VB Decompiler

Manual del usuario

Español

Visual Basic 5.0/6.0 · .NET

DotFix Software

Contenido

1 Descripción 4

2 Inicio rápido 5

3 Ventana principal 6

4 Visualización del código 8

5 Mejora del código con IA 11

6 Apertura y guardado 15

6.1 Abrir un archivo 15

6.2 Trabajo con bases de datos de análisis 16

6.3 Guardar la lista de procedimientos 17

7 Funciones del programa 18

7.1 Ir a una dirección 18

7.2 Búsqueda de cadenas 19

7.3 Referencias a cadenas 20

7.4 Descompilación desde una dirección 22

7.5 Modificación de datos 23

7.6 Ofuscación 25

8 Opciones del programa 26

8.1 Opciones del programa 26

8.2 Formularios 27

8.3 Descompilación 28

8.4 Desensamblador 30

8.5 Esquema de colores 31

8.6 Funciones avanzadas 33

8.7 Selección de idioma 37

9 Plugins 38

10 Reconstrucción visual de formularios 42

11 Trazado 46

12 Funciones de análisis 48

13 Limitaciones de la ofuscación 50

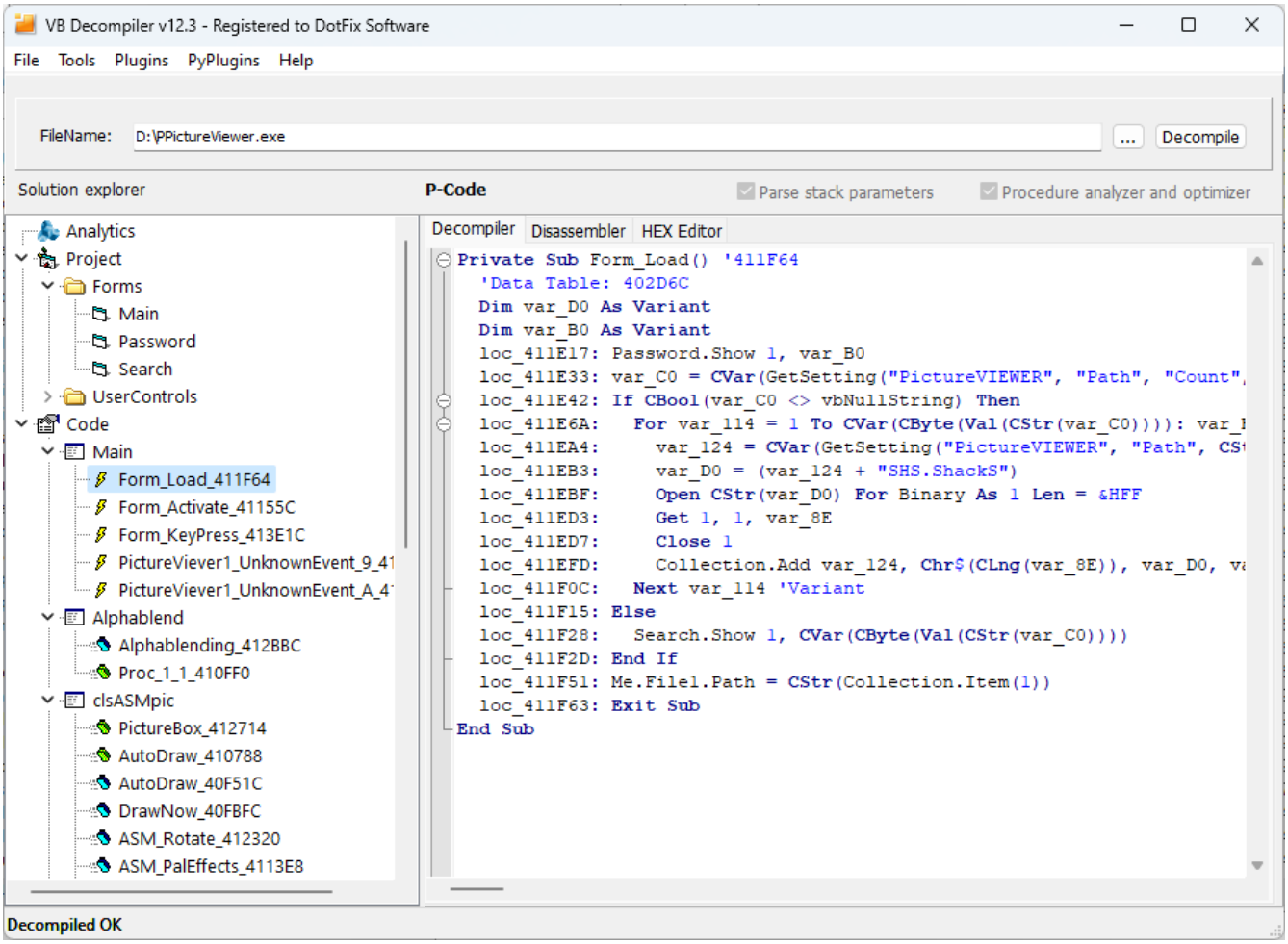
14 Soporte técnico 52

15 Registro 52

16 Acerca del desarrollador 53

1 Descripción

VB Decompiler es una herramienta de ingeniería inversa y recuperación del código fuente para programas Visual Basic 5.0/6.0 y ensamblados .NET. Combina descompilación de alto nivel, desensamblado de bajo nivel, inspección de metadatos, un editor HEX integrado y reconstrucción de la interfaz visual. Los usos típicos incluyen mantenimiento de software heredado, respuesta a incidentes, análisis de malware e investigación del comportamiento no documentado de aplicaciones.



Representaciones de código admitidas

- **VB5/6 P-Code:** reconstruye sentencias de Visual Basic a partir de las instrucciones P-Code del compilador y las operaciones del entorno de ejecución.
- **VB5/6 Native Code:** desensambla instrucciones x86 y utiliza un emulador interno, firmas de tiempo de ejecución y heurísticas para reconstruir código similar a Visual Basic.
- **.NET:** desensambla el Common Intermediate Language (CIL, también llamado MS IL) y descompila patrones IL admitidos en código similar a C#, independientemente del lenguaje fuente .NET original. También se puede desensamblar código nativo x86 y x64 en ensamblados .NET.

El motor de análisis .NET no requiere el entorno de ejecución .NET ni la reflexión para el análisis de código o reconstrucción de formularios visuales.

Herramientas de análisis

Utilice búsqueda de procedimientos y texto, referencias cruzadas y renombrado de variables, y Referencias a cadenas para investigar el comportamiento. El visor reconstruye formularios VB clásicos, Windows Forms .NET y ventanas, páginas y controles de usuario WPF almacenados en recursos BAML compilados.

Las licencias con características de trazado proporcionan emulación paso a paso del código nativo VB5/6 y P-Code. Las licencias con características analíticas proporcionan informes categorizados para aplicaciones VB5/6 y .NET. Ayuda IA (AI Helper) puede procesar código VB5/6 o C# recuperado utilizando la integración de Ollama o la API de OpenAI.

Interpretación del código recuperado

La compilación elimina información como comentarios, muchos nombres de variables locales y algunos metadatos de tipo y procedimiento. La optimización y ofuscación pueden cambiar aún más la estructura del código. El listado reconstruido admite análisis y recuperación de código. Compruebe las conclusiones importantes mediante el desensamblaje, los metadatos, las referencias a cadenas y los procedimientos circundantes.

2 Inicio rápido

1. Seleccione **Archivo > Abrir programa (File > Open program)** y abra el archivo EXE, DLL o OCX para analizar. El procesamiento comienza automáticamente. Espere a que se llene el árbol del proyecto y termine el procesamiento.
2. Abra un procedimiento en **Explorador (Solution explorer)**. Compare las vistas **Decompiler** y **Desensamblador (Disassembler)**; utilice **HEX Editor** para inspeccionar los bytes correspondientes.
3. Pulse **Ctrl+T** para encontrar un procedimiento por su nombre o parte de este. Utilice **Ctrl+F** para buscar texto mostrado, o **Herramientas > Referencia a cadenas (Tools > String references)** para investigar cadenas y los procedimientos que las utilizan.
4. Abra un formulario recuperado y seleccione **Diseño (lectura) (Designer (read only))** para una vista previa visual. Para VB clásico, Windows Forms y WPF, utilice eventos recuperados para navegar al código del manejador correspondiente.
5. Guarde el análisis a través de **Archivo > Base de datos (File > Data Base)** cuando necesite reanudar más tarde o compartir los resultados con otro analista.

Elegir un modo de descompilación

La descompilación rápida procesa procedimientos cuando se abren. Active **Anál. prototipos (Analyze prototypes)** para un análisis preliminar adicional de las firmas de procedimientos privados. Para buscar todo el texto descompilado o exportar un proyecto completo, desactive la Descompilación Rápida y procese el archivo nuevamente.

Guardar código fuente recuperado

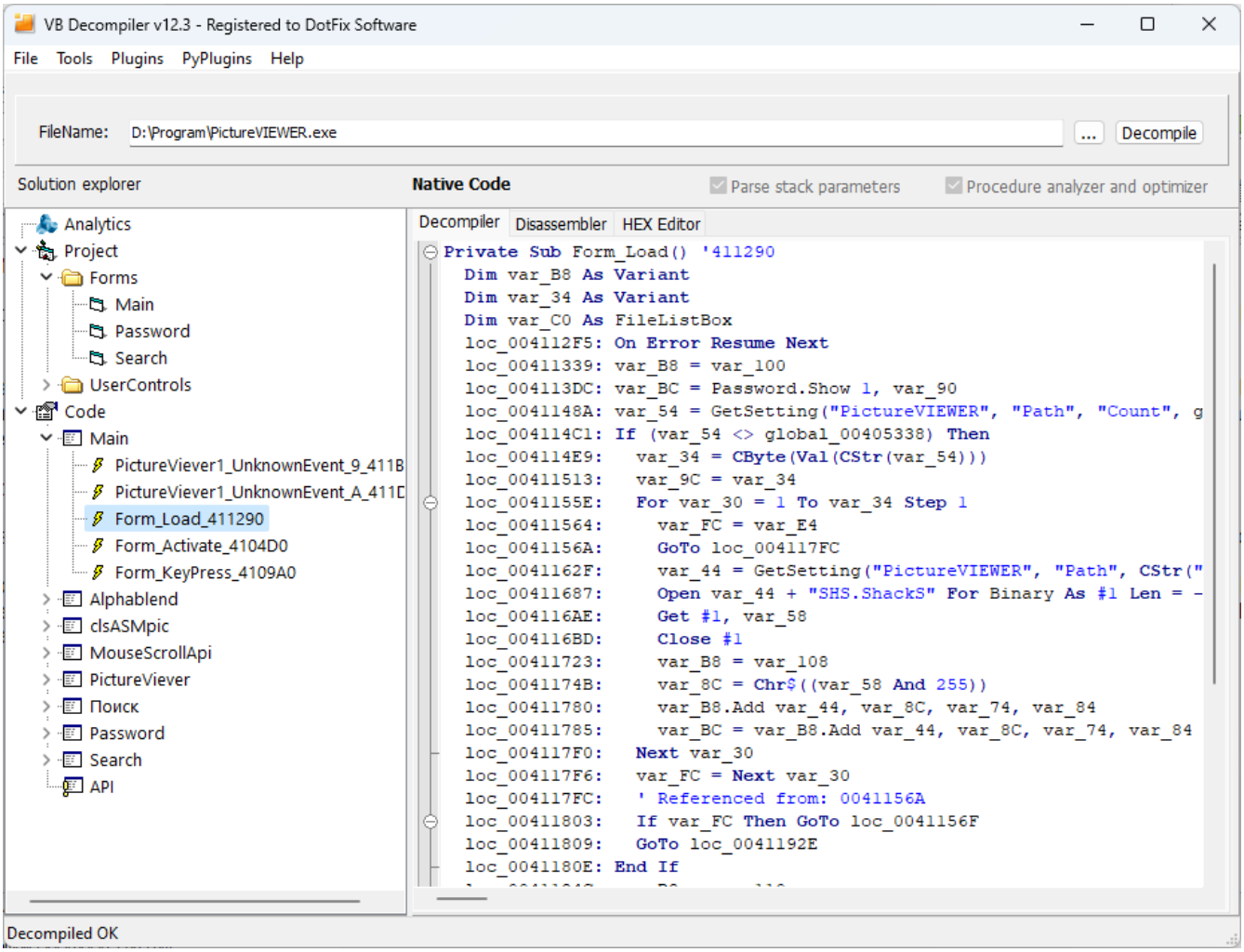
Para exportar los resultados de la descompilación VB5/6, seleccione **Archivo > Grabar proyecto decompilado (File > Save decompiled project)** y elija una carpeta de salida. La exportación incluye formularios y módulos recuperados; los archivos FRX contienen imágenes asociadas y otros datos binarios del formulario.

Antes de abrir el código exportado en un IDE, corrija manualmente todas las imprecisiones de la descompilación y las incompatibilidades del código fuente recuperado.

Vea Reconstrucción visual de formularios, Visualización de código y Mejora del código con IA para los flujos de trabajo detallados.

3 Ventana principal

Explorador (Solution explorer) organiza el proyecto recuperado en formularios, módulos, clases y procedimientos. La vista adyacente muestra el código, formulario o metadatos seleccionados. La opción **Estilo de ventana Visual Studio (Window style as in Visual Studio)** coloca el árbol a la derecha y el código a la izquierda.



Vistas de código y datos

- **Decompiler:** código reconstruido de alto nivel.
- **Desensamblador (Disassembler):** las instrucciones x86/x64 correspondientes, P-Code de VB o IL de .NET, con direcciones.
- **HEX Editor:** los bytes subyacentes del archivo, con navegación por dirección virtual o desplazamiento en el archivo.
- **Diseño (lectura) (Designer (read only)):** un formulario reconstruido y sus propiedades recuperadas, cuando es posible.
- **Ayuda IA (AI Helper):** procesamiento de IA opcional del procedimiento actualmente descompilado.

Las pestañas disponibles dependen del objeto seleccionado. El indicador de tipo de código distingue **P-Code**, **Native Code** y **.NET Assembly**.

Analizar parámetros pila (Parse stack parameters)

Para P-Code, este control combina operaciones de pila y llamadas en expresiones con argumentos. Desactivarlo expone las operaciones intermedias y puede ayudar cuando un prototipo desconocido consume valores inesperados de la pila. Para código nativo, activarlo utiliza el emulador para reconstruir expresiones de nivel superior; desactivarlo expone la salida a nivel de ensamblaje. Para .NET, el procesamiento de la pila contribuye a la reconstrucción de IL como expresiones similares a C#.

Optimizador de procedimiento (Procedure analyzer and optimizer)

Este control simplifica la lista reconstruida combinando expresiones y patrones reconocidos de flujo de control y eliminando el código auxiliar del compilador que se ha reconocido. Se aplica al código VB5/6 y .NET. El optimizador de .NET puede reconstruir patrones `while`, `do...while` y `foreach` soportados, condiciones anidadas, expresiones de operadores y patrones de controladores de cadenas interpoladas. Estas transformaciones son realizadas por el descompilador en sí mismo y no requieren el Asistente de IA.

Si una expresión o rama es poco clara, compare la salida con la optimización desactivada y examine el código desensamblado. En el modo **Decompilación rápida (Fast decompilation)** sin caché, estos controles pueden cambiarse mientras se inspecciona un procedimiento. Con descompilación completa o salida en caché, procese el archivo nuevamente después de cambiar la configuración.

Navegación y acceso por teclado

Los botones de flecha sobre el árbol navegan entre procedimientos previamente vistos; **Esc** regresa al procedimiento anterior. Los botones Atrás y Adelante del ratón utilizan la misma historia, incluida la posición recordada en un procedimiento.

Pulse **Alt+1** para cambiar el foco entre Explorador y la vista activa. Pulse **Enter** en el árbol para abrir el objeto seleccionado. En el Editor HEX, **Tab** se mueve entre los campos hexadecimal y de texto y luego al siguiente control. Las etiquetas de la interfaz están asociadas con sus campos de entrada para la navegación de lectores de pantalla.

Vea Visualización de código para atajos contextualizados, referencias cruzadas y renombrado, y Funciones de análisis para la vista Análisis (Analytics).

4 Visualización del código

Las vistas de código combinan resaltado de sintaxis, bloques colapsables, navegación basada en direcciones y referencias cruzadas. Utilice el listado de alto nivel para comprender un procedimiento y el desensamblaje y los bytes del archivo para verificar cómo se obtuvo ese resultado.

Resaltado y navegación de identificadores

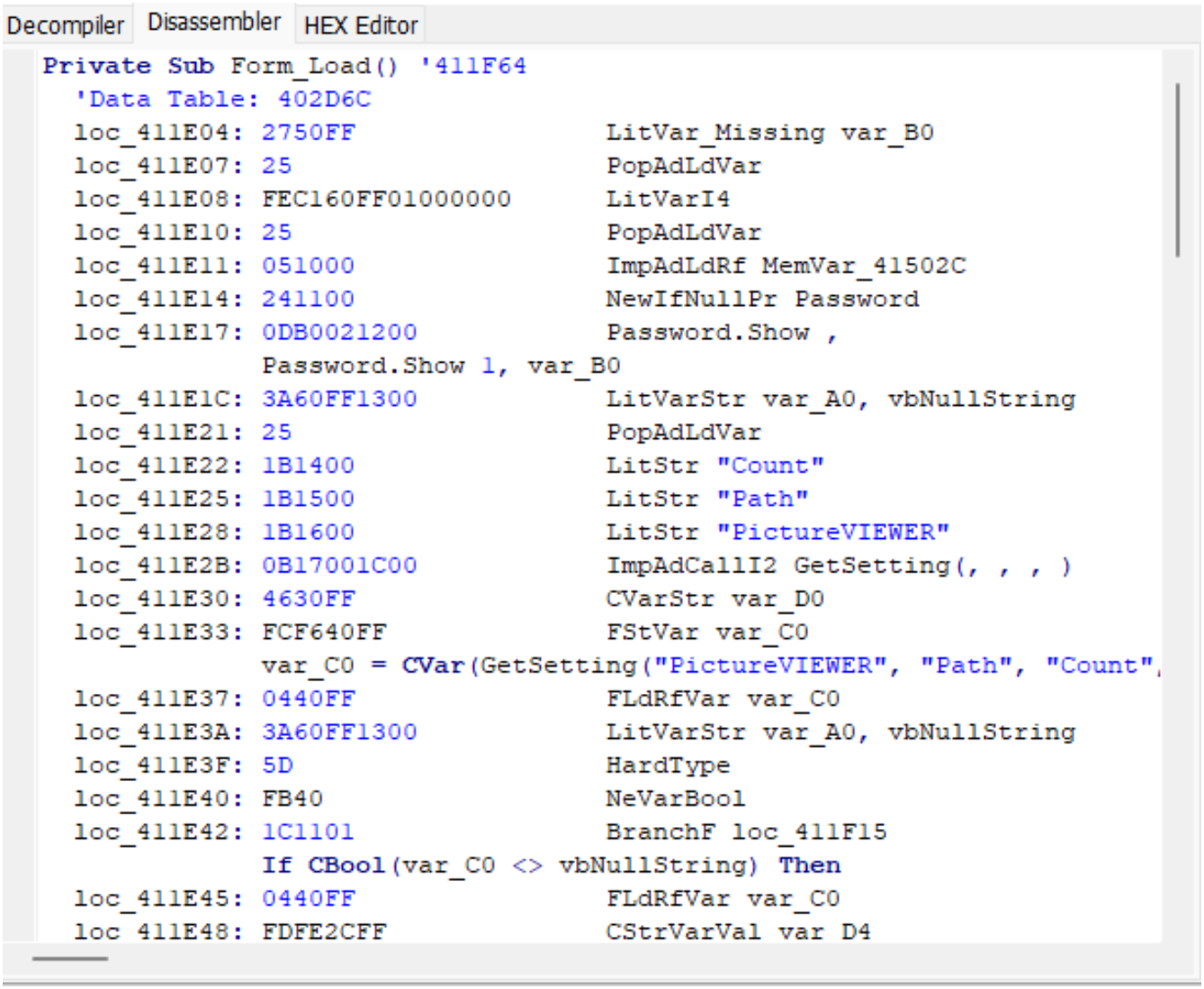
Coloque el cursor sobre un registro, variable o nombre de función para resaltar las ocurrencias coincidentes en el listado actual. Las variables locales como `var_XX` y `str_XX` también se resaltan cuando se utilizan como base de una expresión de miembro, por ejemplo `str_1.Value`. Seleccionar un identificador `global_XX` resalta sus ocurrencias dentro del procedimiento actual; este resaltado es independiente de la búsqueda de referencias en todo el proyecto.

