VB Decompiler

用户手册

简体中文

Visual Basic 5.0/6.0 · .NET

DotFix Software

目录

1 说明 4

2 快速入门 5

3 主窗口 6

4 查看代码 8

5 AI 代码增强 10

6 打开与保存 14

6.1 打开文件 14

6.2 使用分析数据库 15

6.3 保存过程列表 16

7 程序功能 17

7.1 跳转到地址 17

7.2 字符串搜索 18

7.3 字符串引用 19

7.4 从指定地址反编译 21

7.5 修改数据 22

7.6 混淆 23

8 程序选项 25

8.1 程序选项 25

8.2 窗体 26

8.3 反编译 27

8.4 反汇编器 29

8.5 配色方案 30

8.6 高级功能 32

8.7 语言选择 36

9 插件 37

10 可视化窗体重建 41

11 跟踪 44

12 分析功能 46

13 混淆限制 48

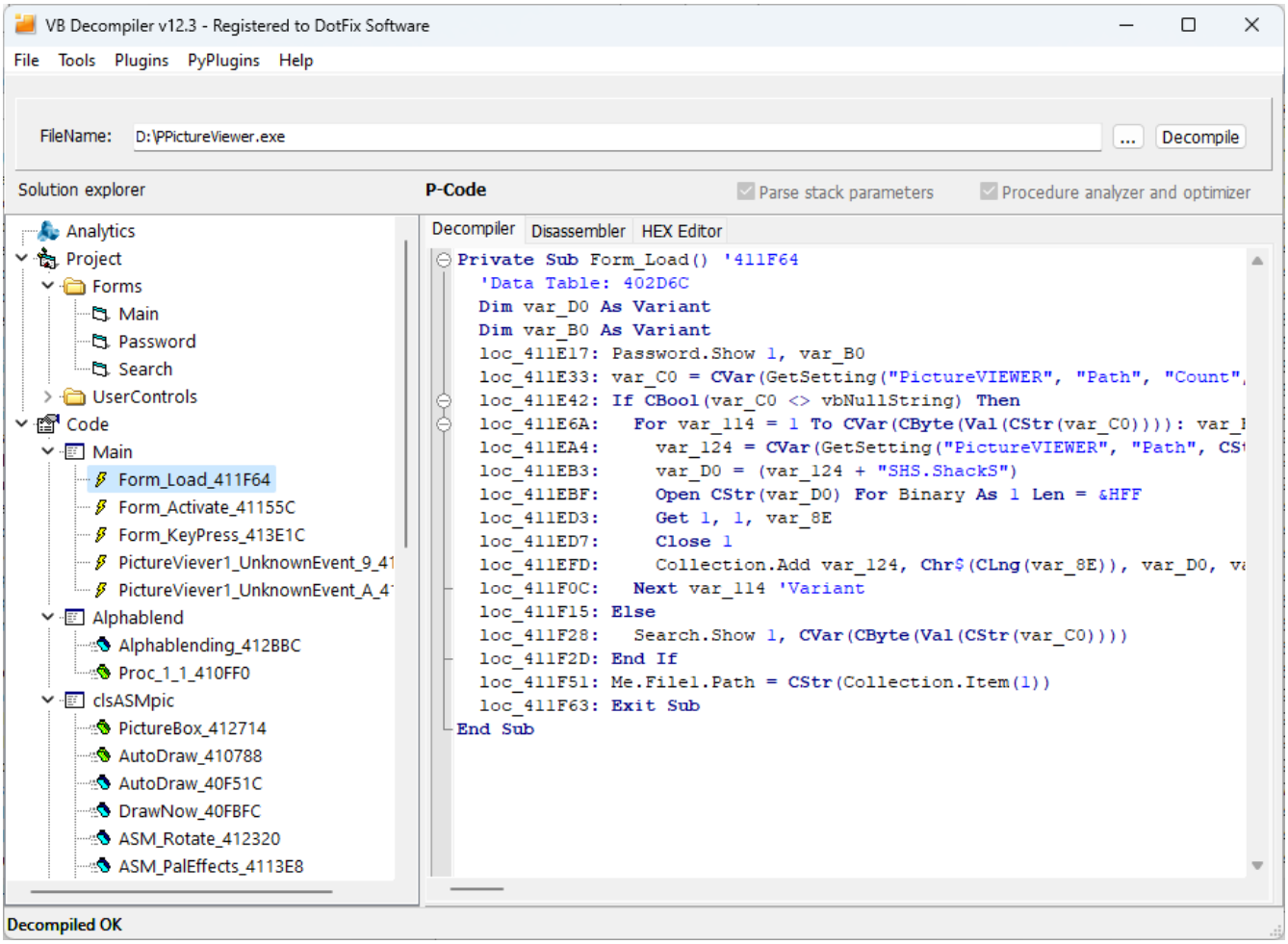
14 技术支持 50

15 注册 50

16 关于开发者 51

1 说明

VB Decompiler 是一款针对 Visual Basic 5.0/6.0 程序和 .NET 程序集的逆向工程与源码恢复工具。它结合了高级反编译、底层反汇编、元数据检查、集成的十六进制编辑器以及可视化界面重构功能。典型用途包括遗留软件维护、安全事件响应、恶意软件分析以及对未文档化的应用程序行为进行调查。



支持的代码表示形式

- **VB5/6 P-Code:** 从编译器的 P-Code 指令和运行时操作中重构 Visual Basic 语句。
- **VB5/6 Native Code:** 反汇编 x86 指令，并使用内部模拟器、运行时特征码和启发式方法恢复类似 Visual Basic 的代码。
- **.NET:** 反汇编通用中间语言（CIL，也称为 MS IL），并将支持的 IL 模式反编译为类似 C# 的代码，无论原始 .NET 源代码语言为何。此外，.NET 程序集中的原生 x86 和 x64 代码也可以进行反汇编。

.NET 分析引擎在进行代码分析或 可视化界面重构时不需要 .NET 运行时或反射功能。

分析工具

使用 过程和文本搜索、交叉引用和变量重命名以及 字符串引用 来调查行为。可视化查看器重构了经典 VB 窗体、.NET Windows Forms 以及 WPF 窗口、页面和用户控件，这些控件存储在编译的 BAML 资源中。

具有 跟踪功能 的许可证提供对 VB5/6 原生代码和 P-Code 的逐步模拟。具有 分析功能 的许可证为 VB5/6 和 .NET 应用程序提供分类报告。AI 助手 可以使用 Ollama 或 OpenAI API 集成来处理恢复的 VB5/6 或 C# 代码。

解释恢复的代码

编译会移除诸如注释、许多局部变量名称以及某些类型和过程元数据等信息。优化和混淆可能会进一步改变代码结构。重构的代码清单支持代码分析和源码恢复。请结合反汇编代码、元数据、字符串引用以及相关过程的代码验证重要结论。

2 快速入门

1. 选择**文件 > 打开程序 (File > Open program)**并打开要分析的 EXE、DLL 或 OCX 文件。处理将自动开始。等待项目树填充完成且处理结束。
2. 在**方案浏览器 (Solution explorer)**中打开一个过程。比较**反编译器 (Decompiler)**和**反汇编 (Disassembler)**视图；使用**十六进制编辑器 (HEX Editor)**检查相应的字节。
3. 按**Ctrl+T**通过其名称或部分名称查找过程。使用**Ctrl+F**搜索显示的文本，或使用**工具 > 字符串的引用 (Tools > String references)**调查字符串及其使用的过程。
4. 打开恢复的窗体并选择**设计器（只读） (Designer (read only))**以进行可视化预览。对于经典 VB、Windows Forms 和 WPF，使用恢复的事件导航到相应的事件处理程序代码。
5. 通过**文件 > 数据库 (File > Data Base)**保存分析结果，以便稍后继续或与其他分析师共享结果。

选择反编译模式

快速反编译在您打开过程时处理它们。启用**分析原型 (Analyze prototypes)**以对私有过程签名进行额外的初步分析。若要搜索所有反编译的文本或导出完整项目，请禁用快速反编译并再次处理文件。

保存恢复的源代码

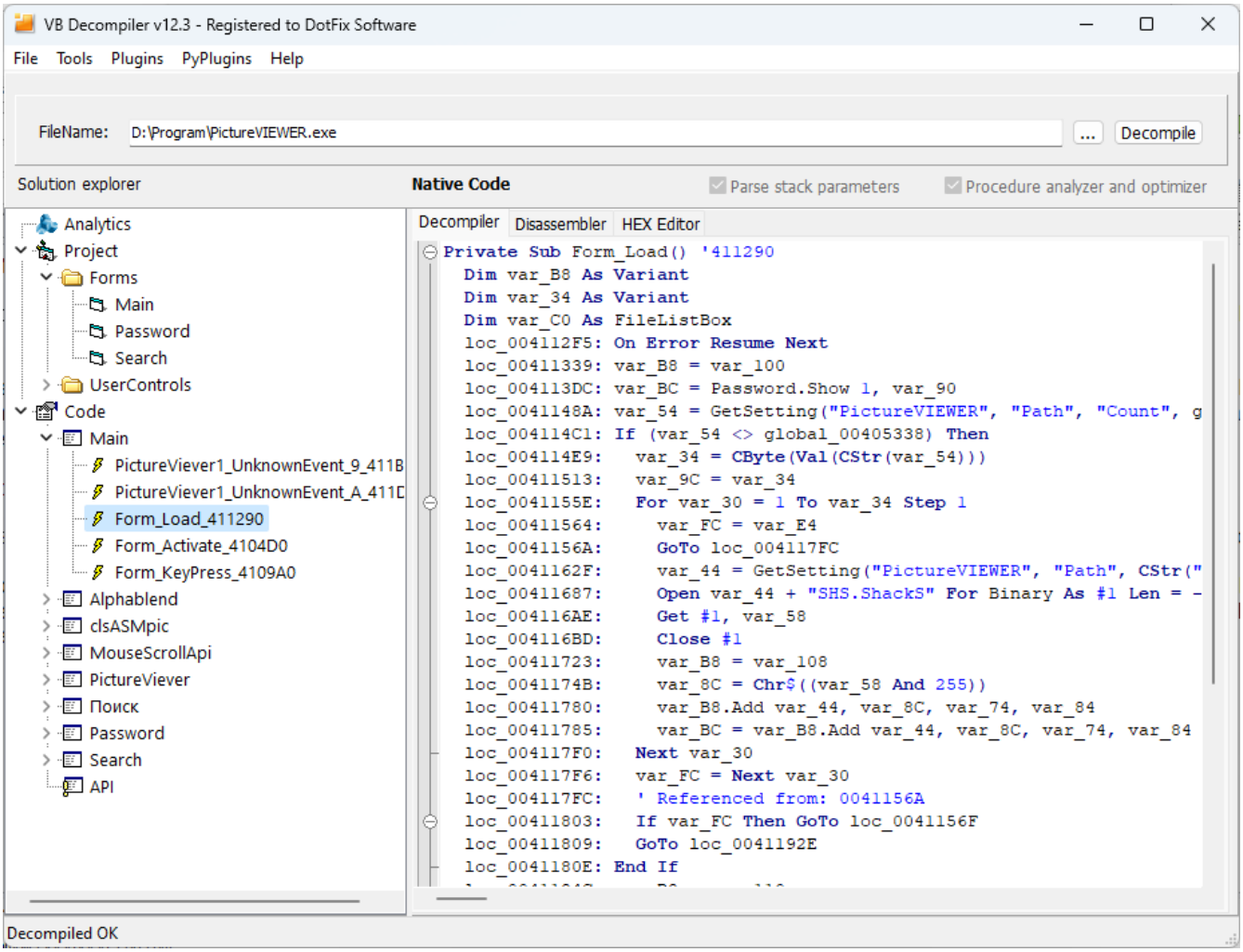
要导出 VB5/6 反编译结果，请选择**文件 > 保存反编译的项目 (File > Save decompiled project)**并选择输出文件夹。导出内容包括恢复的窗体和模块；FRX 文件包含关联的图像和其他二进制窗体数据。

在 IDE 中打开导出的代码之前，必须先手动修正恢复的源代码中所有反编译不准确之处和不兼容问题。

有关详细工作流程，请参阅可视化窗体重建、代码查看和AI 代码增强。

3 主窗口

方案浏览器 (Solution explorer) 将恢复的项目组织为窗体、模块、类和过程。相邻的视图显示所选的代码、窗体或元数据。**Visual Studio 窗口布局 (Window style as in Visual Studio)** 选项将树状结构置于右侧，代码置于左侧。



代码和数据视图

- **反编译器 (Decompiler):** 高级重构代码。
- **反汇编 (Disassembler):** 相应的 x86/x64 指令、VB P-Code 或 .NET IL，并带有地址。
- **十六进制编辑器 (HEX Editor):** 底层文件字节，可通过虚拟地址或文件偏移进行导航。
- **设计器（只读） (Designer (read only)):** 重构的窗体及其恢复的属性（在支持的情况下）。
- **AI 助手 (AI Helper):** 当前反编译过程的可选 AI 处理。

可用的选项卡取决于所选对象。代码类型指示器区分 **P-Code**、**Native Code** 和 **.NET Assembly**。

解析堆栈参数 (Parse stack parameters)

对于 P-Code，此控件将堆栈操作和调用组合为带有参数的表达式。关闭后会显示中间操作，并在未知原型消耗意外堆栈值时提供帮助。对于 Native Code，启用它使用模拟器重构高级表达式；禁用后会显示汇编级输出。对于 .NET，堆栈处理有助于将 IL 重构为类似 C# 的表达式。

程序分析器和优化器 (Procedure analyzer and optimizer)

此控件通过组合表达式和识别的控制流模式，并移除识别出的编译器辅助代码来简化重构列表。它适用于 VB5/6 和 .NET 代码。.NET 优化器可以重构支持的 `while`、`do...while` 和 `foreach` 模式、嵌套条件、运算符表达式和插值字符串处理程序模式。这些转换由反编译器本身执行，不需要 AI 助手。

如果表达式或分支不清楚，请比较禁用优化后的输出并检查反汇编。在 **快速反编译 (Fast decompilation)** 模式下（无缓存），您可以在检查过程时更改这些控件。对于完整反编译或缓存输出，在更改设置后再次处理文件。

导航和键盘访问

树上方的箭头按钮用于导航之前查看的过程；**Esc** 返回到上一个过程。鼠标“后退”和“前进”按钮使用相同的历史记录，包括过程中记住的位置。

按 **Alt+1** 可在Solution explorer和活动视图之间切换焦点。在树中按 **Enter** 可打开所选对象。在十六进制编辑器中，**Tab** 在十六进制和文本字段之间移动，然后移动到下一个控件。界面标签与其输入字段关联，以支持屏幕阅读器导航。

有关上下文相关的快捷键、交叉引用和重命名的信息，请参阅 查看代码，有关分析视图的信息，请参阅 分析功能。

4 查看代码

代码视图结合了语法高亮、可折叠的代码块、基于地址的导航以及交叉引用。使用高级代码清单来理解过程，并使用反汇编和文件字节来检查该结果是如何获得的。

标识符的高亮与导航

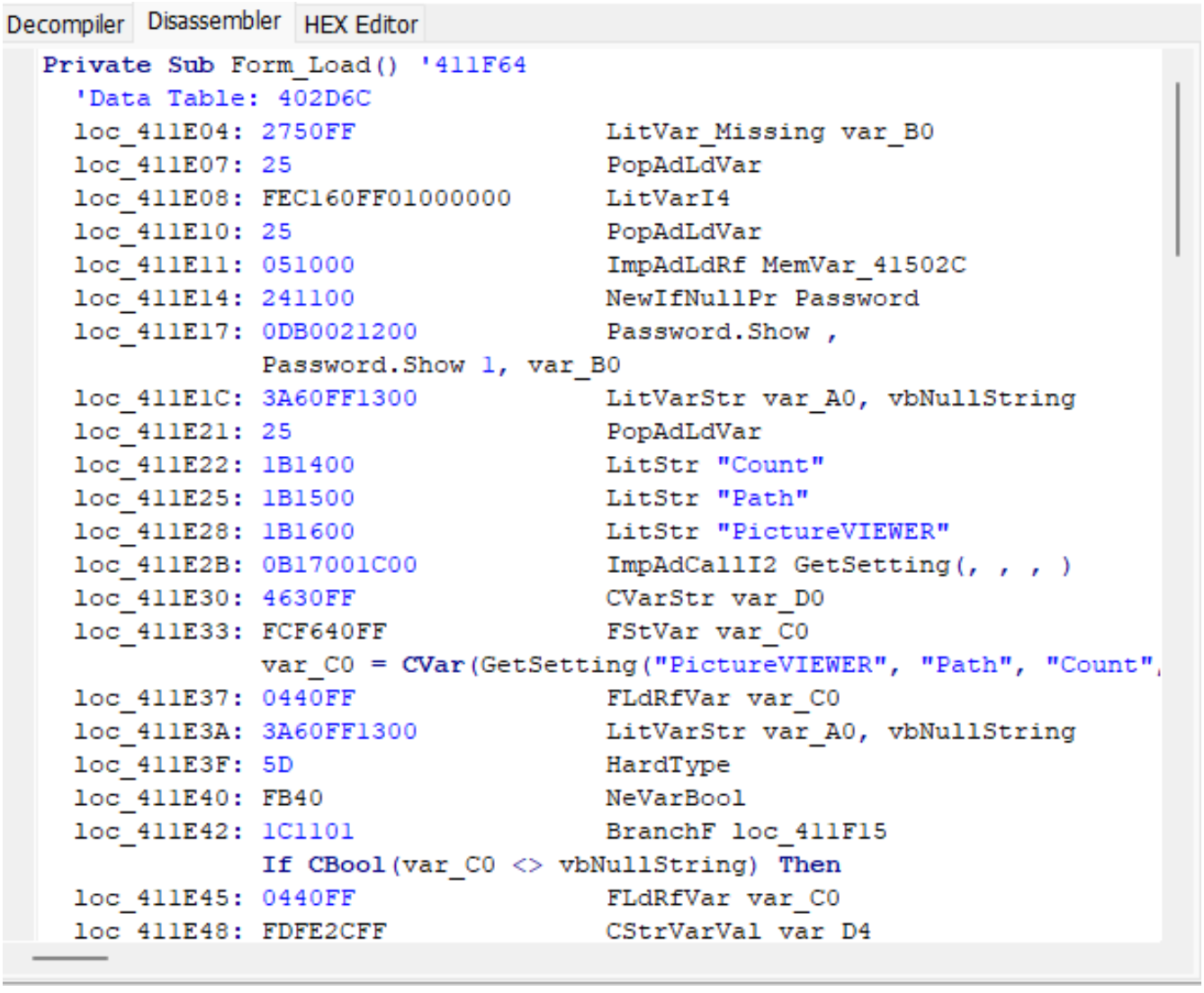
将光标置于寄存器、变量或函数名称上以高亮显示当前列表中匹配的实例。当用作成员表达式的基础时，例如 `str_1.Value`，局部变量（如 `var_XX` 和 `str_XX`）也会被高亮显示。选择 `global_XX` 标识符会高亮显示其当前过程中的实例；此高亮与项目范围的引用搜索是分开的。

