VB Decompiler

Manual do usuário

Português

Visual Basic 5.0/6.0 · .NET

DotFix Software

Sumário

1 Descrição 4

2 Início rápido 5

3 Janela principal 6

4 Visualização de código 8

5 Aprimoramento de código com IA 11

6 Abertura e salvamento 15

6.1 Abrir arquivo 15

6.2 Trabalhar com bancos de dados de análise 17

6.3 Salvar a lista de procedimentos 18

7 Funções do programa 19

7.1 Ir para um endereço 19

7.2 Pesquisa de strings 20

7.3 Referências a strings 21

7.4 Descompilação a partir de um endereço 23

7.5 Alteração de dados 24

7.6 Ofuscação 26

8 Opções do programa 27

8.1 Opções do programa 27

8.2 Formulários 28

8.3 Descompilação 29

8.4 Desassemblador 31

8.5 Esquema de cores 32

8.6 Recursos avançados 34

8.7 Seleção de idioma 38

9 Plugins 39

10 Reconstrução visual de formulários 43

11 Rastreamento 47

12 Recursos analíticos 49

13 Restrições da ofuscação 51

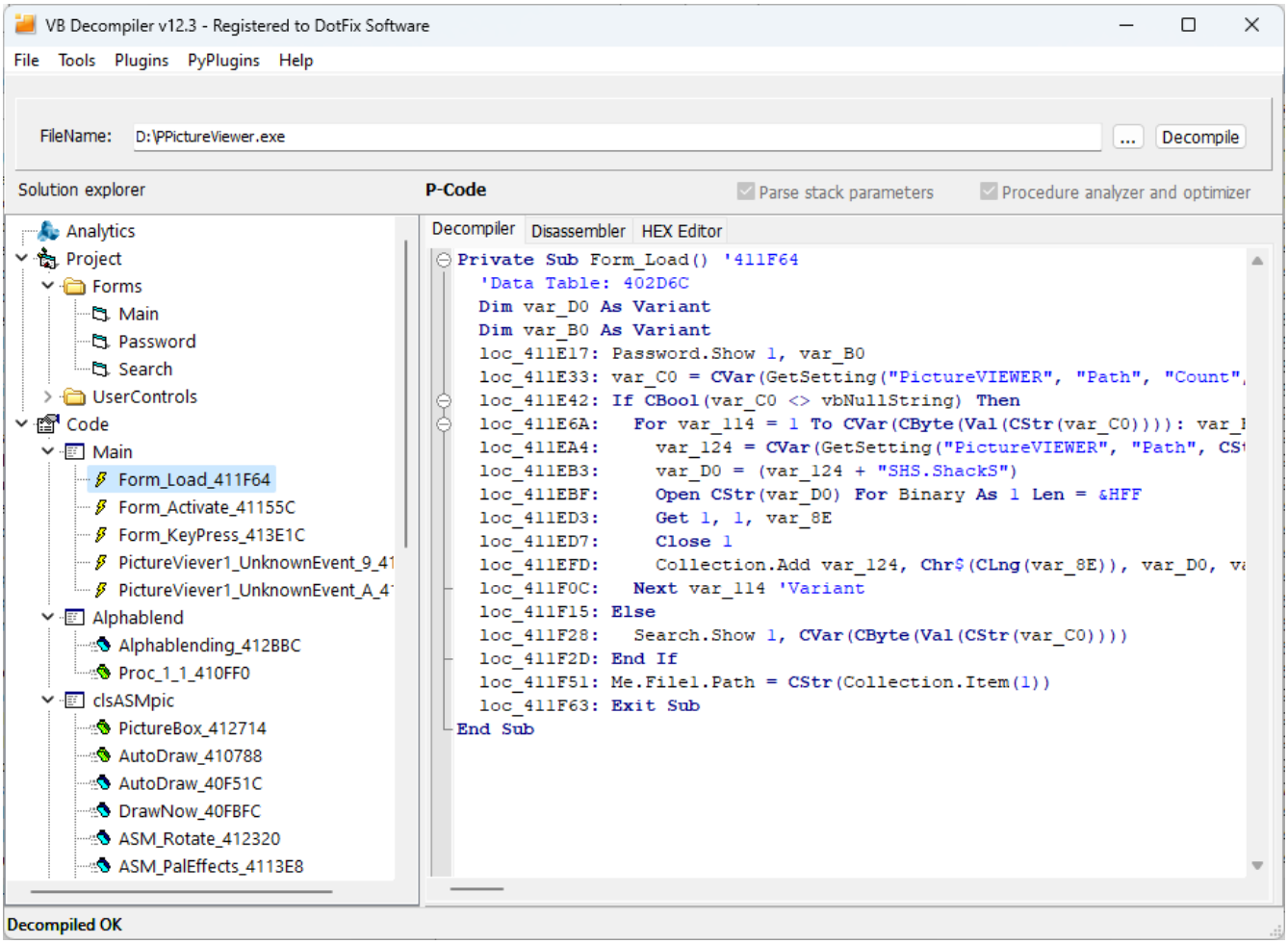
14 Suporte técnico 53

15 Registro 53

16 Sobre o desenvolvedor 54

1 Descrição

VB Decompiler é uma ferramenta de engenharia reversa e recuperação de código-fonte para programas Visual Basic 5.0/6.0 e assemblies .NET. Ele combina descompilação de alto nível, desassemblagem de baixo nível, inspeção de metadados, um editor HEX integrado e reconstrução de interface visual. Usos típicos incluem manutenção de software legado, resposta a incidentes, análise de malware e investigação de comportamento de aplicativos não documentados.



Representações de código suportadas

- **VB5/6 P-Code:** reconstrói instruções de Visual Basic a partir das instruções P-Code do compilador e das operações do runtime.
- **VB5/6 Native Code:** desassembla instruções x86 e usa um emulador interno, assinaturas de tempo de execução e heurísticas para reconstruir código semelhante ao Visual Basic.
- **.NET:** desassembla o Common Intermediate Language (CIL, também chamado de MS IL) e descompila padrões IL suportados em código semelhante a C#, independentemente da linguagem de origem .NET. Código x86 e x64 nativo em assemblies .NET também pode ser desassemblado.

O motor de análise .NET não requer o runtime .NET ou Reflection para análise de código ou reconstrução de formulário visual.

Ferramentas de análise

Use pesquisa de procedimentos e texto, referências cruzadas e renomeação de variáveis e Referências a strings para investigar o comportamento. O visualizador reconstrói formulários VB clássicos, Windows Forms .NET e janelas WPF, páginas e controles de usuário armazenados em recursos BAML compilados.