Haga doble clic en una llamada a procedimiento reconocida para abrir el procedimiento llamado. Las constantes similares a direcciones pueden, en su lugar, llevar a los bytes correspondientes en el Editor HEX. Utilice **Esc**, las flechas de navegación o los botones Atrás/Adelante del ratón para moverse a través del historial de visualización.

Combinación de código descompilado y desensamblado

Haga clic derecho en el listado descompilado y seleccione **Al desensamblador (Copy to disassembler)**, o presione **D** en la vista de código. Esto fusiona las instrucciones reconstruidas con el listado de bajo nivel para el procedimiento actual. Para preservar su posición, coloque el cursor en la dirección relevante `loc_` antes de invocar el comando.

Mantenga habilitado **Add Addresses** en las configuraciones de descompilación al utilizar este flujo de trabajo. El Desensamblador siempre muestra direcciones; Add Addresses controla la vista Decompiler.



Renombrado de variables

Para VB5/6 P-Code y Native Code, haga clic derecho en una variable y seleccione **Renombrar (Rename variable)**, o coloque el cursor sobre la variable y presione **N**. Introduzca un nombre de análisis significativo. El descompilador verifica colisiones de nombres y palabras clave reservadas. Para revertir un renombrado, utilice **Deshacer nom. (Undo renaming)** o **Ctrl+Shift+Z**. Este comando es independiente del deshacer de edición de texto ordinaria.

El renombrado hace que el listado recuperado sea más fácil de analizar; no recupera un nombre que fue eliminado por la compilación. Trate los nombres asignados por el analista como anotaciones al comunicar hallazgos.

Búsqueda de llamantes y referencias a variables globales

Para encontrar llamantes, haga clic derecho en un procedimiento en Explorador (Solution explorer) y seleccione **Buscar refer. (Find references)**, o presione **X** con ese procedimiento seleccionado en el árbol. Los resultados listan los procedimientos que contienen llamadas reconocidas a él. Haga doble clic en un resultado para inspeccionar al llamante. Este flujo de trabajo soporta VB5/6 y .NET; el soporte para .NET se añadió en la versión 12.4.

Para investigar una variable global, coloque el cursor sobre su identificador en la vista de código y presione **X**. La misma tecla, por lo tanto, busca referencias a procedimientos en el árbol y referencias a variables globales en el código. Una lista de referencias reconocidas no establece que no haya usos adicionales indirectos, dinámicamente resueltos u ofuscados.

Atajos de teclado sensibles al contexto

Enfoque	Atajo	Acción
Código	N	Renombrar la variable bajo el cursor (VB5/6).
Código	X	Buscar referencias a la variable global bajo el cursor.
Código	D	Copiar el código descompilado actual al Desensamblador.
Código	F4	Rastrear a la dirección en la línea actual; requiere funciones de trazado.
Código	Ctrl+Shift+Z	Deshacer el renombrado de la variable bajo el cursor.
Árbol	X	Buscar referencias al procedimiento seleccionado.
Árbol	G	Abrir el diálogo para ir a una sección.
Árbol	Ctrl+C	Copiar la ruta completa del árbol del objeto seleccionado.
Árbol	Space or Insert	Marcar o desmarcar el elemento seleccionado.

Los atajos de una letra dependen de la vista enfocada y no reemplazan la escritura normal en campos de entrada. Utilice **Alt+1** para cambiar el enfoque. Consulte Búsqueda de cadenas para Ctrl+T y Ctrl+F, Ir a dirección para rutas del árbol y direcciones, y Trazado para limitaciones del emulador.

5 Mejora del código con IA

Ayuda IA (AI Helper) procesa el código que VB Decompiler ha reconstruido ya. Soporta C# descompilado de ensamblados .NET y código Visual Basic 5.0/6.0 desde P-Code o Código Nativo. Puede utilizar un modelo local a través de Ollama o conectarse a un servicio mediante la integración de la API de OpenAI.

La IA puede sugerir flujos de control más claros, nombres de variables, declaraciones y expresiones. No puede restaurar fiablemente información ausente del binario ni reparar un fragmento que el descompilador haya interpretado incorrectamente. Compare las sugerencias con el listado descompilado original, el desensamblaje y los procedimientos circundantes antes de utilizarlas en el código fuente recuperado o en un informe de investigación.

Procesamiento de un procedimiento

1. Abrir el procedimiento requerido en el Explorador (Solution explorer) y revisar su listado en **Decompiler**.
2. Cambiar a **Ayuda IA** y configurar el proveedor, el host y el modelo como se describe a continuación.
3. Iniciar el procesamiento con el botón de acción junto al selector del modelo.
4. Revisar el código devuelto en Ayuda IA y compararlo con la salida del descompilador.

El procesamiento con IA se ejecuta en un hilo separado. Utilice **Cancelar (Cancel)** si el modelo tarda demasiado en responder. Las opciones adicionales de conexión pueden contraerse para dejar más espacio para el código.

Utilizar un modelo local a través de Ollama

1. Instalar Ollama desde su página de descarga oficial e iniciarla.
2. Descargar el modelo recomendado ejecutando `ollama pull qwen3:8b` en una terminal.
3. Seleccionar el proveedor local LLM/Ollama en Ayuda IA. Establecer **Host Ollama (Ollama host)** en la dirección del servidor; la dirección local predeterminada es `http://localhost:11434`.
4. Seleccionar el modelo instalado en **Mod. (Model)** e iniciar el procesamiento.

Recomendamos `qwen3:8b`. Los prompts integrados de Ayuda IA están adaptados a este modelo ligero. Los modelos más recientes pueden tardar más en procesar el código sin mejorar el resultado.

La selección de modelos no está restringida: puede utilizar cualquier modelo instalado en la instancia de Ollama seleccionada. Si el formato de respuesta no se reconoce, el analizador puede dejar etiquetas propias del modelo u otro texto auxiliar en el resultado mostrado.

Hardware para la ejecución local con Ollama

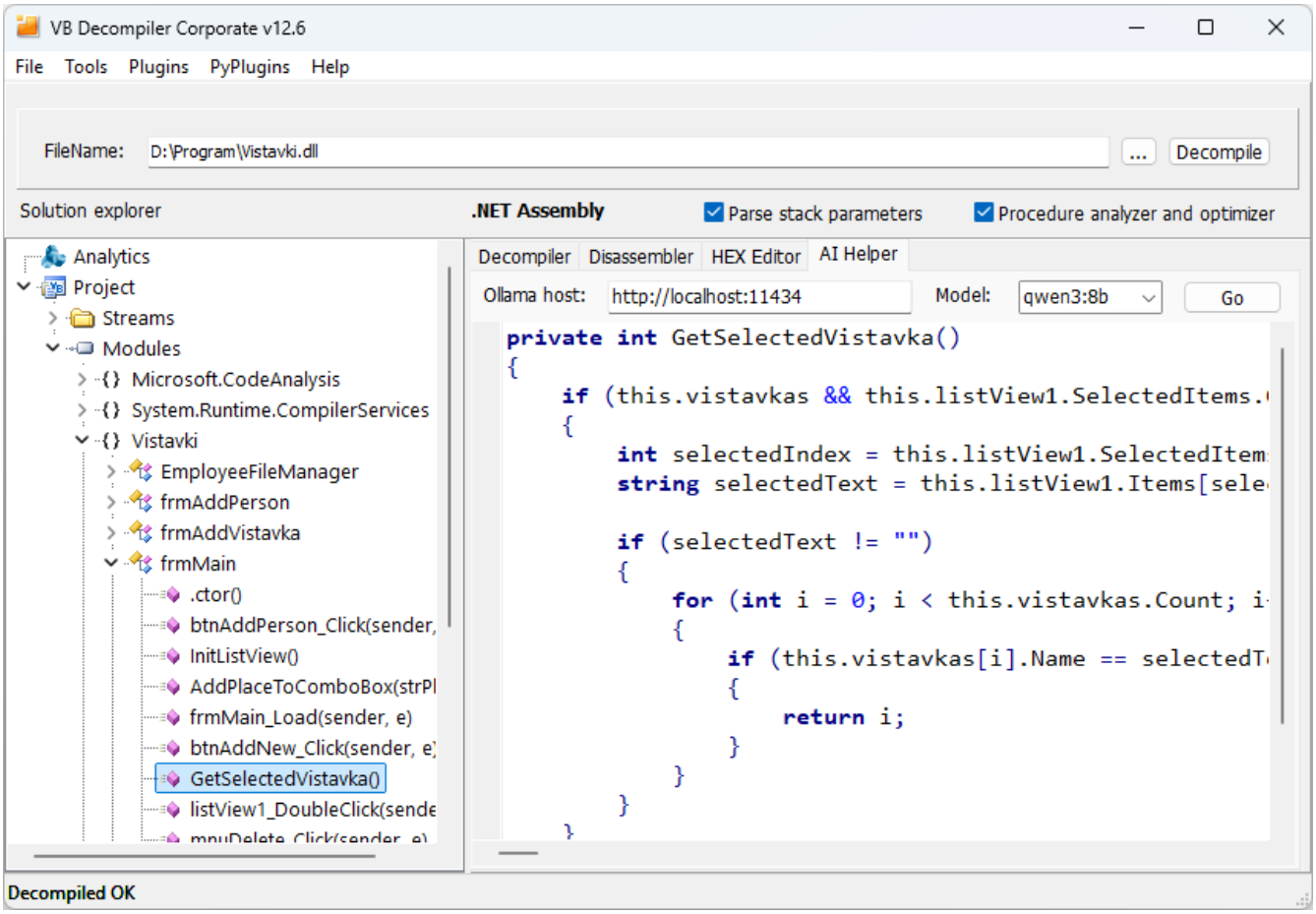
El procesamiento local con IA se realiza en el equipo que ejecuta Ollama y requiere muchos más recursos que la descompilación. Las siguientes configuraciones prácticas suponen `qwen3:8b` con pesos `Q4_K_M`, caché de contexto estándar `f16` y una solicitud a la vez. El contexto se mide en tokens: 4K son 4096 tokens y 128K son 131072 tokens. El límite incluye las instrucciones, el código fuente y la respuesta generada.

Componente	Mínimo práctico: contexto 4K	Recomendado: contexto 128K
Sistema operativo	Windows 10 (22H2) o Windows 11 de 64 bits	Windows 10 (22H2) o Windows 11 de 64 bits
Procesador	CPU x64 moderna con AVX2 y al menos 4 núcleos físicos	CPU x64 moderna con AVX2 y al menos 8 núcleos físicos
Memoria del sistema (RAM)	16 GB	64 GB
Tarjeta gráfica	GPU compatible con Ollama con al menos 8 GB de VRAM dedicada	GPU compatible con Ollama con al menos 32 GB de VRAM dedicada
Almacenamiento	SSD con al menos 20 GB libres para Ollama y este modelo	SSD con al menos 20 GB libres para Ollama y este modelo

Seleccione el contexto con el control deslizante de los ajustes de Ollama. Una tarjeta de 8 GB adecuada para 4K no basta para la configuración de 128K indicada arriba. La descarga del modelo ocupa unos 5.2 GB, pero el consumo total de memoria es mayor. A 128K, solo la caché `f16` requiere aproximadamente 18 GiB, además de los pesos y los búferes de trabajo. Deje memoria disponible para Windows y otras aplicaciones.

Con suficiente RAM, Ollama puede utilizar la CPU, pero el procesamiento del código puede ser muy lento. Los sistemas Pentium M y otros equipos de antigüedad similar no son adecuados para ejecutar `qwen3:8b` localmente. La memoria compartida de las antiguas Intel HD Graphics no sustituye a la VRAM dedicada indicada arriba. Estos requisitos corresponden al equipo con Ollama; un servidor remoto o un servicio API externo utiliza su propio hardware.

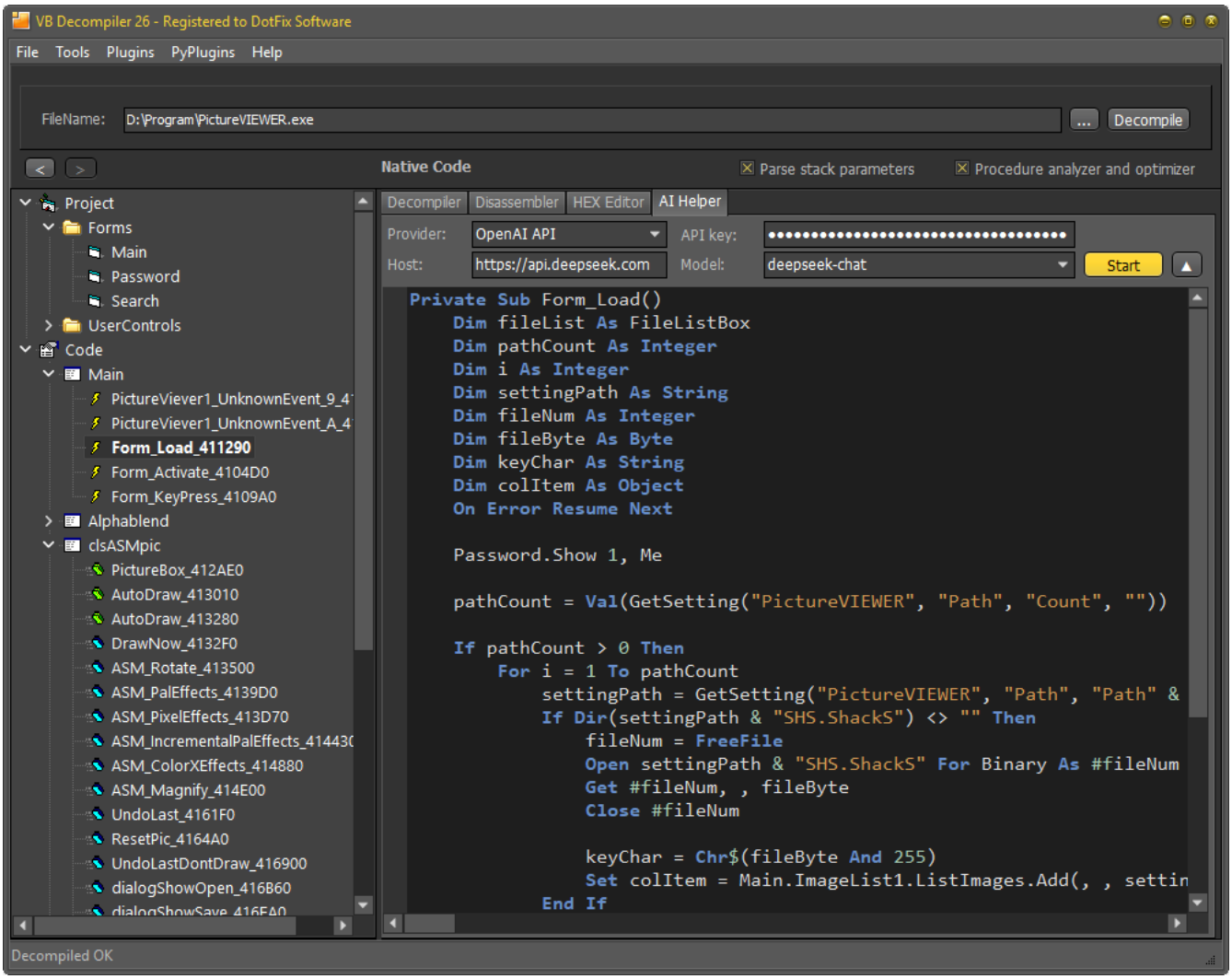
Para procedimientos cortos, empiece con 4K. Un contexto mayor aumenta el consumo de memoria y el tiempo de procesamiento de entradas largas. Si el procesamiento es demasiado lento, cancélelo, reduzca el contexto y compruebe el uso de la GPU con `ollama ps`. El consumo real depende de la versión de Ollama, el backend y las demás tareas en ejecución.



Utilizar la integración de la API de OpenAI

Establecer **Servicio (Provider)** en **OpenAI API**. Configurar **Host**, **CI. API (API key)** y **Mod.** para el servicio previsto. Introduzca una clave API válida y un identificador de modelo disponible para esa clave. Mantenga las claves API fuera de capturas de pantalla y material de análisis compartido.

Ayuda IA también soporta DeepSeek a través del proveedor **OpenAI API**. Introduzca el host de DeepSeek, la clave API y el identificador de modelo en los campos correspondientes. La captura de pantalla a continuación muestra esta configuración.



Dónde se procesa el código

Ayuda IA envía el código descompilado actual al servicio configurado. Para un flujo de trabajo local exclusivo, utilice un servidor Ollama ejecutado localmente y un modelo ejecutado localmente. Un host Ollama en otra máquina, un modelo Ollama en la nube o un servicio API externo pueden transferir el código fuera del puesto de análisis.

Para la recuperación del código fuente confidenciales o investigaciones de seguridad, seleccione el punto final según los requisitos de gestión de datos de su organización. No infiera la operación sin conexión únicamente por la presencia de Ollama o una URL localhost.

Solución de problemas

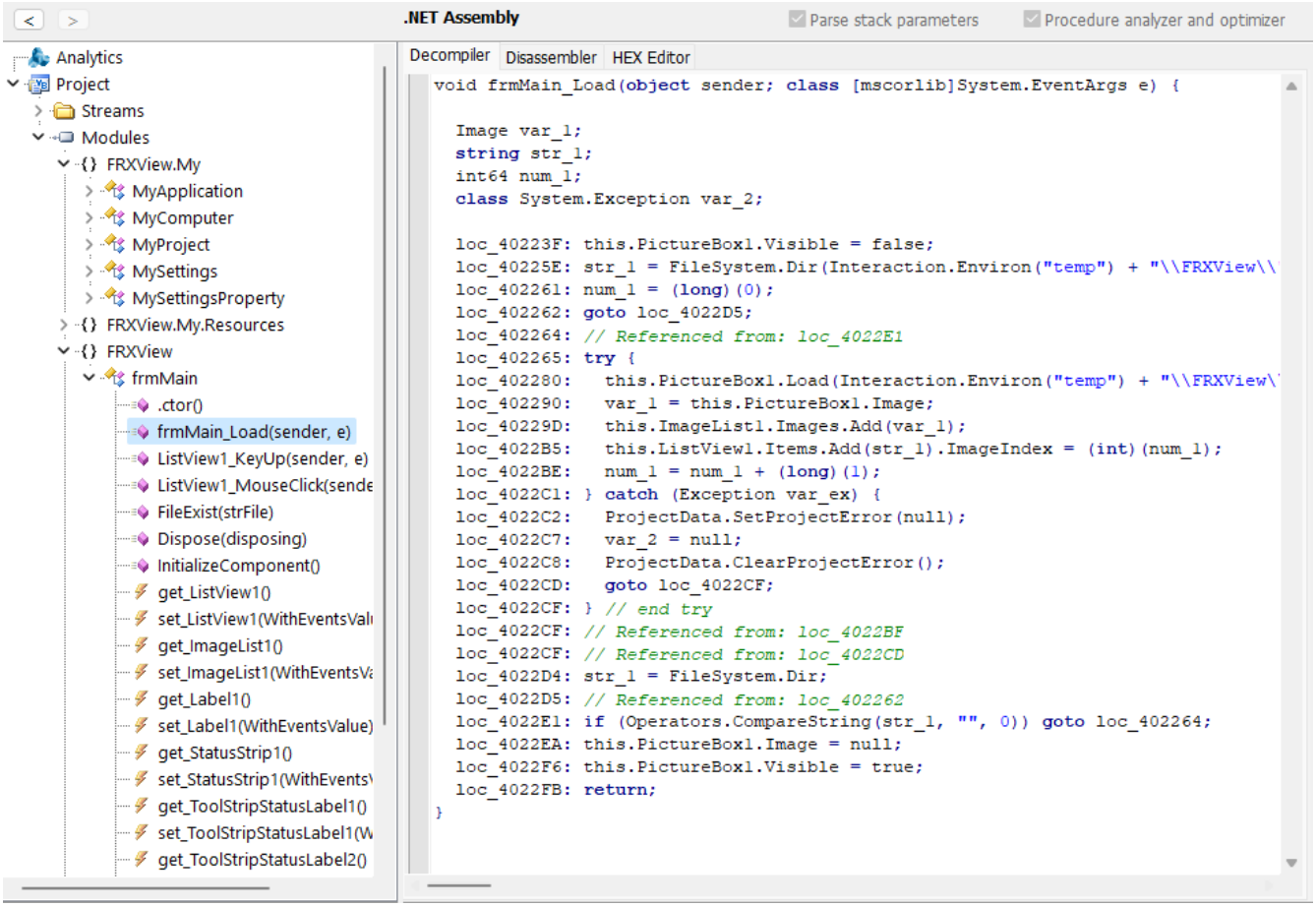
- Si falta un modelo local, confirme que está instalado en la instancia Ollama especificada por el campo host.
- Si no hay respuesta, verifique que el servicio seleccionado esté ejecutándose y sea accesible, y que el punto final, credencial e identificador de modelo correspondan al mismo proveedor.
- Si el procesamiento es demasiado lento, cancele. Al utilizar otro modelo local, pruebe el modelo recomendado `qwen3:8b`.
- Si un modelo desconocido deja etiquetas u otro texto auxiliar en su respuesta, pruebe `qwen3:8b` o elimine ese texto manualmente mientras revisa el resultado.
- Si el resultado cambia una condición, llamada, constante o tipo de manera inesperada, compárelo con el listado de bajo nivel. La legibilidad por sí sola no establece un comportamiento equivalente.