双击识别的过程调用以打开被调用的过程。地址常量可以引导至 HEX 编辑器中的相应字节。使用 **Esc**、导航箭头或鼠标的前进/后退按钮来浏览查看历史。

合并反编译和反汇编代码

右键单击反编译代码清单并选择 **复制到反汇编器 (Copy to disassembler)**，或在代码视图中按 **D**。这将当前过程的重建语句与底层代码清单合并。为了保留您的位置，在调用命令之前将光标置于相关的 `loc_` 地址上。

使用此工作流程时，请保持 **添加地址 (Add Addresses)** 开启（位于反编译设置）。反汇编器始终显示地址；添加地址 控制 反编译器 (Decompiler) 视图中的地址显示。



重命名变量

对于 VB5/6 P-Code 和本机代码，右键单击变量并选择 **变量改名 (Rename variable)**，或将光标置于变量上并按 **N**。输入有意义的分析名称。反编译器检查名称冲突和保留关键字。要撤销重命名，请使用 **撤销重命名 (Undo renaming)** 或 **Ctrl+Shift+Z**。此命令独立于普通文本编辑的撤销。

重命名使恢复的列表更易于分析；它不会恢复因编译而被移除的名称。在与他人交流发现时，将分析师分配的名称视为注释。

查找调用者和全局变量引用

要查找调用者，在Solution explorer中右键单击过程并选择 **查找引用 (Find references)**，或在该过程的树视图中选中时按 **X**。结果列出了包含对其识别调用的过程。双击结果以检查调用者。此工作流支持 VB5/6 和 .NET；.NET 支持是在版本 12.4 中添加的。

要调查全局变量，在代码视图中将光标放在该变量的标识符上并按 **X**。因此，相同的键用于在树中查找过程引用以及在代码中查找全局变量引用。已识别的引用列表并不能证明不存在其他间接、动态解析或混淆的使用。

上下文相关快捷键

焦点	快捷键	操作
代码	N	重命名光标下的变量（VB5/6）。
代码	X	查找光标下全局变量的引用。
代码	D	将当前反编译的代码复制到反汇编器。
代码	F4	跟踪到当前行的地址；需要跟踪功能。
代码	Ctrl+Shift+Z	撤销光标下变量的重命名。
树	X	查找所选过程的引用。
树	G	打开跳转到节点的对话框。
树	Ctrl+C	复制所选对象的完整树路径。
树	Space or Insert	标记或取消标记所选项目。

单字母快捷键取决于当前聚焦的视图，并且不会替换输入字段中的正常键入。使用 **Alt+1** 来切换焦点。有关 Ctrl+T 和 Ctrl+F，请参阅 字符串搜索；有关树路径和地址，请参阅 跳转到地址；有关模拟器限制，请参阅 跟踪。

5 AI 代码增强

AI 助手 (AI Helper) 处理 VB Decompiler 已重构的代码。它支持从 .NET 程序集反编译的 C# 以及来自 P-Code 或 Native Code 的 Visual Basic 5.0/6.0 代码。您可以通过 Ollama 使用本地模型，或通过 OpenAI API 集成连接到服务。

AI 可以建议更清晰的控制流、变量名、声明和表达式。它无法可靠地恢复二进制文件中不存在的信息，或修复反编译器错误解释的片段。在将建议用于源代码恢复或调查报告之前，请将其与原始反编译代码清单、反汇编代码及相关过程的代码进行比较。

处理过程

1. 在Solution explorer中打开所需的过程，并检查其 **反编译器 (Decompiler)** 代码清单。
2. 切换到 **AI 助手** 并按照下文描述配置提供者、主机和模型。
3. 使用模型选择器旁边的操作按钮开始处理。
4. 在 AI 助手 中审查返回的代码，并将其与反编译器的输出进行比较。

AI 处理在单独的线程中运行。如果模型响应时间过长，请使用 **取消 (Cancel)**。可以折叠其他连接设置以腾出更多空间用于显示代码。

通过 Ollama 使用本地模型

1. 从其 官方下载页面 安装 Ollama 并启动它。
2. 在终端中运行 `ollama pull qwen3:8b` 下载推荐的模型。
3. 在 AI 助手 中选择本地 LLM/Ollama 提供者。将 **Ollama host** 设置为服务器地址；默认本地地址为 `http://localhost:11434`。
4. 在 **模型 (Model)** 中选择已安装的模型并开始处理。

我们推荐使用 `qwen3:8b`。AI 助手 的内置提示词针对这一轻量级模型进行了适配。较新的模型可能处理代码更慢，却无法改善结果。

模型选择不受限制：可以使用所选 Ollama 实例中安装的任何模型。对于不熟悉的响应格式，响应解析器可能无法滤除模型特有的标签或其他辅助文本，导致其保留在显示的结果中。

Ollama 本地推理的硬件配置

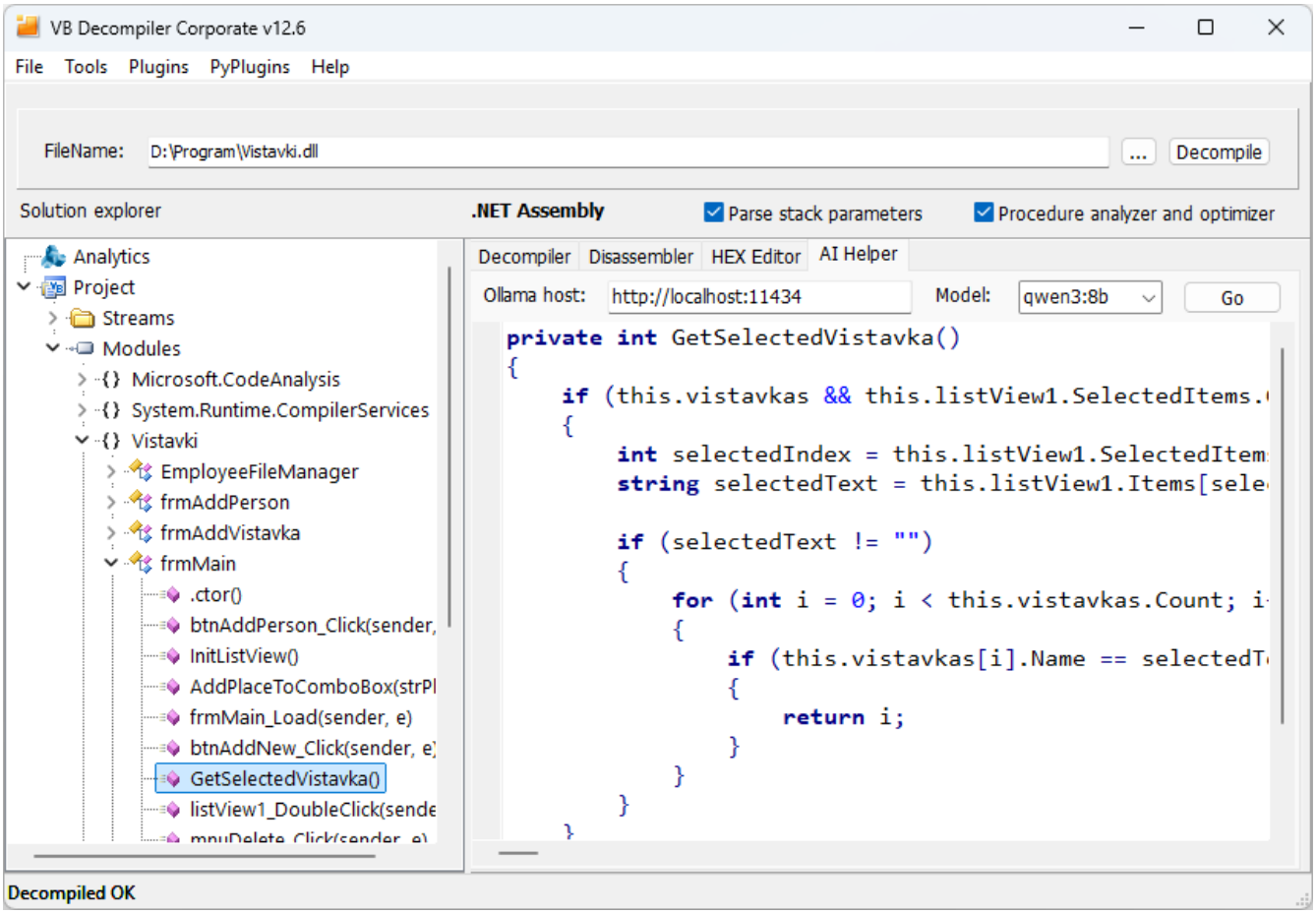
本地 AI 处理在运行 Ollama 的计算机上进行，所需资源明显高于反编译。下表给出实用配置建议，适用于采用 `Q4_K_M` 权重量化和标准 `f16` 上下文缓存的 `qwen3:8b`，且每次只处理一个请求。上下文以 token 计量：4K 为 4096 个 token，128K 为 131072 个 token。此容量包含模型指令、源代码和生成的回答。

组件	实用最低配置：4K 上下文	建议配置：128K 上下文
操作系统	64 位 Windows 10（22H2）或 Windows 11	64 位 Windows 10（22H2）或 Windows 11
处理器	支持 AVX2 的现代 x64 CPU，至少 4 个物理核心	支持 AVX2 的现代 x64 CPU，至少 8 个物理核心
系统内存（RAM）	16 GB	64 GB
显卡	兼容 Ollama 的 GPU，至少 8 GB 独立显存	兼容 Ollama 的 GPU，至少 32 GB 独立显存
存储	SSD，至少预留 20 GB 空间用于 Ollama 和此模型	SSD，至少预留 20 GB 空间用于 Ollama 和此模型

通过 Ollama 设置中的滑块选择上下文大小。适用于 4K 的 8 GB 显卡不足以满足上述 128K 配置。模型下载大小约为 5.2 GB，但总内存需求更高。在 128K 下，仅 `f16` 上下文缓存就需要约 18 GiB，此外还需要模型权重和工作缓冲区所占的内存。请为 Windows 和其他应用程序保留内存。

在 RAM 充足时，Ollama 可以使用 CPU 计算，但代码处理可能非常缓慢。Pentium M 及同年代的老旧计算机不适合在本地运行 `qwen3:8b`。旧款 Intel HD Graphics 使用的共享内存不能替代上表要求的独立显存。这些要求针对运行 Ollama 的计算机；远程服务器或外部 API 服务使用其自身的硬件。

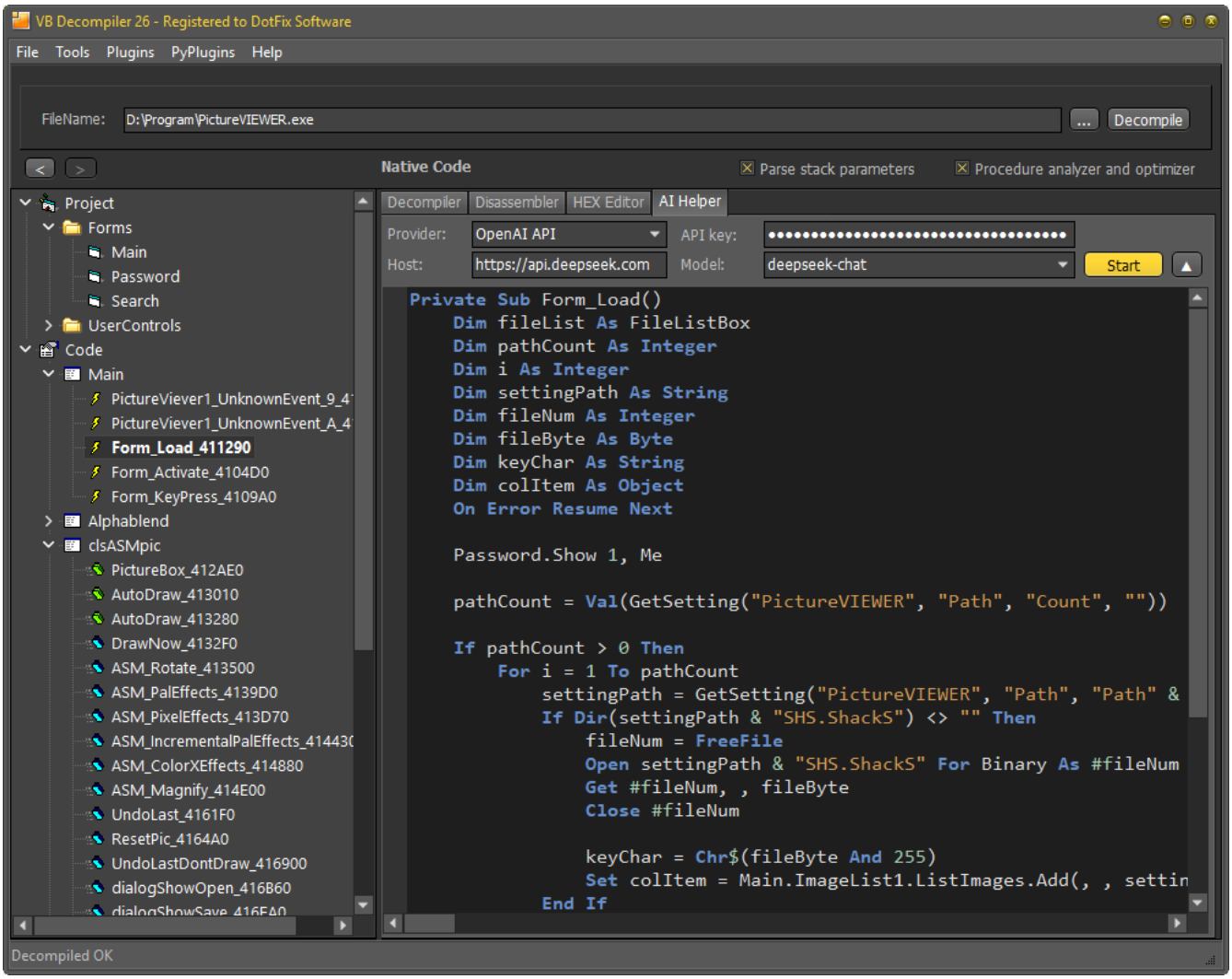
处理较短的过程时，请从 4K 开始。增大上下文会增加内存占用以及长输入的处理时间。处理过慢时，请取消操作、减小上下文，并使用 `ollama ps` 检查 GPU 使用情况。实际内存占用取决于 Ollama 版本、计算后端及其他正在运行的任务。



使用 OpenAI API 集成

将 **服务商 (Provider)** 设置为 **OpenAI API**。配置 **地址 (Host)**、**API密钥 (API key)** 和 **模型** 以适用于目标服务。输入有效的 API 密钥以及该密钥可用的模型标识符。请勿在屏幕截图或共享的分析材料中泄露 API 密钥。

AI 助手 还支持通过 **OpenAI API** 提供者使用 DeepSeek。在相应字段中输入 DeepSeek 主机、API 密钥和模型标识符。下图显示了此配置。



代码处理位置

AI 助手 将当前反编译的代码发送到配置的服务器。对于仅本地工作流，请使用本地运行的 Ollama 服务器和本地执行的模型。位于另一台机器上的 Ollama 主机、Ollama 云模型或外部 API 服务可将代码传输到分析工作站之外。

对于机密源代码恢复或安全调查，请根据您的组织数据处理要求选择端点。不要仅根据存在 Ollama 或 localhost URL 就推断离线操作。

故障排除

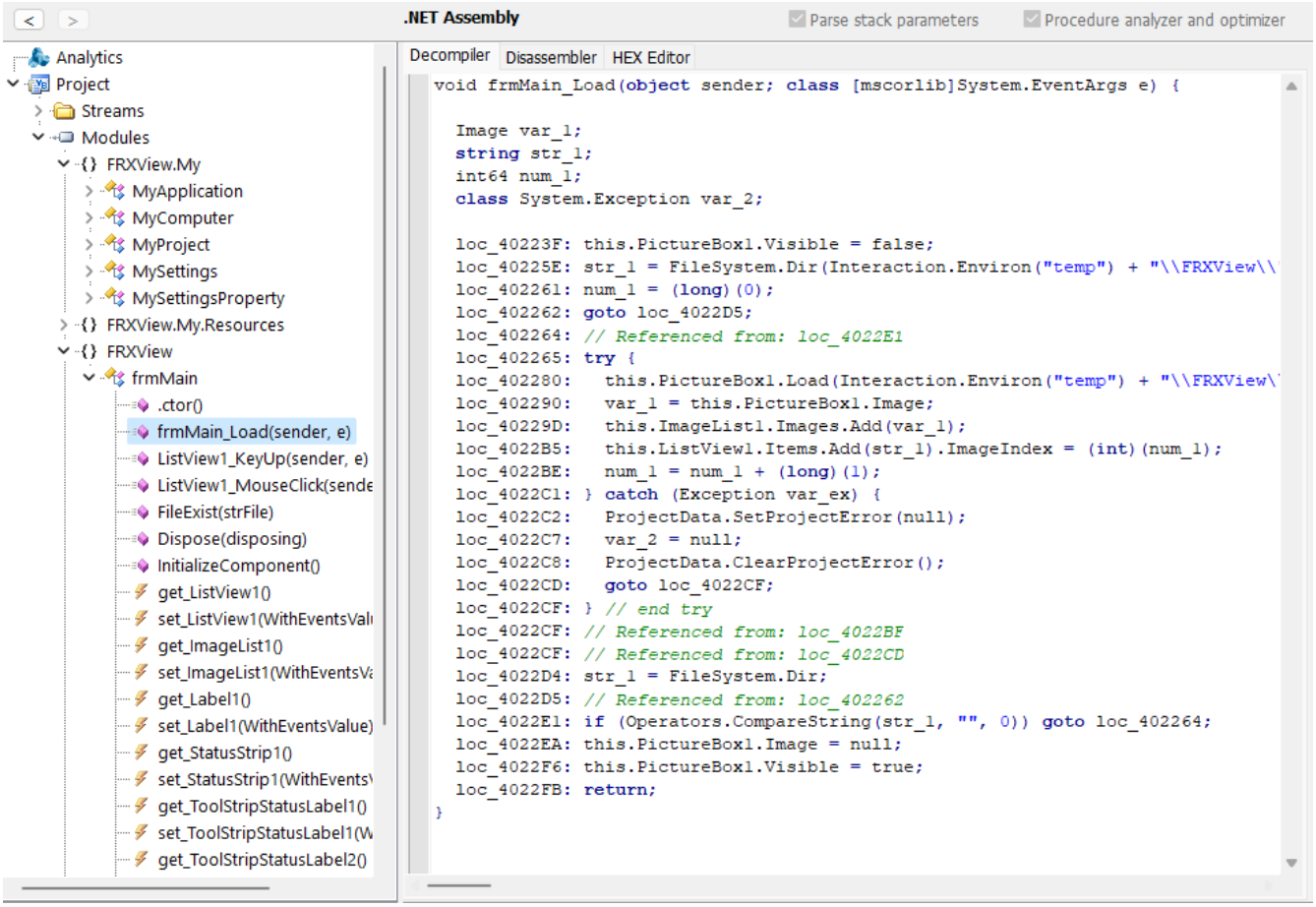
- 如果缺少本地模型，请确认它已安装在主机字段指定的 Ollama 实例中。
- 如果没有响应，请检查所选服务是否正在运行且可访问，并且端点、凭据和模型标识符对应于同一提供者。
- 如果处理速度过慢，请取消操作。当使用其他本地模型时，尝试推荐的 `qwen3:8b` 模型。
- 如果不熟悉的模型在其响应中留下标签或其他辅助文本，请尝试 `qwen3:8b` 或在审查结果时手动删除该文本。
- 如果结果意外更改了条件、调用、常量或类型，请对照底层代码清单进行验证。代码更易读并不能证明其行为与原代码等效。