Licenças com recursos de Tracing fornecem emulação passo a passo do Código Nativo VB5/6 e P-Code. Licenças com recursos Analíticos fornecem relatórios categorizados para aplicativos VB5/6 e .NET. O Ajuda IA (AI Helper) pode processar código VB5/6 ou C# recuperado usando integração com Ollama ou a API OpenAI.

Interpretando o código recuperado

A compilação remove informações como comentários, muitos nomes de variáveis locais e alguns metadados de tipo e procedimento. Otimização e ofuscação podem alterar ainda mais a estrutura do código. A listagem reconstruída auxilia a análise e a recuperação de código. Verifique as conclusões importantes com base em desassemblagem, metadados, referências de string e procedimentos circundantes.

2 Início rápido

1. Selecione **Arquivo > Abrir programa (File > Open program)** e abra o arquivo EXE, DLL ou OCX para análise. O processamento inicia automaticamente. Aguarde a árvore do projeto ser preenchida e que o processamento termine.
2. Abra um procedimento em **Explorador (Solution explorer)**. Compare as visualizações **Descompilador (Decompiler)** e **Disassembler**; use o **Editor HEX (HEX Editor)** para inspecionar os bytes correspondentes.
3. Pressione **Ctrl+T** para encontrar um procedimento por seu nome ou parte de seu nome. Use **Ctrl+F** para pesquisar texto exibido, ou **Ferramentas > Referências de texto (Tools > String references)** para investigar strings e os procedimentos que as utilizam.
4. Abra um formulário recuperado e selecione **Editor (só leitura) (Designer (read only))** para uma pré-visualização. Para VB clássico, Windows Forms e WPF, use eventos recuperados para navegar até o código do manipulador correspondente.
5. Salve a análise através de **Arquivo > Base de dados (File > Data Base)** quando precisar retomá-la mais tarde ou compartilhar os resultados com outro analista.

Escolhendo um modo de descompilação

Descompilação Rápida processa procedimentos quando você os abre. Ative **Análise protótipos (Analyze prototypes)** para uma análise preliminar adicional das assinaturas de procedimentos privados. Para pesquisar todo o texto descompilado ou exportar um projeto completo, desative a Descompilação Rápida e processe o arquivo novamente.