Ollama y los proveedores de modelos externos son productos de terceros. Su disponibilidad, comportamiento del modelo y políticas de servicio son independientes de VB Decompiler.

6 Apertura y guardado

6.1 Abrir un archivo

Antes de iniciar el proceso de descompilación, es necesario seleccionar un archivo para su análisis. Para ello, existe una opción **Abrir programa (Open program)** en el menú **Archivo (File)**. **VB Decompiler** soporta la apertura de archivos EXE/OCX/DLL creados en versiones **Visual Basic** 5.0 o 6.0, así como aplicaciones compiladas con **Visual Studio .NET**. En este último caso, el lenguaje de origen no es relevante, ya que Visual Studio .NET compila las aplicaciones independientemente del lenguaje original en un único formato interno - **MS IL**.



Tras abrir el archivo, el descompilador verifica si el programa pertenece a los entornos de desarrollo especificados. Si el programa abierto fue desarrollado en un entorno no soportado (por ejemplo, **Delphi** o **Visual C++**), se muestra un mensaje indicando que dicho entorno no es compatible y que no es posible descompilar el programa. Si aparece este mensaje, puede contactarnos y podríamos ser capaces de ayudarle a descompilar su programa. Contamos con desarrollos internos para la descompilación de aplicaciones **Delphi** y **FreePascal** y podemos ofrecer servicios de descompilación para dichos archivos como servicio.

Si el programa abierto está empaquetado con un **empaquetador de EXE**, ofuscador o protegido, el descompilador ofrecerá intentar desempaquetar este archivo. Esta función solo está disponible en versiones de Windows de 32 bits y se proporciona sin ninguna garantía. Si es posible desempaquetar, el archivo se lanzará, se desempaquetará en memoria, se guardará en disco con un nombre diferente y se abrirá para su descompilación. **¡Atención!** Nunca intente desempaquetar programas maliciosos o cualquier otro código que pueda dañar su sistema utilizando el desempaquetador integrado, ya que al desempaquetar se ejecutará el programa que se está abriendo y su ejecución puede dañar el sistema o los datos.

Nota importante: bajo ninguna circunstancia debe intentar descompilar o desempaquetar software sin contar con la autorización necesaria, y si el titular de los derechos de autor prohíbe la descompresión y análisis de este archivo mediante licencia. Esto puede violar las leyes de su país.

*Visual Studio .NET, Visual C++ y Visual Basic son marcas registradas de Microsoft Corporation. Delphi es una marca registrada de Embarcadero.

Otras formas de abrir archivos

Puede arrastrar un archivo desde el Explorador hacia VB Decompiler. También puede arrastrar archivos con extensiones desconocidas o sin extensión de esta manera; el reconocimiento sigue dependiendo del contenido real del archivo. Los accesos directos de Windows **.lnk** soportados se pueden resolver a sus archivos referenciados. Las bases de datos VDD guardadas también se pueden abrir mediante arrastrar y soltar o desde la línea de comandos.

6.2 Trabajo con bases de datos de análisis

Esta característica está disponible a través de la sección **Base de datos (Data Base)** en el menú **Archivo (File)** y sirve para habilitar el guardado de los resultados de la descompilación en un archivo comprimido separado. Esto puede ser útil para una carga más rápida de los resultados de la descompilación en el futuro sin necesidad de volver a descompilar el código.

En caso de que necesite volver a analizar el programa o compartir estos resultados con un colega, no es necesario realizar la descompilación nuevamente. En el caso de una descompilación completa (con la opción **Decompilación rápida (Fast decompilation)** desactivada), este proceso puede requerir más tiempo y recursos. Es mucho más conveniente guardar previamente el código descompilado en el **Database**.

VB Decompiler utiliza un formato de base de datos comprimido interno, lo que permite una optimización significativa del espacio necesario para guardar los resultados. Además, existe compatibilidad con bases de datos guardadas por la mayoría de las versiones anteriores de VB Decompiler, lo que le permite almacenar información sin arriesgarse a no poder abrir dicha base de datos en la nueva versión del producto.

Abrir y transferir bases de datos VDD

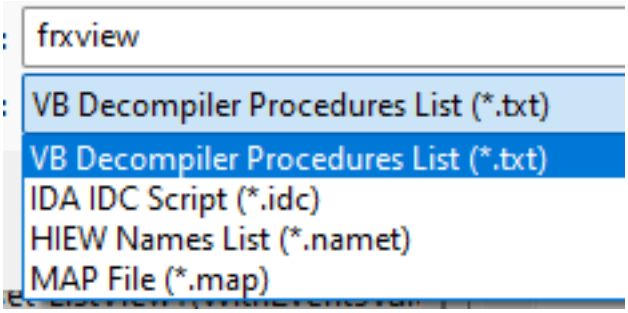
Las bases de datos de análisis guardadas utilizan la extensión **.vdd**. Puede abrir una base de datos VDD a través del flujo de trabajo de apertura de archivos, arrastrarla sobre la ventana del descompilador o pasar su ruta en la línea de comandos.

Al localizar el binario original analizado, VB Decompiler verifica tanto su ruta guardada como el directorio que contiene la base de datos. Para la colaboración, mantenga el binario correspondiente junto al archivo VDD cuando no esté disponible su ruta original. Preserve la compilación exacta analizada para que las direcciones y los bytes del archivo sigan siendo significativos.

6.3 Guardar la lista de procedimientos

Esta operación se realiza llamando a la opción **Grabar lista de procedimientos (Save procedures list)** en el menú **Archivo (File)** y es una herramienta poderosa para analizar programas, ya que permite a las utilidades externas asistir al **VB Decompiler**.

Actualmente, las direcciones, nombres de funciones y procedimientos utilizados en el programa procesado pueden guardarse en varios formatos. Puede elegir cualquiera de estas opciones especificando el tipo de archivo en el diálogo estándar de guardar.



Guardado en formato TXT le permite guardar la lista de procedimientos en un archivo de texto simple. Esto puede ser útil para comparar procedimientos entre diferentes versiones de un programa, detectar cambios de direcciones y así sucesivamente.

Guardado como script IDC le permite cargar las direcciones de funciones y procedimientos en el desensamblajer **IDA Pro**. En casos donde el programa analizado se compila en **Native Code**, esta característica puede ser muy útil.

Guardado en formato NAMET le permite simplificar el análisis utilizando la utilidad **HIEW**. Copie el archivo obtenido en la carpeta con el programa analizado. Luego abra el programa en HIEW y presione la tecla F12. Aparecerá una ventana con una lista vacía de procedimientos. Presione la tecla F12 nuevamente si la dirección del archivo NAMET coincide con la creada - simplemente presione Enter. Las direcciones de los procedimientos se cargarán en HIEW, lo que le permitirá navegar por los métodos utilizando la lista presionando la tecla F12.

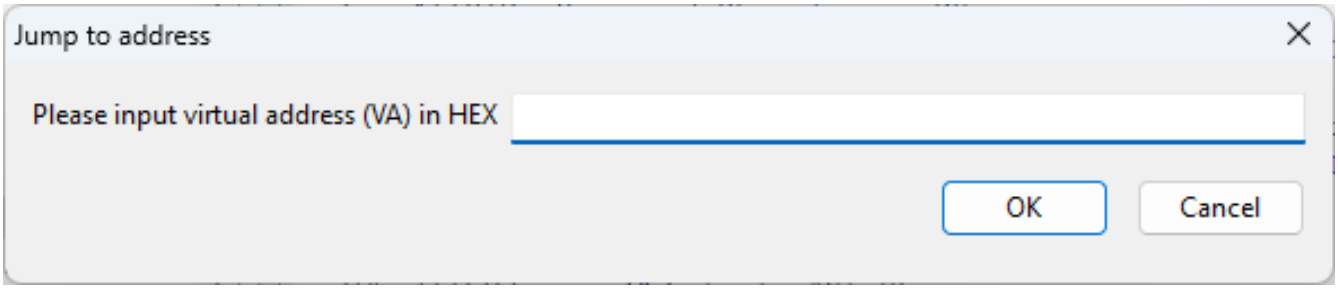
Guardado en formato MAP genera un archivo de mapa estándar que generalmente crean los compiladores. Este archivo puede utilizarse con muchos depuradores y desensambladores para marcar el inicio de las funciones del programa. Dado que los programas escritos en Visual Basic a menudo se desensamblan incorrectamente por software de terceros, el archivo MAP generado en **VB Decompiler** simplificará enormemente el proceso. Este formato es soportado a través de complementos en diversos productos como **OllyDbg**, **IDA Pro** y otros.

7 Funciones del programa

7.1 Ir a una dirección

Ir a una dirección virtual

Seleccione **Herramientas > Ir a dirección virtual (Tools > Jump to virtual address)** o presione **Ctrl+G**. Introduzca la dirección virtual del código que desea inspeccionar. También puede seleccionar una dirección en la lista y activar Ctrl+G. El descompilador abre el código asociado a esa dirección.



Una dirección virtual (VA) identifica una ubicación en el espacio de direcciones de la imagen ejecutable e incluye su base de imagen. Un desplazamiento de archivo identifica una posición de byte en el archivo en el disco. No sustituya uno por el otro: el Editor HEX soporta ambas formas de navegación.

Volver a través del historial de navegación

Utilice los botones de flecha situados encima del árbol para moverse hacia atrás y hacia adelante a través de los procedimientos previamente visualizados. **Esc** vuelve al procedimiento anterior. Los botones Atrás y Adelante del ratón también utilizan este historial, manteniendo la posición dentro del procedimiento.

Compartir una ubicación mediante su ruta en el árbol

En el Explorador (Solution explorer), haga clic con el botón derecho en el objeto requerido y seleccione **Copiar nombre de sección (Copy the name of section)**, o presione **Ctrl+C** mientras el árbol tiene el foco. Esto copia la ruta completa del nodo seleccionado.

Para volver a visitar esa ubicación, seleccione **Ir a la sección (Jump to the section)** en el menú contextual del árbol, o presione **G** en el árbol, y pegue la ruta copiada en el diálogo. Esto es útil cuando se entrega una investigación a otro analista o se vuelve a un proyecto grande. Utilice la ruta con el archivo analizado correspondiente; una ruta de otra compilación puede no identificar ya el mismo código.

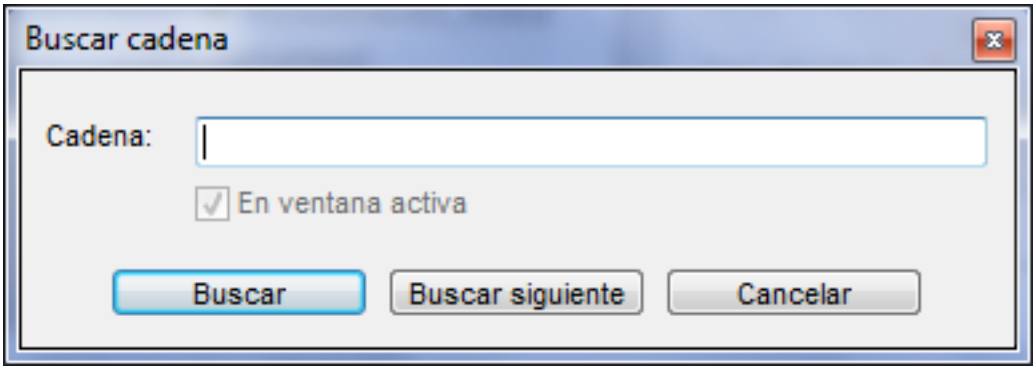
7.2 Búsqueda de cadenas

Buscar un procedimiento por nombre

Seleccione **Herramientas > Buscar rutina (Tools > Find procedure)** o presione **Ctrl+T**. Introduzca el nombre completo del procedimiento o una parte distintiva de él. La búsqueda abarca el árbol del proyecto, por lo que no es necesario saber en qué módulo o clase se encuentra el procedimiento. Si coinciden varios nombres, la lista de resultados contiene todas las entradas coincidentes; haga doble clic en la entrada requerida para abrirla.

Buscar texto mostrado o todo el proyecto

Seleccione **Herramientas > Find string (Tools > Find string)** o presione **Ctrl+F** para abrir el cuadro de diálogo de búsqueda de texto. Con **En ventana activa (In active window)** seleccionado, la búsqueda se aplica al código actualmente mostrado o a los datos del formulario. Desmarque esta casilla de verificación para buscar en todo el proyecto descompilado.



Introduzca el texto en **Cadena (String)** y seleccione **Buscar (Find)**. Utilice **Buscar siguiente (Find next)** para continuar buscando el mismo texto. **Cancelar (Cancel)** cierra el cuadro de diálogo.

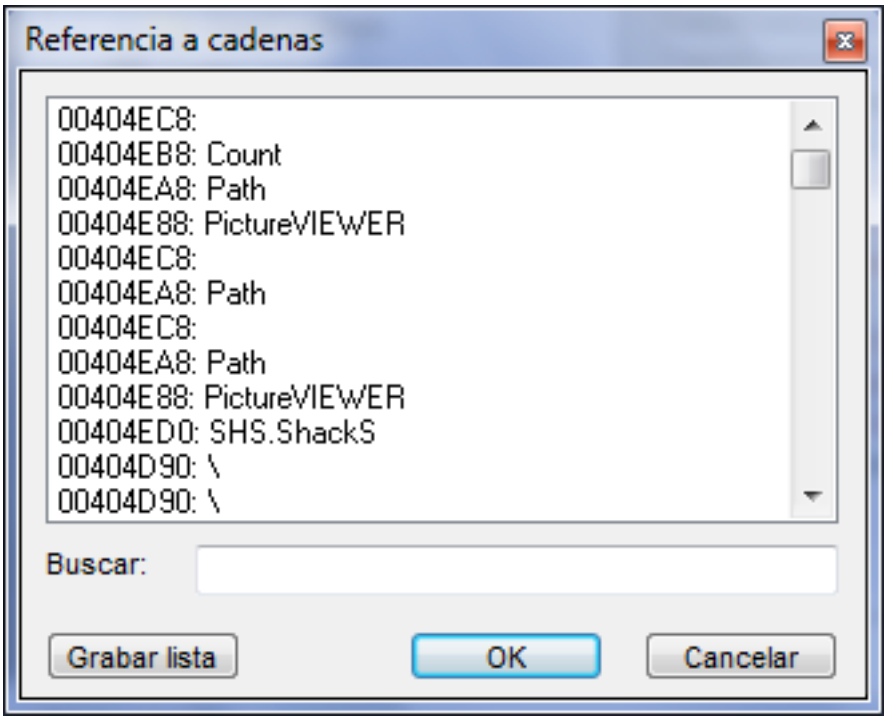
La búsqueda de todo el texto descompilado requiere una descompilación completa. Desactive **Decompilación rápida (Fast decompilation)** en Funciones avanzadas y procese el archivo primero. La búsqueda de procedimientos por nombre es una operación separada sobre la estructura del proyecto.

Buscar referencias a cadenas

Si necesita cadenas literales y el código que las utiliza, abra Referencias a cadenas. Su campo **Buscar (Search)** y su botón **Buscar siguiente** realizan la búsqueda en la lista de cadenas. Las referencias a cadenas pueden recopilarse mediante una desensamblaje preliminar en modo Descompilación rápida; no requieren buscar cada procedimiento completamente descompilado.

7.3 Referencias a cadenas

En el código de cualquier programa, es común encontrar datos de cadena (claves del registro, nombres de archivos, mensajes de error, etc.). Puede ser muy útil ver la lista de cadenas utilizadas por procedimientos y funciones en aplicaciones descompiladas. Eso es exactamente para lo que está diseñada nuestra función **Referencia a cadenas (String references)**, accesible a través del menú **Herramientas (Tools)**. Para utilizarla, simplemente navegue a **Herramientas** -> **Referencia a cadenas**, y si su programa contiene datos de cadena, estos se mostrarán como se muestra a continuación:

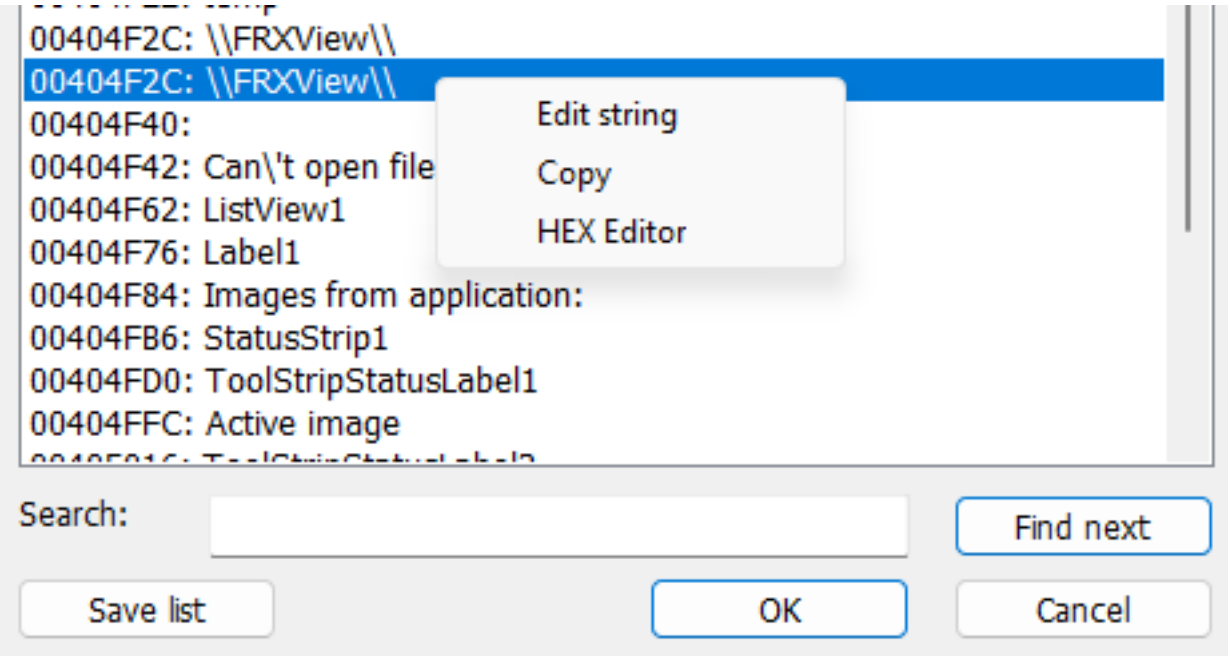


Encuentre la línea deseada en la lista y haga doble clic sobre ella. Se localizará el segmento de código que contiene esa línea, y se abrirá el procedimiento correspondiente en la ventana de código. Hacer doble clic nuevamente en la línea continúa la búsqueda de esta línea dentro del código del programa. Al hacer clic en 'Aceptar', se encuentra la línea resaltada en el código y se cierra el cuadro de diálogo de selección de filas. Cancelar simplemente cierra el cuadro de diálogo.

La función para mostrar **Referencia a cadenas** solo está disponible en la versión licenciada del software y no es accesible para la versión **Lite**.

Nota importante: las cadenas se muestran no solo para el módulo activo como en las versiones 1.x del programa, sino para todo el programa. Este cambio se realizó para facilitar el análisis de programas con un gran número de formularios y módulos.

