migration is completed.

これらの操作は選択可能であり、ActiveX コントロールのラッパーを準備せずに次の段階に進むこともできます。こういうケースでは、最終的には、VB.NET の上に一つ以上の、認識できない ActiveX コントロールのためのプレースホルダーとして機能する赤い四角が表示されます。AxWrapperGen を使って ActiveX コントロールを移行しないという決定はいくつかのケースで有益です。例えば、ActiveX コントロールが全プロジェクトで2、3回しか使用されない場合、ユーザーはとにかく変換が終わった後でネイティブのコントロールに置き換えることを計画します。

6. Reach the zero-compilation-errors stage／コンパイルエラーゼロステージに到達する

In this step you get rid of all the compilation errors in your VB.NET code, so that you can finally launch it.

この段階でユーザーは VB.NET ソースのすべてのコンパイルエラーを取り除きます。そうすれば最終的にアプリケーションを実行することができます。

If VB Migration Partner were a "traditional" conversion tool – like the Upgrade Wizard or any other conversion software on the market – you'd need to fix each and every compilation error in the VB.NET code. In other words, you wouldn't run the conversion tool any longer and just work on a "snapshot" of the converted code.

VB Migration Partner が Upgrade Wizard や市販されている他のコンバージョンツールのような従来のコンバージョンツールであったなら、VB.NET ソースのコンパイルエラーはすべて個々に修正しなければなりません。言いかえれば、ユーザーはそれ以上コンバージョンツールを実行しませんし、変換されたソースの「スナップショット」に対して作業するだけです。

Fortunately for you, VB Migration Partner is **not** a traditional conversion tool. Rather, it's a sophisticated software that accompanies you during all the phases in your migration initiative, from the initial planning to the test stage. The key of VB Migration Partner's power is the so-called **convert-test-fix** methodology.

幸いにも、VB Migration Partner は従来のコンバージョンツール**ではありません**。むしろ、マイグレーション開始からテスト工程の計画初期段階まで使用できる洗練されたソフトウェアです。VB Migration Partner の真価はいわゆる **convert-test-fix** メソドロジーにあります。

In a nutshell, the convert-test-fix methodology boils down to the following, simple statement: ***never, ever manually modify the converted VB.NET***. Instead, you apply one or more pragmas to the original VB6 code and re-run the migration. The great thing about pragmas is that you can apply them to the solution-, project-, file-, method- and variable-level. Quite often a few project-level pragmas can eliminate all compilation errors – or at least the large majority of them.

簡単に言うと、convert-test-fix メソドロジーは結局、次の簡単な記述で表せます。***絶対に変換後の VB.NET ソースを手修正しません***。その代わり、ユーザーは VB6 ソースに対してひとつ以上のプラグマを適用して、マイグレーションを再実行します。プラグマのすごいところは、ソリューション、プロジェクト、ファイルそして変数のレベルまで適用できることです。プロジェクトレベルのかなり多くのプラグマはすべての、もしくは少なくとも大部分のコンパイルエラーを取り除きます。

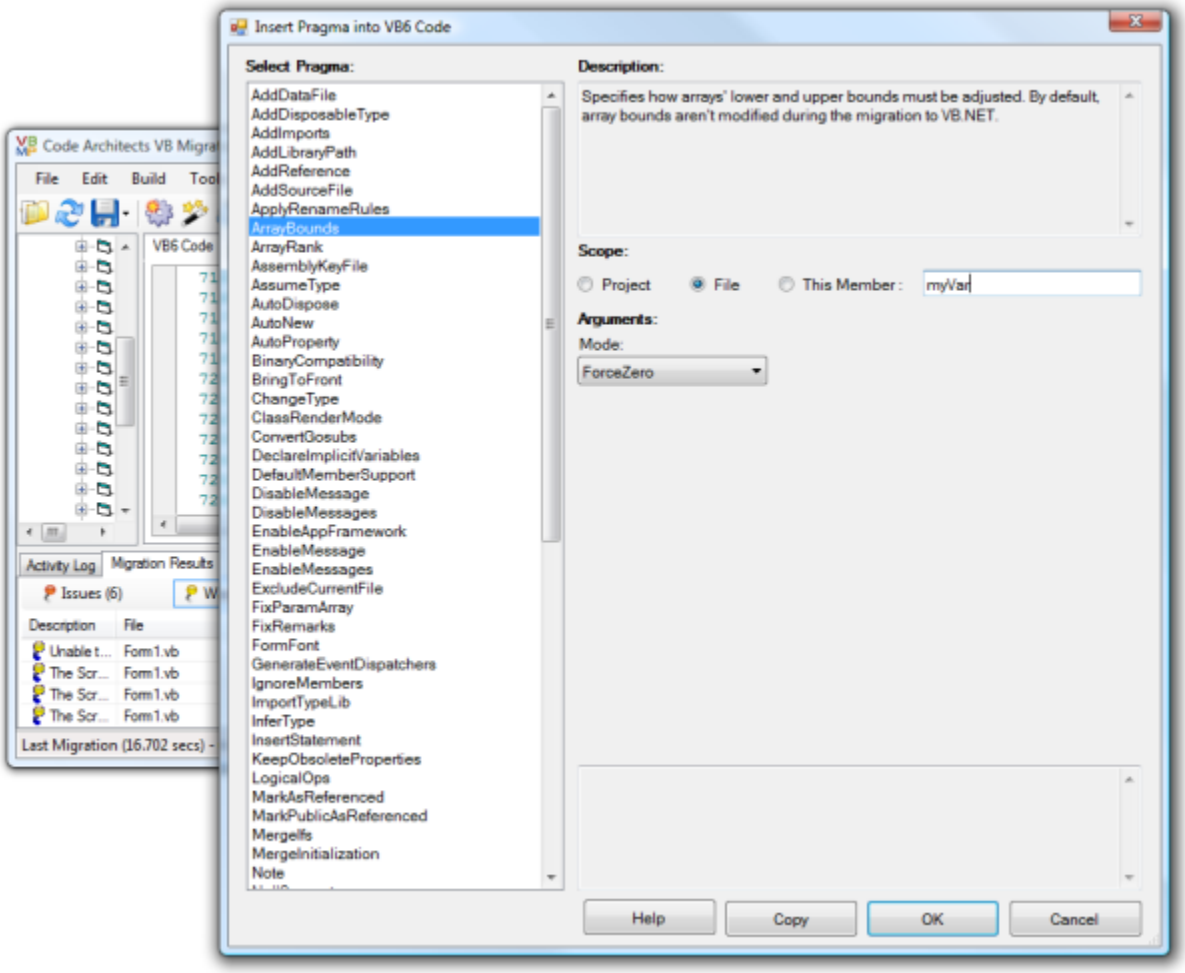
Basically, there are three ways for adding pragmas:

基本的に、プラグマを追加するには 3 つの方法があります。

1. type the pragma in the VB6 editor, save the file, then switch to VB Migration Partner, reload the project (using the Reload command in the File menu), and re-run the migration.
   VB6 のエディターでプラグマを記述し、ファイルを保存し、VB Migration Partner に切り替えて、（File メニューの Reload コマンドを使用して）プロジェクトを読み込み、マイグレーションを再実行する。
2. use VB Migration Partner’s integrated code editor to insert the pragma in the original VB6 code, save, and re-run the migration.
   VB6 ソースにプラグマを挿入するために VB Migration Partner に統合されたコードエディタを使い、保存して、変換を再実行する。
3. (for project-level pragmas only) use Notepad or another text editor to enter the pragma in a *.pragmas file in the same folder as the .vbp file, then reload the project using the Reload command in the File menu, and re-run the migration.
   （プロジェクトレベルのプラグマに対しては）vbp ファイルと同じディレクトリにある pragmas ファイルにプラグマを記入するために Notepad または他のテキストエディタを使い、File メニューの Reload コマンドを使用してプロジェクトを読み込み、変換を再実行する。

You can use the Insert Pragma command (in Edit menu) to have VB Migration Partner help you in selecting the right pragma for a specific purpose. In scenario A, use the Copy button and then paste the code right in the VB6 editor. In scenario B, click on the OK button to insert the code in VB Migration Partner’s integrated editor.

特定の目的のために正しいプラグマを選択するために(Edit メニューの)Insert Pragma コマンドを使用することができます。シナリオ A は、Copy ボタンを使い、VB6 エディタ上でコードを貼り付けます。シナリオ B は、VB Migration Partner に統合されたエディタ上でコードを挿入するために OK ボタンをクリックします。



To reach the zero-compilation-error stage as quickly as possible, it is essential that you become familiar with the many pragmas that VB Migration Partner. Our online documentation provides many details about pragmas, therefore it makes little sense to rehash all that information in this article. However, our experience is that the following pragmas are most useful in this phase of the migration:

コンパイルエラーゼロの段階にできるだけ早く到達するには、多くの VB Migration Partner のプラグマに慣れることが必須となります。弊社のオンラインドキュメントはプラグマに関する多くの詳細な記事を提供しており、この記事の情報のすべてを作り直すことはほとんど意味がありません。しかし、弊社の経験では、次のプラグマは移行におけるこの段階で最も有効です。

ImportTypeLib

is useful if VB Migration Partner fails to correctly recognize a type library.