Salvando código-fonte recuperado

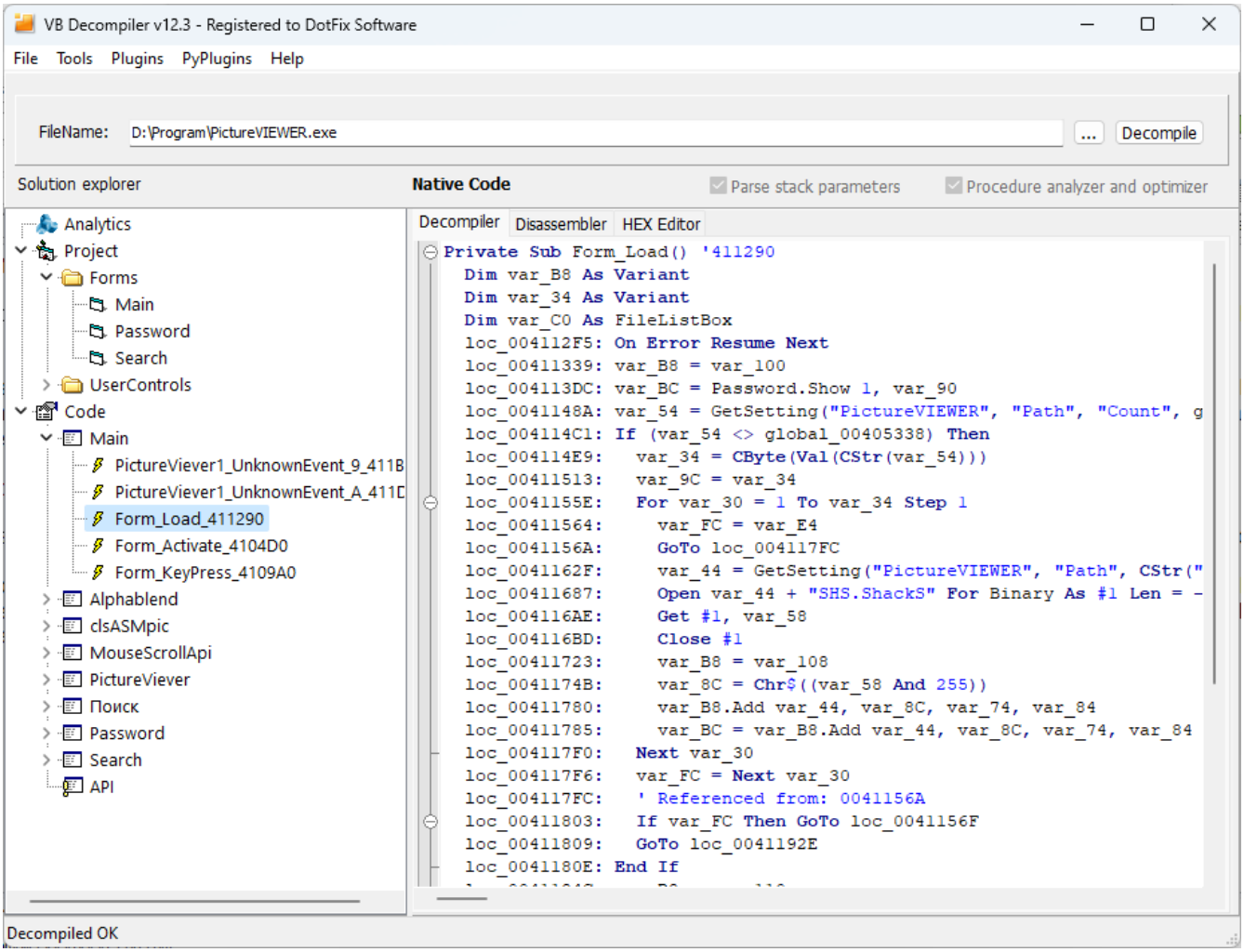
Para exportar os resultados da descompilação VB5/6, selecione **Arquivo > Salvar projeto descompilado (File > Save decompiled project)** e escolha uma pasta de saída. A exportação inclui formulários e módulos recuperados; arquivos FRX contêm imagens associadas e outros dados binários do formulário.

Antes de abrir o código exportado em uma IDE, corrija manualmente todas as imprecisões da descompilação e as incompatibilidades presentes no código-fonte recuperado.

Veja Reconstrução visual de formulário, Visualização de código e Aprimoramento de código com IA para os fluxos de trabalho detalhados.

3 Janela principal

Explorador (Solution explorer) organiza o projeto recuperado em formulários, módulos, classes e procedimentos. A visualização adjacente exibe o código, formulário ou metadados selecionados. A opção **Estilo de janela Visual Studio (Window style as in Visual Studio)** posiciona a árvore à direita e o código à esquerda.



Visualizações de código e dados

- **Descompilador (Decompiler):** código reconstruído de alto nível.
- **Disassembler:** as instruções x86/x64 correspondentes, P-Code do VB ou IL .NET, com endereços.
- **Editor HEX (HEX Editor):** os bytes subjacentes do arquivo, com navegação por endereço virtual ou deslocamento de arquivo.
- **Editor (só leitura) (Designer (read only)):** um formulário reconstruído e suas propriedades recuperadas, quando suportado.
- **Ajuda IA (AI Helper):** processamento opcional de IA do procedimento atualmente descompilado.

As abas disponíveis dependem do objeto selecionado. O indicador de tipo de código distingue **P-Code**, **Native Code** e **.NET Assembly**.