Ollama 和外部模型提供者是第三方产品。它们的可用性、模型行为和服务策略独立于 VB Decompiler。

6 打开与保存

6.1 打开文件

在开始反编译过程之前，您需要选择一个用于分析的文件。为此，**文件 (File)** 菜单中包含一个 **打开程序 (Open program)** 部分。**VB Decompiler** 支持打开由 **Visual Basic** 版本 5.0 或 6.0 创建的 EXE/OCX/DLL 文件，以及使用 **Visual Studio .NET** 编译的应用程序。在后一种情况下，源代码语言无关紧要，因为 Visual Studio .NET 将应用程序无论其原始语言如何都编译为单一的内部格式 - **MS IL**。



打开文件后，反编译器会检查该程序是否属于指定的开发环境。如果打开的程序是在不支持的环境中开发的（例如 **Delphi** 或 **Visual C++**），则会显示一条消息，说明不支持该环境且无法反编译该程序。如果出现此消息，您可以联系我们，我们可能能够协助反编译您的程序。我们在 **Delphi** 和 **FreePascal** 应用程序的反编译领域拥有内部开发成果，并可为此类文件提供反编译服务。

如果打开的程序使用 **EXE 加壳工具**、混淆器或保护机制进行了打包，反编译器将提示尝试解包该文件。此功能仅在 32 位的 Windows 版本中可用，且不提供任何保证。如果解包成功，文件将在内存中解包，以不同名称保存到磁盘，并打开进行反编译。**注意！** 切勿尝试使用内置解包器解包恶意程序或任何其他可能危害您系统的代码，因为在解包过程中，正在打开的程序将被执行，其在您的 PC 上执行的后果可能极其严重。

重要说明： 在任何情况下都不应尝试在未取得必要授权的情况下反编译或解包软件，如果版权所有人通过许可证禁止对该文件进行解压和分析。这可能违反您所在国家的法律。

*Visual Studio .NET、Visual C++ 和 Visual Basic 是 Microsoft Corporation 的注册商标。Delphi 是 Embarcadero 的注册商标。

其他文件打开方式

您可以将文件从资源管理器拖放到 VB Decompiler。具有未知扩展名或无扩展名的文件也可以通过这种方式提交；识别仍取决于文件的实际内容。支持的 Windows **.lnk** 快捷方式可解析为其引用的文件。保存的 VDD 数据库 也可以通过拖放或命令行打开。

6.2 使用分析数据库

此功能可通过**数据库 (Data Base)**部分在**文件 (File)**菜单中启用，用于将反编译结果保存到单独的压缩文件中。这对于未来更快地加载反编译结果非常有用，而无需重新反编译代码。

如果您需要返回分析程序或与同事共享这些结果，则无需再次执行反编译。在完整反编译（禁用**快速反编译 (Fast decompilation)** 选项）的情况下，此过程可能需要更长时间并消耗更多资源。预先将反编译代码保存到**Database**要方便得多。

VB Decompiler使用内部压缩数据库格式，可显著减少保存分析结果所需的空间。此外，它与大多数先前版本的 VB Decompiler 保存的数据库兼容，允许您存储信息而无需担心无法在新版本中打开此类数据库的风险。

打开和转移 VDD 数据库

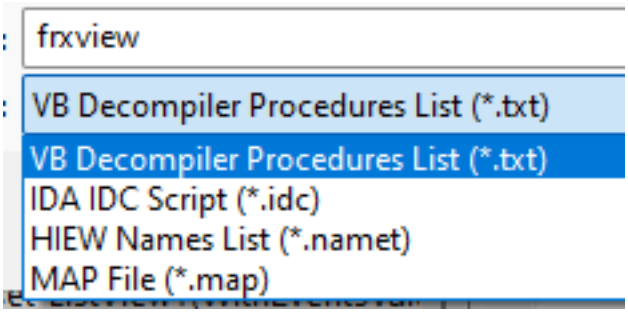
已保存的分析数据库使用**.vdd**扩展名。您可以通过文件打开工作流程打开 VDD 数据库，将其拖放到反编译器窗口中，或在命令行上传递其路径。

在定位原始分析的二进制文件时，VB Decompiler 会检查其保存的路径和包含数据库的目录。为了协作，当原始路径不可用时，请在 VDD 文件旁边保留相应的二进制文件。请保留与分析时完全相同的二进制文件版本，以确保地址和文件字节相互对应。

6.3 保存过程列表

此操作通过调用**保存程序列表 (Save procedures list)**选项在**文件 (File)**菜单中执行，是分析程序的强大工具，因为它允许外部实用程序协助**VB Decompiler**。

目前，处理程序中使用的地址、函数名称和过程可以以几种格式保存。您可以通过在标准保存对话框中指定文件类型来选择这些选项中的任何一个。



保存为 TXT 格式允许您将过程列表保存到简单的文本文件中。这对于比较程序不同版本之间的过程、检测地址变化等非常有用。

保存为 IDC 脚本允许您将函数和过程地址加载到**IDA Pro**反汇编器中。在分析的程序被编译为**Native Code**的情况下，此功能非常有用。

保存为 NAMET 格式允许您使用**HIEW**实用程序简化分析。将获得的文件复制到包含分析程序的文件夹中。然后在 HIEW 中打开该程序并按下 F12 键。将出现一个具有空过程列表的窗口。如果 NAMET 文件的地址与创建的地址匹配，请再次按下 F12 键 - 只需按 Enter。过程地址将加载到 HIEW 中，允许您通过按 F12 键使用列表导航方法。

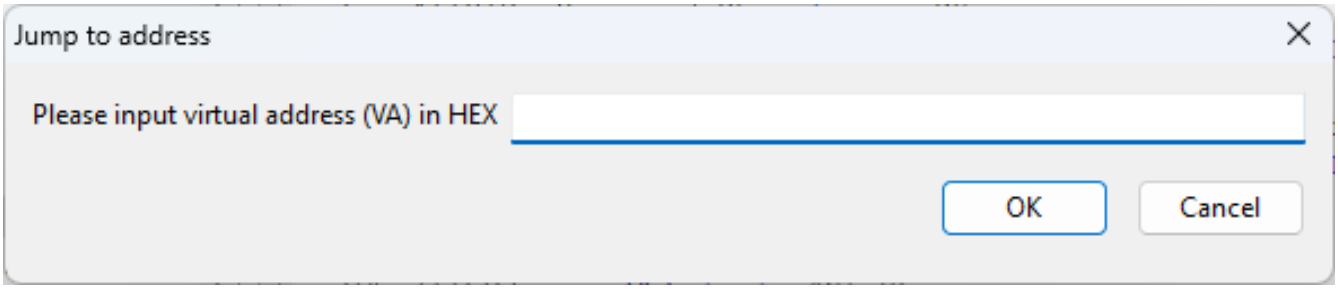
保存为 MAP 格式生成通常由编译器创建的标准 MAP 文件。此文件可用于许多调试器和反汇编器以标记程序函数的开始。由于用 Visual Basic 编写的程序经常由第三方软件错误地反汇编，在**VB Decompiler**中生成的 MAP 文件将大大简化该过程。此格式通过插件在各种产品中支持，例如**OllyDbg**、**IDA Pro**和其他产品。

7 程序功能

7.1 跳转到地址

跳转到虚拟地址

选择**工具 > 跳到虚拟地址 (Tools > Jump to virtual address)**或按**Ctrl+G**。输入要检查的代码的虚拟地址。您还可以在列表中选择一个地址并调用 Ctrl+G。反编译器打开与该地址关联的代码。



虚拟地址（VA）标识可执行映像的地址空间中的位置，并包括其映像基址。文件偏移量标识磁盘上文件中的字节位置。不要将两者互换：十六进制编辑器支持这两种形式的导航。

通过导航历史返回

使用树上方的箭头按钮向后和向前移动以查看之前查看的过程。**Esc**返回到上一个过程。鼠标“后退”和“前进”按钮也使用此历史记录，同时保留过程中的位置。

通过其树路径共享位置

在Solution explorer中，右键单击所需对象并选择**复制区段的名称 (Copy the name of section)**，或在树具有焦点时按**Ctrl+C**。这将复制所选节点的完整路径。

要重新访问该位置，请在树的上下文菜单中选择**跳转到节 (Jump to the section)**，或在树中按**G**，并将复制的路径粘贴到对话框中。这在将调查交给另一位分析师或返回大型项目时非常有用。请将该路径与对应的分析文件一起使用；来自其他构建的路径可能不再标识相同的代码。

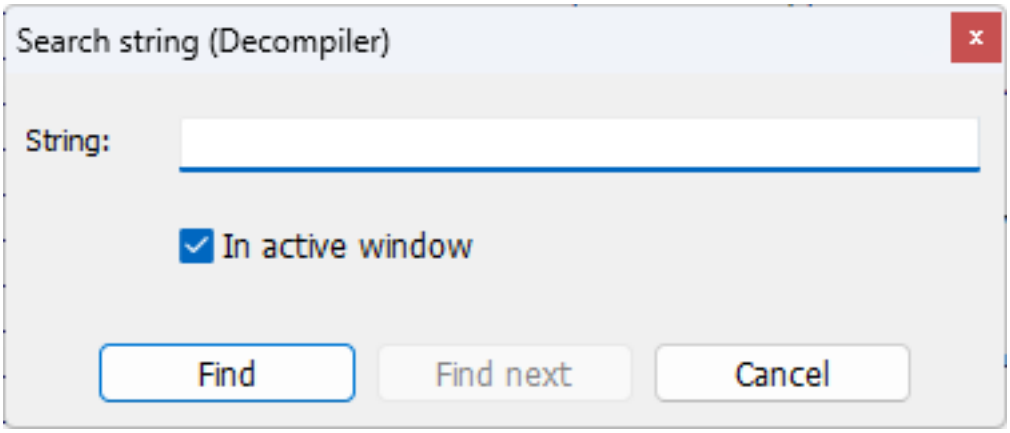
7.2 字符串搜索

按名称查找过程

选择**工具 > 查找过程 (Tools > Find procedure)**或按下**Ctrl+T**。输入完整的过程名称或其显著部分。搜索范围涵盖项目树，因此您无需知道哪个模块或类包含该过程。如果多个名称匹配，结果列表将包含所有匹配的条目；双击所需的条目以打开它。

搜索显示的文本或整个项目

选择**工具 > Find string (Tools > Find string)**或按下**Ctrl+F**以打开文本搜索对话框。选中**在活动窗口 (In active window)**时，搜索将应用于当前显示的代码或窗体数据。清除此复选框以在整个反编译项目中进行搜索。



在**字符串 (String)**中输入文本并选择**查找 (Find)**。使用**查找下一个 (Find next)**继续搜索相同的文本。**取消 (Cancel)**关闭对话框。

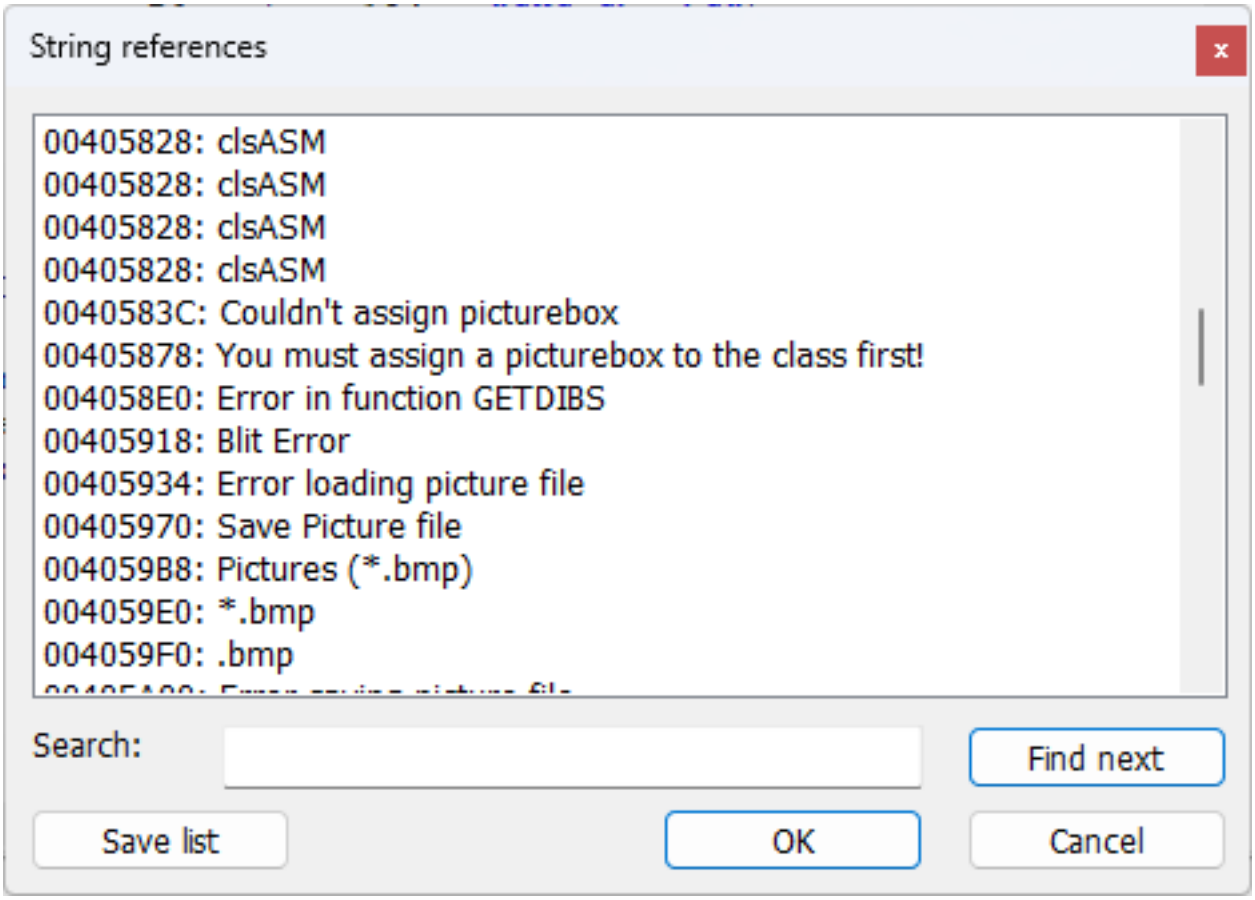
搜索全部反编译文本需要完整反编译。禁用 **快速反编译 (Fast decompilation)**（位于高级功能），然后重新处理文件。按名称搜索过程是针对项目结构执行的另一项操作。

搜索字符串引用

如果您需要字符串字面量以及使用它们的代码，请打开字符串引用。其**搜索 (Search)**字段和**查找下一个**按钮用于搜索字符串列表。字符串引用可以通过在快速反编译模式下进行初步反汇编来收集；它们不需要搜索每个完全反编译的过程。

7.3 字符串引用

在任何程序的代码中，您经常会遇到字符串数据（注册表键、文件名、错误消息等）。查看已反编译应用程序中过程和函数所使用的字符串列表非常有用。这正是我们的**字符串的引用 (String references)** 功能所设计的用途，可通过**工具 (Tools)**菜单访问。要使用它，只需导航到**工具** -> **字符串的引用**，如果您的程序包含任何字符串数据，它们将如下所示显示：

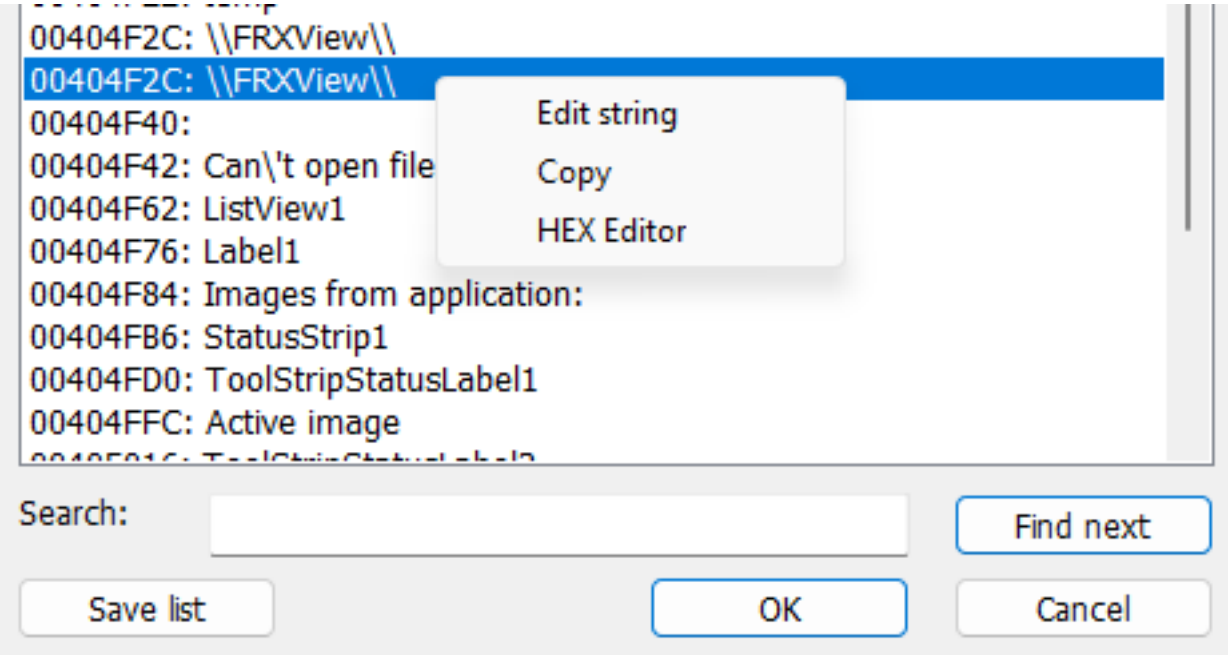


在列表中找到所需的行并双击它。将定位包含该行的代码段，并在代码窗口中打开相应的过程。再次双击该行将在程序代码中继续搜索此行。单击“确定”将在代码中找到高亮的行并关闭行选择对话框。取消将简单地关闭对话框。