このプラグマは VB Migration Partner がタイプライブラリを正しく認識できない場合に役立ちます。

SetName

allows you to assign a different name to a variable or member, in case the original VB6 name clashes with a VB.NET or .NET Framework member.

このプラグマは、元の VB6 の名前が VB.NET や.NET Framework で使用するとクラッシュする場合に、ユーザーに変数やメンバーに異なる名前を割り当てられるようにします。

ArrayBounds and ShiftIndexes

can fix problems with arrays having nonzero lower index.

このプラグマはインデックスの下限が 0 でない配列の問題を修正します。

ArrayRank

can be necessary if VB Migration Partner hasn't enough information to infer the number of dimensions of an array.

このプラグマは VB Migration Partner が配列の大きさを推測する情報が十分得られない場合に必要となります。

PreProcess and PostProcess

allow you to tweak the original VB6 code or the generated VB.NET code, respectively, to get rid of various errors. (Our Knowledge Base contains many examples of how these pragmas can be useful.)

このプラグマは様々なエラーを取り除くために、ユーザーに元の VB6 ソースや生成された VB.NET ソースをそれぞれ微調整できるようにします。(弊社のナレッジベースにはこれらのプラグマをどのように役立てるかの例が数多く含まれています。)

MergeInitialization

can be necessary if VB Migration Partner fails to correctly merge a variable with the first assignment to it.

このプラグマは VB Migration Partner が変数を、最初に割り当てられた変数に正しく統合できない場合に必要となります。

ClassRenderMode

should be used if you are migrating a public classed used as an interface, but VB Migration Partner can't generate the correct interface because the current ActiveX DLL project doesn't include an Implements statement that references the interface.

このプラグマはインターフェースとして使用されているパブリッククラスを変換するときに使用されるべきですが、VB Migration Partner は正しいインターフェースを生成することはできません。理由は、ActiveX DLL のプロジェクトがインターフェースを参照する Implements ステートメントを含んでいないからです。

Even if the following pragmas don't directly reduce the number of compilation errors, it's a good idea to include them in the early migration attempts, so that you can test the behavior of code that is more similar to its final form:

次のプラグマは直接的にコンパイルエラーを減らすわけではありませんが、マイグレーション初期に含めることはいい考えです。これによって、最終型と似たソースの動作をテストすることができます。

DeclareImplicitVariables

is useful for modules that lack an Option Explicit statement.

このプラグマは Option Explicit ステートメントが不足しているモジュールに対して役立ちます。

ConvertGosubs

attempts to convert Gosubs into calls to separated methods.

このプラグマは Gosubs を分離したメソッドの呼び出しに変換します。

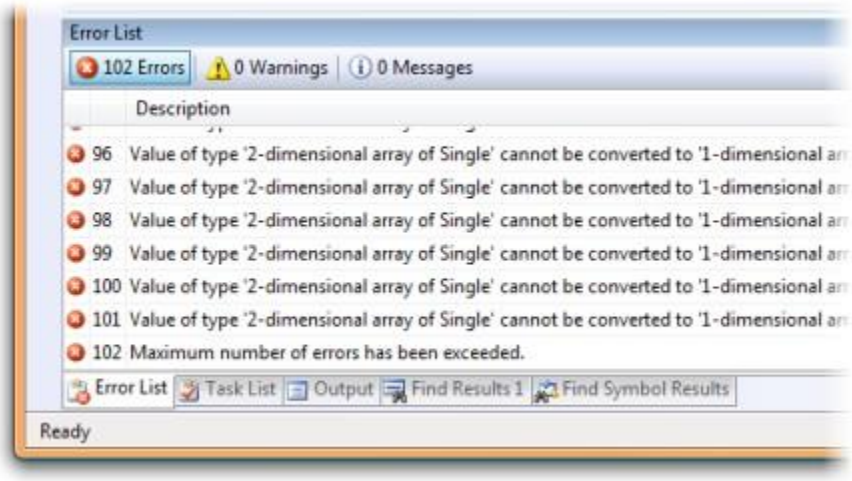
MergeIfs

refactors nested IFs into a single IF that uses the AndAlso operator.

このプラグマは入れ子になった IF を AndAlso 演算子を使う単純な IF に分解します。

Once again, the compile-test-fix methodology is an iterative process: you add one or more pragmas, convert to VB.NET and then compile the project (or load it into Visual Studio) to have a look at all compilation errors. Just remember that the VB.NET compiler never displays more than 102 compilation errors, therefore if you read “Maximum number of errors has been exceeded” in the Error List window you can’t actually know how many errors you’ll have to deal with.

もう一度言いますが、compile-test-fix メソドロジーは繰り返し作業です。ユーザーは一つ以上のプラグマを追加し、VB.NET に変換し、コンパイルエラーを確認するためにプロジェクトをコンパイル(または Visual Studio に読み込み)します。VB.NET コンパイラは 102 個以上のエラーを表示できないことは忘れないでください。これによって、エラーリストウインドウのエラーの数が最大値を超えていた場合、ユーザーが対処しなければならないエラーの数を把握することは実際には不可能です。



Here’s a simple tip that might significantly reduce your efforts to reach the zero-compilation-error stage: look for the following migration warnings in the generated VB.NET code:

ここに、コンパイルエラー0 の段階に到達するための作業を著しく減らせるかもしれない簡単な方法を記述します。生成された VB.NET ソースのマイグレーションの警告を探してください。

#0501:
Member isn’t used anywhere in current application.

メンバーはこのアプリケーションのどこにも使用されません。

#0511:
Member is referenced only by members that haven’t found to be used in current application.

メンバーはこのアプリケーションで使用されている箇所が見つからないメンバーからのみ参照されます。

#0521:
Unreachable code detected

使用されていないソースコードです。

These messages typically mark portions of VB.NET code that will be never executed and that should be dropped during the migration process. In general you'll take care of this code during the optimization step, but if the code contains one or more portions that are never used yet contain many compilation errors, it might make sense to drop those portions now. Instead of deleting the code in the original VB6 code, you can use theRemoveUnusedMembers and RemoveUnreachableCode pragmas with the Remarks argument, so that the code will be temporarily commented until you come to a final decision about whether you should really delete it.

これらのメッセージは通常、決して実行されない移行工程の間に削除されるべき VB.NET コードの箇所に印をつけます。一般的に最適化の段階でこのコードに気をつけますが、そのコードが一つ以上の決して使用されない箇所を含みかつ多数のコンパイルエラーを含んでいる場合、それらの箇所をその場で削除することは道理にかなっているかもしれません。元の VB6 コードのコードを削除する代わりに、RemoveUnusedMembers プラグマと RemoveUnreachableCode プラグマを Remarks 引数と共に使用して、そのコードを、本当に削除するべきかどうかを最終的に決定するまで一時的にコメント化することができます。

7. Reach the zero-runtime-errors stage／ランタイムエラーゼロステージに到達する

If the VB.NET code has no compilation errors, you can run it inside Visual Studio and check that it behaves as expected and that no unexpected errors occur at runtime.

VB.NET コードにコンパイルエラーがない場合、Visual Studio で実行して想定通りに動作することを確認したり、実行時に予期せぬエラーが発生しないことを確認することができます。

As for previous step, you should abide by the convert-test-fix methodology and refrain from manually editing the VB.NET code. Instead, when you find an unexpected or unwanted runtime behavior you should insert or more pragmas in the original VB6 code and then run VB Migration Partner once again.

前のステップに関しては、convert-test-fix メソドロジーを受け入れて、手作業で VB.NET コードを編集することを控えるべきです。その代わり、実行時に予期せぬ望ましくない動作を発見した場合は元の VB6 コードにプラグマを挿入して VB Migration Partner をもう一度実行するべきです。

The pragmas that are most likely to be useful to complete this stage are:

この段階を完了するためにもっとも役立ちそうなプラグマは以下のプラグマです。

DefaultMemberSupport
allows to infer a variable's default member at runtime and is useful with Object variables and with Variant variables that contain an object reference.

このプラグマによって、実行時に変数の初期メンバを推論することができ、かつオブジェクト変数やオブジェクト参照を含むバリアント変数に対して役立ちます。

InsertStatement, ReplaceStatement, ParseReplace, PreInclude and PostInclude
insert or replace VB.NET code in the generated project.

これらのプラグマは変換されたプロジェクトに VB.NET コードを挿入したり、置換したりします。

AutoNew

ensures that an auto-instancing (As New) variable preserves its semantics in VB.NET.

このプラグマは自動インスタンス化(As New)変数が VB.NET においてもその意味を持ち続けることを保証します。

UseSystemString

should be used with all Type…End Type structures that are passed to Declare methods.

このプラグマは、Declare に渡されるすべての Type…End Type 構造と共に使用されるべきです。

NullSupport

should be used for Variant expressions that deal with Null values.