Analisar parâmetros do stack (Parse stack parameters)

Para P-Code, este controle combina operações de pilha e chamadas em expressões com argumentos. Desativá-lo expõe operações intermediárias e pode ajudar quando um protótipo desconhecido consome valores inesperados da pilha. Para código Nativo, ativá-lo usa o emulador para reconstruir expressões de nível superior; desativá-lo expõe a saída em nível de assembly. Para .NET, o processamento da pilha contribui para a reconstrução do IL como expressões semelhantes ao C#.

Analisador e otimizador de procedimento (Procedure analyzer and optimizer)

Este controle simplifica a lista reconstruída combinando expressões e padrões reconhecidos de fluxo de controle e removendo o código auxiliar do compilador que foi reconhecido. Aplica-se a código VB5/6 e .NET. O otimizador .NET pode reconstruir padrões suportados `while`, `do...while` e `foreach`, condições aninhadas, expressões de operadores e padrões de manipuladores de strings interpoladas. Essas transformações são realizadas pelo próprio descompilador e não requerem o Assistente de IA.

Se uma expressão ou ramificação for ambígua, compare a saída com a otimização desativada e examine a desassemblagem. No modo **Descompilação rápida (Fast decompilation)** sem cache, esses controles podem ser alterados enquanto você inspeciona um procedimento. Com descompilação completa ou saída em cache, processe o arquivo novamente após alterar as configurações.

Navegação e acesso via teclado

Os botões de seta acima da árvore navegam entre procedimentos visualizados anteriormente; **Esc** retorna ao procedimento anterior. Os botões Back (Voltar) e Forward (Avançar) do mouse usam o mesmo histórico, incluindo a posição lembrada em um procedimento.

Pressione **Alt+1** para alternar o foco entre o Explorador e a visualização ativa. Pressione **Enter** na árvore para abrir o objeto selecionado. No Editor HEX, **Tab** move entre os campos hexadecimal e de texto e depois para o próximo controle. Os rótulos da interface estão associados aos seus campos de entrada para navegação por leitores de tela.

Veja Visualização de código para atalhos contextualizados, referências cruzadas e renomeação, e Recursos analíticos para a visualização Análise (Analytics).

4 Visualização de código

As visualizações de código combinam realce sintático, blocos expansíveis, navegação baseada em endereço e referências cruzadas. Use a listagem de alto nível para entender um procedimento e a desassemblagem e bytes do arquivo para verificar como aquele resultado foi obtido.

Realce e navegação de identificadores

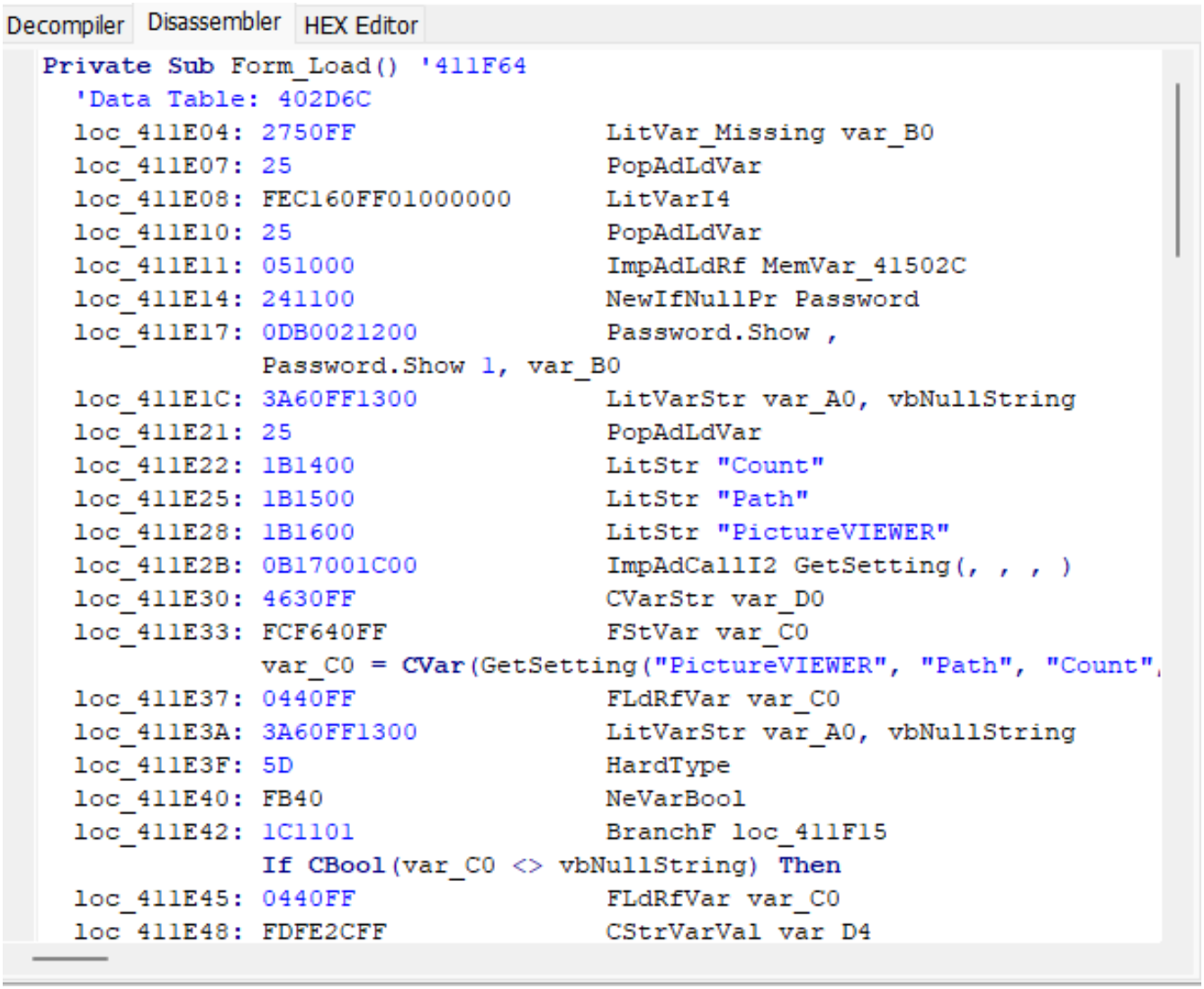
Coloque o cursor sobre um registrador, variável ou nome de função para realçar ocorrências correspondentes na listagem atual. Variáveis locais, como `var_XX` e `str_XX`, também são realçadas quando usadas como base de uma expressão de membro, por exemplo `str_1.Value`. Selecionar um identificador `global_XX` realça suas ocorrências dentro do procedimento atual; este realce é separado de uma pesquisa de referência em todo o projeto.