VB Decompiler también admite la edición de cadenas. Para llamar al cuadro de diálogo de edición de cadena, haga clic derecho en la cadena que desea cambiar y seleccione **Editar cadena (Edit string)** del menú contextual que aparece. Aparecerá un cuadro de diálogo para ingresar una nueva cadena. Después de cambiar la cadena, haga clic en **Ok** y la cadena se modificará directamente en el archivo editado. Recomendamos crear una copia de seguridad del archivo que se está examinando antes de ello, porque no se realiza ninguna copia de seguridad durante esta acción y cancelar esta operación no es posible.



Nota importante: al editar cadenas, recuerde que la longitud de la nueva cadena no debe exceder la longitud de la antigua. Esto está relacionado con el formato de los archivos binarios de aplicaciones donde no se reserva espacio para insertar una cadena más larga.

También puede ver y editar parcialmente la cadena en el editor HEX. Para ello, haga clic derecho en la cadena y seleccione Editor HEX del menú que aparece. En el editor HEX, hay funciones para deshacer y rehacer cambios, lo que facilita la experimentación al modificar los datos de cadena de las aplicaciones.

La ventana con la lista de cadenas también tiene un campo de texto llamado **Buscar (Search)** y un botón **Buscar siguiente (Find next)**. Esta funcionalidad será útil para encontrar cadenas dentro de la lista, especialmente al tratar con proyectos grandes donde la búsqueda visual manual a través de la lista es extremadamente difícil.

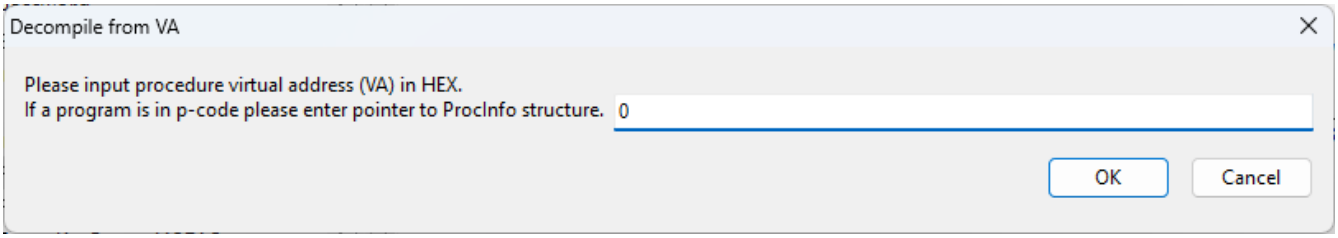
Descompilación rápida y listas de cadenas grandes

Las referencias de cadena pueden recopilarse mediante desensamblado preliminar, incluido en el modo **Decompilación rápida (Fast decompilation)**. Esto difiere de buscar todo el texto completamente descompilado. La versión 26.2 expande el diálogo más allá de 32 000 referencias y mejora la gestión de memoria para listas grandes.

Para ensamblados .NET, las cadenas de usuario del heap #US pueden inspeccionarse con sus ubicaciones de archivo. La lista representa cadenas disponibles para análisis estático; las cadenas encriptadas o generadas dinámicamente pueden requerir investigación separada.

7.4 Descompilación desde una dirección

En el menú **Herramientas (Tools)**, existe una función muy útil: **Decompilar desde VA (Decompile from VA)** (descompilar cualquier código desde la dirección virtual proporcionada). La característica principal de esta función es que no requiere que el archivo a descompilar tenga una estructura correcta y sea procesado mediante el método estándar. Por supuesto, en este caso, el código se descompilará con ciertas limitaciones. Por ejemplo, si las funciones importadas u objetos OLE no han sido pre-procesados, el código será poco informativo en esos puntos.



Para descompilar un fragmento específico de código dentro de un archivo, simplemente debe especificar la dirección virtual de ese código (si hablamos de **Native Code** o **ensamblado .NET**), o proporcionar una referencia a la estructura **ProcInfo** (si hablamos del clásico **Visual Basic P-Code**). La singularidad de trabajar con tales **P-Code** es que la estructura **ProcInfo** contiene un array de datos utilizados por el código. Sin este array, es imposible determinar con qué funciones trabaja el código, a qué cadenas se refiere, y así sucesivamente. En el caso de trabajar con **ensamblados .NET**, tras introducir la dirección, el descompilador le ofrecerá elegir entre descompilar el código como código intermedio **MS IL** o desensamblar este código. Esta opción se debe al hecho de que los ensamblados .NET pueden contener tanto código intermedio MS IL (**código administrado**) como código máquina (llamado **código no administrado**).

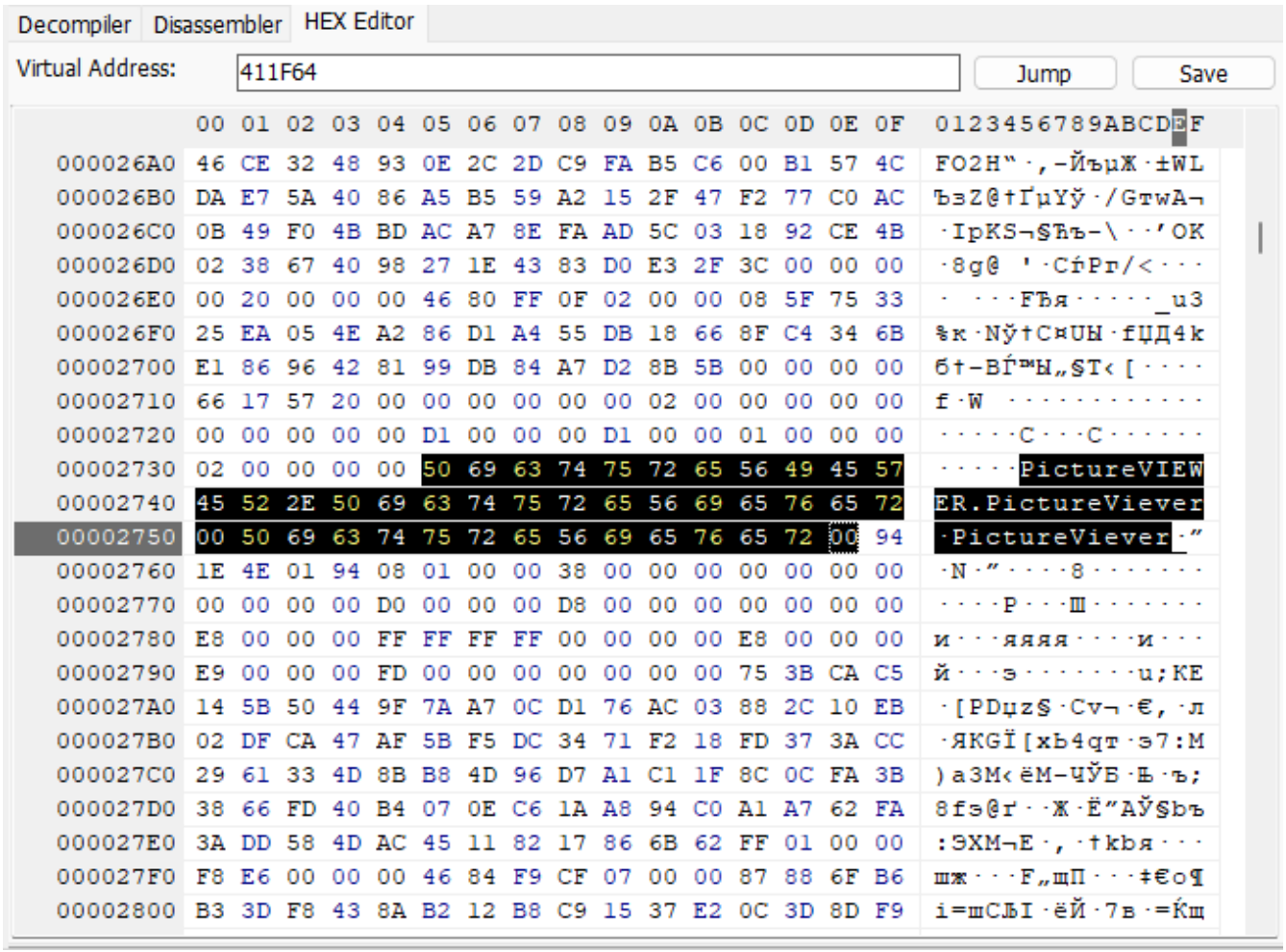
Si el archivo ya está abierto en el descompilador, solo necesita llamar a la función **Decompilar desde VA** y especificar la dirección para procesar. Si el archivo aún no ha sido abierto, esto se sugerirá automáticamente y el archivo se abrirá en modo simplificado.

La función **Decompilar desde VA** es particularmente útil cuando el código deseado no se descompila automáticamente o si el archivo está parcialmente dañado, impidiendo que se analice con el procedimiento habitual. Por supuesto, debe entenderse que el resultado de la descompilación en este caso será peor que con una descompilación estándar.

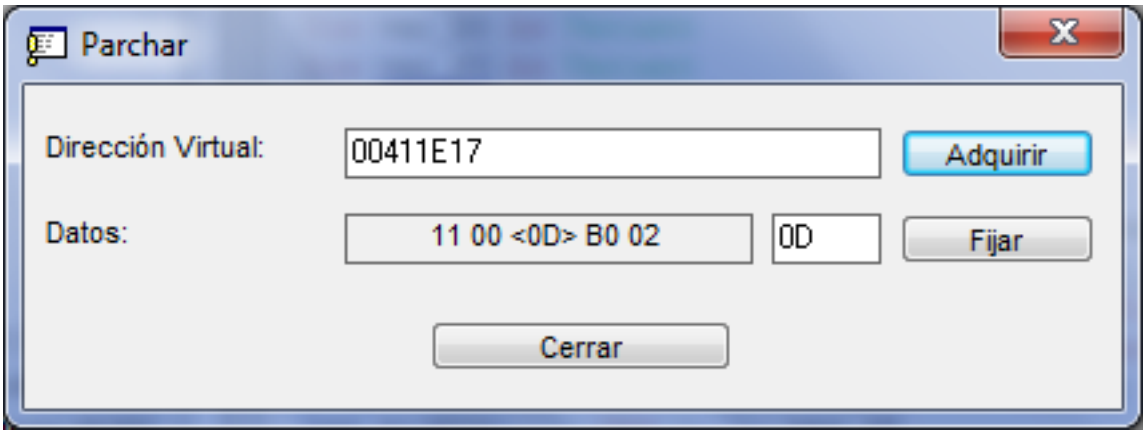
7.5 Modificación de datos

A partir de la versión 9.4, **VB Decompiler** incluye una herramienta moderna y práctica para editar datos binarios dentro de archivos: un editor hexadecimal completo **HEX Editor**. Es accesible desde la ventana principal del programa junto al código descompilado en la pestaña **HEX Editor** para cada procedimiento o formulario.

El editor hexadecimal **HEX Editor** permite editar los bytes del archivo tanto individualmente como por bloques. Para navegar a la dirección deseada dentro del archivo analizado, puede utilizar direcciones virtuales (incluidas las **Image Base**) o desplazamientos (una dirección regular que indica el desplazamiento de los datos necesarios respecto al inicio del archivo).



En versiones anteriores del descompilador (antes de la versión 9.4), la función para editar archivos se aplicaba mediante una herramienta más primitiva: Data Patcher. Funcionaba de la siguiente manera. En el menú **Herramientas (Tools)** -> **Parchar datos (Patch data)**, existía un instrumento llamado patcher que podía utilizarse en casos donde, tras analizar el código del programa, era necesario modificar alguna parte. Para cambiar los bytes necesarios sin utilizar herramientas externas, **VB Decompiler** contaba con funcionalidad integrada. La ventana del patcher se veía de esta forma:



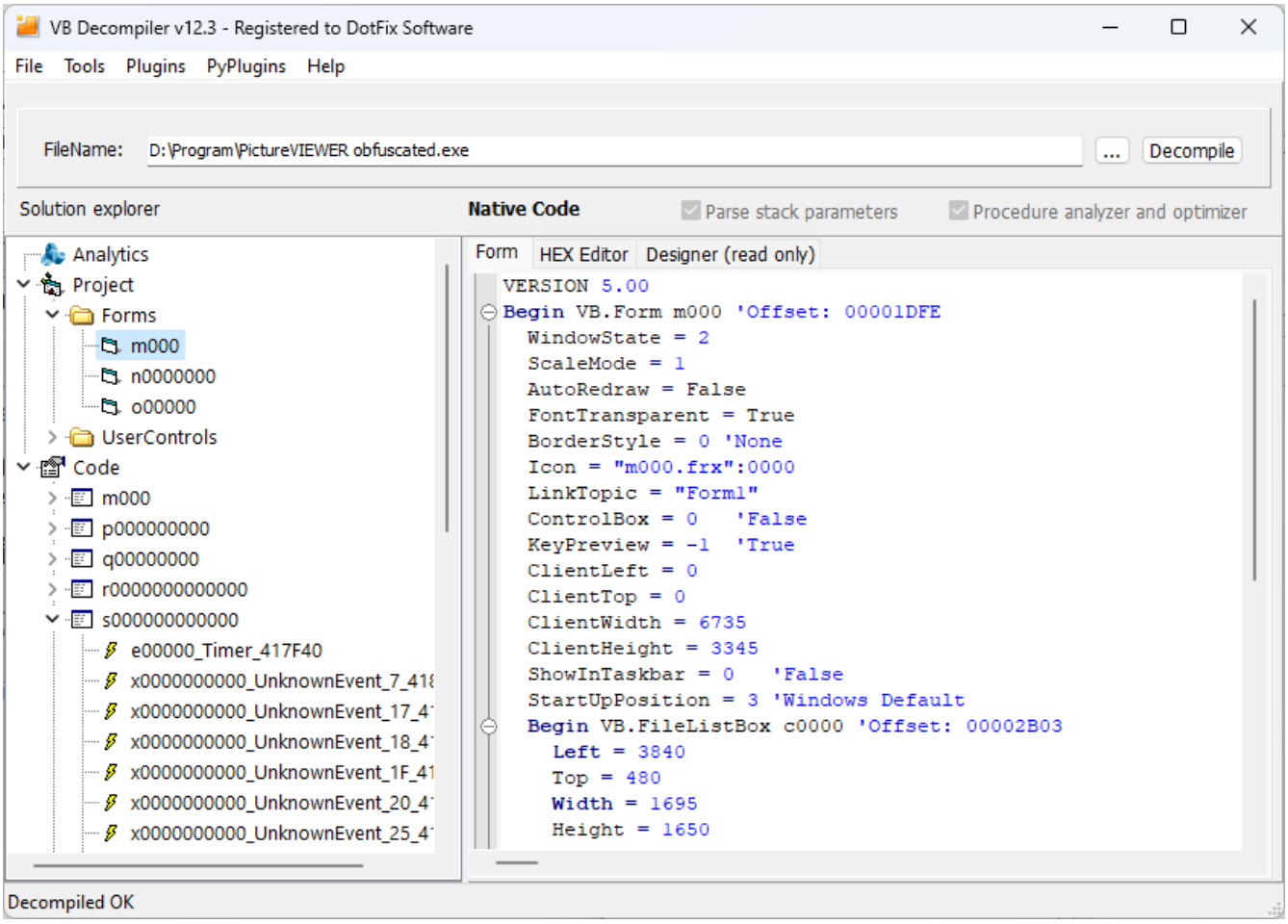
Trabajar con el patcher era extremadamente sencillo. En el campo **Dirección Virtual (Virtual Address)**, se introducía la dirección para editar un byte, luego se pulsaba **Enter** o se hacía clic en **Adquirir (Get)**. Los bytes de datos situados cerca de la dirección seleccionada se mostraban en el campo Data, y el byte seleccionado podía modificarse; después, era necesario hacer clic en **Fijar (Set)**, y el byte introducido se cambiaba.

Si en Opciones del programa existía la opción para **Decompilación rápida (Fast decompilation)** habilitada y la opción **Código decompilado a la memoria cache (Cache decompiled code)** deshabilitada, entonces todos los cambios se reflejaban en esa función donde se modificaron los bytes. Para ello, era suficiente volver a abrir esa misma función haciendo doble clic sobre su nombre en el árbol.

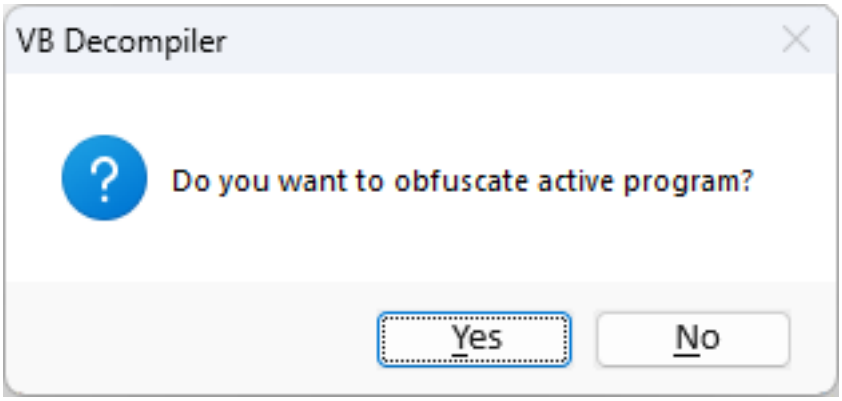
Nota importante: esta funcionalidad está diseñada para editar únicamente sus archivos de programa o aquellos para los cuales tiene los derechos correspondientes, así como para investigar código malicioso en laboratorios antivirus. Antes de cambiar el código de otra persona, asegúrese de revisar el acuerdo de licencia respecto a la existencia de tal derecho.

7.6 Ofuscación

Si está desarrollando programas en **Visual Basic 5.0/6.0**, puede aumentar su protección contra la descompilación. Para lograrlo, se incrusta el **Obfuscator** dentro del descompilador, capaz de eliminar todos los nombres de funciones, formularios, módulos y objetos del archivo binario de su programa, así como diversas firmas. Esto dificultará el análisis del programa, ya sea utilizando el descompilador proporcionado u otras soluciones. Además, la mayoría de los productos de terceros se negarán a descompilar este archivo por completo.



Para aplicar esta protección, abra su proyecto en el descompilador y seleccione **Ofuscar proyecto (Obfuscate project)** desde el menú **Herramientas (Tools)**. Este proceso no toma mucho tiempo, por lo que recomendamos ofuscar sus proyectos antes de publicarlos.



Esta función también está integrada en nuestra solución integral para la protección de software - **DotFix NiceProtect**, que se puede descargar desde www.niceprotect.com.

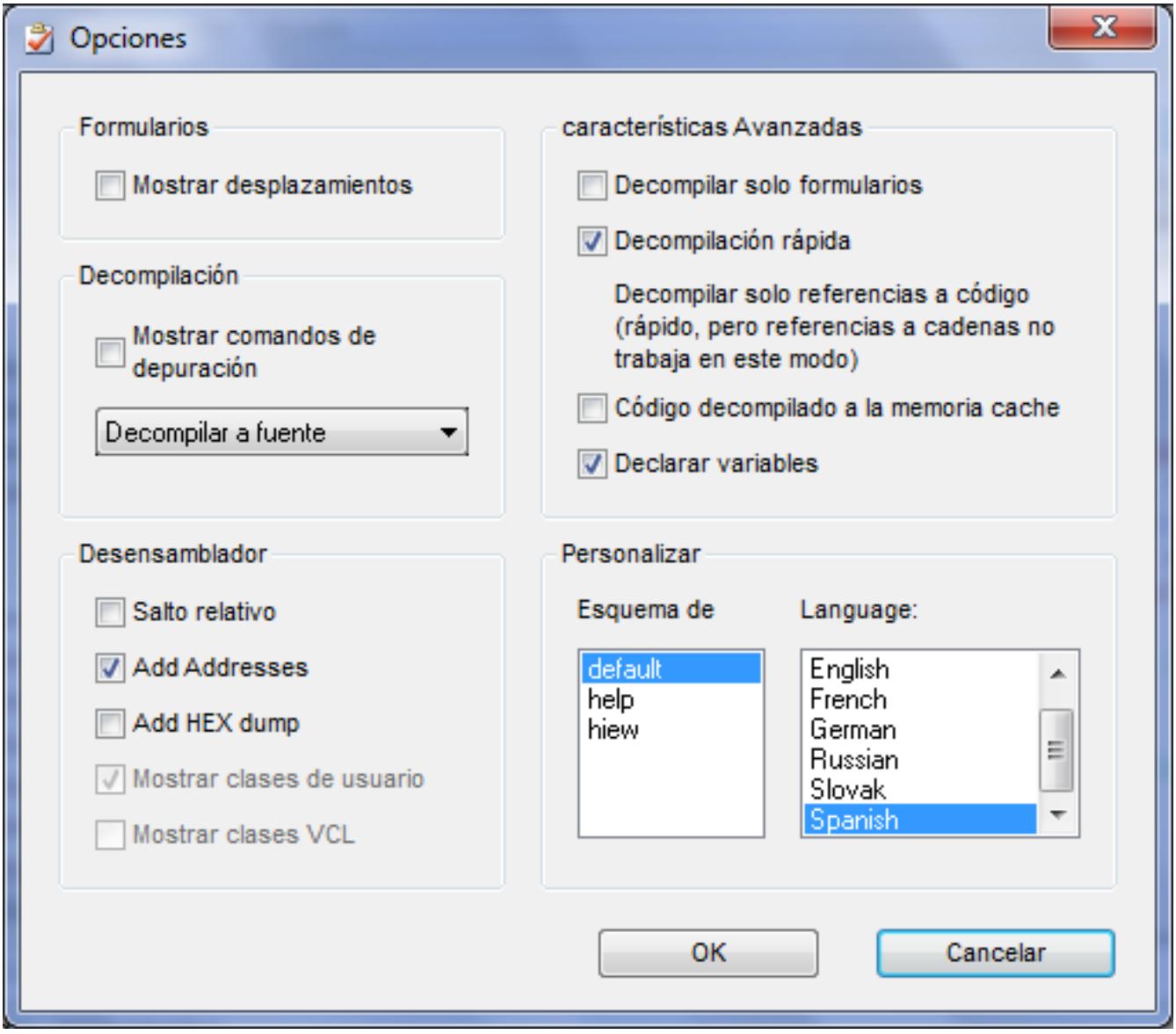
La ofuscación está disponible solo para usuarios con una licencia comercial del **VB Decompiler**.