このプラグマは NULL 値を扱うバリアント式に使用されるべきです。

AddDataFile

copies one or more data files into the .NET project folder and avoids “File not found” errors.

このプラグマは一つ以上のデータファイルを.NET プロジェクトフォルダにコピーして「ファイルが見つからない」エラーを回避します。

WrapDeclareWithCallbacks

is used to prevent the “orphaned delegate” problem that may occur when calling a Declare method that takes a delegate object (the AddressOf keyword).

このプラグマは(AddressOf キーワードの)デリゲートオブジェクトを受け取る Declare メソッドが呼ばれたときに発生する「デリゲートの孤立」を防ぐために使用されます。

Not all the runtime errors can be avoided or fixed by means of the above pragmas, though. The causes for unexpected errors can be just too many, but fortunately the VB Migration Partner’s support library automatically prevents most of these problems. All the known problems that the support library doesn’t handle automatically are documented in our Knowledge Base.

とはいえ、実行時エラーのすべてが上記のプラグマによって回避または修正できるわけではありません。予期せぬエラーの原因は非常に多いのですが、幸い VB Migration Partner のサポートライブラリは自動的にこれらの問題の多くを防ぎます。サポートライブラリが自動的に処理しない既知の問題は弊社のサポート技術情報に記述されています。

A common problem that you solve in this step is Null values. VB6 and VB.NET greatly differ in how Null values are handled. In VB6 you can pass a Null value to most string functions, including Left, Mid, and Len (but not Left$ or Mid$), in which case the function returns Null. The .NET equivalent for Null is DBNull.Value, except you get an exception if you pass this value to any string function. Typically you solve this issue by a combination of the NullSupport and ReplaceStatement pragmas, and you can also use the ChangeType and SetType pragmas to convert Variant variables into VB6Variant elements.

このステップで利用者が解決する一般的な問題は NULL 値です。VB6 と VB.NET は NULL 値の取り扱いにおいて大きく異なっています。VB6 では NULL 値を、Left、Mid、Len（Left$や Mid$は除く）というほとんどの文字ファンクションに渡すことができ、その場合の戻り値は NULL です。.NET において NULL に等しいのは DBNull.Value です。ただし、この値を文字ファンクションに渡す場合は式を使う場合を除きます。通常は NullSupport プラグマと ReplaceStatement プラグマの組み合わせを使用してこの問題を解決します。また、バリアント変数を VB6Variant 要素に変換するために ChangeType プラグマと SetType プラグマも使用することができます。

Also, keep in mind that unsupported methods and properties don't cause a runtime exception, by default. This means that – for example – the DrawMode property is simply ignored. This behavior is intentional, because it allows you to run and test the VB.NET code, even if with reduced functionality. However, after completing the first set of tests, you might want to change the behavior by adding the following line of code in the VB.NET project:

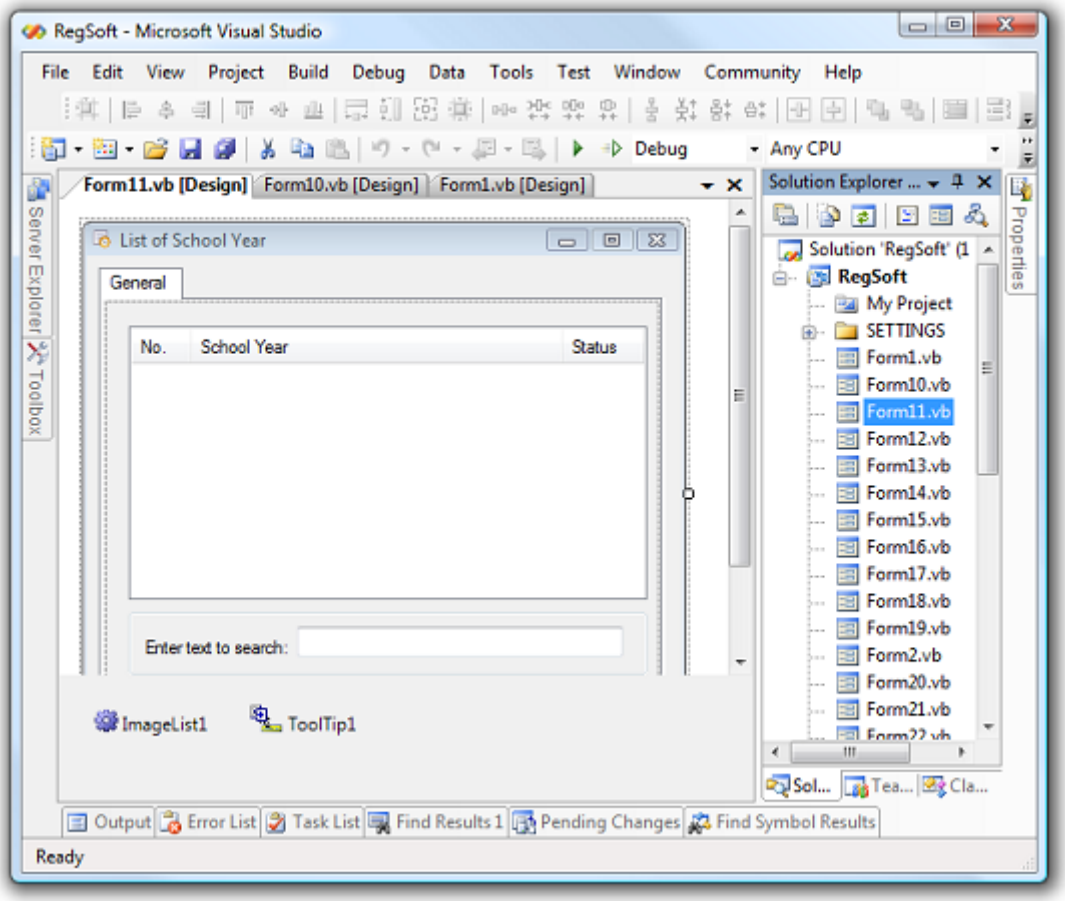
また、デフォルトではサポートされないメソッドとプロパティは実行時エラーを発生させないことに注意してください。これは、―例えば―DrawMode プロパティが単純に無視されるということを意味します。この動作は、たとえ機能性が損なわれても、それによって VB.NET コードを実行し検証することができるという理由による意図的なものです。しかし、最初の検証が完了した後では、VB.NET プロジェクトに以下のコードを追加して動作を変更したいと思うかもしれません。

```
VB6Config.ThrowOnUnsupportedMembers = True
```

Interestingly, VB Migration Partner's support library provides the ability to automatically generate a log for any fatal exception, which you enable as follows:

興味深いことに、VB Migration Partner のサポートライブラリは、致命的なエラーに対して自動的にログを生成する機能を持っています。それは以下のようにして有効にします。

```
VB6Config.LogFatalErrors = True
```



8. Achieve functional equivalence with original VB6 code／元の VB6 コードに対して機能的等価を達成する

After you successfully run the VB.NET project without any unexpected exception you can focus on refining the VB.NET code to achieve full functional equivalence with the original VB6 code.

予期しないエラーが発生せずに VB.NET プロジェクトを実行した後、元の VB6 コードと機能的に完全に等しくなるように VB.NET コードを修正する作業に入ることができます。

VB Migration Partner's support library ensures that this step takes much less time than a traditional conversion tool. However, you might need to add a few pragmas to account for minor differences, especially those related to the user interface.

VB Migration Partner のサポートライブラリは、このステップに掛ける時間が従来のコンバージョンツール以下であること保証します。しかしながら、特にユーザーインターフェースに関する小さな違いを解決するために2、3のプラグマを追加する必要があるかもしれません。

For example, a common problem in the user interface is caused by VB.NET not supporting system fonts. VB Migration Partner must therefore convert them into similar (but not perfectly identical) true type fonts, which tend to occupy slightly more space. If a caption matches perfectly in a VB6 Label or Button control's client area, odds are that a portion of the caption will be invisible after the migration to VB.NET.

例えば、ユーザーインターフェースにおいて通常起こる問題は VB.NET がサポートしないシステムフォントに起因します。そのため VB Migration Partner はそれらを（完全には一致しない）類似の true type フォントに変換しなければなりません。なおかつ、それらのフォントは若干余計にスペースを取る傾向があります。見出しが、VB6 のラベルやボタンコントロールのクライアント領域に完全に一致している場合、VB.NET に変換された後で見出しの一部が見えなくなるという違いがあります。

These are the pragmas that are most useful during this step:

以下はこのステップで最も有益なプラグマです。

EnableAppFramework
generates Windows XP-like user interface, specifies a splash screen, and more.

このプラグマは、スプラッシュスクリーン等を指定して、Windows XP のようなユーザーインターフェースを生成します。

FormFont, ReplaceFont, and ReplaceFontSize
allow you to fix issues related to fonts, for example when a caption isn't fully visible inside a Label or Button.

これらのプラグマをによって、フォントに関連する問題—例えば、ラベルやボタンの内側の見出しが完全に見えない場合—を解決することができます。

WriteProperty
permits to account for minor user-interface issues, for example to adjust the size or position of a control.