Clique duas vezes em uma chamada de procedimento reconhecida para abrir o procedimento chamado. Constantes semelhantes a endereços podem, em vez disso, levar aos bytes correspondentes no Editor HEX. Use **Esc**, as setas de navegação ou os botões Voltar/Avançar do mouse para navegar pelo histórico de visualização.

Combinação de código descompilado e desassemblado

Clique com o botão direito na listagem descompilada e selecione **Copiar para disassembler (Copy to disassembler)**, ou pressione **D** na visualização de código. Isso funde as instruções reconstruídas com a listagem de baixo nível para o procedimento atual. Para preservar sua posição, coloque o cursor no endereço relevante `loc_` antes de invocar o comando.

Mantenha **Adicionar Endereços (Add Addresses)** ativado nas configurações de descompilação ao usar este fluxo de trabalho. O Desassemblador sempre exibe endereços; Adicionar Endereços controla a visualização Descompilador (Decompiler).



Renomeação de variáveis

Para VB5/6 P-Code e Native Code, clique com o botão direito em uma variável e selecione **Renomear var. (Rename variable)**, ou coloque o cursor sobre a variável e pressione **N**. Digite um nome de análise significativo. O descompilador verifica colisões de nomes e palavras-chave reservadas. Para reverter uma renomeação, use **Desfazer nome (Undo renaming)** ou **Ctrl+Shift+Z**. Este comando é independente do desfazer de edição de texto comum.

A renomeação torna a listagem recuperada mais fácil de analisar; ela não recupera um nome que foi removido pela compilação. Trate nomes atribuídos pelo analista como anotações ao comunicar descobertas.

Encontrar chamadores e referências a variáveis globais

Para encontrar chamadores, clique com o botão direito em um procedimento no Explorador (Solution explorer) e selecione **Buscar refer. (Find references)**, ou pressione **X** com aquele procedimento selecionado na árvore. Os resultados listam os procedimentos contendo chamadas reconhecidas a ele. Clique duas vezes em um resultado para inspecionar o chamador. Este fluxo de trabalho suporta VB5/6 e .NET; o suporte ao .NET foi adicionado na versão 12.4.

Para investigar uma variável global, coloque o cursor sobre seu identificador na visualização de código e pressione **X**. A mesma tecla, portanto, pesquisa referências de procedimento na árvore e referências a variáveis globais no código. Uma lista de referências reconhecidas não estabelece que não há usos adicionais indiretos, dinamicamente resolvidos ou ofuscados.