显示**字符串的引用** 的功能仅在软件的授权版本中可用，而不可用于**Lite** 版本。

重要提示： 字符串不仅针对活动模块（如 1.x 程序中的版本）显示，而是针对整个程序显示。此更改是为了便于分析具有大量窗体和模块的程序而做出的。

VB Decompiler 还支持编辑字符串。要调用编辑字符串对话框，右键单击要更改的字符串，并从出现的上下文菜单中选择**编辑字符串 (Edit string)**。将出现一个用于输入新字符串的对话框。更改字符串后，单击**Ok**，字符串将直接在编辑的文件中修改。我们建议在检查文件之前创建备份，因为在此操作期间不会创建备份，并且无法撤销此操作。



重要提示： 编辑字符串时，请记住新字符串的长度不应超过旧字符串的长度。这与二进制应用程序文件的格式有关，其中没有为插入较长的字符串预留空间。

您还可以在十六进制编辑器中查看和部分编辑字符串。为此，右键单击字符串并从出现的菜单中选择十六进制编辑器。在十六进制编辑器中，有撤销和重做更改的功能，这简化了修改应用程序字符串数据时的实验。

具有字符串列表的窗口还有一个名为**搜索 (Search)** 的文本字段和一个**查找下一个 (Find next)** 按钮。此功能对于在列表中查找字符串非常有用，特别是在处理大型项目时，通过列表手动进行视觉搜索极其困难。

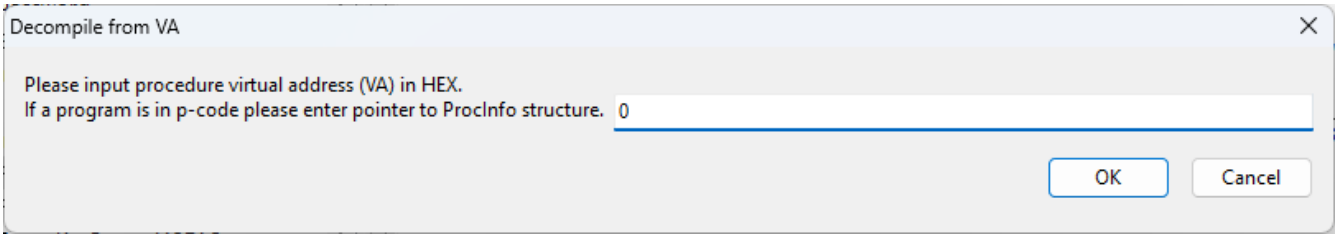
快速反编译和大字符串列表

可以通过初步反汇编收集字符串引用，包括在**快速反编译 (Fast decompilation)** 模式下。这与搜索所有完全反编译的文本不同。26.2 版本使对话框支持超过 32,000 个引用，并改进了大列表的内存处理。

对于 .NET 程序集，可以检查来自#US 堆的用户字符串及其文件位置。该列表代表可用于静态分析的字符串；加密或动态生成的字符串可能需要单独调查。

7.4 从指定地址反编译

在**工具 (Tools)**菜单中，有一个非常实用的功能——**从 VA 反编译 (Decompile from VA)**（从提供的虚拟地址反编译任意代码）。该功能的主要特点是，它不要求被反编译的文件具有正确的结构，也不需要通过标准方式进行处理。当然，在这种情况下，代码的反编译将受到一定限制。例如，如果未预先处理导入函数或OLE对象，则相关代码片段可能无法提供足够的信息。



若要反编译文件中的特定代码段，只需指定该代码的虚拟地址（如果涉及**Native Code**或**.NET 程序集**），或提供**ProcInfo**结构引用（如果涉及经典**Visual BasicP-Code**）。与**P-Code**相关的独特之处在于，**ProcInfo**结构包含代码所使用的数据数组。没有该数组，就无法确定代码使用哪些函数、引用哪些字符串等。在处理**.NET 程序集**时，输入地址后，反编译器将提供选择：是将代码作为**MS IL** 中间代码进行反编译，还是将其反汇编。此选项是因为.NET程序集可以同时包含MS IL 中间代码（**托管代码**）和机器码（即所谓的**非托管代码**）。

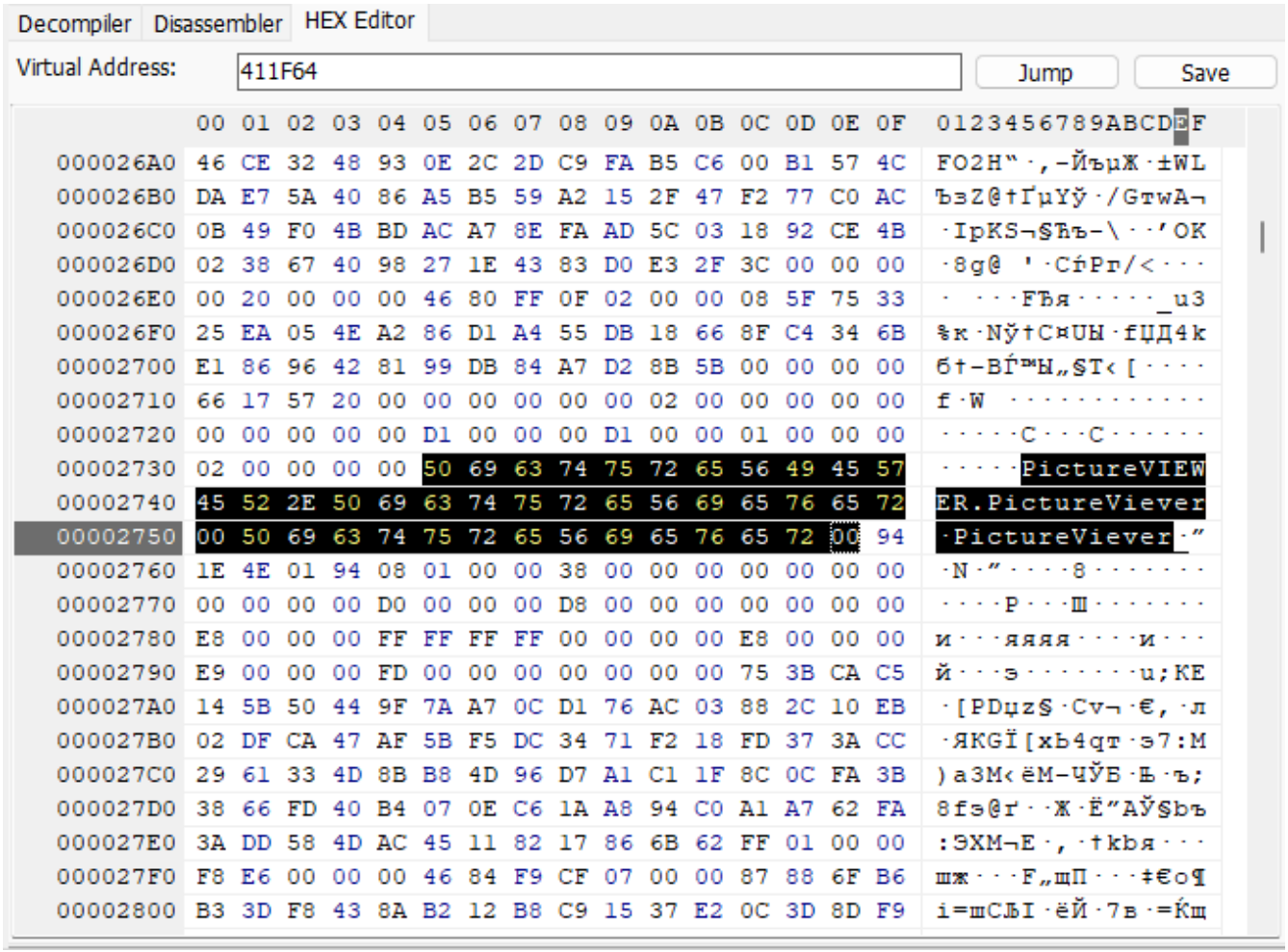
如果文件已在反编译器中打开，只需调用**从 VA 反编译**函数并指定处理地址即可。如果尚未打开文件，将自动提示，并以简化模式打开该文件。

从 VA 反编译函数在所需代码无法自动反编译或文件部分损坏（导致无法按常规方式分析）时特别有用。当然，应理解在这种情况下反编译的结果不如标准反编译理想。

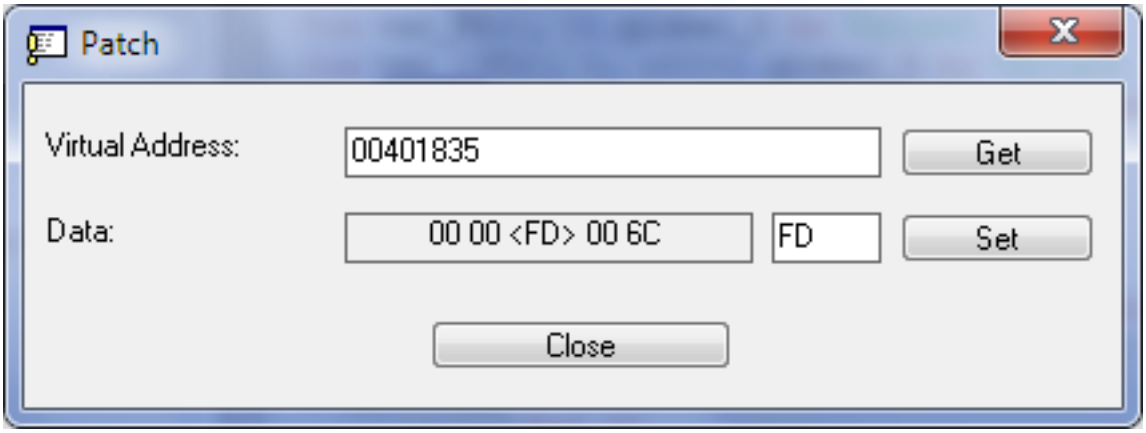
7.5 修改数据

从版本 9.4 开始，**VB Decompiler** 配备了一种用于在文件内编辑二进制数据的现代化且便捷的工具——功能完备的**十六进制编辑器 (HEX Editor)**。该工具可从主程序窗口访问，位于每个过程或窗体的**十六进制编辑器**选项卡中，与反编译的代码并列显示。

十六进制编辑器允许您按字节或按块编辑文件。要导航到检查文件中的目标地址，您可以使用虚拟地址（包括 **Image Base**）或偏移量（表示相对于文件开头所需数据偏移量的常规地址）。



在反编译器的早期版本（版本 9.4 之前）中，文件编辑功能使用更原始的工具“数据补丁器”实现。其工作原理如下：在**工具 (Tools)**菜单->**补丁数据 (Patch data)**中，存在一个名为“补丁器”的工具，可用于在分析程序代码后需要修改某些部分的情况。为避免使用外部工具并更改所需的字节，**VB Decompiler**内置了此功能。补丁器的窗口如下所示：



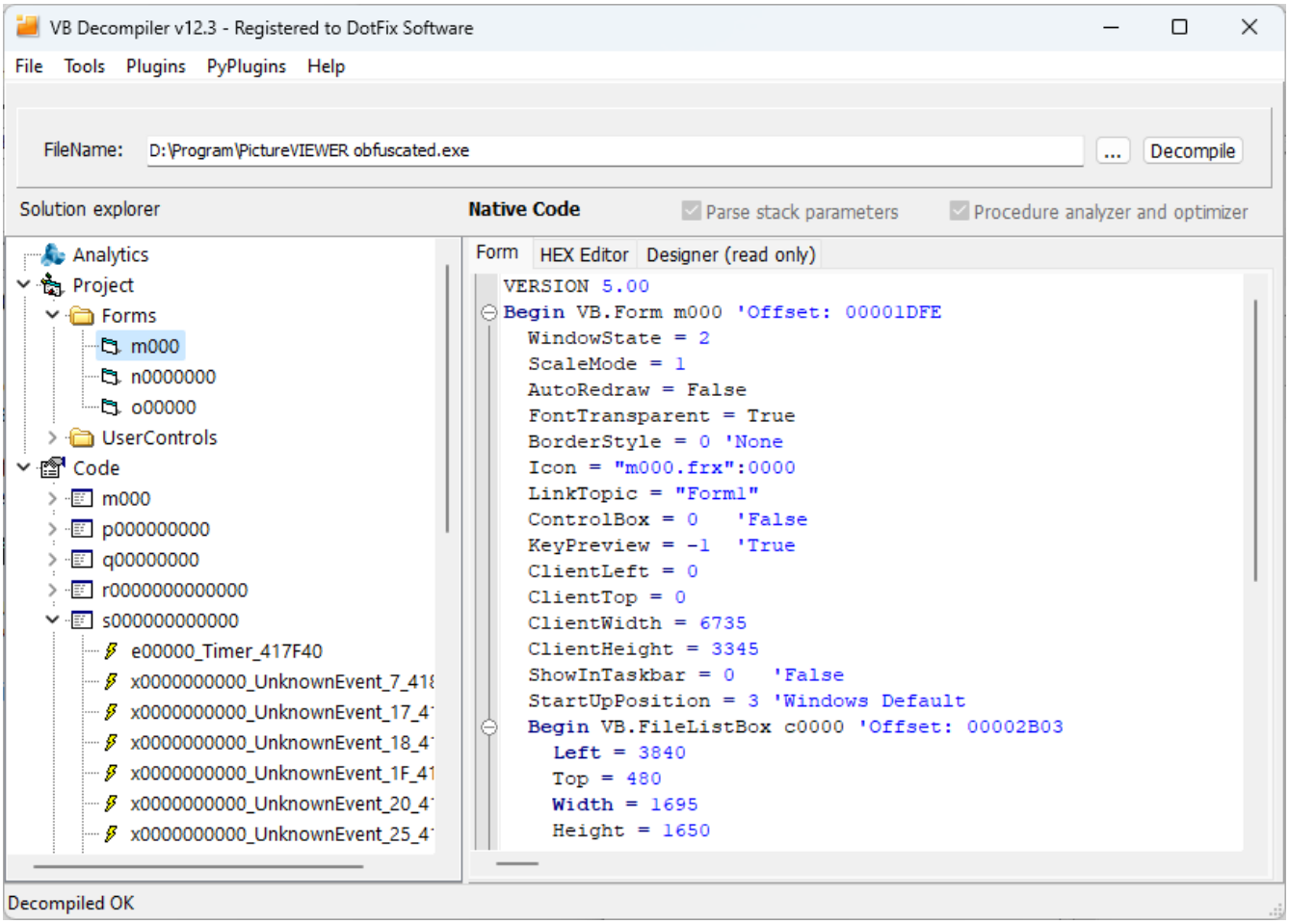
使用补丁器非常简单。在**虚拟地址 (Virtual Address)**字段中，您输入要编辑字节的地址，然后按**Enter**或点击**Get**。所选地址附近的数据字节将显示在“数据”字段中，您可以更改选定的字节，之后必须点击**Set**，输入的字节即被更改。

如果在程序选项中启用了**快速反编译 (Fast decompilation)**选项并禁用了**缓存反编译代码 (Cache decompiled code)**选项，则所有更改都会反映在该函数中（即修改字节的函数）。为此，只需双击树中的该函数名称重新打开该函数即可。

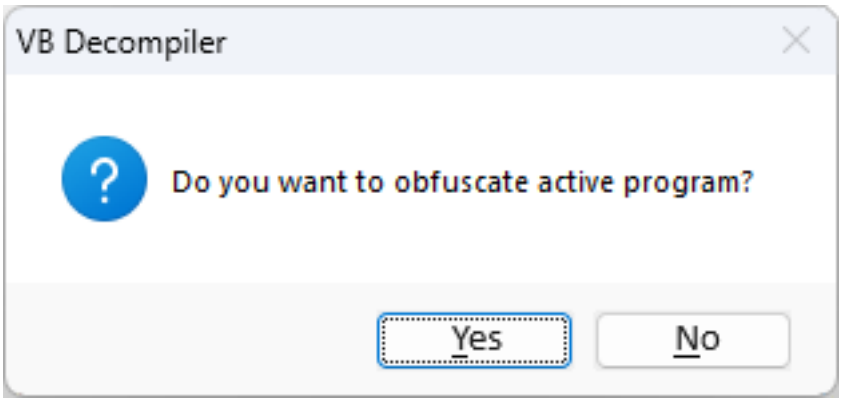
重要提示：此功能仅用于编辑您自己的程序文件或您拥有相应权限的文件，以及在反病毒实验室中研究恶意代码。在更改他人的代码之前，请查阅有关是否存在此类权利的许可协议。

7.6 混淆

如果您正在开发**Visual Basic 5.0/6.0**程序，您可以提高其防反编译的保护级别。为此，**Obfuscator**被嵌入到反编译器中，能够从您的程序的二进制文件中移除所有函数、窗体、模块和对象的名称以及各种签名。这会增加使用本反编译器或其他工具分析程序的难度。此外，大多数第三方产品将拒绝反编译此文件。



要应用此保护，请在反编译器中打开您的项目，并从**工具 (Tools)**菜单中选择**混淆项目 (Obfuscate project)**。此过程不需要太多时间，因此我们建议在发布之前对您的项目进行混淆。