このプロパティは、—例えば、コントロールのサイズや場所を調整を必要とする—ユーザーインターフェースの小さな問題を解決します。

BringToFront and SendToBack
can be useful if a control is mistakenly hidden by other controls on the converted VB.NET form.

これらのプラグマは、変換後の VB.NET フォーム上でコントロールが他のコントロールによって誤って隠された場合に有益です。

FixParamArray
should be used with methods that modify the elements of a ParamArray argument, to preserve the by-reference semantics.

このプラグマは、「参照による」という意味を維持するために、ParamArray 引数の要素を変更するメソッドと一緒に使用されるべきです。

UseByVal
is often necessary to prevent exceptions when exiting a method that has received read-only properties.

このプラグマは、読み取り専用プロパティを受け取ったメソッドが存在する場合、例外を抑止するために頻繁に使用されます。

AddDisposableType and AutoDispose
help you to ensure that objects are released correctly and avoid exception when accessing a resource such as a file or a database connection.

これらのプラグマによって、ファイルやデータベースのようなリソースにアクセスする場合にオブジェクトが正しく解放され、また例外を回避することを保証することができます。

ThreadSafe
is necessary when migrating a DLL that can be used in multi-threaded scenarios, for example by ASP.NET clients.

このプラグマは、マルチスレッドシナリオで使用される DLL—例えば、ASP.NET クライアントによって使用される—を変換する場合に必要です。

It almost goes without saying that you can be sure that you have fully achieved complete functional equivalence only after stressing the VB.NET application with an exhaustive QA test, which usually takes a lot of time and energies. For this reason, you might want to postpone the QA test to the last step.

完全な品質保証検査—普通は非常に多くの時間と労力を使う—によって VB.NET アプリケーションをテストした後でのみ完全に機能的等価を達成したことを保証できることは言うまでもありません。そのため、品質保証検査を最後に回したいと思うかもしれません。

9. Optimize the converted VB.NET code／変換された VB.NET コードを最適化する

However, the ultimate goal of any migration process is to improve the application, for example by making it faster or more scalable. This is the kind of things you accomplish in this last step.

しかし、あらゆるマイグレーションの究極の目的はアプリケーションを改善すること—例えば、さらなる高速化や拡張性の向上—です。これはこの最後のステップで成し遂げる事柄です。

VB Migration Partner can be very helpful in this phase, because it emits warnings that draw your attention on the portions of your code that might require additional care, for example:

VB Migration Partner はこの工程でとても役立ちます。その理由は追加作業を必要とするかもしれないコードの箇所に注意を引くための警告を発行するためです。以下がその例です。

#0531:
You can replace call to unmanaged method with a .NET member.

呼び出しを.NET メンバのアンマネージドメソッドに置き換えることができます。

#0561:
A symbol was defined without an explicit “As” clause.

記号は明示的な「As」節を使用せずに定義されています。

#0571:
String concatenation inside a loop. Consider declaring the variable as a StringBuilder6 object.

ループ内部の文字連結です。変数を StringBuilder6 オブジェクトとして宣言することを検討してください。

#05B1:

The variable wasn't explicitly declared.

その変数は明示的に宣言されていません。

#07F1:

Type library is never used in current project. Consider deleting it from VB6 project references.

タイプライブラリは現在のプロジェクトで使用されていません。VB6 プロジェクトの参照から削除することを検討してください。

You can avoid these messages by either modifying the original VB6 code or by inserting a pragma. The pragmas that are most useful in code optimization are:

VB6 コードを変更するかプラグマを挿入して、これらのメッセージを回避することができます。コード最適化に最も有益なプラグマは以下のプラグマです。

RemoveUnusedMembers and RemoveUnreachableCode

delete portions of VB.NET code that aren't actually used.

これらのプラグマは実際には使用されない VB.NET コードを削除します。

InferType and SetType

allow you to generate a specific type for Object and Variant variables.

これらのプラグマによってオブジェクトやバリアント変数用の特定の型を生成することができます。

ApplyRenameRules

generates code that is compliant with .NET Framework naming guidelines.

このプラグマは.NET Framework の命名基準に準拠するコードを生成します。

BinaryCompatibility

ensures that the generated .NET DLL is compatible with existing COM clients.

このプラグマは、生成された.NET DLL が既存の COM クライアントと互換性があることを保証します。

AssemblyKeyFile

generates assemblies that are signed with a strong name.

このプラグマは、厳密な名前で署名されたアセンブリを生成します。

AutoProperty

converts a field into a property, for better data encapsulation.

このプラグマは、よりよいデータのカプセル化のために、フィールドをプロパティに変換します。

UseTryCatch

attempts to replace On Error statements with Try... Catch blocks.

このプラグマは、On Error 文を Try... Catch ブロックに置き換えようとします。

AddImports

allows you to simplify long namespaces in code.

このプラグマによって、コード上の長い名前空間を簡素化することができます。

Arguably, the most effective optimization technique has to do with string concatenation (see warning 0571). This article explains how you can take advantage of the StringBuilder6 class to speed up string operations by two or three orders of magnitude.

ほぼ間違いなく、最も効果的な最適化技法は文字連結(警告 0571 参照)と関係がある。こちらの記事は文字列操作を 100 倍から 1000 倍高速化するための StringBuilder6 クラスの利用方法について説明しています。

If you are migrating an N-tier application that is split in many DLLs, you might also want to improve the startup time by the **VB6Config.ParsingAssembliesAtStartup** property to False. For more info, read this article.

多数の DLL で分割された N 層アプリケーションを移行している場合、**VB6Config.ParsingAssembliesAtStartup** プロパティを False にすることで起動時間を改善したいと考えるかもしれません。詳細はこちらの記事を読んでください。

In the optimization step you should also attempt to remove any dependency from unmanaged code, for example:

最適化の段階では、以下のように、アンマネージドコードへの依存を取り除くことも検討すべきです。

1. discard any reference to COM type libraries and replace them with references to .NET DLLs.
   COM タイプライブラリへの参照は破棄して、.NET DLL への参照に置き換えてください。
2. modify the wrapper classes generated in step 5, so that the class inherits from a native .NET control rather than the original ActiveX control.
   ステップ 5 で生成されたラッパークラスを修正して、そのクラスが、元の ActiveX コントロールではなくネイティブの.NET コントロールから継承するようにしてください。
3. revise your ADO, DAO, or RDO code and use ADO.NET instead.
   ADO、DAO、RDO のコードを修正して、代わりに ADO.NET を使ってください。
4. attempt to replace calls to Declare statements with native .NET objects and methods.
   Declare 文の呼び出しをネイティブの.NET オブジェクトとメソッドに置換することを検討してください。

10. Extend the VB.NET application／VB.NET アプリケーションを機能拡張する

Strictly speaking, at this point the migration from VB6 has been completed and you can start selling it or deploy it at your existing customers' site. In practice, however, you often wish to add new features to the application before releasing it to your customers.

厳密に言えば、この時点で VB6 からの移行は完了していますので、それを販売することも既存の顧客サイトに配布することも可能です。しかし、実際には、顧客にリリースする前にアプリケーションに対して新しい機能を追加したい場合が多いです。

A converted application can be extended with new classes exactly like a VB.NET project built from scratch inside Visual Studio. You can add new controls to existing (migrated) forms or add new forms that weren't included in the original VB6 project. The one thing you cannot do is dropping controls from the VB Migration Partner library onto a plain .NET form.

移行されたアプリケーションは、Visual Studio でゼロから作成された VB.NET プロジェクトとまったく同じように、新しいクラスで拡張することができます。既存の（変換された）フォームに新しいコントロールを追加したり、元の VB6 プロジェクトには含まれていなかった新しいフォームを追加することができます。たった一つ不可能なことは VB Migration Partner のライブラリから何もない.NET フォームの上にコントロールを配置することです。

All the controls in VB Migration Partner's library – with few exceptions – inherit from a native .NET control. For example, the VB6TextBox control inherits from the System.Windows.Forms.TextBox control. This article shows how you can leverage the extra features offered by the .NET controls, thanks to inheritance and the special**NetObject** property.

VB Migration Partner のライブラリのコントロールはすべて—2、3の例外はありますが—ネイティブの.NET コントロールを継承しています。例えば、VB6TextBox コントロールは System.Windows.Forms.TextBox コントロールを継承しています。こちらの記事は、.NET コントロールによって提供された—継承と特別な **NetObject** プロパティによる—余分な機能の使用方法を説明しています。

For example, the VB6ComboBox control exposes several properties and methods that are missing in the VB6 ComboBox – including the DropDownWidth, DropDownHeight, and FormatString properties – because it inherits them from the .NET ComboBox. You can assign these properties in the code-behind portion of the form by using a WriteProperty pragma, or at runtime using an InsertStatement pragma.