Atalhos sensíveis ao contexto

Foco	Atalho	Ação
Código	N	Renomear a variável sob o cursor (VB5/6).
Código	X	Encontrar referências à variável global sob o cursor.
Código	D	Copiar o código descompilado atual para o Desassemblador.
Código	F4	Rastrear até o endereço na linha atual; requer recursos de Rastreamento.
Código	Ctrl+Shift+Z	Desfazer a renomeação da variável sob o cursor.
Árvore	X	Encontrar referências ao procedimento selecionado.
Árvore	G	Abrir a caixa de diálogo para ir a uma seção.
Árvore	Ctrl+C	Copiar o caminho completo da árvore do objeto selecionado.
Árvore	Space or Insert	Marcar ou desmarcar o item selecionado.

Atalhos de letra única dependem da visualização focada e não substituem a digitação normal em campos de entrada. Use **Alt+1** para alterar o foco. Veja Pesquisa de strings para Ctrl+T e Ctrl+F, Ir para um endereço para caminhos da árvore e endereços, e Rastreamento para limitações do emulador.

5 Aprimoramento de código com IA

Ajuda IA (AI Helper) processa código que o VB Decompiler já reconstituiu. Ele suporta C# descompilado de assemblies .NET e código Visual Basic 5.0/6.0 a partir de P-Code ou Native Code. Você pode usar um modelo local através do Ollama ou conectar-se a um serviço através da integração com a API OpenAI.

A IA pode sugerir fluxos de controle mais claros, nomes de variáveis, declarações e expressões. Ela não pode restaurar confiavelmente informações ausentes no binário nem reparar um fragmento que o descompilador interpretou incorretamente. Compare as sugestões com a listagem descompilada original, a desassemblagem e os procedimentos circundantes antes de usá-las em código-fonte recuperado ou em um relatório de investigação.

Processando um procedimento

1. Abra o procedimento necessário no Explorador (Solution explorer) e revise sua listagem em **Descompilador (Decompiler)**.
2. Mude para **Ajuda IA** e configure o provedor, o host e o modelo conforme descrito abaixo.
3. Inicie o processamento com o botão de ação ao lado do seletor de modelo.
4. Revise o código retornado no Ajuda IA e compare-o com a saída do descompilador.

O processamento com IA ocorre em uma thread separada. Use **Cancelar (Cancel)** se o modelo levar muito tempo para responder. As configurações adicionais de conexão podem ser recolhidas para deixar mais espaço para o código.

Usando um modelo local através do Ollama

1. Instale o Ollama em sua página oficial de download e inicie-o.
2. Baixe o modelo recomendado executando `ollama pull qwen3:8b` em um terminal.
3. Selecione o provedor local LLM/Ollama no Ajuda IA. Defina **Host Ollama (Ollama host)** para o endereço do servidor; o endereço local padrão é `http://localhost:11434`.
4. Selecione o modelo instalado em **Mod. (Model)** e inicie o processamento.

Recomendamos `qwen3:8b`. Os prompts integrados do Ajuda IA são adaptados a esse modelo leve. Modelos mais recentes podem levar mais tempo para processar o código sem melhorar o resultado.

A escolha do modelo não é restrita: é possível usar qualquer modelo instalado na instância do Ollama selecionada. Se o formato da resposta não for reconhecido, o analisador poderá deixar tags específicas do modelo ou outro texto auxiliar no resultado exibido.