此功能也集成到我们的综合软件保护解决方案**DotFix NiceProtect**中，可以从www.niceprotect.com下载。

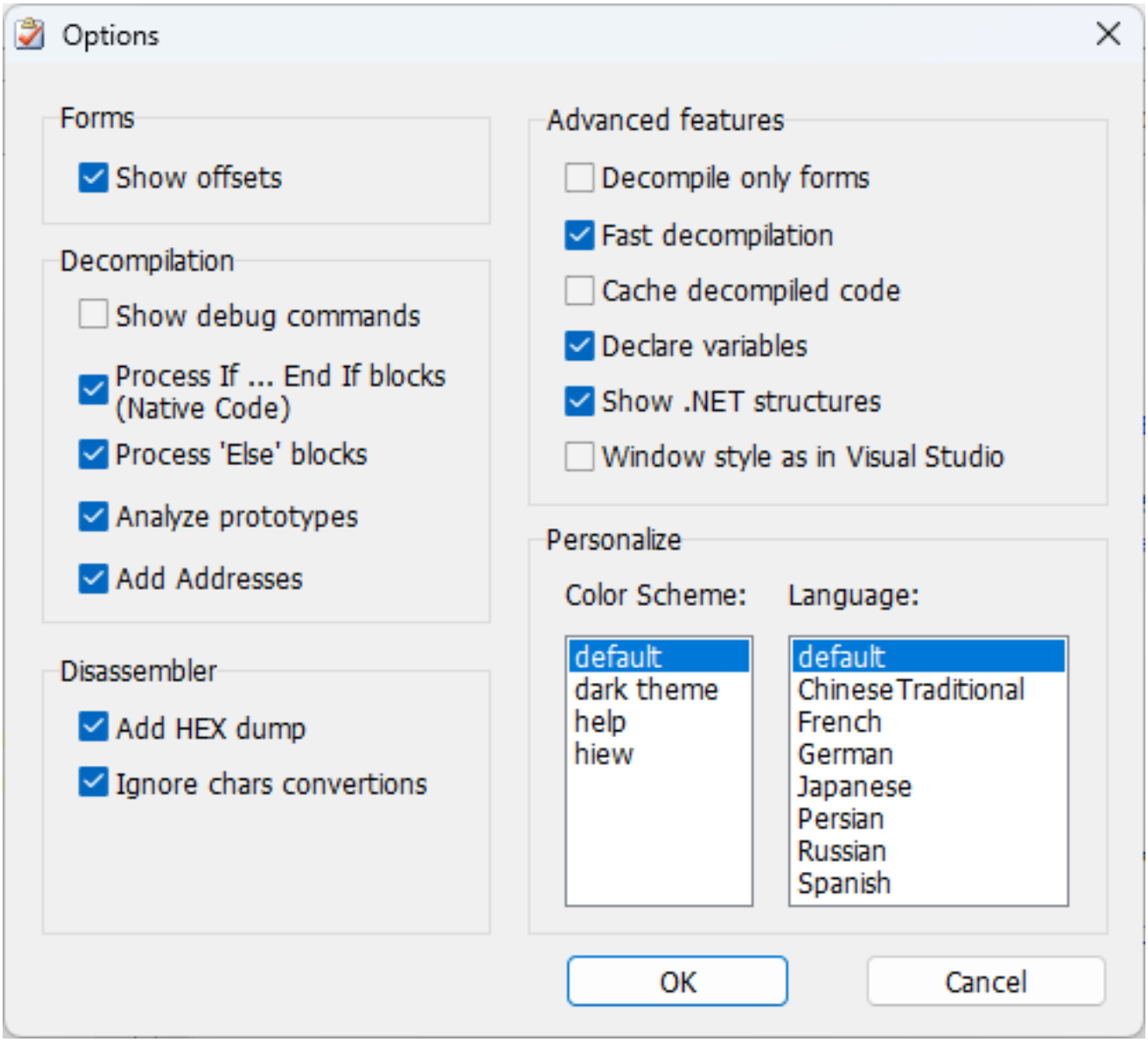
混淆功能仅供持有商业许可证的**VB Decompiler**用户使用。

混淆限制

8 程序选项

8.1 程序选项

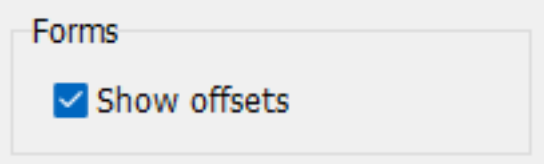
若要更改程序设置，请进入 **工具 (Tools)** 菜单 > **选项 (Options)**。在生成的对话框中，您将看到以下选项：



窗体
反编译
反汇编器
配色方案 高级功能 语言选择

8.2 窗体

您可以为窗体更改以下反编译设置：



在此设置部分中，您可以启用在窗体对象旁边显示偏移量。这在例如在没有源代码的情况下将程序本地化为另一种语言时可能是必要的。**显示偏移 (Show offsets)** 将允许您找到这些对象在程序中的位置、它们的名称和属性。

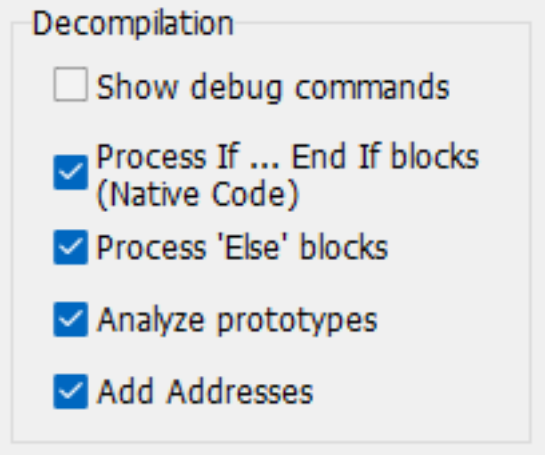
之后，您可以使用 HEX 编辑器 修改这些值。请注意，如果您的知识仅限于基本用户级别且不熟悉二进制文件的内部工作原理 - 不建议启用此选项。因为对文件进行的任何不正确修改都会使其无法运行，而不必要的地址只会妨碍您。

可视化预览和事件导航

窗体选项与 **设计器（只读） (Designer (read only))** 查看器是分开的。有关经典 VB 窗体、Windows Forms、WPF/BAML 支持、事件处理程序导航以及每种界面技术的限制，请参阅 可视化窗体重建。

8.3 反编译

打开**工具 > 选项 (Tools > Options)**以配置反编译。在交互式比较设置时，请使用无缓存的快速反编译；否则在更改选项后重新处理文件。



显示调试命令 (Show debug commands)

对于经典 VB P-Code，包含编译器和调试器操作，这些操作否则不会包含在高级语言代码清单中。当检查底层指令序列而非仅重构的程序逻辑时启用它。

处理 If ... End If 区块 (Native Code) (Process If ... End If blocks (Native Code))

将识别的原生代码条件跳转重构为`If...End If`块。当需要检查原始分支模式或分析结构看似不正确的过程时禁用它。

处理 'Else' 块 (Process 'Else' blocks)

使用额外的启发式方法区分 Else 分支与嵌套条件和 GoTo 跳转。对于原生代码，这需要对 If 块进行处理。保持启用以进行正常阅读；当重构的分支不明确时，将其禁用进行比较。

分析原型 (Analyze prototypes)

对过程参数和返回值执行初步启发式分析。这对于快速反编译特别有用：标准模块中的私有函数和过程可能缺少显式的签名元数据。初步分析需要额外时间，但避免仅根据当前已知的堆栈内容处理调用。启发式签名仍然是一种推断，特别是当元数据、外部类型或代码不完整时。

添加地址 (Add Addresses)

在重构的语句旁边显示虚拟地址。保持启用以将代码与反汇编、文件字节和其他工具的发现相关联。它对于**复制到反汇编器 (Copy to disassembler)**合并高级语句与其低级对应项也是必需的。反汇编器始终独立于此设置显示地址。

理解原生代码重构

原生代码引擎识别支持的`Select Case`模式、多维数组访问、COM 调用和 VB 运行时操作。错误处理可以重构为`On Error Resume Next`、`On Error GoTo`和`On Error GoTo 0`。编译器优化和缺失的类型信息仍可能限制结果；当高级输出不完整时检查反汇编。

版本 26.2 扩展了 Variant 类型、多维 SafeArray 读写以及`VarPtr`数组元素重构。输出可能会将数组引用标识为`SafeArray`，反映分析引擎识别的运行时结构。分支状态隔离和临时变量分析也改善了复杂 If/Else 代码的重构。版本 26.5 进一步改进了 Select Case 和 VB5/VB6 运行时函数处理。

COM 和 ActiveX 类型信息

VB Decompiler 首先使用其内置的已知接口知识库。对于未知的外部 COM 对象，它可以通过系统注册表解析库、类和接口标识符，并解析相应的 DLL/OCX 类型库。这为早期绑定和支持的后期绑定调用提供方法和属性签名，包括返回的接口类型。

如果缺少所需的类型库或其注册无法解析，某些调用可能保持未识别，例如`UnkVCall`。因此，在比较不同分析机器的输出时，需要考虑缺失依赖项的影响。扩展的 IID 知识库包括 Office 自动化、ADO/DAO、XML 和 HTTP、脚本、Shell、消息传递和目录服务接口；它并不意味着可以解析每个可能的 COM 调用。

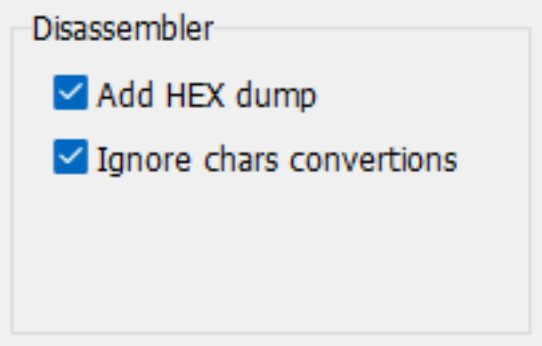
仅有限的 UDT 元数据保留在某些 VB 二进制文件中。反编译器可以恢复诸如结构大小和字符串字段计数等信息，但不一定能恢复完整的原始字段布局。不要将部分声明解释为完整的源代码级定义。

.NET 重构和优化

支持的 IL 被重构为类似 C# 的表达式和控制流。主窗口中的 **程序分析器和优化器 (Procedure analyzer and optimizer)** 控件可以简化支持的循环、条件、操作符和临时变量模式。在遇到不支持或混淆的片段时检查 IL 和元数据。此处理独立于AI 助手 (AI Helper)。

8.4 反汇编器

此选项块负责反汇编机器代码：



添加十六进制转储 (Add HEX dump) 在每条反汇编指令前显示机器代码，这在分析后使用 HEX 编辑器 修改二进制文件时可能是必要的。

忽略字符转换 (Ignore chars convertions) 用于禁用将非英文字符串数据转换为Unicode 数值编码。启用此选项可使显示的字符串更易阅读。当需要Unicode 数值编码 表示时请禁用它。

显示用户类 (Show user classes) 包含在 **Delphi / Free Pascal** 应用程序内部解析用户类。此选项不适用于 **Visual Basic** 和 **C#** 应用程序。

显示VCL类 (Show VCL classes) 包含反编译 **VCL** 库的所有内部类。建议禁用此选项，以防止反汇编器分析数千个在常规程序分析中通常不需要的内部辅助类。此选项仅用于反编译 **Delphi** 应用程序的内部目的，并旨在与其他产品统一，但在 **VB Decompiler** 中未实现。

Delphi 和 **Free Pascal** 应用程序的部分反编译目前仅作为单独服务（而非软件）提供。

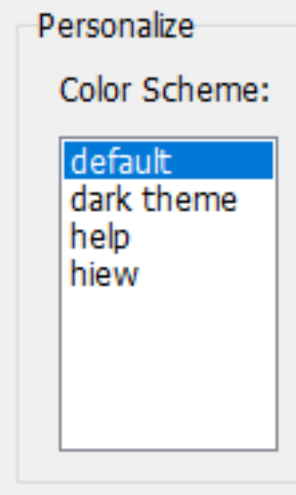
VB P-Code 和 .NET IL

添加十六进制转储 也支持底层 P-Code 代码清单。需要在同一个清单中对照恢复的语句与 P-Code 或原生指令时，请在 反编译器 (Decompiler) 视图中使用 **复制到反汇编器 (Copy to disassembler)**。

对于 .NET 程序集，反汇编器可显示适用的 IL 或本机 x86/x64 方法。恢复的局部变量签名可包括数组、泛型、值类型和 TypeSpec 引用。当需要检查代码旁边的元数据表和堆时，请启用 **显示 .NET 结构 (Show .NET structures)**。后续版本扩展了对混淆元数据的支持，包括 FieldPtr 表；但这并不保证成功恢复每个损坏或混淆的程序集。

8.5 配色方案

本节定义了查看反编译器中的代码时的配色方案和语法高亮：



预设值存储在 VB Decompiler 安装目录中的.ini 文件内的"colors"文件夹中。文件名决定了程序设置部分中的预设名称。这些文件格式简单，您可以轻松自行创建或编辑现有文件。

配置格式如下：

```
[Colors]
Background = decimal color code (for example: 1973790)
Foreground = decimal color code (for example: 14671840)
KeyWords = decimal color code (for example: 13471849)
LocationsAndArguments = decimal color code (for example: 11849128)
Procedures = decimal color code (for example: 5676246)
; next properties is only available in version 12 or later
UnderCursorBackground = decimal color code (for example: 6986069)
UnderCursorForeground = decimal color code (for example: 1973790)
UseDarkMode = 0 or 1 (0 for Light Theme, 1 for Dark Theme)
Colors2 = 0 or 1 (using extended colors at Colors2 section)
```

```
[Font]
Name = font name (for example: Consolas)
Size = font size (for example: 11)
KeyWordsBold = 0 or 1 (font bold)
LocationsAndArgumentsBold = 0 or 1 (font bold)
ProceduresBold = 0 or 1 (font bold)
```

```
; this section is only available in version 12 or later
[Colors2]
TVFontName = Solution Explorer treeview font name or list of font names, separated by comma (for example:
    Segoe UI, Tahoma, Arial)
TVFontSize = size of the font (for example: 10)
FontName = Decompiler and Disassembler font name or list of font names, separated by comma (for example:
    Courier New, Consolas, Liberation Mono, monospace)
FontSize = size of the font (for example: 11)
Background = BackColor in hexadecimal for Decompiler and Disassembler tab (for example: 1E1E1E)
Foreground = ForeColor in hexadecimal for Decompiler and Disassembler tab (for example: DFDFE0)
; code highllighting
Key = color in hexadecimal (for example: CD9069)
Comment = color in hexadecimal (for example: CD9069)
Char = color in hexadecimal (for example: B4CDA8)
Float = color in hexadecimal (for example: B4CDA8)
Hex = color in hexadecimal (for example: B4CDA8)
Number = color in hexadecimal (for example: B4CDA8)
String = color in hexadecimal (for example: 569CD6)
Symbol = color in hexadecimal (for example: DFDFE0)
; HEX Editor font and colors
HexFontName = font name or list of font names, separated by comma (for example: Courier New, Consolas,
    Liberation Mono, monospace)
HexFontSize = size of the font (for example: 11)
HexBackground = color in hexadecimal (for example: 1E1E1E)
HexForeground = color in hexadecimal (for example: DFDFE0)
HexGrid = color in hexadecimal (for example: 1E1E1E)
HexActiveFieldBackground = color in hexadecimal (for example: 1E1E1E)
HexOffsetBackground = color in hexadecimal (for example: 1E1E1E)
HexCurrentOffset = color in hexadecimal (for example: B4CDA8)
HexOddColumn = color in hexadecimal (for example: B4CDA8)
HexEvenColumn = color in hexadecimal (for example: B4CDA8)
HexOffset = color in hexadecimal (for example: 569CD6)
```

颜色代码以 32 位真彩色格式指定。Colors 部分使用十进制（为了与 VB Decompiler 旧版本向后兼容），Colors2 使用十六进制。

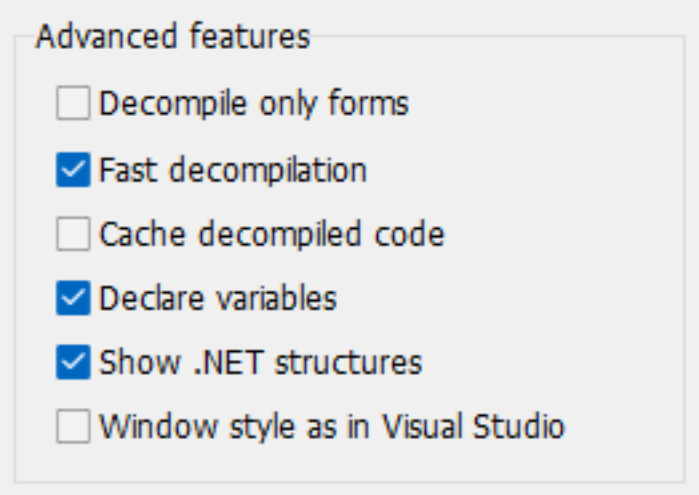
分析代码时更改主题

主题系统支持暗色模式，并且可以在不重启 VB Decompiler 的情况下应用所选主题，包括在编辑其 INI 文件之后。扩展配置中的字体设置可以包含逗号分隔的后备名称，这有助于 Windows 和兼容环境之间等效字体不同的情况。

光标下方的颜色控制标识符高亮显示。有关局部变量、对象成员和全局变量高亮显示的信息，请参阅 查看代码。

8.6 高级功能

这些设置控制项目的处理方式以及恢复信息的展示方式。



只反编译窗体 (Decompile only forms)

当仅需恢复窗体和 UserControl 定义时，禁用代码反编译。有关可视查看器及其特定技术功能，请参阅 可视化窗体重建。

快速反编译 (Fast decompilation)

启用后，VB Decompiler 首先处理项目结构，并在您打开各个过程时对其进行反编译。这减少了调查选定过程时的初始等待时间。禁用 **缓存反编译代码 (Cache decompiled code)** 后，您可以更改分析设置并重新打开过程以获取新的代码清单。

启用 **分析原型 (Analyze prototypes)**（位于反编译设置），对私有过程的签名进行初步启发式分析。这会增加初始处理时间，但无需完整反编译即可改善参数重建。

快速反编译不提供项目的完整可搜索高级代码清单。禁用它并在使用整个项目的 文本搜索 或导出完整反编译项目之前再次处理文件。字符串引用 则不同：它们可以通过初步反汇编收集，并且在快速反编译模式下可用。

完全反编译

禁用快速反编译后，反编译器预先处理项目中的过程。这需要更长时间，但使完整生成的代码清单可用于文本搜索和导出。对反编译设置的更改需要再次处理文件。

缓存反编译代码

在快速反编译模式下保留已生成的代码清单。这避免了重复反编译打开的过程，但缓存结果不反映后续分析设置的变化。当交互式比较 **解析堆栈参数 (Parse stack parameters)** 或 **程序分析器和优化器 (Procedure analyzer and optimizer)** 设置时，请禁用缓存。