例えば、VB6ComboBox コントロールは、.NET の ComboBox から継承したので、VB6 にはなかった複数のプロパティとメソッド—DropDownWidth プロパティ、DropDownHeight プロパティ、FormatString プロパティを含む—を持っています。これらのプロパティを、WriteProperty プラグマを使って、または実行時に InsertStatement プラグマを使って、フォームのコードビハインド部分に割り当てることができます。

Of course, you can also just manually edit the VB.NET form and code, however we recommend that you continue to use pragmas and the convert-test-fix methodology as long as possible, or at least until you are absolutely certain that you can throw the VB6 source away and focus exclusively on the VB.NET version of your application.

もちろん、VB.NET フォームやコードを手作業で修正することもできますが、私たちはできる限り、もしくは、少なくとも VB6 ソースを捨てて VB.NET のバージョンのみを対象に作業できると完全に確信するまでは、プラグマと convert-test-fix メソドロジーを使い続けることを推奨します。

Wrapping up／まとめ

This document explains how you can convert a VB6 application of average complexity into VB.NET using VB Migration Partner in ten *relatively easy* steps. Obviously, not all the steps described here equally easy.

本書は平均的な複雑さの VB6 アプリケーションを VB.NET に変換する方法を*比較的やさしい* 10 のステップで説明したものです。はっきりしていることは、ここに述べられたステップすべてが簡単であるとは言えないことです。

Likewise, each migration project is a story of its own, therefore it's quite difficult to predict how long each step will take. In simpler projects you might be able to skip a few steps entirely, for example manual editing of the VB6 code (step 4) or the optimization phase (step 10).

同様に、それぞれのプロジェクト移行は独自の歴史を持っているため、各ステップにどの程度時間がかかるかを予測することはかなり難しいことです。比較的単純なプロジェクトでは、2、3のステップ—例えば、VB6 コードの手修正（ステップ 4）や最適化の工程（ステップ 10）—を飛ばすことも可能かもしれません。

In all case, the most effective route towards a successful migration begins when you send us an email with the report from the VB6 Bulk Analyzer. From that point on, we can give you a hand.

いかなる場合においても、移行成功への最短の道のりは、VB6 Bulk Analyzer から得られたレポートをつけて弊社にメールを送信するときに始まります。そこから、私たちは参加することができます。

注記：
日本国内でのご使用につきましては弊社（インフォーテック）の無料診断サイトよりお問い合わせください。

WHITE PAPERS／ホワイトペーパー

Comparing VB Migration Partner with Upgrade Wizard／VB Migration Partner とアップグレード・ウィザードの比較

A common question coming from VB6 developers is how well VB Migration Partner compares with Upgrade Wizard, the conversion tool provided for free inside Microsoft Visual Studio, **in terms of compilation errors and warnings.**

VB6 技術者から寄せられる共通的な質問は VB Migration Partner が、**コンパイルエラーと警告に関して、**Microsoft Visual Studio が、無償提供している変換ツールである、Upgrade Wizard に比べてどのぐらい優れているかということです。

This document answers this question with two complementary approaches. In the first part, it illustrates the results you can get when running both programs over a number of VB6 projects. In the second part it describes how both programs solve (or fail to solve) the most common issues you face when migrating VB6 apps to .NET.

このドキュメントは、この疑問に二つの相補的なアプローチによって回答します。まず最初の部分で、多数の VB6 プロジェクトに対して両方のプログラムを実行し、得られた結果について説明します。次に、両方のプログラムが VB6 から.NET へのマイグレーションの際に直面する共通の課題に対する解決方法（または解決不可能なケース）について述べます。

Test #1: Migrate open source VB6 projects／検証 #1 : VB6 のオープンソースプロジェクトを変換する

Instead of just providing a series of unverifiable numbers, we run both tools against a group of open source code samples which offer a variety of challenges, including rarely-used controls, data-binding, graphic methods, and drag-and-drop.

客観的ではない検証結果を数多く提供するよりも、弊社はほとんど使用されないコントロールやデータバインディング、グラフィックメソッド、ドラッグアンドドロップを含む、様々な課題を含むオープンソースの一群に対して両方のツールを実行します。

We didn't edit any executable statement in the original VB6 nor in the converted VB.NET code. In about half of these cases, however, we made a second pass through VB Migration Partner after adding all the pragmas that were necessary to reach a fully functional .NET project.

弊社は元の VB6 の実行可能ステートメントおよび変換後の VB.NET のソースコードには一切手を加えておりません。しかし、それらの約半数において弊社は、必要なすべてのプラグマを追加することで、VB Migration Partner を使って、完全な機能を備えた.NET プロジェクトに変換することができました。

We benchmarked the Upgrade Wizard installed with Microsoft Visual Studio 2005 and release 1.00.06 of VB Migration Partner, on a Intel Core Duo 7800 @ 2.66GHz running Microsoft Windows Vista 64-bit SP1. **(NOTE: we ran these tests before Microsoft Visual Studio 2008 and VB Migration Partner version 1.10 were released.)**

弊社は、Intel Core Duo 7800 @ 2.66GHz で Microsoft Windows Vista 64-bit SP1 が動作する環境で、Microsoft Visual Studio 2005 と共にインストールされた Upgrade Wizard と VB Migration Partner のリリース 1.00.06 をベンチマークテストにかけました。
（※弊社はこれらのテストを Microsoft Visual Studio 2008 と VB Migration Partner 1.10 がリリースされる前に行いました。）

The results are available in a Microsoft Excel spreadsheet for all to see.

この結果は Microsoft Excel のスプレッドシートとなっており、誰でも見ることができます。

	VB6 Project		Upgrade Wizard			VB Migration Partner					
Code sample	Source files	Lines of code	Compilation errors/warnings		Time (secs.)	Compilation errors/warnings		Errors/warnings (after pragmas)		Number of pragmas	Time (secs.)
School	33	2661	80	13	80	12	28	0	28	3	15
Grid-Net Waves 3D	4	806	102	0	14	102	0	0	0	1	3
ID Card Maker	4	1139	102	0	29	12	7	0	7	10	6
Classic VB	6	382	41	2	20	0	10	0	10	1	4
Code Library	11	858	7	2	26	0	0	0	0	3	6
PhoneBook	7	526	102	0	22	1	0	0	0	5	6
3D define	1	242	45	1	13	0	0	0	0	0	5
MP3 Player	7	3445	26	22	27	0	5	0	5	3	6
Stars - Virtual Night Sky	1	794	74	4	15	0	0	0	0	4	4
ezDatabase	4	1098	6	3	17	2	0	0	0	4	4
CD Archive	2	534	13	14	21	1	2	0	2	1	8
Archive Explorer	17	3261	17	68	22	1	0	0	0	1	7
A complete Web browser	3	214	48	0	37	0	6	0	6	0	22
Spell checker	2	214	8	1	12	0	0	0	0	0	2
Barcode generator	4	4694	38	2	34	0	4	0	4	2	13
SQL Server Documenter	4	548	9	6	23	0	0	0	0	0	6
Pacman	4	808	24	0	19	1	0	0	0	12	4
Mastermind	3	4291	102	0	37	0	0	0	0	0	8
BC-52	8	1480	23	10	35	0	0	0	0	13	6
Battleship Online	6	1682	10	3	75	0	0	0	0	6	10
EGL 25	8	775	15	3	20	0	1	0	1	0	5
Cool Progress Bar	4	336	20	0	15	0	0	0	0	9	3
LCD Clock	3	227	3	2	14	0	0	0	0	0	3
Type-N-Sign	6	407	13	1	21	0	0	0	0	2	5
A common calculator	1	172	4	0	12	0	0	0	0	0	3
Photo Album	3	232	17	0	12	0	0	0	0	0	3
Expression Evaluator	3	415	0	8	11	0	0	0	0	0	3
Caro	2	249	14	2	10	4	0	0	0	1	3
Tetris	1	495	14	0	16	6	0	0	0	1	3
Mouse Recorder	7	710	13	6	15	0	0	0	0	0	4
Ald WinInvaders	3	428	0	5	15	0	0	0	0	0	3
FMStocks Core	14	1277	8	21	24	0	1	0	1	0	6
Total	186	35400	998	199	763	142	64	0	64	82	189

A summary of the results:
検証結果要約検証

- VB Migration Partner generates nearly **5x fewer compilation** errors than the Upgrade Wizard before adding a single pragma. In four cases, Upgrade Wizards generates just too many compile errors for Visual Studio to display (the max number is 102 errors), therefore the **actual ratio is even higher.**
  VB Migration Partner の変換後のコンパイルエラーはプラグマを追加する前でも Upgrade Wizard の **1/5** にすぎません。4 つのケースにおいて、Upgrade Wizards による変換後のコンパイルエラーは Visual Studio 上で表示できる上限(102)以上でしたので、**実際の割合は遥かに多いと考えられます。**

- When comparing Upgrade Wizard and VB Migration Partner, the mere number of compilation errors in generated code is quite misleading, because in most cases, VB Migration Partner permits you to get rid of all these errors with just **one or two** pragmas.
  Upgrade Wizard と VB Migration Partner を比較するとき、コンパイルエラーの数はあまり当てになりません。なぜなら、多くのケースにおいて、VB Migration Partner を使えば、**一つか二つ**のプラグマを適用することによってすべてのエラーを取り除くことができるからです。