Restricciones de ofuscación

8 Opciones del programa

8.1 Opciones del programa

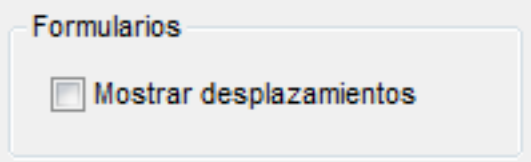
Para cambiar la configuración del programa, vaya al menú **Herramientas (Tools)** > **Opciones (Options)**. En el cuadro de diálogo resultante, verá las siguientes opciones:



Formularios
Descompilación
Desensamblador
Esquema de colores Funciones avanzadas Selección de idioma

8.2 Formularios

Puede modificar las siguientes opciones de descompilación para formularios:



En esta sección de opciones, puede habilitar la visualización de desplazamientos junto a los objetos de formulario. Esto puede ser necesario, por ejemplo, al localizar un programa en otro idioma sin acceso al código fuente. **Mostrar desplazamientos (Show offsets)** le permitirá encontrar la ubicación dentro del programa donde se encuentran estos objetos, sus nombres y propiedades.

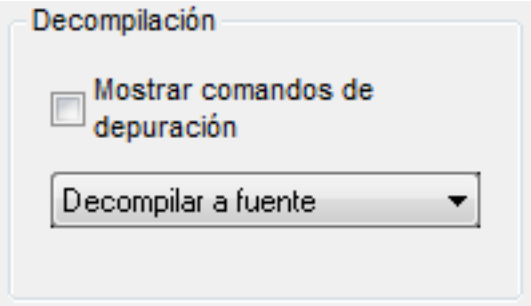
Posteriormente, utilizando un Editor HEX, puede modificar estos valores. Tenga en cuenta que si su conocimiento se limita al nivel básico de usuario y no está familiarizado con el funcionamiento interno de los archivos binarios, se recomienda no habilitar esta opción. Cualquier modificación incorrecta realizada en el archivo lo hará no funcional, y las direcciones innecesarias simplemente le obstaculizarán.

Vista previa visual y navegación de eventos

Las opciones de formulario son independientes del visor **Diseño (lectura) (Designer (read only))**. Consulte Reconstrucción visual de formularios para formularios de VB clásico, soporte de Windows Forms, WPF/BAML, navegación de manejadores de eventos y las limitaciones de cada tecnología de interfaz.

8.3 Descompilación

Abra **Herramientas > Opciones (Tools > Options)** para configurar la descompilación. Al comparar configuraciones de forma interactiva, utilice Descompilación rápida sin caché; de lo contrario, procese el archivo nuevamente después de cambiar una opción.



Mostrar comandos de depuración (Show debug commands)

Para P-Code clásico de VB, incluye operaciones del compilador y del depurador que de otro modo se omitirían en la lista de alto nivel. Habilítelo al inspeccionar la secuencia subyacente de instrucciones en lugar de solo la lógica del programa reconstruida.

Bloques If ... End If (Native Code) (Process If ... End If blocks (Native Code))

Reconstruye saltos condicionales de Código Nativo reconocidos como bloques `If...End If`. Deshabilítelo cuando necesite inspeccionar el patrón original de ramificación o investigar un procedimiento cuya estructura parece incorrecta.

Bloques 'Else' (Process 'Else' blocks)

Utiliza heurísticas adicionales para distinguir las ramas Else de condiciones anidadas y transiciones GoTo. Para Código Nativo, esto requiere el procesamiento de bloques If. Manténgalo habilitado para una lectura normal; compare los resultados con esta opción deshabilitada cuando una rama reconstruida sea ambigua.

Anál. prototipos (Analyze prototypes)

Ejecuta un análisis heurístico preliminar de los argumentos y valores de retorno de los procedimientos. Esto es particularmente útil con la Descompilación Rápida: las funciones y procedimientos privados en módulos estándar pueden carecer de metadatos de firma explícitos. El análisis preliminar toma tiempo adicional, pero evita tratar una llamada únicamente según el contenido de pila actualmente conocido. Una firma heurística sigue siendo una inferencia, especialmente cuando los metadatos, tipos externos o el código son incompletos.

Add Addresses

Muestra direcciones virtuales junto a las instrucciones reconstruidas. Manténgalo habilitado para correlacionar el código con el desensamblaje, los bytes del archivo y los hallazgos de otras herramientas. También es necesario para **Al desensamblador (Copy to disassembler)** para fusionar instrucciones de alto nivel con sus contrapartes de bajo nivel. El Desensamblador siempre muestra direcciones independientemente de esta configuración.

Comprensión de la reconstrucción de Código Nativo

El motor de Código Nativo reconoce patrones `Select Case` compatibles, acceso a arrays multidimensionales, llamadas COM y operaciones del runtime de VB. El manejo de errores puede ser reconstruido como `On Error Resume Next`, `On Error GoTo` y `On Error GoTo 0`. La optimización del compilador y la información de tipo faltante aún pueden limitar el resultado; inspeccione el desensamblaje cuando la salida de alto nivel sea incompleta.

La versión 26.2 amplió el tipado Variant, lecturas y escrituras de SafeArray multidimensionales y la reconstrucción `VarPtr` para elementos de array. La salida puede identificar una referencia a un array como `SafeArray`, reflejando la

estructura del runtime reconocida por el motor de análisis. El aislamiento del estado de rama y el análisis de variables temporales también mejoran la reconstrucción de código If/Else complejo. La versión 26.5 mejora aún más el manejo de Select Case y funciones del runtime VB5/VB6.

Información de tipos COM y ActiveX

VB Decompiler utiliza primero su base de conocimientos integrada para interfaces conocidas. Para un objeto COM externo desconocido, puede resolver identificadores de biblioteca, clase e interfaz a través del registro del sistema y analizar la biblioteca de tipos DLL/OCX correspondiente. Esto proporciona firmas de método y propiedad, incluyendo tipos de interfaz devueltos, para llamadas early-bound y late-bound compatibles.

Si una biblioteca de tipos requerida está ausente o su registro no puede resolverse, algunas llamadas pueden permanecer sin identificar, por ejemplo como `UnkVCall`. Las dependencias faltantes son relevantes al comparar la salida entre máquinas de análisis. La base de conocimientos IID ampliada incluye automatización de Office, ADO/DAO, XML y HTTP, scripting, shell, mensajería e interfaces de servicio de directorio; no implica que cada llamada COM posible pueda resolverse.

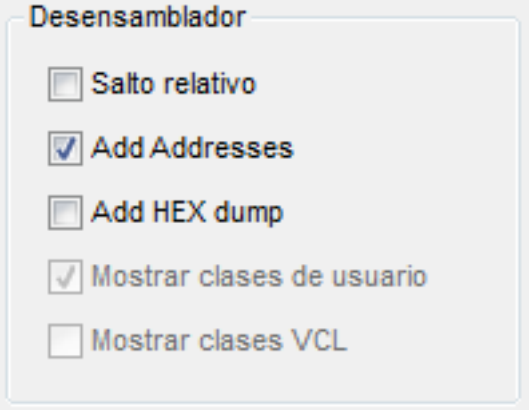
Solo metadatos limitados de UDT sobreviven en algunos binarios de VB. El descompilador puede recuperar información como el tamaño de una estructura y el conteo de campos de cadena sin recuperar la disposición completa original de los campos. No interprete una declaración parcial como una definición a nivel de fuente completa.

Reconstrucción y optimización .NET

El IL compatible se reconstruye como expresiones similares a C# y flujo de control. El control **Optimizador de procedimiento (Procedure analyzer and optimizer)** en la ventana principal puede simplificar patrones compatibles de bucles, condicionales, operadores y variables temporales. Inspeccione el IL y los metadatos donde quede un fragmento no compatible u ofuscado. Este procesamiento es independiente del Ayuda IA (AI Helper).

8.4 Desensamblador

Este bloque de opciones es responsable del desensamblaje de código máquina:



Volcado HEX (Add HEX dump) muestra los códigos máquina antes de cada comando desensamblado, lo cual puede ser necesario únicamente al modificar el archivo binario después del análisis en un Editor HEX.

Omitir conv. de carac. (Ignore chars convertions) se utiliza para desactivar la conversión de datos de cadena no ingleses a Unicode numérico. Habilitar esta opción hace que las cadenas mostradas sean más fáciles de leer. Desactívela cuando necesite la representación Unicode numérica.

Mostrar clases de usuario (Show user classes) incluye el análisis de clases de usuario dentro de aplicaciones **Delphi** / **Free Pascal**. Esta opción no se utiliza para aplicaciones **Visual Basic** y **C#**.

Mostrar clases VCL (Show VCL classes) incluye la descompilación de todas las clases internas de la biblioteca **VCL**. Se recomienda desactivar esta opción para que el desensamblajer no analice miles de clases de servicio, que generalmente son innecesarias al analizar programas. Esta opción solo se utiliza para la descompilación de aplicaciones **Delphi** con fines internos y fue creada para la unificación con otros productos, pero no está implementada en **VB Decompiler**.

La descompilación parcial de aplicaciones **Delphi** y **Free Pascal** se ofrece actualmente exclusivamente como un servicio separado (no como software).

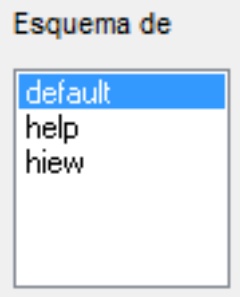
Código P-Code y .NET IL de VB

Volcado HEX también admite el listado P-Code de bajo nivel. Utilice **Al desensamblador (Copy to disassembler)** desde la vista Decompiler para comparar las instrucciones recuperadas con P-Code o instrucciones nativas en un mismo listado.

Para ensamblados .NET, el desensamblajer puede mostrar métodos IL o nativos x86/x64 según corresponda. Las firmas de variables locales recuperadas pueden incluir arrays, genéricos, tipos de valor y referencias a TypeSpec. Habilite **Mostrar estructuras .NET (Show .NET structures)** cuando necesite inspeccionar tablas de metadatos y heaps junto con el código. Las versiones posteriores amplían el soporte para metadatos ofuscados, incluida la tabla FieldPtr; esto no garantiza la recuperación exitosa de cada ensamblado dañado u ofuscado.

8.5 Esquema de colores

Esta sección define el esquema de colores y la resaltado de sintaxis al visualizar código en un descompilador:



Los preajustes se almacenan en la carpeta "colors" dentro de los archivos .ini ubicados en el directorio donde se instaló VB Decompiler. Los nombres de archivo determinan los nombres de los preajustes en la sección de configuración del programa. Estos archivos tienen un formato sencillo, y usted puede crearlos fácilmente o editar los existentes.

El formato de configuración es el siguiente:

```
[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)
```

El código de color se especifica en el formato 32-bit de color verdadero. Decimal en una sección Colors (para la compatibilidad hacia atrás con versiones antiguas de VB Decompiler) y hexadecimal en la sección Colors2.

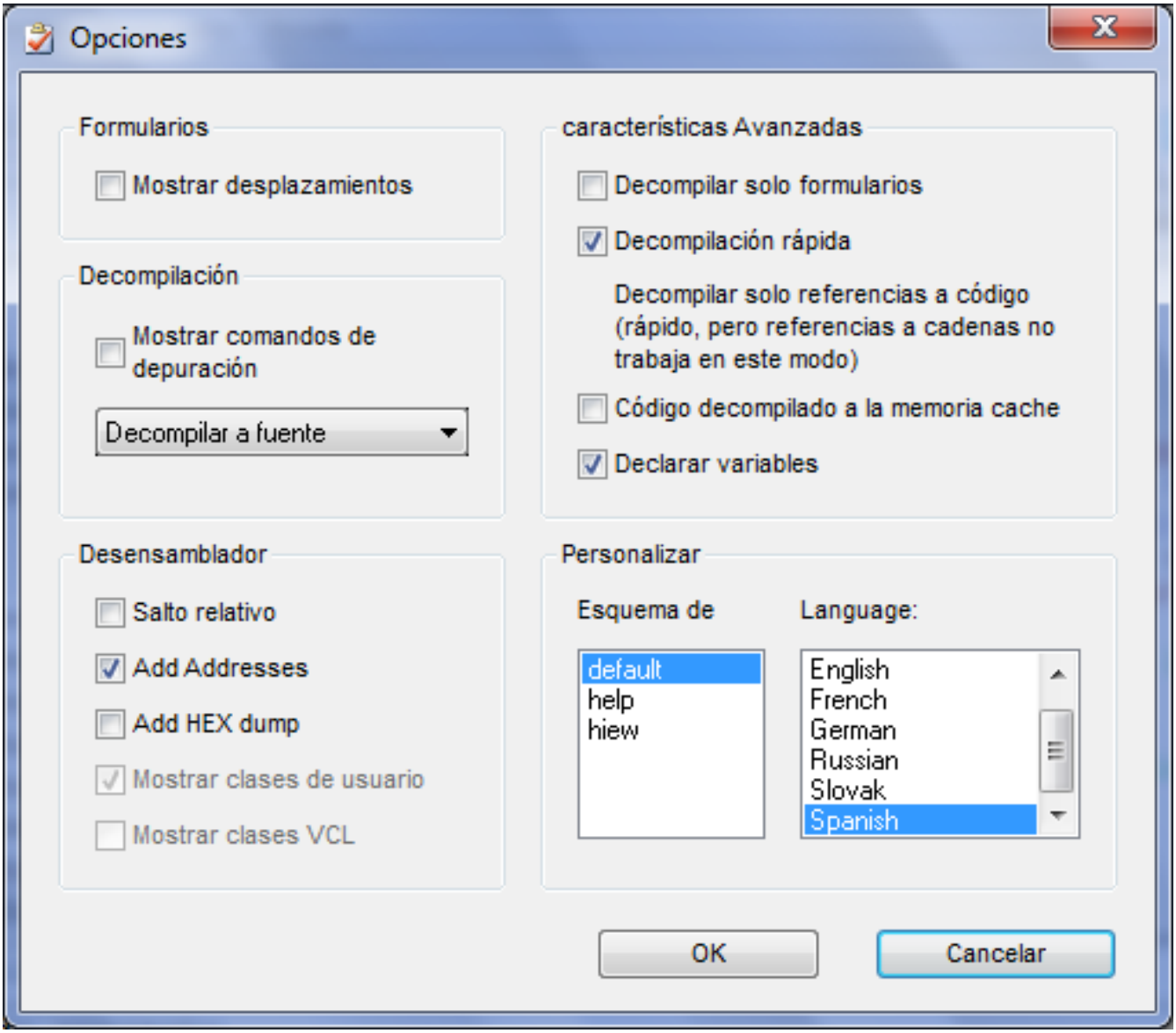
Cambiar temas durante el análisis de código

El sistema de temas soporta modo oscuro y puede aplicar un tema seleccionado sin reiniciar VB Decompiler, incluso después de editar su archivo INI. La configuración extendida puede contener nombres de fuentes de respaldo separadas por comas, lo cual ayuda cuando las fuentes equivalentes difieren entre Windows y entornos de compatibilidad.

Los colores bajo el cursor controlan la resaltado de identificadores. Consulte Visualización de código para la resaltado de variables locales, miembros y globales.

8.6 Funciones avanzadas

Estas configuraciones controlan cómo se procesa el proyecto y cómo se presenta la información recuperada.



Decompilar solo formularios (Decompile only forms)

Deshabilita la descompilación del código cuando solo se necesitan las definiciones recuperadas de formulario y UserControl. Para el visor y sus capacidades específicas de tecnología, consulte Reconstrucción de formularios visuales.

Decompilación rápida (Fast decompilation)

Cuando está habilitado, VB Decompiler procesa primero la estructura del proyecto y descompila los procedimientos individuales a medida que se abren. Esto reduce el tiempo de espera inicial al investigar procedimientos seleccionados. Con **Código decompilado a la memoria cache (Cache decompiled code)** deshabilitado, puede cambiar las configuraciones de análisis y volver a abrir un procedimiento para obtener un nuevo listado.

Habilite **Anál. prototipos (Analyze prototypes)** en Configuraciones de descompilación para realizar un análisis heurístico preliminar de las firmas de procedimientos privados. Esto añade tiempo de procesamiento inicial pero puede mejorar la reconstrucción de argumentos sin requerir una descompilación completa.

La descompilación rápida no genera de antemano un listado completo de alto nivel del proyecto. Para realizar una búsqueda de texto en todo el proyecto o exportar el proyecto descompilado completo, desactive este modo y procese

el archivo de nuevo. Las Referencias a cadenas son diferentes: pueden recopilarse mediante desensamblaje preliminar y están disponibles en el modo de descompilación rápida.

Descompilación completa

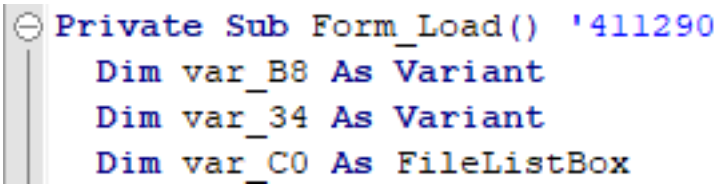
Cuando la descompilación rápida está deshabilitada, el descompilador procesa los procedimientos del proyecto con antelación. Esto tarda más tiempo, pero hace disponible el listado completo generado para búsqueda de texto y exportación. Los cambios en las configuraciones de descompilación requieren procesar el archivo nuevamente.

Código decompilado a la memoria cache

Mantiene los listados ya generados en el modo de descompilación rápida. Esto evita volver a descompilar repetidamente un procedimiento abierto, pero los resultados almacenados no reflejan cambios posteriores en las configuraciones de análisis. Cuando se comparan interactivamente las configuraciones **Analizar parámetros pila (Parse stack parameters)** o **Optimizador de procedimiento (Procedure analyzer and optimizer)**, deshabilite el almacenamiento en caché.