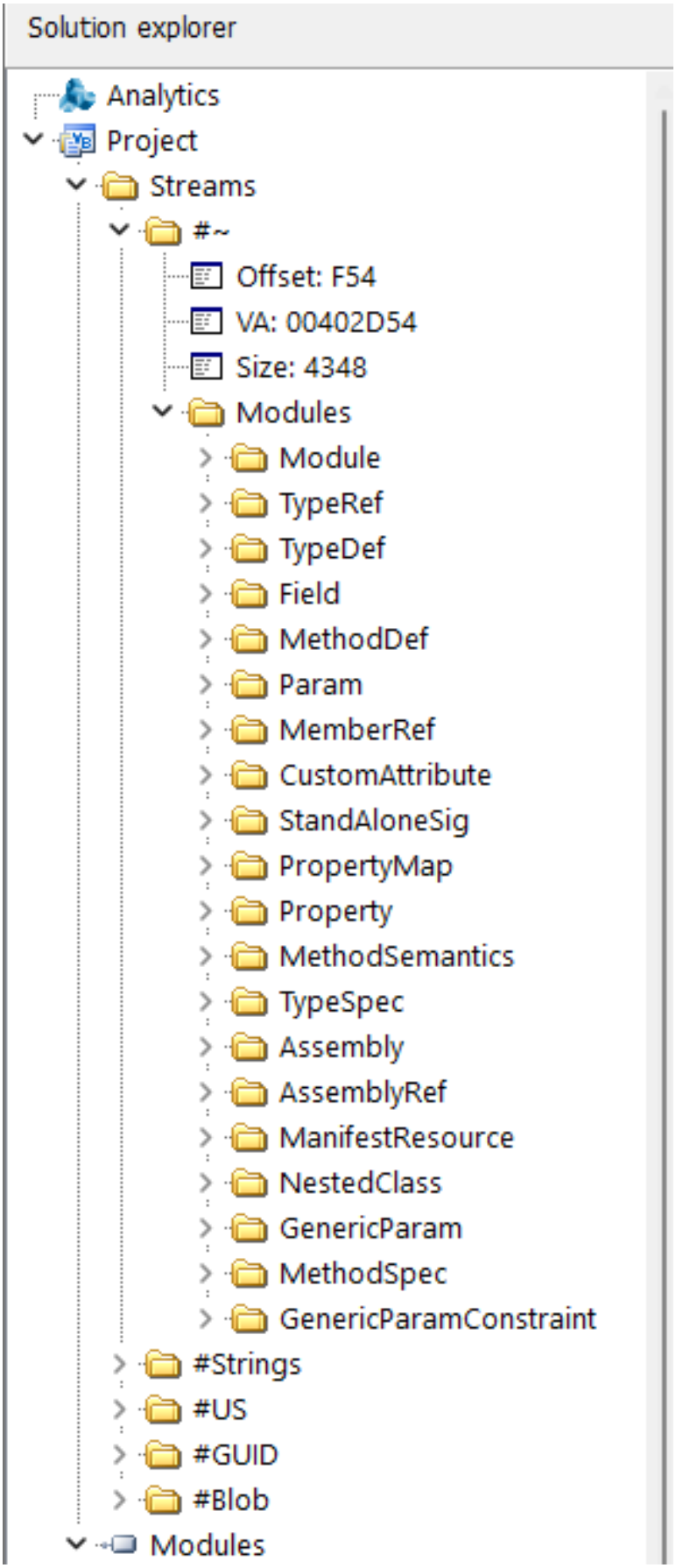
声明变量 (Declare variables)

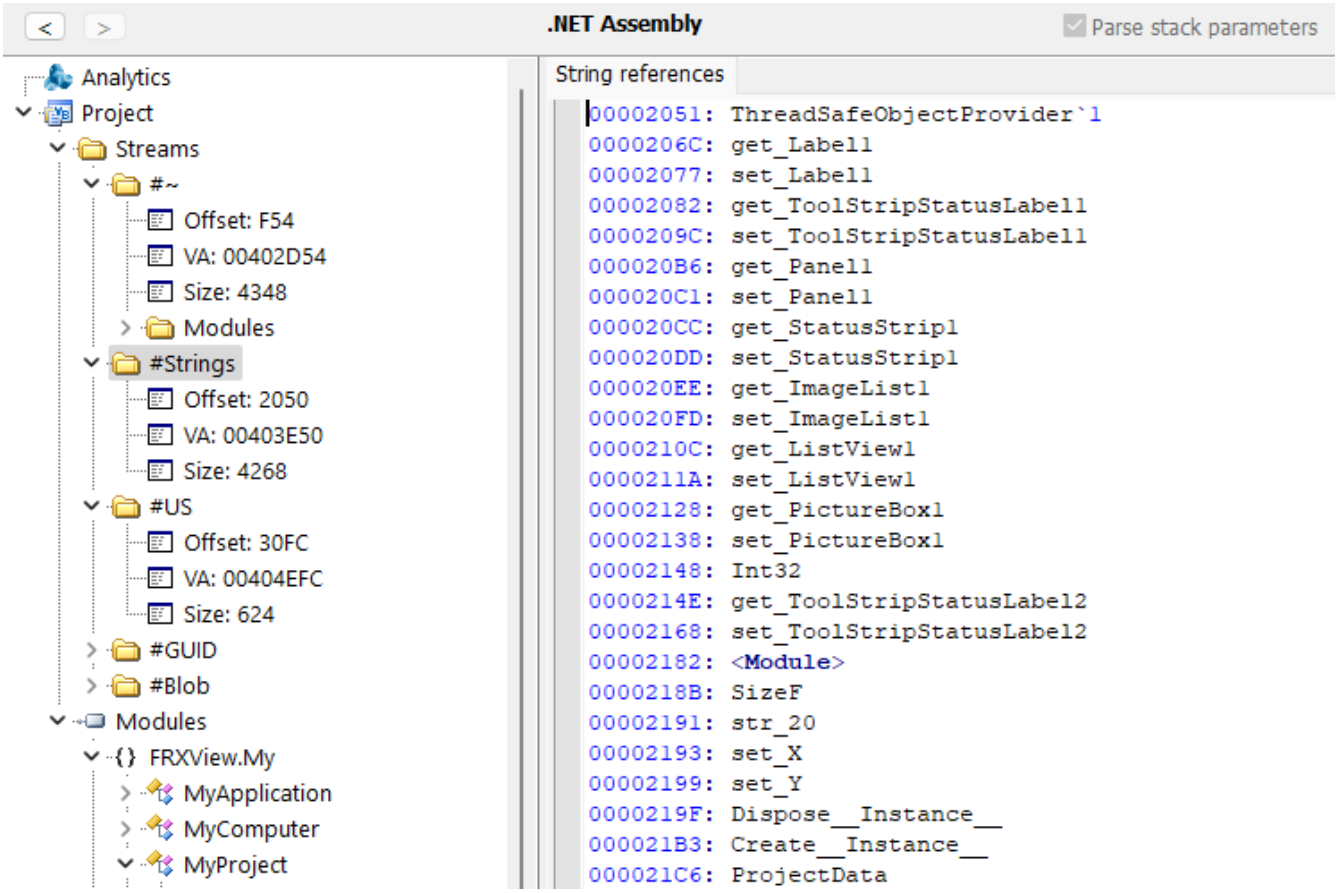
在每个反编译过程开始时为恢复的变量添加声明块。这对于源代码恢复和类型检查很有用。如果只需快速了解过程逻辑，可以隐藏变量声明。恢复的类型受反编译器可用的元数据和分析限制。

```
Private Sub Form_Load() '411290
  Dim var_B8 As Variant
  Dim var_34 As Variant
  Dim var_C0 As FileListBox
```

显示 .NET 结构 (Show .NET structures)

在 方案浏览器 (Solution explorer) 中显示 .NET 元数据表和堆。使用它来检查与代码并列的结构和签名。选择表可显示其内容；#US 堆包含用户字符串。当您不需要元数据导航时禁用该选项。


Solution explorer
Analytics
Project
Streams
#~
Offset: F54
VA: 00402D54
Size: 4348
Modules
Module
TypeRef
TypeDef
Field
MethodDef
Param
MemberRef
CustomAttribute
StandAloneSig
PropertyMap
Property
MethodSemantics
TypeSpec
Assembly
AssemblyRef
ManifestResource
NestedClass
GenericParam
MethodSpec
GenericParamConstraint
#Strings
#US
#GUID
#Blob
Modules



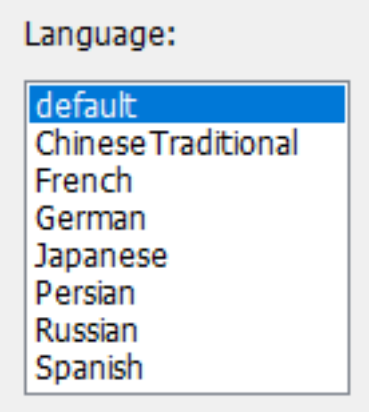
Visual Studio 窗口布局 (Window style as in Visual Studio)

将代码放在左侧，将 方案浏览器 放在右侧。这会更改工作区布局，不改变恢复的代码。

8.7 语言选择

VB Decompiler 为全球使用的超过 20 种语言提供界面本地化。文档也以多种语言提供。

您可以在 **工具 (Tools)** -> **选项 (Options)** 菜单下的 **个性化 (Personalize)** 设置中选择程序的优选语言界面。选择语言后，需要重启 **VB Decompiler** 以使整个程序正确应用语言设置。如果不想重启程序，您可以继续使用 **VB Decompiler**；然而，某些文本仍可能以前一种语言显示。



帮助语言会根据您计算机的系统语言自动选择。若要阅读其他语言的说明手册，请从安装 **VB Decompiler** 的文件夹中打开相应的帮助文件。PDF 是本地化说明手册的主要格式；某些语言仍保留 CHM 帮助文件。

英文说明手册是所有翻译的来源。如果您在本地化说明手册中发现错误或不一致之处，请向技术支持报告。

访问说明手册

如果缺少本地帮助文件或所需的 PDF 查看器，VB Decompiler 可以打开在线文档。对于离线调查环境，请提前准备好所需的本地帮助文件。

9 插件

插件通过原生 DLL 扩展 VB Decompiler。已安装的插件可从**插件 (Plugins)**菜单中访问。回调 API 提供对恢复的项目信息和代码的访问权限，允许插件检查或修改受支持的内部结构。

VB/DE Decompiler MCP Server 2.0.2

将 MCP 客户端连接到反编译器中已打开的项目。无需运行被分析的程序，即可读取模块、窗体、过程和还原的代码。

下载适用于 x86 主程序的 MCP Server 2.0.2

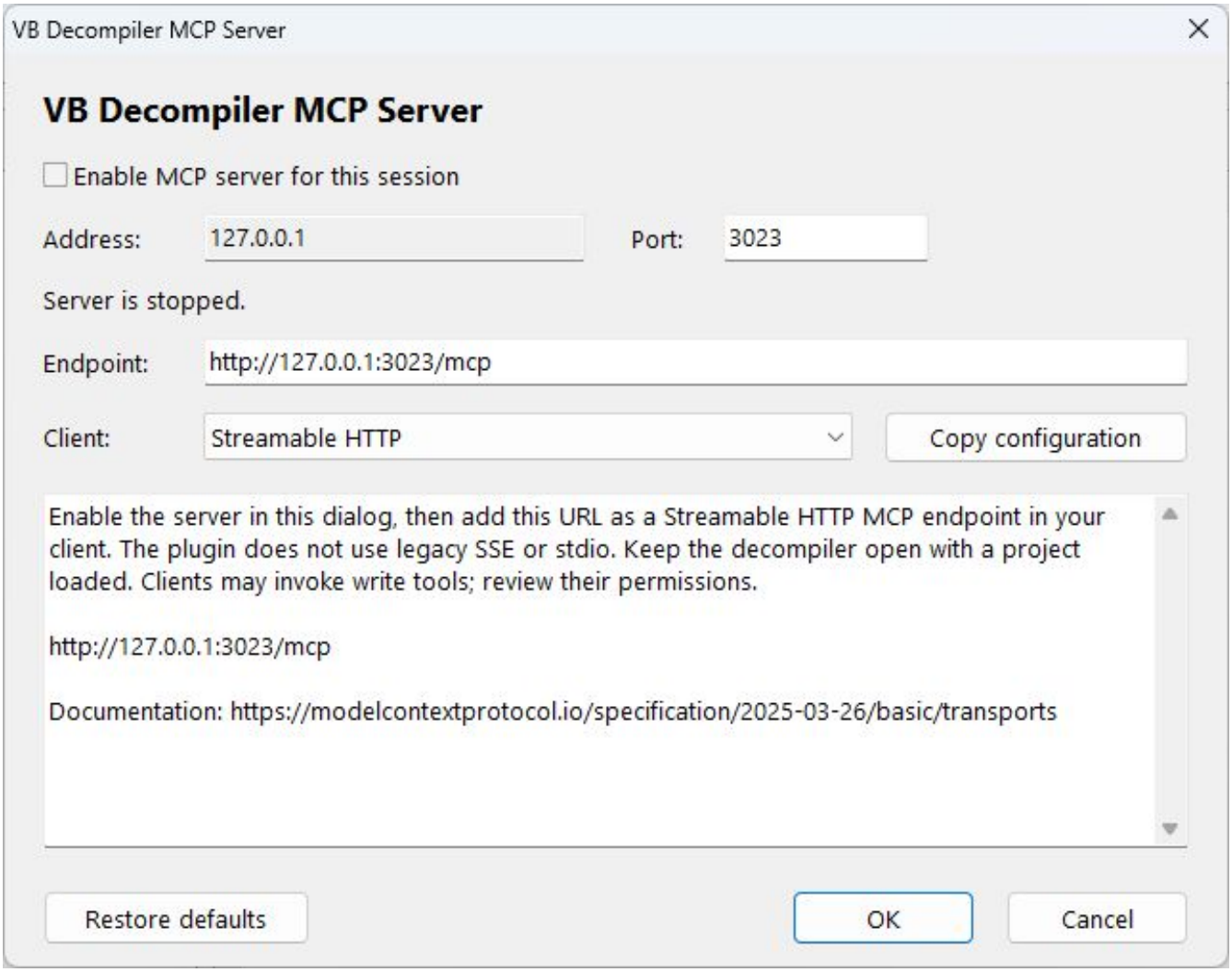
`vb_get_form` 返回窗体索引 `form_index`、窗体名称 `form_name` 和属性文本 `form_text`。如果无法获取名称，`form_name` 为空字符串。 MCP 工具参考和示例

使用要求

安装包内含适用于 32 位 Windows 版反编译器的 x86 插件。可用的工具取决于产品版本、版本类型和当前项目。在较早的版本中，部分读取操作只能访问已缓存的代码。

安装与启动

1. 关闭反编译器。解压安装包，将 `vbmcp.dll` 复制到安装目录下的 `plugins` 文件夹。如果已有旧版 DLL，请替换它。
2. 启动反编译器并打开项目。选择 **Plugins → VB/DE Decompiler MCP Server**。
3. 勾选 **Enable MCP server for this session**，单击 **OK**，然后使用窗口中显示的地址。



VB Decompiler 中的 MCP Server 设置。 插件会识别加载它的反编译器，并在窗口标题中显示产品名称。

连接 MCP 客户端

使用 Streamable HTTP。默认地址为 `http://127.0.0.1:3023/mcp`。如果同时使用两个反编译器，请为它们指定不同的端口。

Codex

```
codex mcp add vbmcp --url http://127.0.0.1:3023/mcp
```

Claude Code

```
claude mcp add --transport http vbmcp http://127.0.0.1:3023/mcp
```

VS Code

在 VS Code 中，可以运行 **MCP: Add Server** 并选择 HTTP。

```
{
  "servers": {
    "vbmcp": {
      "type": "http",
      "url": "http://127.0.0.1:3023/mcp"
    }
  }
}
```

客户端文档 · Codex · Claude Code · VS Code

检查连接

打开带有窗体的示例项目，将 `limit` 设为 `5`，调用 `vb_list_forms`。此版本的工具名称以 `vb_` 开头。

本地访问与项目修改

服务器仅监听 `127.0.0.1`，不进行用户身份验证。本地客户端可以读取项目数据，也可以修改代码清单、名称和标记。请保留客户端的写入确认。要停止服务，请取消勾选并单击 **OK**。每次启动反编译器时，服务器默认关闭。

插件在本地运行。MCP 客户端可能根据自身设置将结果发送给模型服务提供方。

原项目作者 july008 (MIT).