- Considering all the pragmas used in all samples delivers we get an average of one pragma every 430 line of code; in a real application you need even fewer pragmas because a single pragma can affect all the files and statements in the project.
  サンプルすべてに使用されたプラグマについて考えると、弊社は平均 430 行に1つのプラグマを使用しました。しかし、実際のアプリケーションにおいては、ユーザーが必要とするプラグマはもっと少なくなります。理由は、プラグマによってはプロジェクト内の全ファイル、全コードに効果があるからです。

- The Upgrade Wizard processes about 46 LOCs per second on the average, VB Migration Partner runs at 187 LOCs per second, i.e. **4 times faster**. However, these timings are misleading, because at the end of the process VB Migration Partner generates code statistics and reports that the Upgrade Wizard doesn't.
  Upgrade Wizard は平均して毎秒 46 行を処理していますが、VB Migration Partner は毎秒 187 行処理しています。これは**約 4 倍の速さ**です。しかし、この時間も正確ではありません。なぜなら、VB Migration Partner は処理の最後に統計レポートを生成するからです。こういう統計レポートは Upgrade Wizard にはありません。

Interestingly, most of the compilation errors you get from VB Migration Partner come from the first two code samples – School and Grid-Net Waves 3D. If you exclude them from the stats, it turns out that VB Migration Partner averages at **one compilation error every 1140 LOCs** before adding a single pragma, and is therefore**about 30 times better than the Upgrade Wizard** (which delivers one compilation error every 40 LOCs). On large, real-world VB6 projects we've seen that the actual ratio is between 8x and 12x.

興味深いことに、VB Migration Partner による変換後のコンパイルエラーの大半はサンプルコードの 1 番目(School)と 2 番目(Grid-Net Waves 3D)によるものです。この二つを除くと、プラグマを追加する前の VB Migration Partner による変換後のコンパイルエラーの平均値は **1140 行につき1つ**となります。すなわち、**Upgrade Wizard に比べると約 1/30** となります。(Upgrade Wizard のコンパイルエラーの平均値は 40 行につき1つ。)弊社で確認した、より大きな実際の VB6 プロジェクトでは、1/8～1/12 となりました。

It's important to bear in mind that compilation errors and warnings tell only a part of the story, because even a VB.NET project with zero compilation errors might raise one or more runtime errors. In other words, the samples that show zero compilation errors after the migration with the Upgrade Wizard might require a lot of additional work to run correctly. By comparison, the column labeled as Number of pragmas indicates how many pragmas were needed to have a **fully functional** VB.NET application that raises no runtime errors.

重要なことは、コンパイルエラーと警告についての検証結果は VB Migration Partner の優位性の一つにすぎないということです。なぜなら、コンパイルエラーのない VB.NET プロジェクトであっても、一つ以上のランタイムエラーを発生させる可能性があるからです。言いかえれば、Upgrade Wizard によって変換されたコンパイルエラーがないサンプルであっても、正常に動作させるためには非常に多くの労力を必要とするかもしれないということです。それに比べて、Number of pragmas 列は VB.NET アプリケーションがランタイムエラーを起こさずに**完全に動作する**ために要したプラグマの数を表しています。

An important note: These values are to be used only to compare VB Migration Partner's speed and precision as relative to the Upgrade Wizard, not as a way to measure VB Migration Partner's performance in absolute terms. When running on large, real-world projects, VB Migration Partner tends to be up to 7-8x times faster than Upgrade Wizard.

重要事項: これらの数値は VB Migration Partner の速度と正確性を Upgrade Wizard と比較するためにのみ使用されるものであって、VB Migration Partner の性能を計測するための絶対的な尺度ではありません。実際の巨大なプロジェクトで実行した場合、VB Migration Partner は Upgrade Wizard に比べて約7～8倍の速さで処理するようです。

Along the same lines, the number of pragmas that are necessary to reach a fully functional VB.NET application tend to decrease – in percentage over the total number of executable lines – when the VB6 application gets larger, because often one single pragma scoped at the project-level can solve all similar occurrences of a given compile or runtime error.

同じように、VB6 アプリケーションが大きければ大きいほど、VB.NET アプリケーションが完全に動作するために必要となるプラグマの数（有効行数に対する割合）も少なくなる傾向があります。なぜなら、有効範囲がプロジェクトレベルのひとつのプラグマは類似のコンパイルエラーやランタイムエラーをすべて解決してしまうことがあるからです。

Test #2: Aivosto's compatibility checklist／検証 #2: Aivosto 社製互換性チェックリスト

Another way to compare VB Migration Partner with Upgrade Wizard is checking which migration issues either software can solve automatically.

VB Migration Partner と Upgrade Wizard をを比較する二つ目の方法はどのツールがどのマイグレーション課題を自動的に解決することができるかを確認することです。

VB Project Analyzer is a popular tool available on Aivosto's web site. It performs code analysis, dead code detection and removal, coding and naming rule enforcement. It can find common programming errors (including memory leaks caused by undisposed API handles), can optimize your code much faster than the fastest and smartest developer, and can generate a thorough documentation of all classes, forms, and members (including cross-reference data to detect who call whom). Best of all, it works with VB6, VB.NET, and VBA.

VB Project Analyzer は Aivosto 社のウェブサイトで公開されている著名なツールです。このツールはコード分析、デッドコードの検出と除去、コーディングルールとネーミングルールの適用を行うことができます。また、一般的なプログラミングエラー（API ハンドルを解放しないことによるメモリリークを含む）を見つけることができ、最高レベルの技術者よりも素早くコードを最適化することができるばかりか、クラス、フォーム、メンバーのすべてについての完全なドキュメント（何が何を呼び出しているのかを確認するためのクロスリファレンスを含む）を生成することができます。何よりも、このツールは VB6 でも、VB.NET でも VBA でも動作します。

If you are preparing your VB6 apps for migration to .NET, Aivosto's VB Project Analyzer is also very helpful, because it can automatically spot most VB6 language elements that the Update Wizard doesn't convert correctly to VB.NET. The online help includes the list of all compatibility checks that VB Project Analyzer performs. Please refer to the original list for an explanation of each compatibility issue.

もし、.NET へ移行する VB6 アプリケーションがあるのであれば Aivosto 社の VB Project Analyzer もまた非常に役に立ちます。なぜなら、Update Wizard が VB.NET に正しく変換できない VB6 の言語的要素を自動的にマークすることができるからです。また、オンラインヘルプには VB Project Analyzer が生成した互換性チェックリストが含まれています。各互換性の課題の説明についてはオリジナルのリストを参照してください。

NOTES

VBMP stands for VB Migration Partner, **UW** stands for Upgrade Wizard

✔ means that a given feature is supported by VB Migration Partner (VBMP)

✔ * means that the feature appears in Aivosto compatibility check but is supported by Upgrade Wizard 2008 (UW)

✔ P means that VBMP supports the feature only partially: for example, it doesn't cause a compilation error yet it doesn't ensure functional equivalence at runtime.

注記

VBMP は VB Migration Partner を意味し、UW は Upgrade Wizard を意味します。

✔ は与えられた課題が VB Migration Partner によって対応されていることを意味しています。

✔ * は Aivosto 社の互換性チェックに現れる課題ではあるものの Upgrade Wizard 2008 によって対応されていることを意味しています。

✔ P は VBMP が一部対応している課題であることを意味しています。例えば、コンパイルエラーは起こさないものの、実行時に完全に動作することを保証していない場合などです。

Add-in model changed in VB.NET VB.NET ではアドインモデルが変更されました。	VBMP is unable to migrate add-ins because the IDE object model is too different, but it emits a warning. IDE オブジェクトモデルが違い過ぎるため、VBMP はアドインを変換することができませんが、警告メッセージは出力します。
ADO required for data binding in VB.NET VB.NET では ADO はデータバインディングを必要とします。	✔ VBMP supports DAO, RDO and ADO data-binding, including binding with DataEnvironment objects, ADO data source classes, and simple-bound data consumer classes. VBMP は DataEnvironment オブジェクトや ADO データソースクラス、単純データバインドのコンシューマクラスを含む、DAO、RDO、ADO のデータバインディングに対応しています。
Array must start at 0 in VB.NET VB.NET では配列は 0 から始めなければなりません。	✔ VBMP supports several strategies for migrating arrays with non-zero LBound. Not only does it fix the array declaration, it can even modify the index used to reference individual array elements. VBMP は下限が 0 ではない配列の移行に対して複数の対応方法を持っています。配列の宣言を固定するだけでなく、個々の配列要素を参照するために使用されるインデックスを修正することもできます。
As Any not allowed in VB.NET VB.NET では As Any は許可されていません。	✔ VBMP correctly converts As Any arguments by producing one or more overloads of the Declare statement. VBMP は Declare ステートメントのオーバーロードを一つ以上生成することによって As Any を正しく変換します。
As New doesn't auto-instantiate if object released in VB.NET VB.NET では、As New は、オブジェク	✔ VBMP optionally supports the lazy-instancing feature of As New variables. VBMP は As New 変数の遅延インスタンス化対応をオプション対応にしています。