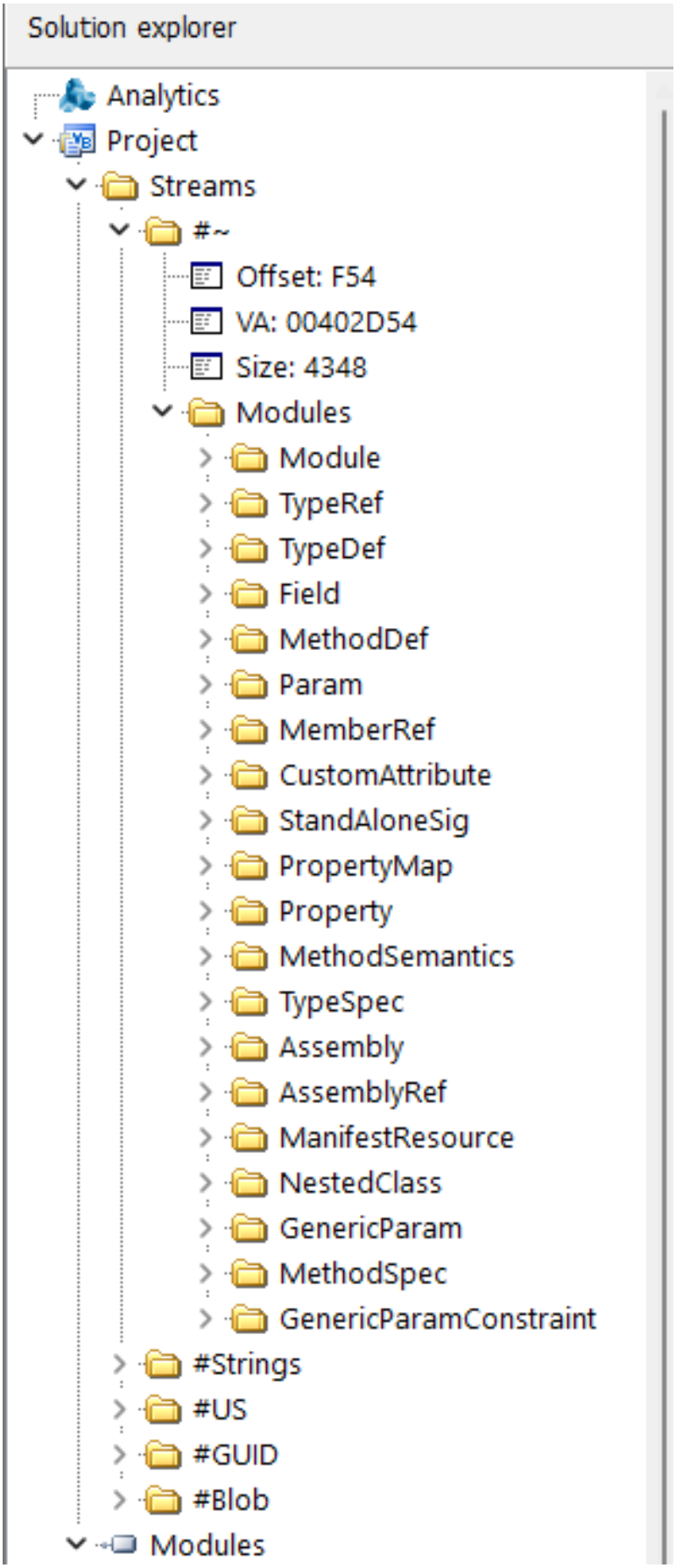
Declarar variables (Declare variables)

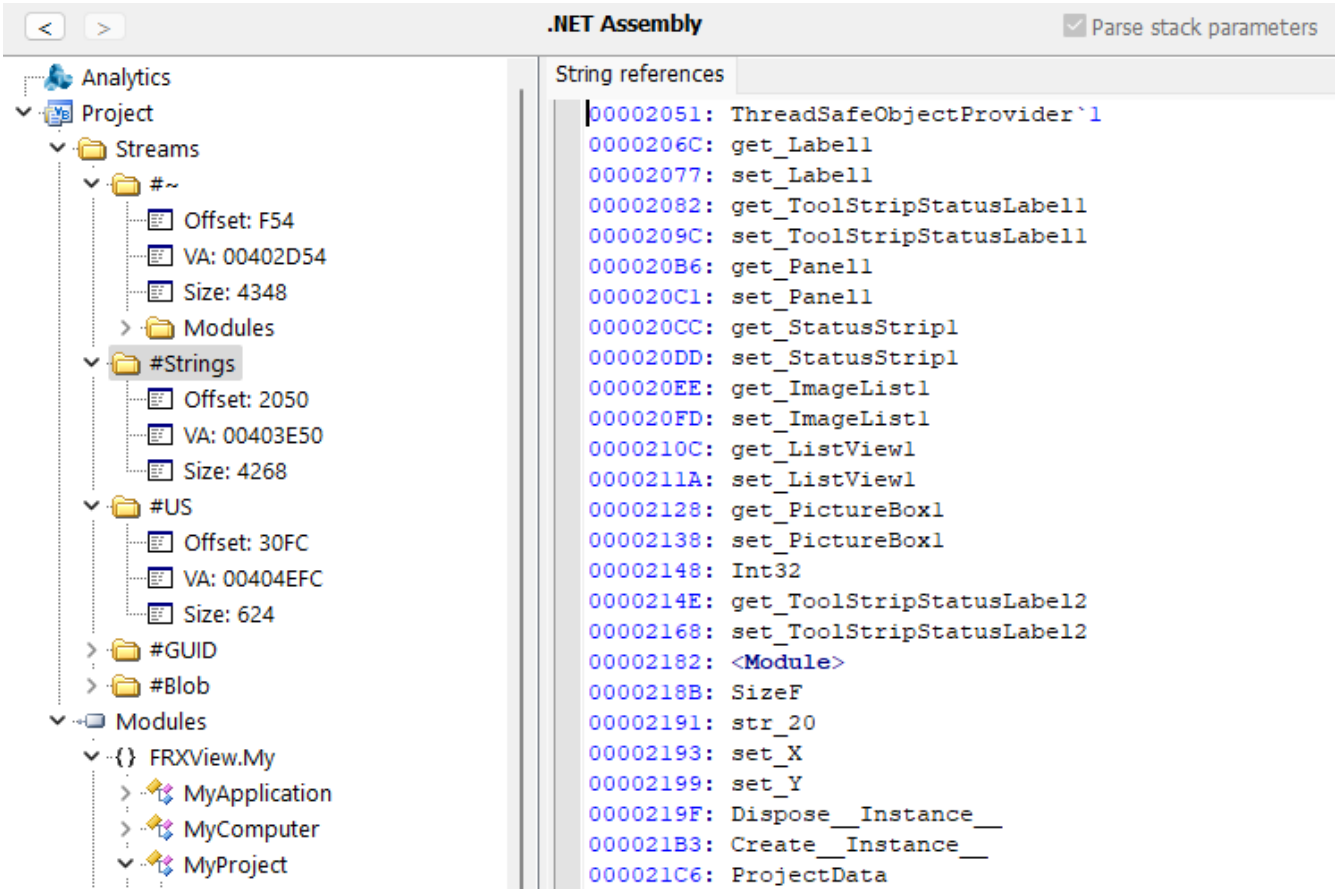
Añade un bloque de declaración para las variables recuperadas al inicio de cada procedimiento descompilado. Esto es útil para la recuperación del código fuente y la inspección de tipos. Para obtener una visión concisa de la lógica del procedimiento, puede ocultar las declaraciones. Los tipos recuperados están sujetos a los metadatos y análisis disponibles para el descompilador.



Mostrar estructuras .NET (Show .NET structures)

Expone las tablas y heaps de metadatos .NET en el Explorador (Solution explorer). Úselo para inspeccionar estructuras y firmas junto con el código. Seleccionar una tabla muestra su contenido; el heap #US contiene cadenas de usuario. Deshabilite la opción cuando no necesita navegación de metadatos.


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



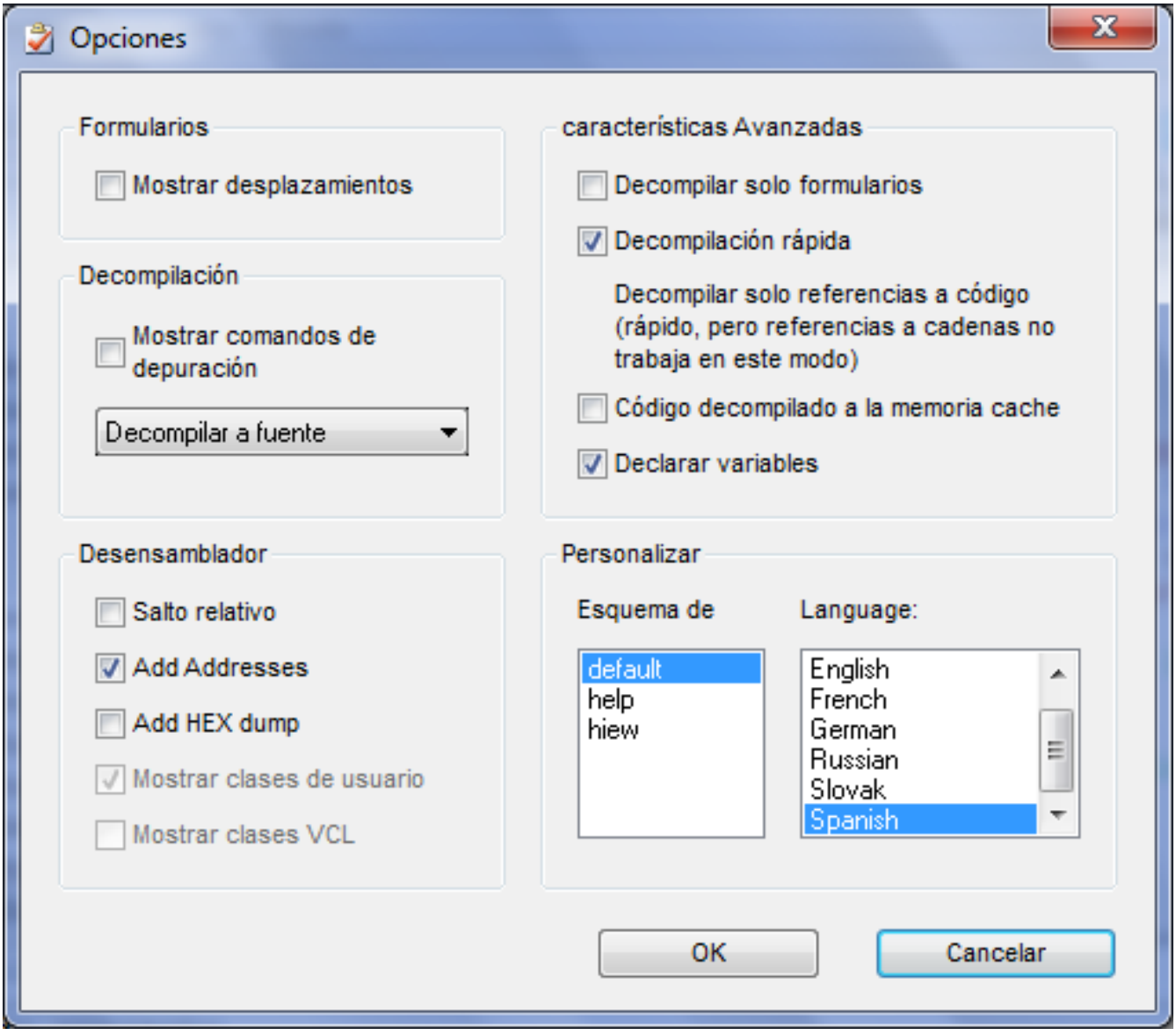
Estilo de ventana Visual Studio (Window style as in Visual Studio)

Coloca el código a la izquierda y Explorador a la derecha. Esto cambia la distribución del espacio de trabajo sin modificar el código recuperado.

8.7 Selección de idioma

VB Decompiler proporciona localizaciones de interfaz para más de 20 idiomas utilizados a nivel mundial. La documentación también está disponible en múltiples idiomas.

Puede elegir el idioma de interfaz que prefiera en el menú **Herramientas (Tools)** -> **Opciones (Options)**, en la sección **Personalizar (Personalize)**. Después de elegir el idioma, es necesario reiniciar **VB Decompiler** para la aplicación adecuada de las configuraciones del idioma en todo el programa. Si reiniciar el programa no es deseable, puede continuar utilizando **VB Decompiler**; sin embargo, algunos textos aún pueden mostrarse en el idioma anterior.



El idioma de la ayuda se selecciona automáticamente en función del idioma del sistema de su computadora. Para leer el manual en otro idioma, abra el archivo de ayuda correspondiente desde la carpeta donde **VB Decompiler** está instalado. PDF es el formato principal para los manuales localizados; los archivos de ayuda CHM siguen disponibles para algunos idiomas.

El manual en inglés es la fuente de todas las traducciones. Si nota un error o inconsistencia en un manual localizado, por favor informe a soporte técnico.

Acceso al manual

Si un archivo de ayuda local o el visor PDF requerido no están disponibles, VB Decompiler puede abrir la documentación en línea. Para un entorno de investigación sin conexión, mantenga los archivos de ayuda locales necesarios disponibles con anticipación.

9 Plugins

Los complementos extienden VB Decompiler mediante DLL nativas. Los complementos instalados están disponibles desde el menú **Agregados (Plugins)**. La API de devolución de llamada proporciona acceso a la información y al código del proyecto recuperado, permitiendo que un complemento inspeccione o modifique las estructuras internas compatibles.

VB/DE Decompiler MCP Server 2.1.0

Conecta un cliente MCP a un proyecto ya abierto. Puedes consultar módulos, formularios, procedimientos y código recuperado sin ejecutar el programa analizado.

Descargar MCP Server 2.1.0 para hosts x86 · Código fuente en Delphi

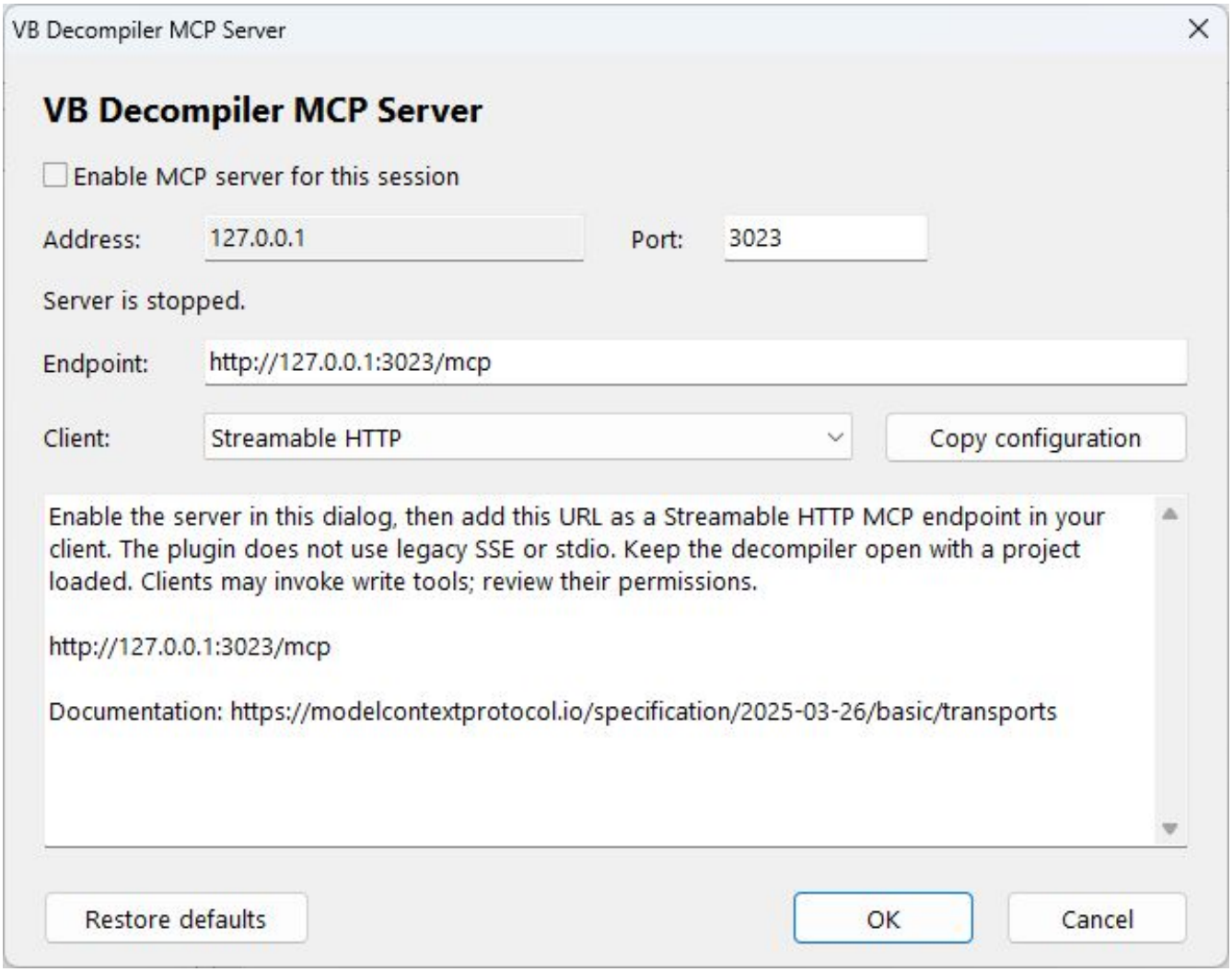
`vb_get_form` devuelve el índice del formulario `form_index`, su nombre `form_name` y el texto de sus propiedades `form_text`. Si el nombre no está disponible, `form_name` es una cadena vacía. Busque unidades o clases con `vb_list_modules` y `query`, y texto en los cuerpos con `vb_search_module_code`. Referencia de comandos MCP y ejemplos

Requisitos

El archivo incluye el complemento x86 para el descompilador de Windows de 32 bits. Las herramientas disponibles dependen de la versión, la edición y el proyecto abierto. En versiones anteriores algunas lecturas solo usan los listados guardados en caché.

Instalación e inicio

1. Cierra el descompilador. Extrae el archivo y copia `vbmcp.dll` a la carpeta `plugins` de la instalación. Si ya existe una DLL anterior, sustitúyela.
2. Inicia el descompilador y abre un proyecto. Elige **Plugins** → **VB/DE Decompiler MCP Server**.
3. Marca **Enable MCP server for this session**, pulsa **OK** y usa la dirección que muestra el cuadro de diálogo.



Configuración del servidor MCP en VB Decompiler. El complemento detecta qué descompilador lo ha cargado y muestra el nombre del producto en el título de la ventana.

Conexión de un cliente MCP

Usa Streamable HTTP. La dirección predeterminada es `http://127.0.0.1:3023/mcp`. Si usas los dos productos a la vez, asigna puertos distintos.

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

En VS Code puedes usar **MCP: Add Server** y elegir HTTP.

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

Documentación de los clientes · Codex · Claude Code · VS Code

Comprobación de la conexión

Abre un ejemplo con formularios y llama a `vb_list_forms` con `limit` igual a `5`. Los nombres de las herramientas de esta versión empiezan por `vb_`.

Acceso local y cambios en el proyecto

El servidor escucha solo en `127.0.0.1` y no autentica al usuario. Los clientes locales pueden leer el proyecto o modificar listados, nombres y marcas. Conserva activada la confirmación de las escrituras. Para detener el servidor, desmarca la casilla y pulsa **OK**. El servidor empieza desactivado cada vez que se ejecuta el descompilador.

El complemento funciona localmente. El cliente MCP puede enviar los resultados al proveedor del modelo según su configuración.

Desarrollado en Delphi por Sergey Chubchenko (DotFix Software). El punto de partida fue el plugin MCP original. Se distribuye bajo la licencia MIT.

FRX Image Viewer

FRX Image Viewer 1.2.0 permite visualizar y extraer imágenes registradas por VB Decompiler o DE Decompiler. Use VB Decompiler para gráficos de formularios VB5/6 y .NET, y DE Decompiler para gráficos de DFM binarios de Delphi. Se admiten BMP, PNG, JPEG, GIF, ICO y CUR.