FRX Image Viewer

FRX Image Viewer 1.2.0 可预览并提取 VB Decompiler 或 DE 反编译器 (Decompiler) 注册的图像。VB5/6 和 .NET 窗体图像使用 VB Decompiler，Delphi 二进制 DFM 图像使用 DE 反编译器。支持 BMP、PNG、JPEG、GIF、ICO 和 CUR。

安装与打开

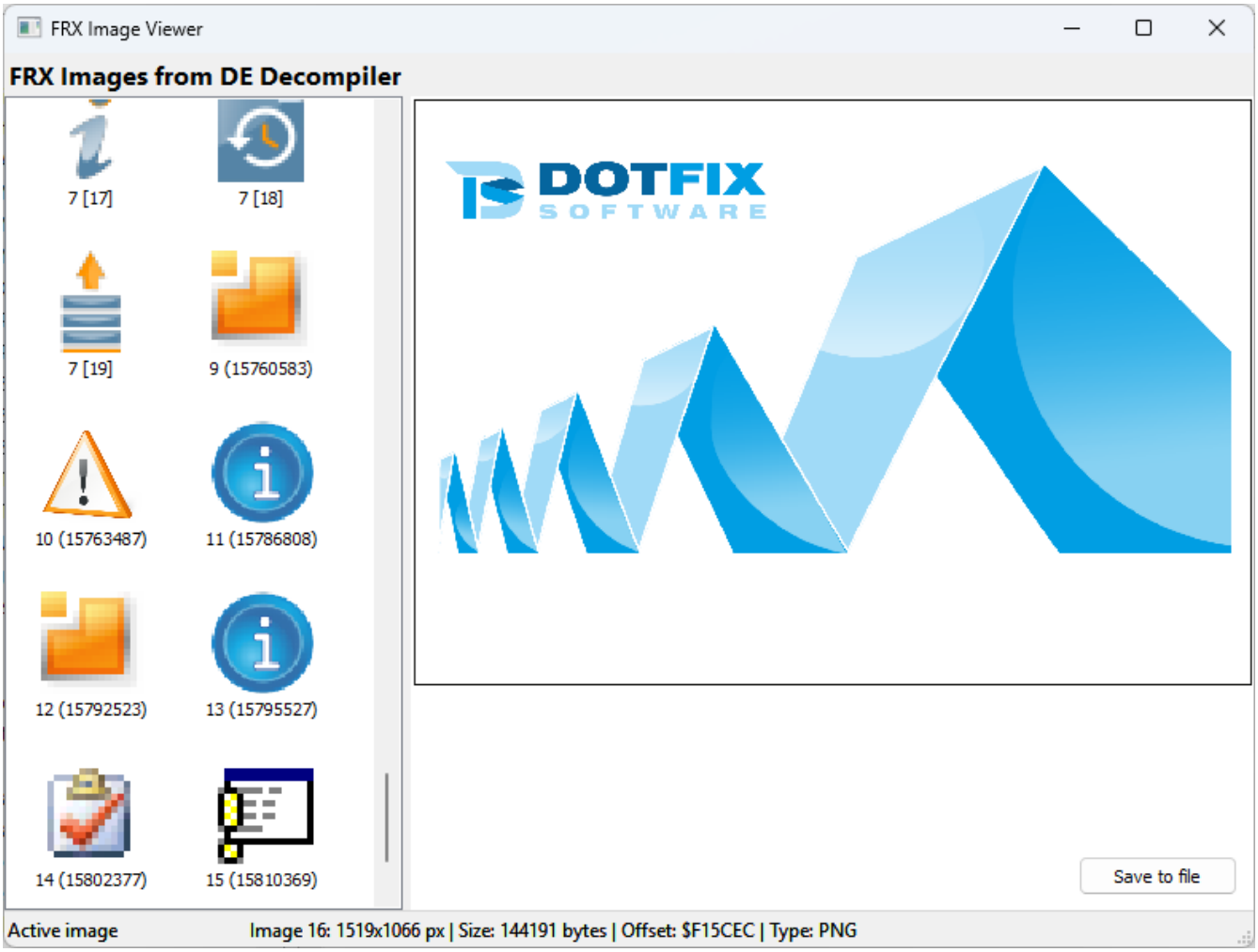
1. 将 `FRXViewer.dll` 复制到宿主安装目录的 `plugins` 子目录，然后重新启动反编译器。DLL 的体系结构必须与宿主应用程序匹配，与被分析文件的体系结构无关。
2. 打开 EXE 或 DLL 并解析其窗体，使宿主能够注册可用图像。
3. 选择 **插件** > **FRX Image Viewer**，然后单击缩略图查看图像。

1.2.0 下载包包含适用于 x86 宿主的 DLL。x64 宿主需要匹配的构建版本。插件要求 Windows 10/11；构建说明见源代码仓库。

预览与导出

缩略图库和预览区域显示可用图像，包括包含多个图像的 ICO/CUR 文件。状态栏显示所选项目、尺寸、数据大小或源位图偏移以及格式。查看器沿用宿主的主题并支持深色模式。

选择 **Save to file** 导出所选项目。独立图像保留原始字节和格式。ImageList 中的单个单元以带有掩码和 Alpha 透明度的 ICO 文件导出；这些 ICO 文件由源位图生成。



DE 反编译器 中的 FRX Image Viewer，显示 Delphi DFM 图像和单个 ImageList 图标。

原生 Delphi ImageList 与兼容性

在兼容宿主中，可识别的原生 Delphi ImageList 显示为单个图标。集合项目中独立编码的图像仍作为普通独立图像。对于不支持的数据流，只要存在原始图像条目，就保留整幅图像的处理方式。

VB Decompiler 26.6 和 DE 反编译器 26.5 支持共享 SDK 修订版 26.6，新增可选命令 `GetFrxImageListInfo` (63) 和 `GetProductName` (64)。较旧宿主保留原始图像 API 和整幅图像的处理方式。功能检测和偏移规则见SDK 参考。

预览和提取不会执行被分析文件，也不会加载其组件。动画 GIF 仅显示第一帧。可用资源取决于宿主成功解析并注册的图像。

开发插件

SDK 包含针对 Visual C++、Rust、Delphi 和 Free Pascal 的示例。请参阅VB Decompiler 插件 API 参考以获取导出的入口点、回调签名、命令标识符、数据表示形式以及示例项目的链接。

DLL 插件提供其名称和作者信息，并通过`VBDecompilerPluginLoad`接收主机回调。该回调用于请求或更新受支持的项目数据。在实现此接口时，请遵循 SDK 的调用约定、指针、字符串编码和缓冲区要求；不要在缺少兼容的原生互操作接口时，直接用托管语言声明替代此接口。

VB Decompiler 在启动时初始化可用的 DLL 插件。为预期的主机架构和 Windows 环境构建插件，并使用该环境验证生成的 DLL。SDK 同时提供对活动视图代码和项目级代码的访问；需要所有反编译过程的操作需要完整反编译，而不仅仅是按需列表。

10 可视化窗体重建

设计器（只读） (Designer (read only)) 选项卡显示从分析的二进制文件中重建的受支持用户界面的可视化视图。它有助于在不手动解释每个恢复的属性或初始化语句的情况下，将接口元素与应用行为联系起来。

打开恢复的窗体

1 打开应用程序并让 VB Decompiler 恢复其项目结构。

2 在Solution explorer中选择恢复的窗体并打开 **设计器（只读）**。

3 选择控件以检查其可用属性和恢复的事件信息。使用事件导航到相应的事件处理程序代码。

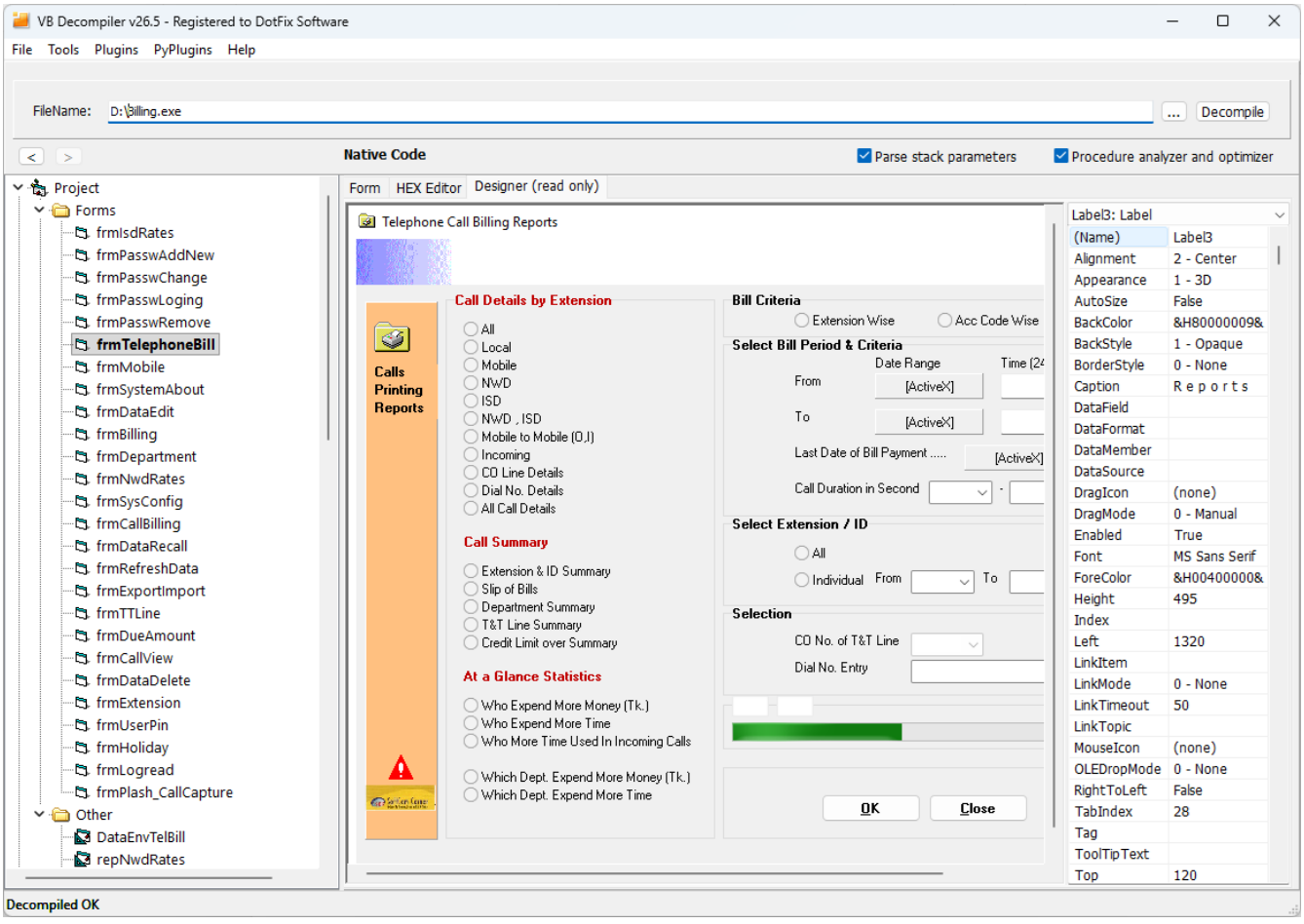
预览是从存储的信息中重建的。它不是原始用户界面的运行实例。不支持的控件、运行时创建的内容、缺少资源以及自定义布局逻辑可能导致不完整或近似的结果。

Visual Basic 5.0/6.0

查看器重构窗体和 UserControl，包括支持的控件属性、字体、颜色和图像。它读取列表、组合框和其他控件使用的二进制窗体数据。图像可以包含图标、光标、位图以及 JPEG、PNG 和 GIF 数据。

可视化菜单重构支持 P-Code 和 Native Code 应用程序。支持的控件包括 `SSTab`、`MSComctlLib.StatusBar`、`MSComctlLib.ListView`、`MSComctlLib.ProgressBar` 和 `MSComctlLib.TreeView`。

要检查经典 VB 控件的处理程序，请选择控件并双击属性列表中的其 **_Event** 字段。从生成的列表中选择所需的事件以打开处理程序。

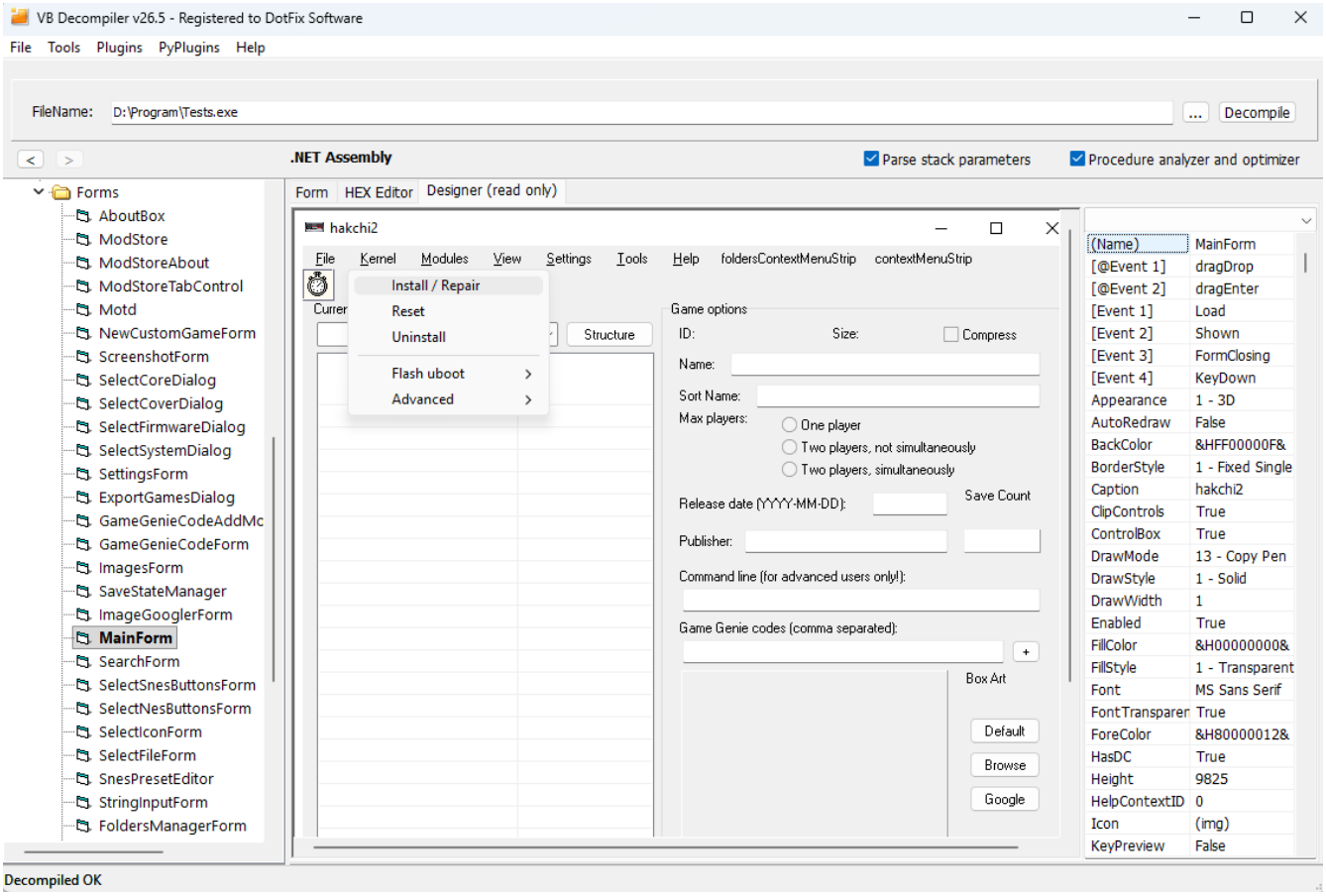


.NET Windows Forms

VB Decompiler 静态分析 `InitializeComponent` 并重构受支持的 Windows Forms 控件及其布局。解析器和渲染器不需要 .NET 运行时或目标应用程序的执行。支持的恢复图像也可以在控件和窗体背景上显示。

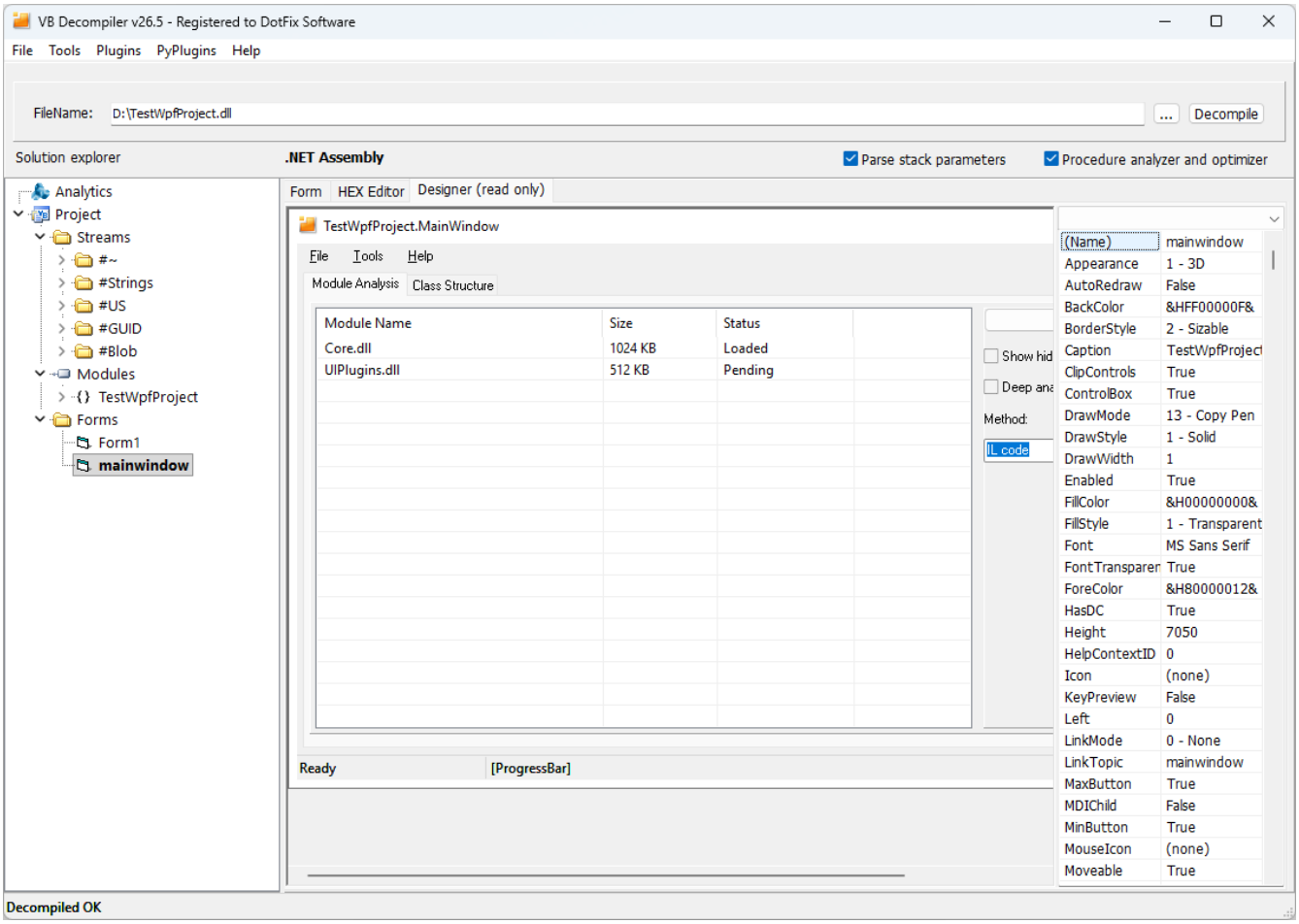
选择控件以检查其恢复的事件，然后双击事件以打开相应的事件处理程序。查看器重构菜单和上下文菜单层次结构；单击恢复的菜单项会打开其相应的 Click 事件。自定义 Windows Forms 事件带有 @ 前缀显示。

查看器处理控件 Z-order、继承的前景和背景颜色、`Color.FromArgb` 值、数值 `FontStyle` 标志、布尔资源属性以及 `MinimumSize` 与 `ClientSize`。