トが解放された場合、自動インスタンス化しません。

As New unsupported for arrays in VB.NET
VB.NET では配列に対する As New をサポートしていません。

VBMP can correctly translate As New arrays by preserving the VB6 semantics.
VBMP は VB6 の動作を維持することによって As New の配列を正しく変換することができます。

ByRef property params unsupported by VB.
ByRef プロパティパラメータはサポートされていません。

VBMP converts ByRef arguments inside properties into ByVal arguments because VB.NET requires it; however it emits a warning if the argument appears to be modified inside the property procedure ? or is passed to another method that can modify it.
VBMP は VB.NET の仕様に従って、ByRef 引数を内部プロパティに変換し ByVal 引数に格納します。しかし、引数がプロパティプロシージャ内部で変更されると推測される場合、または、それが、引数を変更することができるメソッドに渡された場合、警告メッセージを出力します。

ByVal/ByRef not allowed in API calls in VB.NET
VB.NET では API コールで ByVal/ByRef を使用することはできません。

VBMP safely resolves ByVal and ByRef in calls to API methods.
VBMP は API メソッドの呼び出しにおける ByVal と ByRef を安全に解決します。

Circle and Oval unsupported by VB.NET
VB.NET では Circle と Oval はサポートされていません。

VBMP correctly converts Line and Shape controls, and even translates graphic methods such as Line, Circle, PSet, PaintPicture, etc.
VBMP は Line コントロールと Shape コントロールを正しく変換します。また、Line、Circle、PSet、PaintPicture などのようなグラフィックメソッドも変換することができます。

Class Instancing changes in VB.NET
VB.NET ではクラスのインスタンス化は変更されました。

VBMP deals with SingleUse objects as if they were MultiUse, because .NET has no notion of “single use” objects. Global objects are converted correctly.
VBMP はあたかも MultiUse であるかのように SingleUse オブジェクトを取り扱います。理由は、.NET が「single use」という考え方を持っていないためです。Global オブジェクトは正しく変換されます。

COM method not callable from VB.NET
VB.NET から COM メソッドは呼び出せません。

VBMP can’t handle COM module methods.
VBMP は COM モジュールのメソッドをハンドルすることができません。

COM+/MTS not upgradable to VB.NET
COM+/MTS は VB.NET へアップグレードすることができません。

VBMP correctly converts all frequently used MTS/COM+ features into the corresponding .NET features.
VBMP はよく使用される MTS/COM+の機能のすべてを対応する.NET の機能へ正しく変換します。

Conditional block will not upgrade to VB.NET
条件付きブロックは VB.NET へアップグレードすることができません。

VBMP doesn’t migrate code inside an #IF block whose condition is false.
VBMP は条件が false の場合の#IF ブロックの内部コードを変換することができません。

Control unsupported by VB.NET
VB.NET でサポートされていないコントロール

VBMP converts 60+ controls, including all those included in the VB6 toolbox with the only exception of OLE Container. It supports other commonly used controls such as WebBrowser and ScriptControl, and all the windowless controls in the MSWLESS library.
VBMP は、OLE コンテナを除く VB6 のツールボックスに格納することができるものすべてを含む、60 以上

		のコントロールを変換します。また、その他の一般的に使用される WebBrowser や ScriptControl、MSWLESS ライブラリの非ウインドウコントロールすべてをサポートしています。
DDE unsupported by VB.NET VB.NET では DDE はサポートされていません。	P	VBMP supports DDE communications, among migrated VB.NET apps. VBMP は変換された VB.NET アプリケーション間の DDE 通信をサポートします。
Diagonal line unsupported by VB.NET VB.NET ではDiagonal line(斜線)はサポートされていません。	*	VBMP supports the Line control, with any inclination and style. VBMP は傾き、スタイルに関わらず Line コントロールをサポートしています。
DoEvents() returns no value in VB.NET VB.NET では DoEvents()は値を返しません。		VBMP provides a DoEvents6 replacement statement that returns the number of open forms. VBMP は開かれているフォームの数値を返す DoEvents6 ステートメントを提供します。
Drag-and-drop requires rewrite for VB.NET ドラッグアンドドロップは VB.NET 用に書き直す必要があります。		VBMP fully supports OLE drag-and-drop, in both the manual and automatic flavors. Starting with version 1.20, VBMP version also supports “classic” (non-OLE) drag-and-drop. VBMP はマニュアル/自動に関わらず OLE ドラッグアンドドロップに完全対応しています。バージョン 1.20 以降であれば、クラシック(非 OLE)にも対応しています。
Event behavior changes in VB.NET VB.NET ではイベントの動作が変更されています。		VBMP fully supports all these (and other) events, no work is required after update. VBMP はすべてのイベントに完全対応しています。変換後の追加作業はありません。
Event log model differs in VB.NET イベントログモデルが VB.NET では異なっています。		VBMP supports all the Event Log-related properties and methods. VBMP はイベントログに関するすべてのプロパティとメソッドに対応しています。
Initialized arrays in UDTs unsupported by VB.NET VB.NET ではユーザー定義型の初期化配列がサポートされていません。		VBMP correctly initializes UDTs containing arrays, fixed-length strings, and auto-instancing (As New) object variables. It even generates special code to correctly convert assignments between UDTs that contain these members. (UW doesn’t even emit a warning in that case.) VBMP は配列、固定長文字列、自動インスタンス化(As New)オブジェクト変数を含むユーザー定義型を正しく初期化することができます。また、これらのメンバーを含むユーザー定義型同士の代入を正しく変換するための特別なコードを生成することもできます。(UW はこのようなケースにおいて警告メッセージを出力することさえできません。)
MDIForm event unsupported in VB.NET VB.NET では MDIForm イベントがサポートされていません。		VBMP correctly handles mouse-related events inside MDI forms. VBMP は MDI フォーム内部のマウス関連イベントを正しくハンドルすることができます。
Member cannot be default in VB.NET VB.NET ではメンバーをデフォルトにすることができません。		VBMP offers the same degree of support that UW does. In addition, VBMP can convert a default method with parameters into a VB.NET ReadOnly Property and then mark it as the default member of that class. VBMP は UW と同レベルのサポートを提供します。さらに、VBMP はパラメータを伴うデフォルトメソッドを変換して、VB.NET の ReadOnly プロパティに格納し、それをそのクラスのデフォルトメンバにすることができます。
Module not upgradable to VB.NET	P	VBMP doesn’t migrate DHTML and WebClass components. However, it converts UserDocument and

標準モジュールは VB.NET にアップグレードすることができません。		PropertyPages into VB.NET UserControls, thus you have something to work with after the migration even though you'll need additional manual coding to have it work as expected. VBMP は DHTML や WebClass コンポーネントを変換できません。しかし、UserDocument や PropertyPages は VB.NET の UserControls に変換しますので、想定通りに動作させるための手作業によるコーディングが必要となったとしても、変換後にそれらを使って作業を行うことができます。
No control arrays in VB.NET VB.NET にはコントロール配列がありません。	✔	VBMP correctly converts all sorts of VB6 control arrays, including arrays of menus and 3rd-party ActiveX controls. VBMP は、メニューやサードパーティ製 ActiveX コントロールの配列を含む、すべての VB6 のコントロール配列を正しく変換することができます。
Old VB project not upgradable to VB.NET 古い VB プロジェクトは VB.NET にアップグレードできません。		VBMP has the same limitation as UW and requires that you migrate VB3, VB4, and VB5 projects to VB6 before attempting the migration to VB.NET. VBMP は UW と同様の制約を持っていますので、ユーザーは VB.NET へ移行する前に VB3/5/6 から VB6 に移行する必要があります。
OLE Automation unavailable in VB.NET OLE オートメーションは VB.NET ではサポートされていません。	✔ P	VBMP converts OLE Automation features into do-nothing members that are marked as obsolete. Calling these members has no effect or throws an exception, but at least you can start testing other portions of the application without having to fix one or more compilation errors. VBMP は OLE オートメーションを旧式と見なし、何もしないメンバに変換します。これらのメンバを呼び出しても、何も起らず、エラーも発生しませんが、少なくともユーザーはコンパイルエラーを取り除く作業をすることなくアプリケーションの他の箇所のテストを開始することはできます。
ParamArray is ByVal in VB.NET ParamArray は VB.NET では ByVal です。	✔	VBMP can automatically generate code that ensures that ParamArray use by-reference semantics. VBMP は ParamArray が by-reference の意味を持つことを保証するコードを自動的に生成します。
Parameterless default properties unsupported in VB.NET VB.NET ではパラメータを持たないデフォルトプロパティをサポートしていません。	✔	VBMP behaves like UW when the object variable is typed; when the variable uses late binding, VBMP can generate code that determines the default member at runtime. オブジェクト変数が定義されている場合、VBMP は UW のように変換します。すなわち、変数が遅延バインディングを使用している場合、VBMP は実行時にデフォルトメンバを定義するコードを生成することができます。
Property mixes scopes プロパティが複数のスコープを持っています。	✔ *	VBMP correctly converts property procedures with mixed scope. VBMP はスコープが混在するプロパティのプロシージャを正しく変換します。
Property passed ByRef ByRef が渡されるプロパティ	✔	VBMP can convert ByRef parameters into ByVal parameters if the parameter isn't assigned inside the method. VBMP は、パラメータがメソッド内部で割り当てられない場合、ByRef パラメータを ByVal パラメータに変換することができます。
Resource file requires work in VB.NET VB.NET ではリソースファイルは追加作業を必要とします。	✔	VBMP converts both the resource file and all LoadRes* methods; it even converts them to My.Resources members if possible. VBMP はリソースファイルと LoadRes*メソッドの両方を変換します。すなわち、可能であれば、両方とも My.Resources のメンバに変換します。
ScaleMode must be vbTwips for VB.NET VB.NET では ScaleMode は vbTwips	✔	VBMP supports all ScaleMode settings, including 0-vbUser. VBMP は、0-vbUser を含むすべての ScaleMode をサポートします。