Instalación y apertura

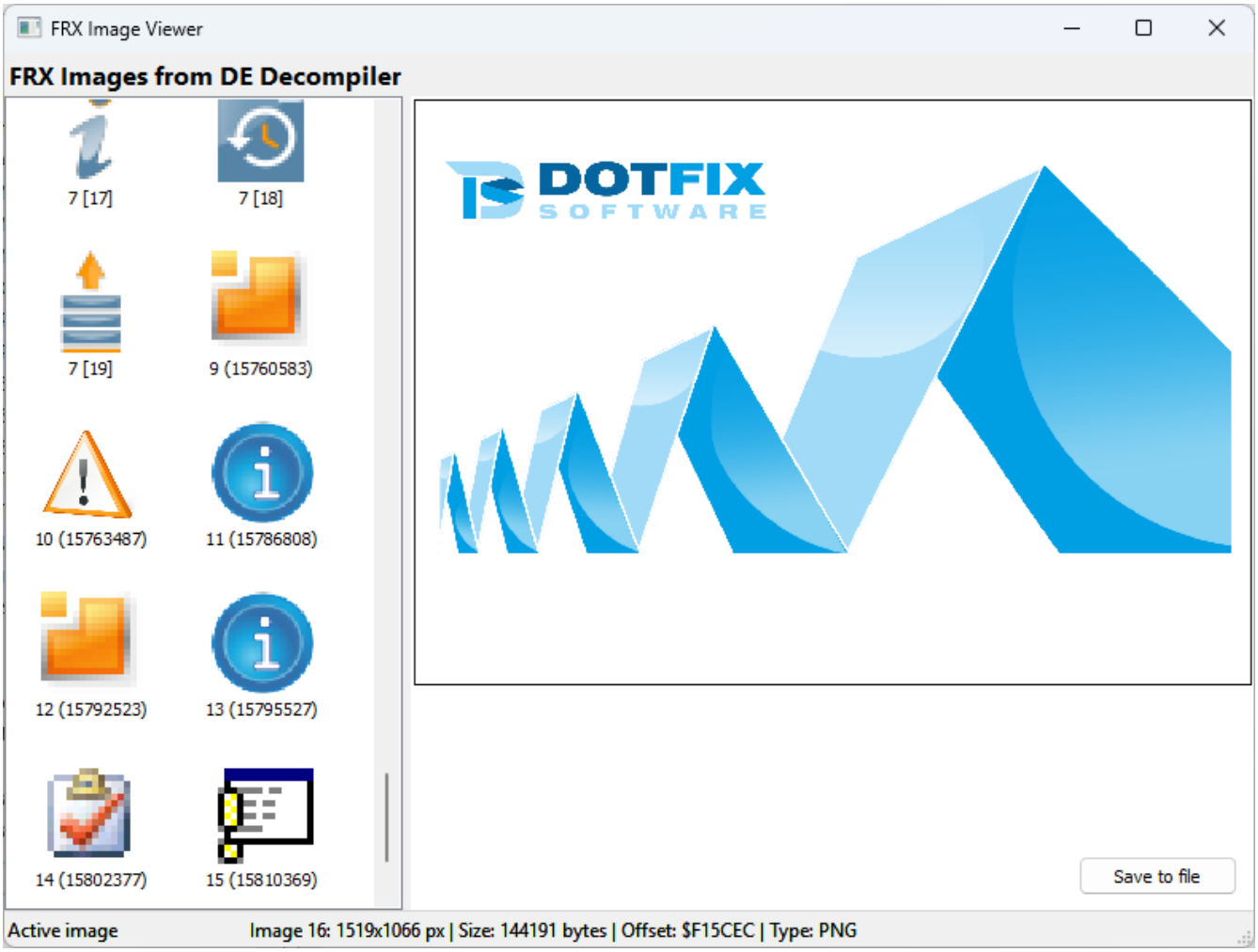
1. Copie `FRXViewer.dll` en la subcarpeta `plugins` de la instalación del host y reinicie el descompilador. La arquitectura de la DLL debe coincidir con la aplicación host, independientemente de la arquitectura del archivo analizado.
2. Abra el EXE o DLL y decodifique sus formularios para que el host registre las imágenes disponibles.
3. Seleccione **Agregados** > **FRX Image Viewer** y elija una miniatura.

La descarga de 1.2.0 contiene una DLL para hosts x86. Un host x64 requiere una compilación correspondiente. El plugin necesita Windows 10/11; el repositorio del código fuente incluye instrucciones de compilación.

Vista previa y exportación

La galería y la vista previa muestran las imágenes disponibles, incluidos los archivos ICO/CUR compuestos. El área de estado indica el elemento, dimensiones, tamaño u offset del mapa de bits de origen y formato. El visor sigue el tema del host y admite el modo oscuro.

Pulse **Save to file** para exportar el elemento seleccionado. Las imágenes independientes conservan sus bytes y formato originales. Las celdas individuales de ImageList se exportan como ICO con sus máscaras y transparencia alfa; estos ICO se generan a partir de los mapas de bits de origen.



FRX Image Viewer en DE Decompiler, con gráficos DFM de Delphi e iconos individuales de ImageList.

ImageList nativas de Delphi y compatibilidad

Con un host compatible, las ImageList nativas de Delphi reconocidas aparecen como iconos individuales. Las imágenes codificadas de forma independiente en elementos de colecciones siguen siendo imágenes separadas normales. Los flujos no admitidos mantienen la visualización de la imagen completa cuando existen entradas de imagen originales.

La revisión compartida SDK 26.6, compatible con VB Decompiler 26.6 y DE Decompiler 26.5, añade los comandos opcionales `GetFrxImageListInfo` (63) y `GetProductName` (64). Los hosts anteriores conservan la API original y la visualización de imágenes completas. Consulte la referencia del SDK para detectar funciones y aplicar las reglas de offsets.

La visualización y extracción no ejecutan el archivo analizado ni cargan sus componentes. Los GIF animados muestran el primer fotograma. Los recursos disponibles dependen de las imágenes decodificadas y registradas correctamente por el host.

Desarrollo de un complemento

El SDK incluye ejemplos para Visual C++, Rust, Delphi y Free Pascal. Consulte la referencia de la API del complemento VB Decompiler para los puntos de entrada exportados, la firma de devolución de llamada, identificadores de comandos, representaciones de datos y enlaces a los proyectos de ejemplo.

Un complemento DLL proporciona su nombre e información del autor y recibe un puntero a la función de devolución de llamada del programa anfitrión mediante `VBDecompilerPluginLoad`. La devolución de llamada se utiliza para solicitar o actualizar los datos compatibles del proyecto. Siga las convenciones de llamada, punteros, codificación de cadenas

y requisitos de búfer del SDK al implementar esta interfaz; si utiliza un lenguaje administrado, asegúrese de que la capa de interoperabilidad respete la interfaz nativa.

VB Decompiler inicializa los complementos DLL disponibles en el inicio. Compile los complementos para la arquitectura del anfitrión prevista y el entorno Windows, y verifique la DLL resultante con ese entorno. El SDK expone tanto el acceso al código de vista activa como al código a nivel de proyecto; las operaciones que requieren todos los procedimientos descompilados necesitan una descompilación completa en lugar de solo una lista bajo demanda.

10 Reconstrucción visual de formularios

La pestaña **Diseño (lectura) (Designer (read only))** muestra una reconstrucción visual de las interfaces compatibles extraídas del binario analizado. Ayuda a conectar los elementos de la interfaz con el comportamiento de la aplicación sin interpretar manualmente cada propiedad recuperada o declaración de inicialización.

Abrir un formulario recuperado

1. Abrir la aplicación y permitir que VB Decompiler recupere su estructura del proyecto.
2. Seleccionar un formulario recuperado en el Explorador (Solution explorer) y abrir **Diseño (lectura)**.
3. Seleccionar un control para inspeccionar sus propiedades disponibles e información de eventos recuperados. Utilizar los eventos para navegar al código del manejador correspondiente.

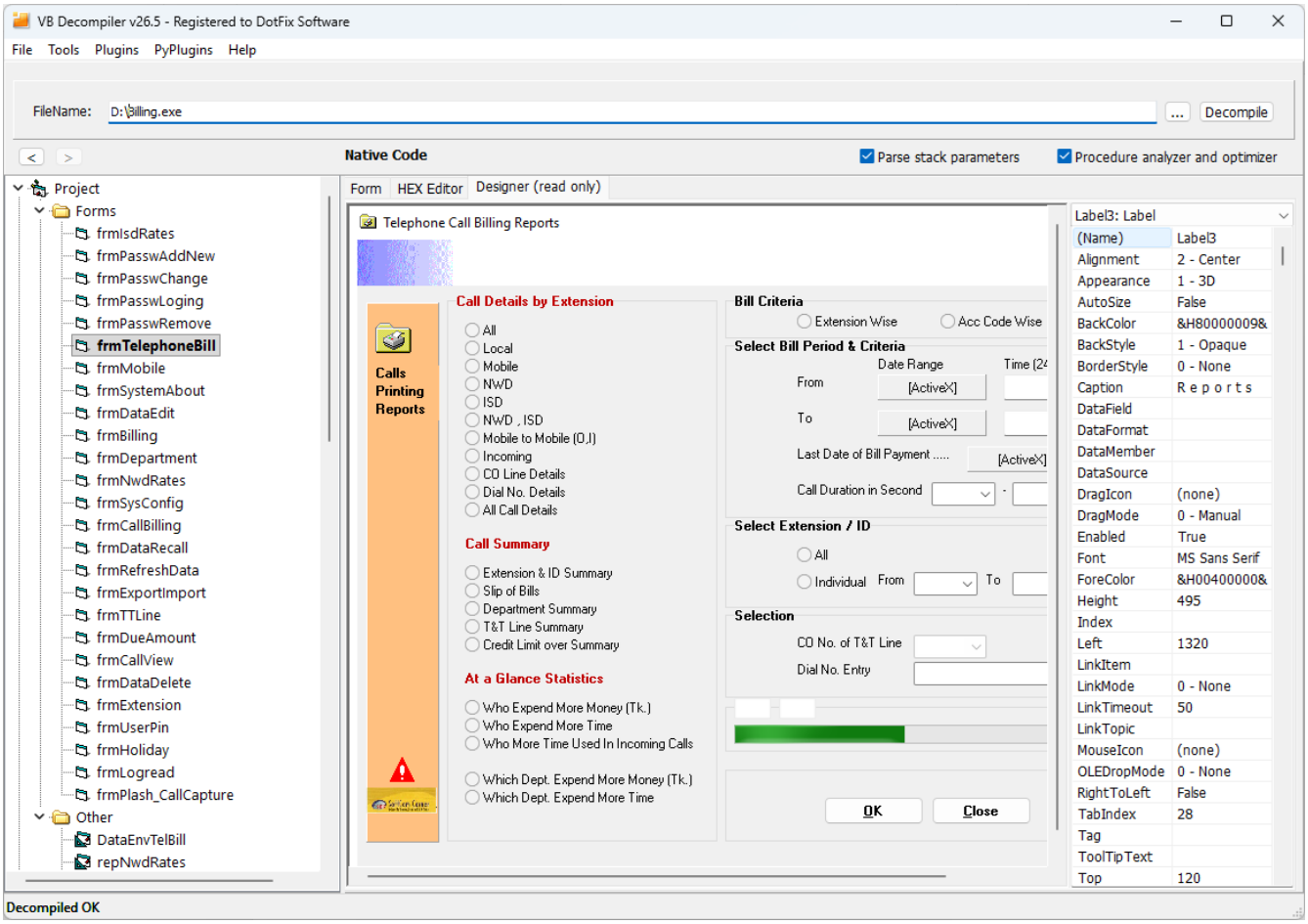
La vista previa se reconstruye a partir de la información almacenada. No es una instancia en ejecución de la interfaz original. Los controles no compatibles, el contenido creado en tiempo de ejecución, los recursos faltantes y la lógica de diseño personalizada pueden producir resultados incompletos o aproximados.

Visual Basic 5.0/6.0

El visor reconstruye formularios y UserControls, incluyendo propiedades de controles compatibles, fuentes, colores e imágenes. Lee los datos binarios del formulario utilizados por listas, cuadros combinados y otros controles. Las imágenes pueden incluir iconos, punteros, mapas de bits, JPEG, PNG y GIF.

La reconstrucción del menú visual soporta tanto aplicaciones en P-Code como en Native Code. Los controles compatibles incluyen `SSTab`, `MSComctlLib.StatusBar`, `MSComctlLib.ListView`, `MSComctlLib.ProgressBar` y `MSComctlLib.TreeView`.

Para inspeccionar el manejador de un control clásico de VB, seleccione el control y haga doble clic en su campo **_Event** en la lista de propiedades. Elija el evento requerido de la lista resultante para abrir el manejador.

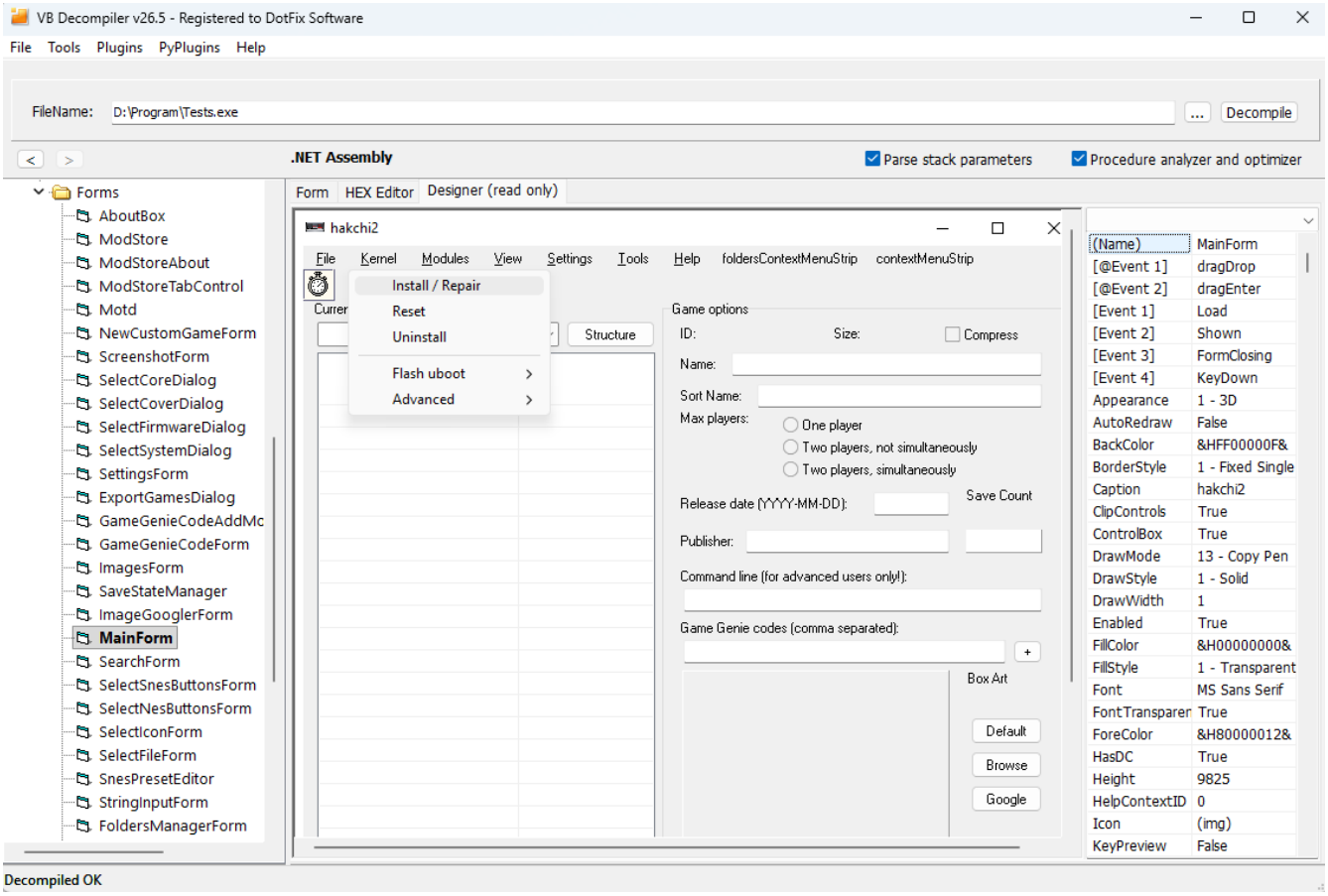


.NET Windows Forms

VB Decompiler analiza estáticamente `InitializeComponent` y reconstruye los controles de Windows Forms compatibles y su diseño. El analizador y el renderizador no requieren el runtime de .NET ni la ejecución de la aplicación objetivo. Las imágenes recuperadas compatibles también pueden mostrarse en los controles y fondos del formulario.

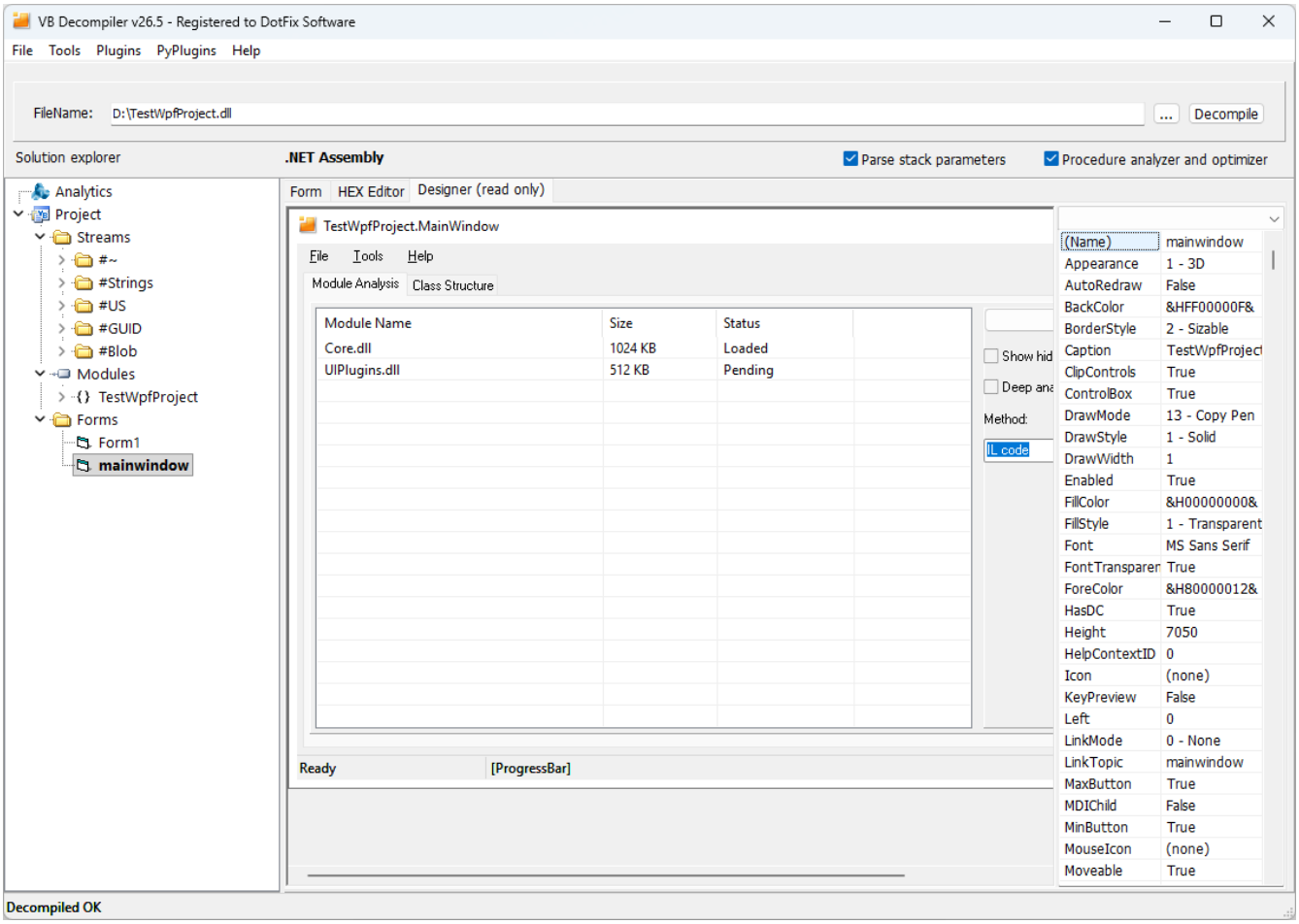
Seleccione un control para inspeccionar sus eventos recuperados, luego haga doble clic en un evento para abrir el manejador correspondiente. El visor reconstruye jerarquías de menús y menús contextuales; hacer clic en un elemento de menú recuperado abre su evento Click correspondiente. Los eventos personalizados de Windows Forms se muestran con el prefijo **@**.

El visor maneja el orden Z de los controles, colores de primer plano y fondo heredados, valores `Color.FromArgb`, banderas numéricas `FontStyle`, propiedades de recursos Booleanas y `MinimumSize` junto con `ClientSize`.



Ventanas, páginas y controles de usuario WPF

El visor reconstruye ventanas, páginas y controles de usuario WPF compatibles a partir de recursos XAML compilados almacenados como **BAML**.



Los contenedores de diseño compatibles incluyen **Grid** con tamaños de fila y columna fijos, Auto y proporcionales (*), **StackPanel**, **DockPanel** y **Canvas**, incluyendo diseños anidados. La reconstrucción tiene en cuenta los tamaños, márgenes, relleno, alineación, orientación, acoplamiento y extensiones de los controles sobre filas y columnas de Grid compatibles.