WPF 窗口、页面和用户控件

查看器从作为 **BAML** 存储的编译 XAML 资源中重构受支持的 WPF 窗口、页面和用户控件。



支持的布局容器包括 **Grid**（具有固定、自动和比例 (*) 的行和列大小）、**StackPanel**、**DockPanel** 和 **Canvas**，包括嵌套布局。重构考虑了支持的大小、边距、内边距、对齐、方向、停靠方式以及 Grid 行列跨度。

除了常见的输入控件外，查看器还可以重构选项卡、分组框、展开面板、菜单层次结构和分隔符、状态栏、ListView 列和支持的静态数据、静态 TreeView 节点以及静态组合框项目。Ribbon 界面、导航面板和停靠面板可以有简化的表示。

在资源数据可用的情况下，查看器恢复名称、可见文本、工具提示、字体、背景颜色、启用和可见状态以及嵌入图像，包括支持的 PNG 透明度和背景不透明度。不支持的自定义控件可以显示为标识原始 WPF 类型的占位符。

WPF 查看器将恢复的控件事件与其处理程序关联起来。使用可视化查看器中的事件信息，即可打开对应的反编译代码。

重建限制： 可视化预览是近似的。模板、样式、绑定、自定义控件和动态生成的内容可能仅部分表示。

检查和提取窗体数据

有关Show offsets 设置及相关二进制数据检查，请参阅 窗体。使用 FRX 图像查看器插件 检查和导出支持的恢复图像资源。

11 跟踪

Tracing features 提供 Visual Basic 5.0/6.0 Native Code 和 P-Code 的逐步模拟功能。使用此功能需要包含跟踪功能的许可证。跟踪器处理的是模拟器中过程的内部表示，并不在工作站的 CPU 上执行目标过程。

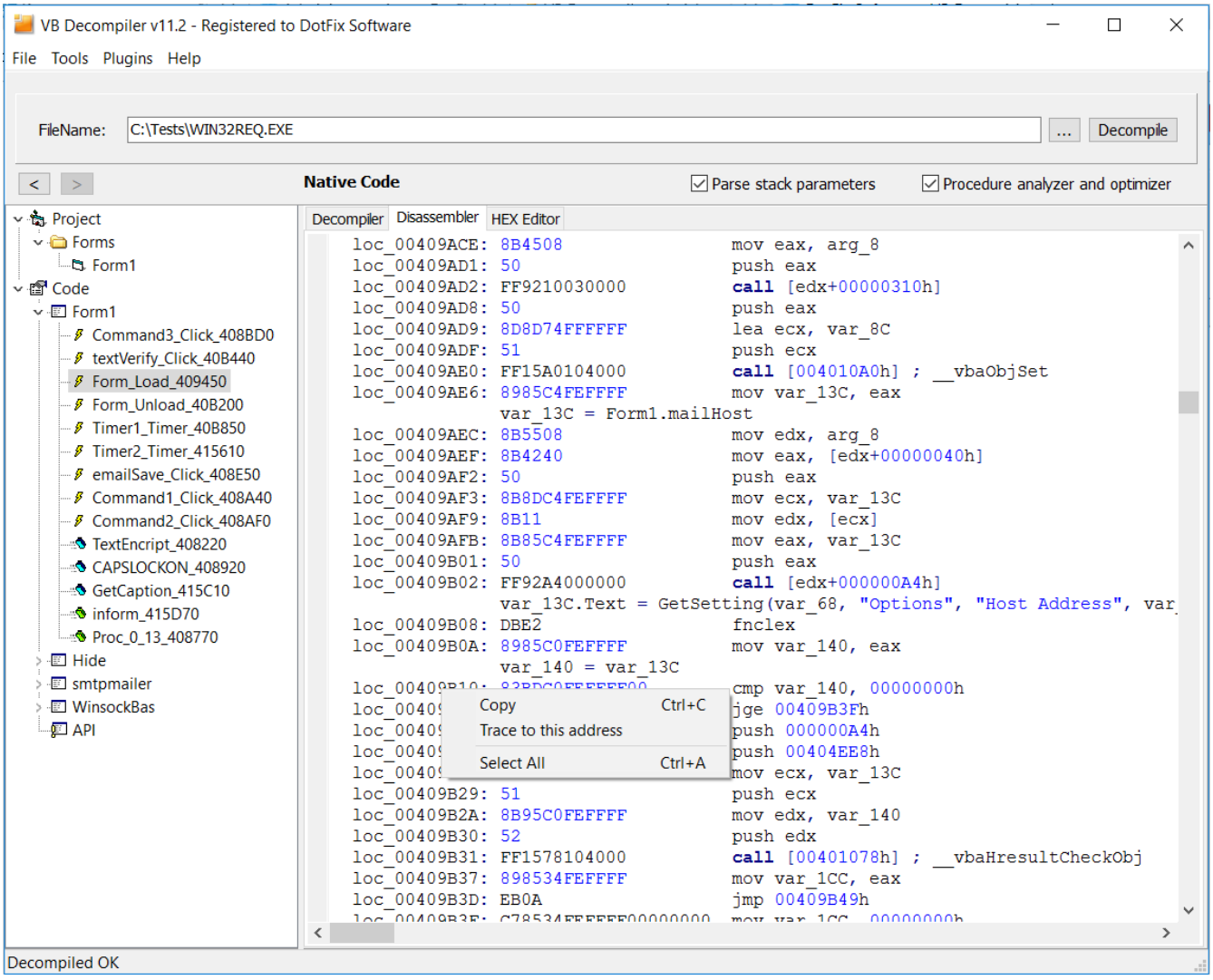
启动跟踪

1 打开一个 VB5/6 过程，并在列表中保持地址可见。

2 在要调查的行上，从代码上下文菜单中调用 **跟踪至此地址 (Trace to this address)**，或按 **F4**。

3 检查跟踪器的代码和状态显示，并使用 **步进 (Next)** 进行推进。完成后取消跟踪。

从版本 26.4 开始，该命令可以在修剪行首空白后解析前导 `loc_` 地址。您可以将光标放置在该行的其他位置，而无需精确地放在其地址标签上。



Native Code 状态

随着模拟的推进，跟踪器会显示 CPU 和 FPU 寄存器、堆栈、变量以及重构的代码。使用这些视图来检查值或表达式是如何推导出来的。对于较大的过程，变量列表搜索和显示当前对象的恢复类型会有所帮助。

P-Code 跟踪

P-Code 跟踪（在版本 11 中引入）显示了变量信息和反编译语句在处理过程中的发展情况。它有助于解释中间 P-Code 操作与优化后的高级代码清单之间的差异。

范围与限制

- 跟踪针对单个过程进行。开始时，模拟寄存器、堆栈和变量均为空状态，而不是某个正在运行的进程的状态。
- 不支持的 Native Code 指令可能会被跳过。不执行外部函数调用；支持的调用约定处理可以调整模拟的堆栈，而无需重现外部函数的行为。
- 跟踪器不提供完整的处理器标志显示。不要将其状态解释为完整的硬件执行跟踪。
- 跟踪采用向前的逐步模拟，不包括反向跟踪或循环迭代的重复执行。
- 此工作流程描述了 VB5/6 的跟踪。.NET 反汇编的存在并不意味着存在 .NET 运行时调试器。

使用跟踪来调查反编译器的重构和中间值。仅凭模拟得到的值，不能确定具有已初始化全局变量、外部依赖项和输入数据的实际进程会产生什么值。

12 分析功能

持有相应许可证时，**Analytic features** 可为分析程序中识别出的操作生成分类报告。这些报告有助于安全分析师在阅读整个代码库之前确定值得调查的过程。它们支持 Visual Basic 5.0/6.0 应用程序，并从版本 11.7 开始支持 .NET 程序集。

生成报告

1 在 **工具 > 选项 (Tools > Options)** 中启用 **分析原型 (Analyze prototypes)**。

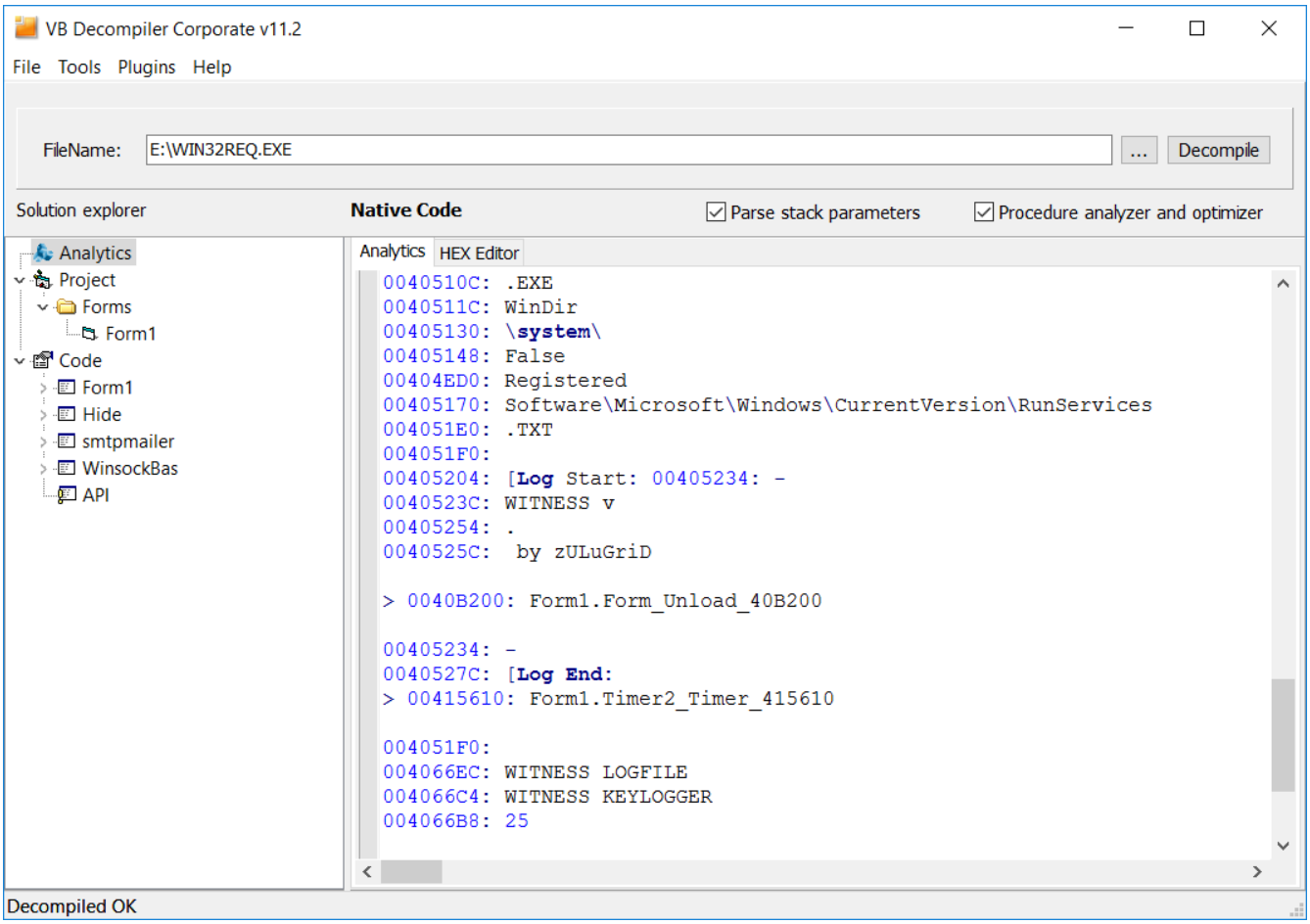
2 打开文件并等待分析完成。使用所需的许可证和设置，报告将自动生成并显示。

3 查看类别，然后打开相关过程并检查其代码和字符串引用。

类别涵盖已识别的操作，例如文件和卷访问、注册表访问、网络活动、进程创建、窗口操作以及选定的 VB 内部函数，如 `CallByName` 和 `VarPtr`。

快速报告和字符串上下文

快速报告列出包含已识别操作的过程的地址和名称。扩展报告添加来自这些过程的可用字符串引用。此上下文可以揭示候选路径、注册表键、主机名或其他值得检查的值。



加密、混淆或在运行时生成的字符串可能不存在。与过程关联的字符串是有用的上下文，但它本身并不能证明它被传递给特定的 API 调用或分支被执行。

导航和保存结果

在报告中双击过程名称以打开其代码。双击过程或字符串前面的地址以在十六进制编辑器中检查其位置。使用对象树上方的 **A** 按钮返回到报告的前一个位置，并使用导航箭头或 **Esc** 重新访问已检查的过程。

根据需要选择和复制报告文本，或选择 **文件 > 保存分析报告 (File > Save analytics report)** 将其保存以供进一步分析。

解释发现

报告列出静态识别出的代码模式和引用。它是用于初步筛查的辅助工具，而不是观察到的执行记录或自动判定文件是恶意的结论。间接调用、未解析的外部接口、损坏的元数据、混淆和运行时生成的值可能会限制覆盖率。在相应的过程、反汇编和文件数据中确认重要发现。

有关 VB5/6 重建的深入检查，请参阅 跟踪。有关字符串字面量及其代码位置，请参阅 字符串引用。

13 混淆限制

VB Decompiler 凭借独特的混淆技术，有效隐藏了我们在对 **Visual Basic 5.0 和 6.0 应用程序** 进行反编译研究时发现的编译器遗留数据。现代编程语言通常会在编译后的文件中存储大量元数据和调试信息，这使得破解者更容易在您的程序中定位所需的功能。为了应对这一威胁，我们专门为 **Visual Basic** 应用程序开发了一种混淆器。

混淆过程详解

该混淆器通过将整个程序分解为其组成部分（包括窗体、模块、类以及按钮和文本框等控件）来工作。随后，每个窗体被进一步分解为带有相关事件的独立控件。接下来，收集有关函数和过程的信息，以及程序中每个对象绑定的事件信息。在此过程中，混淆器检查对象之间的依赖关系，并同步更改它们的名称，使得确定哪个函数属于特定模块或类变得困难。

因此，在程序中找到特定的窗体（如 **frmTrial** 或 **frmRegistration**）将变得显著更加困难。

Visual Basic 5.0/6.0 应用程序的混淆限制

混淆器采用的算法很复杂，旨在分析对象依赖关系，但它仍然不支持所有内容。因此，在某些情况下，您需要事先为混淆过程准备您的程序。下面讨论的要点仅适用于 Visual Basic 5.0 / 6.0 应用程序的混淆。

1. 混淆器不支持对其他对象的设计时引用，这可能导致运行时出现问题。

例如，考虑这样一种情况：您有一个 **TreeView** 控件依赖于关联的 **ImageList**。如果该 **ImageList** 对象的名称硬编码到您的 **TreeView** 属性中，任何混淆尝试都会导致错误，因为原始引用保持不变。此限制也适用于数据库相关组件，例如 **DataSets**

```
Begin ComctlLib.TreeView TreeView1
  Height          =   1932
  Left            =   3600
  TabIndex        =   3
  Top             =   1320
  Width           =   2412
  _ExtentX        =   4255
  _ExtentY        =   3408
  _Version        =   327682
  Style           =   7
  ImageList       =   "ImageList1"
  Appearance      =   1
End
```

此问题可以通过将设计时的赋值移至运行时轻松解决，只需将这些赋值移至 **Form_Load** 事件即可：

```
Private Sub Form_Load()
  TreeView1.ImageList = ImageList1
End Sub
```

对于每个相同实例，都需要应用相同的修正。幸运的是，大多数项目中通常只有少数此类情况。

2. 不支持通过 As New 声明对象（LateCall 调用）

此限制通常不会影响大多数 VB6 项目，但必须牢记这一点。如果您使用窗体作为类并创建窗体实例，可能会熟悉以下场景：

```
Private Sub Form_Load()
  Dim oForm as New frmMain
  oForm.Show
End Sub
```

在这种情况下，VB6 编译器会将名称 **frmMain** 直接插入到 **Form_Load** 函数的代码中。由于我们的混淆器仅修改数据而不修改代码，因此混淆后，嵌入代码中的名称仍保持不变。不幸的是，这会导致程序终止并报错。

很遗憾，我们无法在 **VB Decompiler** 的混淆器中支持此类调用，因为我们需要反汇编和反编译所有函数的整个代码。尽管具备此能力，但反编译是一个极其耗时的过程。对于大型项目，仅分解整个代码库就需要超过半小时。此外，我们还需要分析对象引用并重命名它们，这会增加更多的处理时间。

鉴于这些技术限制，我们决定不在 **VB Decompiler** 的混淆器中实现完整的反编译和代码分析，尤其是因为这只会解决 Visual Basic 5.0 / 6.0 应用程序中极少见的情况，并且不会为大多数客户提供任何好处。

14 技术支持

如果您对本程序有疑问且无法在本帮助指南中找到答案，请联系我们的技术支持团队。此选项仅对已注册用户可用。对于未注册用户，我们很乐意协助您咨询购买**VB Decompiler**许可证的相关事宜。

此外，如果您在程序中遇到任何错误或缺陷，或有关于如何改进本程序的建议，欢迎随时联系我们的技术支持服务并提供您的反馈。

15 注册

要使用**VB Decompiler**的全部功能，您需要购买**Business License**。通过购买单开发者商业许可证或企业许可证，您将获得以下能力：

1. 对于典型的受支持 **Visual Basic 5.0/6.0 P-Code** 应用程序，恢复接近 85% 的代码;
2. 对于典型的受支持 **Visual Basic 5.0/6.0 Native Code** 应用程序，恢复最多 85% 的代码;
3. 将典型的受支持且未混淆的 **.NET** 程序集反编译为类似 **C#** 的代码，恢复率最高可达 95%;
4. 始终拥有最新版本：购买许可证后一年内更新完全免费；
5. 在购买许可证后一年内通过电子邮件接收技术支持；
6. 支持该反编译器的开发。

此外，我们建议考虑为恶意软件分析师和取证购买特殊许可证：

1. 带有跟踪功能的许可证
2. 带有分析功能的许可证

要立即购买反编译器的**许可证**，请在程序的**帮助 (Help)**菜单中选择**订购专业版本 (Order Pro version)**选项。此时，您必须连接到互联网。将打开浏览器窗口以访问程序网站，您可以在该网站上购买许可证。

反编译器的**Business License**仅允许在单台计算机上由**Single Developer**（许可证所有者）使用。如果您需要在多台计算机上使用程序，则需要为每台计算机购买单独的许可证，或考虑购买**Business Enterprise license**。（可在多达20台计算机上使用）。

16 关于开发者

本反编译器由 **DotFix Software** 开发。

产品网站: www.vb-decompiler.org
技术支持邮箱: admin@vb-decompiler.org