でなければなりません。

Setting .Interval does not enable/disable timer in VB.NET VB.NET では、.Interval によってタイマーを enable/disable にすることはできません。	Projects converted by VBMP don't suffer from this issue. VBMP によって変換されたプロジェクトはこの影響を受けません。
String byte functions unavailable in VB.NET VB.NET ではバイト指向関数は使用できません。	VBMP partially support byte-oriented string functions, such as LenB or InStrB; it also support implicit conversion between strings and byte arrays, and conversions between ANSI and Unicode strings. VBMP は一部のバイト指向関数に対応しています。例えば、LenB や InStrB です。すなわち、文字とバイトの配列や ANSI と Unicode の文字列の変換は暗黙的に行われます。
Sub Main not executed in VB.NET VB.NET では Sub Main は実行されません。	VBMP fixes this issue: a VB.NET class library project that is the result of converting a VB6 ActiveX DLL project correctly executes the Sub Main method before any class in the library is instantiated. VBMP はこの問題を解決します。VB6 の ActiveX DLL プロジェクトを正しく変換した結果の VB.NET クラスライブラリプロジェクトは、ライブラリのクラスがインスタンス化する前に Sub Main メソッドを実行します。
Sub Main	.NET program exits at End Sub: VBMP can handle this issue by adding a proper pragma. .NET のプログラムは End Sub で処理を終了します。すなわち、VBMP は、この問題を適切なプラグマを追加することで解決することができます。
TTF/OTF fonts required by VB.NET VB.NET では TTF(True Type Font)/OTF(Open Type Font)が必要とされます。	VBMP allows you to determine how fonts are converted during the migration to VB.NET. VBMP を使えば、VB.NET に変換する際にどのフォントを変換するかを指定することができます。
Type unsupported by VB.NET VB.NET は Type をサポートしていません。	VBMP comes with a VB6FixedString type that perfectly mimics the VB6 fixed-length string type; also, VBMP can convert arrays of fixed-length strings; additionally, fixed-length strings in UDTs can be converted into standard strings and still retain the fixed-length behavior VBMP は VB6 の固定長文字列型を完璧に模倣した VB6FixedString 型を持っています。また、VBMP は固定長文字列の配列も変換することができます。さらに、ユーザー定義型内の固定長文字列についても、固定長文字列としての動作を維持したまま、標準の文字列に変換することができます。
Unavailable in VB.NET VB.NET では有効ではありません。	VBMP supports CVErr, GoSub, Return, vbDataObject, vbUnicode, vbFromUnicode, IsEmpty, and a limited form of LSet that works with UDTs. (It doesn't support VarPtr, ObjPtr, and StrPtr undocumented functions, though.) VBMP は CVErr、GoSub、Return、vbDataObject、vbUnicode、vbFromUnicode、IsEmpty とユーザー定義型で動作する LSet の限定的なフォームをサポートします。
Underscore _names not hidden in VB. アンダースコアで始まる名前は非表示になりません。	VBMP doesn't deal names with a leading underscore in a special way, however it recognizes VB6's hidden members and convert them correctly to VB.NET. VBMP はアンダースコアで始まる名前を特別な方法で扱うことはありませんが、VB6 で非表示のメンバであったことは見分けがつきますし、VB.NET に正しく変換されます。
VB5 project may not upgrade to VB.NET VB5 のプロジェクトは VB.NET にアップグレードすることはできません。	VBMP converts only VB6 projects, therefore VB5 projects must be converted to VB6 first. However, it supports the Common Windows controls released with VB5. VBMP は VB6 のプロジェクトのみ変換できます。そのため、VB5 のプロジェクトはまず、VB6 に変換しなければなりません。しかし、VB5 でリリースされた Common Windows controls はサポートしています。

WebClasses upgrade to ASP.NET WebClasses は ASP.NET にアップグレードします。		VBMP doesn't convert WebClass projects. We strongly believe that such projects should be upgraded to ASP.NET in all cases. VBMP は WebClass プロジェクトを変換しません。弊社はこのようなプロジェクトについてはどのような場合でも ASP.NET にアップグレードすることを強く推奨しています。
Function without type specification 型が定義されていないファンクション	✔	VBMP emits a warning if a function or property has no As clause; you can use the SetType pragma to define the type returned by the VB.NET function without altering the VB6 code. VBMP はファンクションやプロパティが AS を持っていない場合は警告メッセージを出力します。ユーザーは VB.NET ファンクションによって戻ってくる型を定義するために、VB6 のコードを修正することなく、SetType プラグマを使用することができます。
Variable without type specification 型が定義されていない変数	✔	VBMP emits a warning if a variable or parameter has no As clause; however, you can use the SetType pragma to define the type of the VB.NET variable without altering the VB6 code. VBMP は変数やパラメータが AS を持っていない場合は警告メッセージを出力します。ユーザーは VB.NET の変数の型を定義するために、VB6 のコードを修正することなく、SetType プラグマを使用することができます。
ByVal/ByRef missing ByVal/ByRef がありません。	✔	By default, VBMP doesn't take any action if an explicit ByRef/ByVal keyword is missing and converts these parameters as ByRef parameters; however, it emits a warning if a ByRef parameter can be safely converted as a ByVal parameter and you can use the UseByVal pragma to automatically convert such parameters into ByVal parameters. デフォルトでは、ByRef/ByVal が明示的にない場合、VBMP は特別な処理は行わず、これらのパラメータを ByRef パラメータに変換します。しかし、ByRef パラメータを安全に ByVal パラメータとして変換できる場合には警告メッセージを出力するので、ユーザーはそのようなパラメータを ByVal パラメータに自動変換する UseByVal プラグマを使用することができます。
Option Explicit missing Option Explicit がありません。	✔	VBMP doesn't take any specific action if Option Explicit is missing; however, you can use a pragma to automatically create VB.NET variables for all VB6 variables that weren't explicitly declared. VBMP は Option Explicit がない場合、特別な処理を実行しません。しかし、ユーザーは明示的に宣言されていない VB6 変数に対する VB.NET 変数を自動的に作成するプラグマを使用することができます。
Optional parameter missing default value 任意パラメータのデフォルト値がありません。	✔	VBMP automatically add the property default value for Optional parameters if necessary. VBMP は、必要に応じて、プロパティに任意パラメータのデフォルト値を自動的に付加します。
Variable/Parameter with generic type ジェネリック型の変数/パラメータ	✔	VBMP provides statistics about Variant variables and allows you to use the SetType pragma to change the type of a variable or parameter during the conversion to VB.NET, without affecting the existing VB6 code. VBMP はバリアント変数についての測定値を提供してくれるだけでなく、ユーザーは、既存の VB6 のコードに影響を与えることなく、VB.NET へ変化する際に変数やパラメータの型を変更する SetType プラグマを使用することができます。

To recap, **VB Migration Partner can fully or partially handle 44 of the 49 compatibility issues that are left unresolved by the Upgrade Wizard**. The remaining 5 unresolved issues are related to features that just don't make sense under .NET – such as OLE-related properties, the IDE extensibility object model, and WebClasses components.

要点をまとめると、**VB Migration Partner は、Upgrade Wizard によって解決できなかった互換性の問題の 49 個中 44 個まで完全または部分的に解決することができます**。残る 5 つの問題は.NET では意味のない特徴と関連しています。例えば、OLE に関連するプロパティや IDE 拡張オブジェクトモデルや WebClasses コンポーネントです。

Even more important, **VB Migration Partner fixes many more compatibility issues than those listed in this page**. As a matter of fact, VBMP solves many problems that even VB Project Analyzer fails to detect. (Readhere and here for a more exhaustive list of migration problems.)

より重要な点は、**VB Migration Partner がこのページに記載されていない多くの互換性の問題を解決するということです**。実際に、VBMP は VB Project Analyzer が検出できない多くの問題を解決します。(マイグレーション上の課題の包括的なリストについてはこちらとこちらを参照してください。)