Hardware para execução local com o Ollama

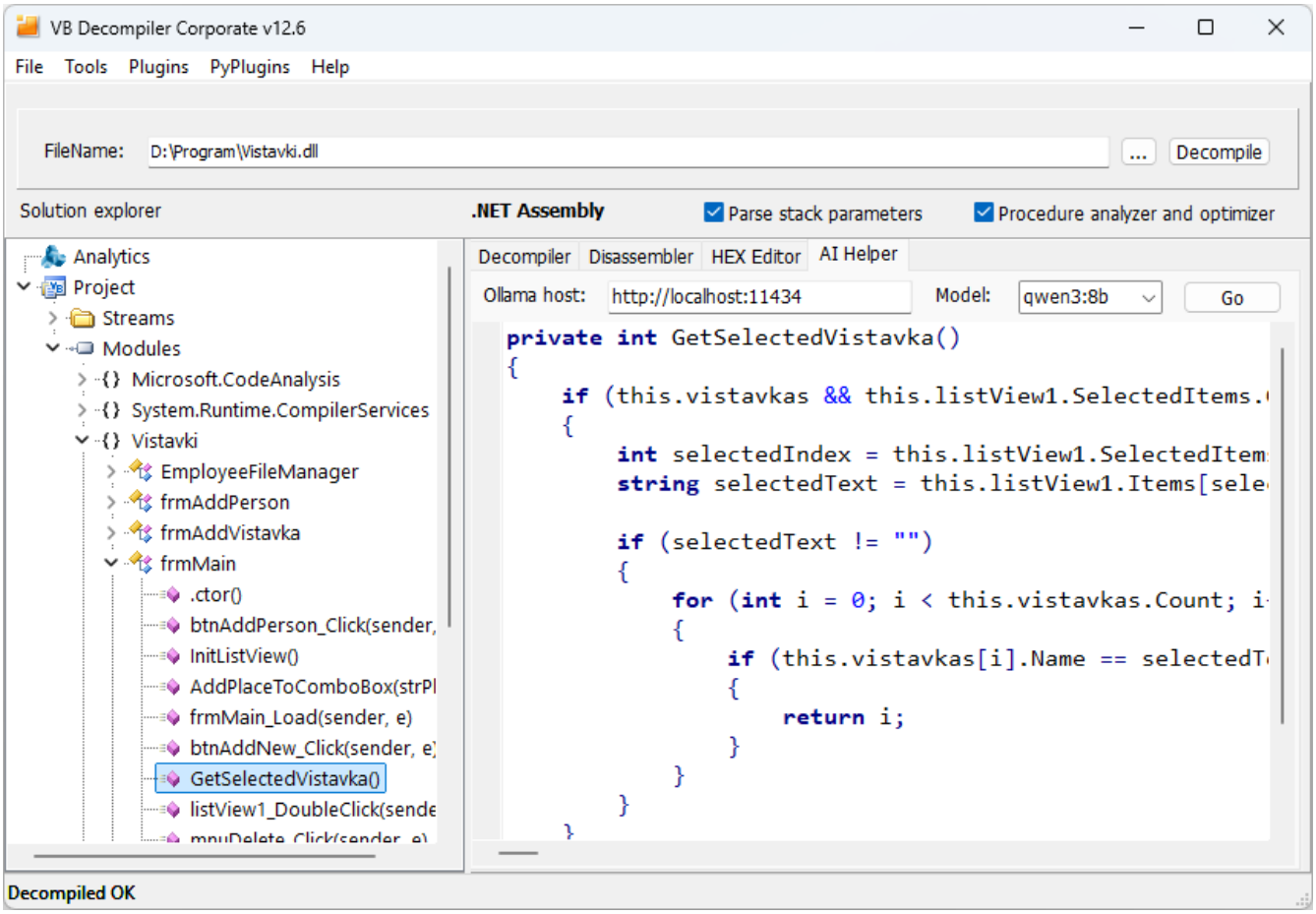
O processamento local com IA ocorre no computador que executa o Ollama e exige muito mais recursos que a descompilação. As configurações práticas abaixo consideram `qwen3:8b` com pesos `Q4_K_M`, cache de contexto padrão `f16` e uma solicitação por vez. O contexto é medido em tokens: 4K corresponde a 4096 tokens e 128K a 131072 tokens. Esse limite inclui as instruções, o código-fonte e a resposta gerada.

Componente	Mínimo prático: contexto 4K	Recomendado: contexto 128K
Sistema operacional	Windows 10 (22H2) ou Windows 11 de 64 bits	Windows 10 (22H2) ou Windows 11 de 64 bits
Processador	CPU x64 moderna com AVX2 e pelo menos 4 núcleos físicos	CPU x64 moderna com AVX2 e pelo menos 8 núcleos físicos
Memória do sistema (RAM)	16 GB	64 GB
Placa de vídeo	GPU compatível com o Ollama com pelo menos 8 GB de VRAM dedicada	GPU compatível com o Ollama com pelo menos 32 GB de VRAM dedicada
Armazenamento	SSD com pelo menos 20 GB livres para o Ollama e este modelo	SSD com pelo menos 20 GB livres para o Ollama e este modelo

Selecione o contexto pelo controle deslizante nas configurações do Ollama. Uma placa de 8 GB adequada para 4K não é suficiente para a configuração de 128K acima. O download do modelo ocupa cerca de 5.2 GB, mas o consumo total de memória é maior. Em 128K, somente o cache `f16` utiliza aproximadamente 18 GiB, além dos pesos e dos buffers de trabalho. Reserve memória para o Windows e outros aplicativos.

Com RAM suficiente, o Ollama pode usar a CPU, mas o processamento do código pode ser muito lento. Sistemas Pentium M e computadores de idade semelhante não são adequados para executar `qwen3:8b` localmente. A memória compartilhada das antigas Intel HD Graphics não substitui a VRAM dedicada indicada acima. Esses requisitos se aplicam ao computador com o Ollama; um servidor remoto ou serviço de API externo usa seu próprio hardware.