Más allá de los controles de entrada comunes, el visor puede reconstruir pestañas, cuadros de grupo, expanders, jerarquías de menús y separadores, barras de estado, columnas de ListView y datos estáticos compatibles, nodos estáticos de TreeView y elementos de cuadro combinado estático. Las interfaces Ribbon, paneles de navegación y paneles acoplados pueden tener representaciones simplificadas.

Donde están disponibles los datos de recursos, el visor recupera nombres, texto visible, textos de ayuda emergentes, fuentes, colores de fondo, estados de habilitación y visibilidad, e imágenes incrustadas, incluyendo transparencia PNG compatible y opacidad de fondo. Los controles personalizados no compatibles pueden aparecer como marcadores de posición que identifican el tipo WPF original.

El visor WPF vincula los eventos recuperados de los controles con sus controladores. Utilice la información de eventos del visor para abrir el código descompilado correspondiente.

Limitaciones de la reconstrucción: la vista previa visual es aproximada. Plantillas, estilos, vinculaciones, controles personalizados y contenido generado dinámicamente pueden representarse parcialmente.

Inspección y extracción de datos del formulario

Vea Formularios para la opción Mostrar desplazamientos (Show offsets) e inspección binaria vinculada. Utilice el complemento FRX Image Viewer para inspeccionar y exportar recursos gráficos recuperados compatibles.

11 Trazado

Tracing features permite la emulación paso a paso de Native Code y P-Code de Visual Basic 5.0/6.0. Se requiere una licencia con funciones de trazado. El trazador utiliza la representación interna del procedimiento en el emulador; no ejecuta el procedimiento en la CPU de la estación de trabajo.

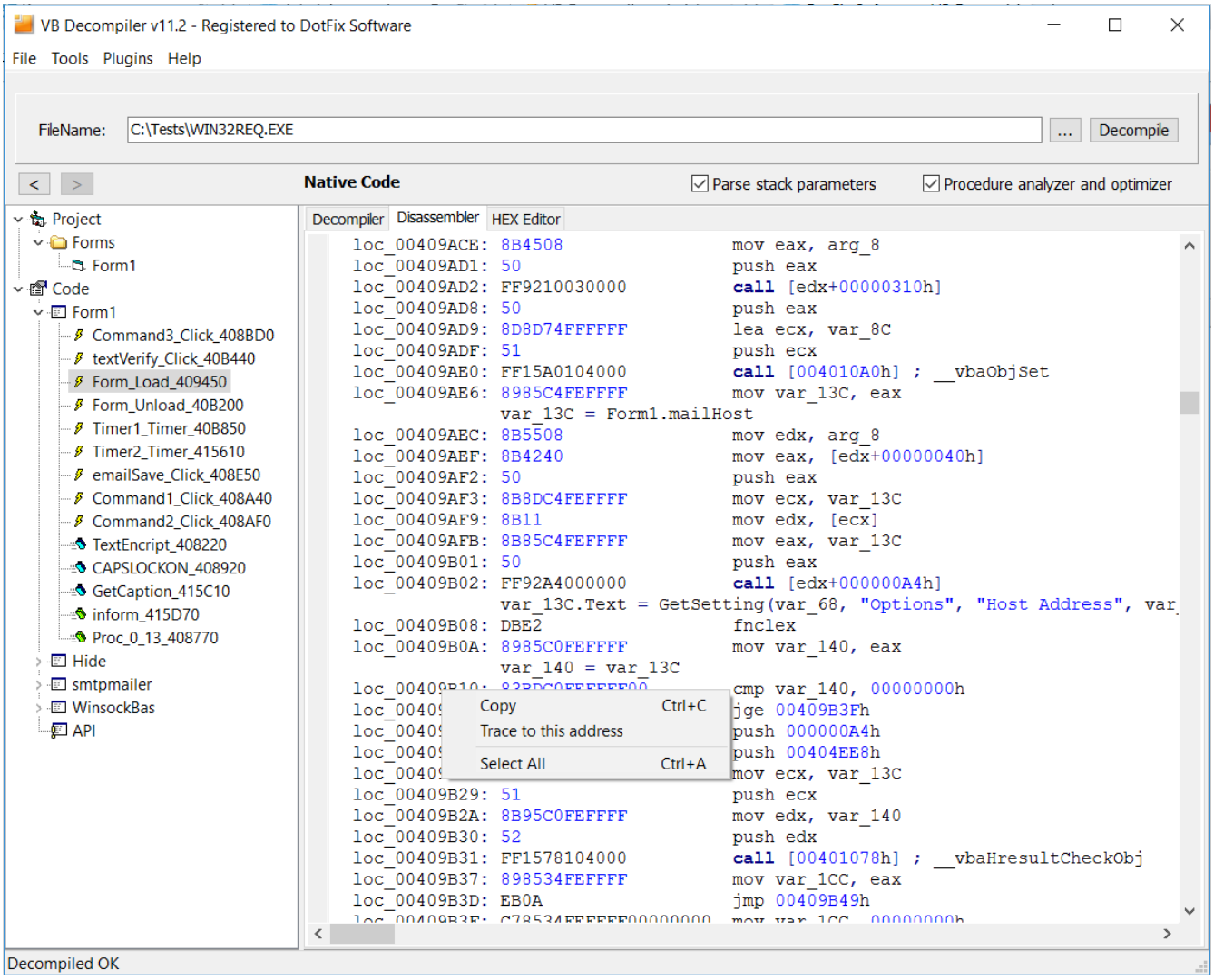
Iniciar un trazado

1 Abrir un procedimiento VB5/6 y mantener las direcciones visibles en la lista.

2 En la línea a investigar, invocar **Trazar hasta aquí (Trace to this address)** desde el menú contextual del código, o presionar **F4**.

3 Inspeccionar las visualizaciones de código y estado del trazador y avanzar utilizando **Paso (Next)**. Cancelar el trazado cuando se haya terminado.

A partir de la versión 26.4, el comando puede resolver la dirección inicial `loc_` después de eliminar los espacios en blanco al inicio de la línea. Puede colocar el cursor en otra parte de esa línea en lugar de exactamente en su etiqueta de dirección.



Estado del Código Nativo

El trazador expone registros de CPU y FPU, la pila, variables y el código reconstruido a medida que avanza la emulación. Utilice estas vistas para examinar cómo se derivó un valor o expresión. La búsqueda en la lista de variables y la visualización del tipo recuperado del objeto actual pueden ayudar con procedimientos más grandes.

Trazado de P-Code

El trazado de P-Code, introducido en la versión 11, muestra cómo evoluciona la información de las variables y se generan las sentencias descompiladas durante el procesamiento. Puede ayudar a explicar las diferencias entre las operaciones intermedias de P-Code y la lista de alto nivel optimizada.

Alcance y limitaciones

- El trazado se aplica a un procedimiento individual. Comienza con un estado vacío de los registros emulados, la pila y las variables, en lugar del estado de un proceso en ejecución.
- Las instrucciones de Código Nativo no compatibles pueden omitirse. Las llamadas a funciones externas no se ejecutan; el manejo de convenciones de llamada compatibles puede ajustar la pila modelada sin reproducir el comportamiento de la función externa.
- El trazador no proporciona una visualización completa de las banderas del procesador. No interprete su estado como un rastro de ejecución de hardware completo.
- El modelo documentado es una emulación paso a paso hacia adelante, sin trazado inverso ni ejecución repetida de iteraciones de bucle.
- Este flujo de trabajo describe el trazado de VB5/6. La presencia de desensamblado .NET no implica un depurador de tiempo de ejecución .NET.

Utilice el trazado para investigar la reconstrucción del descompilador y los valores intermedios. Un valor emulado por sí solo no establece el valor producido por un proceso real con globales inicializados, dependencias externas y datos de entrada.

12 Funciones de análisis

Las licencias con **Analytic features** generan informes categorizados de las operaciones reconocidas en el programa analizado. Estos informes ayudan a los analistas de seguridad a identificar procedimientos que merecen ser investigados antes de leer toda la base de código. Soportan aplicaciones de Visual Basic 5.0/6.0 y, desde la versión 11.7, ensamblados .NET.

Generación de un informe

1 Habilite **Anál. prototipos (Analyze prototypes)** en **Herramientas > Opciones (Tools > Options)**.

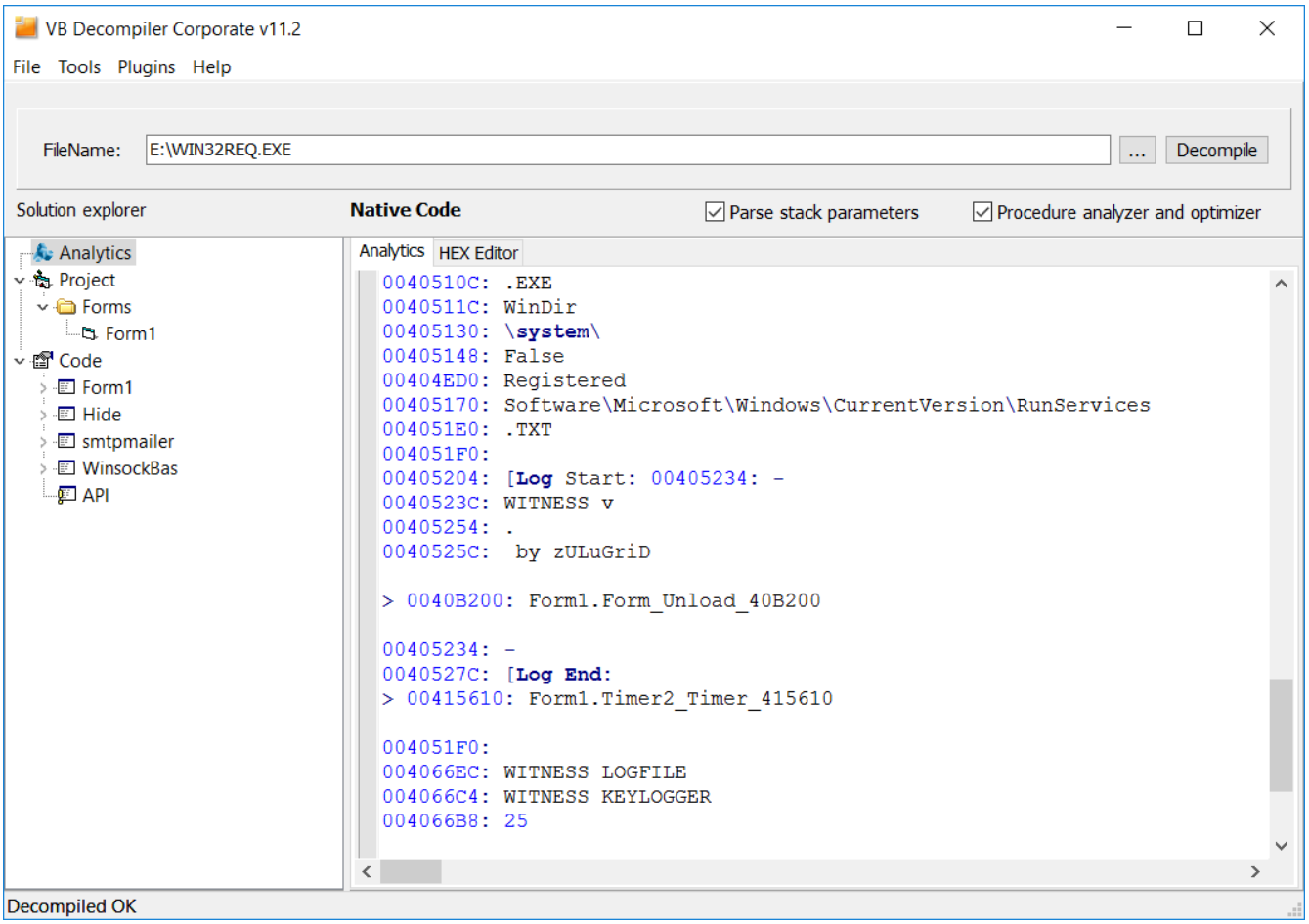
2 Abra el archivo y espere a que finalice el análisis. Con la licencia y los ajustes requeridos, el informe se genera y se muestra automáticamente.

3 Revise las categorías, luego abra los procedimientos relevantes e inspeccione su código y referencias de cadenas.

Las categorías cubren operaciones reconocidas como acceso a archivos y volúmenes, acceso al registro, actividad de red, creación de procesos, operaciones de ventana y funciones internas seleccionadas de VB como `CallByName` y `VarPtr`.

Informe rápido y contexto de cadenas

El informe rápido lista direcciones y nombres de procedimientos que contienen operaciones reconocidas. El informe extendido añade las referencias de cadena disponibles desde esos procedimientos. Este contexto puede revelar rutas candidatas, claves del registro, nombres de host u otros valores que merecen ser examinados.



Las cadenas que están encriptadas, ofuscadas o generadas en tiempo de ejecución pueden estar ausentes. Una cadena asociada a un procedimiento es un contexto útil, pero por sí sola no prueba que se pase a una llamada API particular o que se ejecute una rama.

Navegación y guardado de resultados

Haga doble clic en el nombre de un procedimiento en el informe para abrir su código. Haga doble clic en una dirección anterior a un procedimiento o cadena para inspeccionar su ubicación en el Editor HEX. Utilice el botón **A** por encima del árbol de objetos para volver al informe en la posición anterior, y las flechas de navegación o **Esc** para volver a visitar procedimientos inspeccionados.

Seleccione y copie el texto del informe según sea necesario, o elija **Archivo > Guardar informe (File > Save analytics report)** para guardarlo para un análisis posterior.

Interpretación de hallazgos

El informe identifica patrones de código y referencias reconocidos estáticamente. Es una ayuda de triaje, no un registro de ejecución observada ni un veredicto automático de que un archivo es malicioso. Las llamadas indirectas, interfaces externas sin resolver, metadatos dañados, ofuscación y valores generados en tiempo de ejecución pueden limitar la cobertura. Confirme los hallazgos significativos en los procedimientos correspondientes, el desensamblaje y los datos del archivo.

Para una inspección más profunda de la reconstrucción de VB5/6, consulte Trazado. Para cadenas literales y sus ubicaciones de código, consulte Referencias a cadenas.

13 Limitaciones de la ofuscación

VB Decompiler, equipado con una tecnología de ofuscación única, oculta efectivamente los datos dejados por el compilador en nuestra investigación sobre la descompilación de **aplicaciones de Visual Basic 5.0 y 6.0**. Los lenguajes de programación modernos suelen almacenar cantidades significativas de metadatos e información de depuración dentro de los archivos compilados, lo que facilita a los crackers localizar las funciones deseadas dentro de su programa. Para contrarrestar esta amenaza, hemos desarrollado un ofuscador diseñado específicamente para aplicaciones **Visual Basic**.

El proceso de ofuscación en detalle

El ofuscador funciona descomponiendo el programa entero en sus partes constituyentes, incluyendo formularios, módulos, clases y controles como botones y campos de texto. Cada formulario se disecciona luego en controles individuales con eventos asociados. A continuación, se recopila información sobre funciones y procedimientos, así como eventos vinculados a cada objeto del programa. Durante este proceso, el ofuscador verifica las dependencias entre objetos y altera simultáneamente sus nombres para dificultar la determinación de qué función pertenece a un módulo o clase en particular.

Como resultado, encontrar formularios específicos como **frmTrial** o **frmRegistration** dentro del programa se vuelve significativamente más difícil.

Limitaciones de la ofuscación de aplicaciones Visual Basic 5.0/6.0

El algoritmo empleado por el ofuscador es complejo y diseñado para analizar dependencias de objetos, pero aún no soporta todo. En consecuencia, en algunos casos, necesitará preparar su programa previamente para el proceso de ofuscación. Los puntos discutidos a continuación se aplican exclusivamente a la ofuscación de aplicaciones Visual Basic 5.0 / 6.0.

1. El ofuscador no soporta referencias de tiempo de diseño a otros objetos, lo que puede provocar problemas durante la ejecución.

Por ejemplo, considere un escenario donde tiene un control **TreeView** que depende de un asociado **ImageList**. Si el nombre de este objeto **ImageList** está codificado directamente en las propiedades de su **TreeView**, cualquier intento de ofuscación resultará en errores, ya que la referencia original permanece intacta. Esta limitación también se aplica a componentes relacionados con bases de datos, como **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
```

Este problema se puede resolver fácilmente moviendo las asignaciones de tiempo de diseño al tiempo de ejecución, simplemente moviendo estas asignaciones al evento **Form_Load**:

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

La misma corrección debe aplicarse para cada instancia del mismo. Afortunadamente, generalmente solo hay unos pocos casos de este tipo en la mayoría de los proyectos.

2. No se admiten declaraciones de objetos mediante As New (llamadas LateCall)

Esta limitación típicamente no afecta a la mayoría de los proyectos VB6, pero es esencial tenerlo en cuenta. Si está trabajando con formularios como clases y creando instancias de formularios, puede estar familiarizado con este escenario:

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

En tales casos, el compilador VB6 inserta el nombre **frmMain** directamente en el código de la función **Form_Load**. Dado que nuestro ofuscador solo modifica datos y no código, el nombre incorporado en el código permanece sin cambios después de la ofuscación. Desafortunadamente, esto conduce a una terminación del programa con error.

Desafortunadamente, es imposible para nosotros soportar tales llamadas en el ofuscador de **VB Decompiler**, ya que necesitamos desensamblar y descompilar todo el código de todas las funciones. A pesar de tener esta capacidad, la descompilación es un proceso extremadamente laborioso. Para proyectos grandes, puede tomar más de media hora solo para descomponer toda la base de código. Además, también necesitamos analizar referencias a objetos y renombrarlas, lo que añade aún más tiempo de procesamiento.

Dadas estas limitaciones técnicas, decidimos no implementar la descompilación completa y análisis de código en el ofuscador de **VB Decompiler**, especialmente ya que esto solo abordaría el problema en casos poco frecuentes en aplicaciones Visual Basic 5.0 / 6.0 y no proporcionaría ningún beneficio para la mayoría de los clientes.

14 Soporte técnico

Si tiene preguntas sobre el programa y no puede encontrar respuestas en esta guía de ayuda, por favor contacte a nuestro equipo de soporte técnico. Esta opción solo está disponible para usuarios registrados. Para los usuarios no registrados, nos complace ayudarle con consultas relacionadas con la adquisición de licencias para **VB Decompiler**.

Adicionalmente, si encuentra algún error dentro del programa o tiene sugerencias sobre cómo mejorarlo, no dude en comunicarse con nuestro servicio de soporte técnico con su retroalimentación.

15 Registro

Para utilizar las capacidades completas de **VB Decompiler**, debe adquirir una **Business License**. Al comprar una Licencia comercial para desarrollador único o una Licencia empresarial, recibirá la capacidad de:

1. Recuperar cerca del 85% del código de aplicaciones típicas compatibles de **Visual Basic 5.0/6.0 P-Code**;
2. Recuperar hasta el 85% del código de aplicaciones típicas compatibles de **Visual Basic 5.0/6.0 Native Code**;
3. Descompilar ensamblados **.NET** típicos compatibles y no ofuscados en código similar a **C#**, con una recuperación de hasta el 95%;
4. Contar siempre con la última versión del programa: las actualizaciones son absolutamente gratuitas durante un año tras la compra de la licencia;
5. Recibir soporte técnico por correo electrónico durante un año tras la compra de la licencia;
6. Apoyar el desarrollo de este descompilador.

Adicionalmente, recomendamos considerar la adquisición de licencias especiales para analistas de malware y especialistas en informática forense:

1. licencia con características de trazado
2. licencia con características analíticas

Para adquirir una **licencia** del descompilador ahora mismo, seleccione la opción **Pida la versión Pro (Order Pro version)** en el menú **Ayuda (Help)** del programa. En este momento, debe estar conectado a Internet. Se abrirá una ventana de navegador hacia el sitio web del programa donde puede comprar la licencia.

La **Business License** del descompilador está diseñada para su uso en un solo equipo y únicamente por un **Single Developer** (titular de la licencia). Si necesita utilizar el programa en múltiples equipos, deberá adquirir una licencia separada para cada uno o considerar la adquisición de la **Business Enterprise license** (para un máximo de 20 equipos).

16 Acerca del desarrollador

El descompilador es desarrollado por **DotFix Software**.

Sitio web del producto: www.vb-decompiler.org
Correo de soporte técnico: admin@vb-decompiler.org