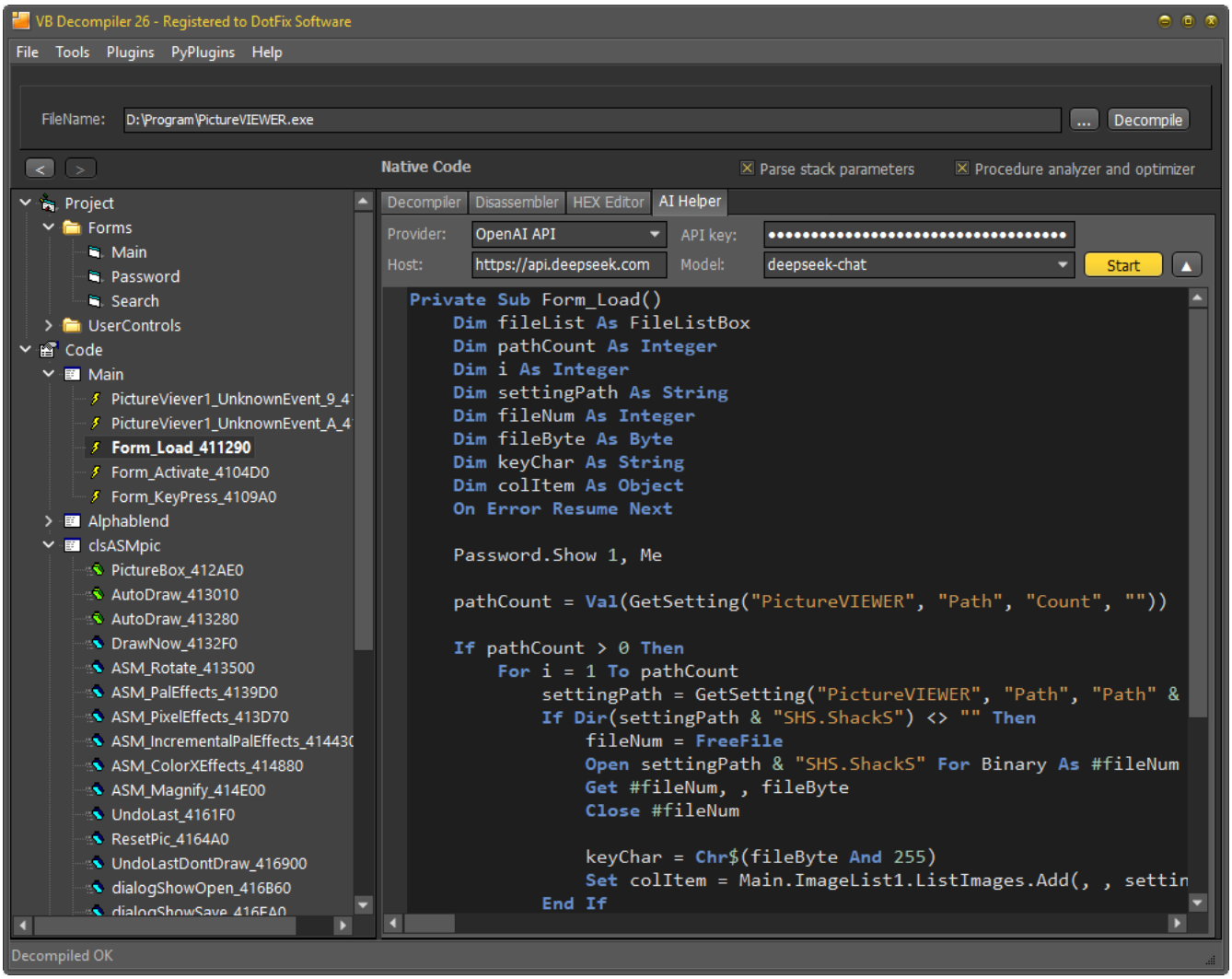
Para procedimentos curtos, comece com 4K. Um contexto maior aumenta o consumo de memória e o tempo de processamento de entradas longas. Se o processamento estiver muito lento, cancele-o, reduza o contexto e verifique o uso da GPU com `ollama ps`. O consumo real depende da versão do Ollama, do backend e das outras tarefas em execução.



Usando a integração com a API OpenAI

Defina **Serviço (Provider)** para **OpenAI API**. Configure **Host**, **Ch. API (API key)** e **Mod.** para o serviço pretendido. Insira uma chave de API válida e um identificador de modelo disponível para essa chave. Mantenha as chaves de API fora de capturas de tela e materiais de análise compartilhados.

O Ajuda IA também suporta o DeepSeek através do provedor **OpenAI API**. Insira o host do DeepSeek, a chave de API e o identificador de modelo nos campos correspondentes. A captura de tela abaixo mostra esta configuração.



Onde o código é processado

O Ajuda IA envia o código descompilado atual para o serviço configurado. Para um fluxo de trabalho apenas local, use um servidor Ollama executado localmente e um modelo executado localmente. Um host Ollama em outra máquina, um modelo Ollama na nuvem ou um serviço de API externo pode transferir o código fora da estação de trabalho de análise.

Para recuperação de código-fonte confidencial ou investigações de segurança, escolha o endpoint conforme os requisitos de tratamento de dados da sua organização. Não infira operação offline apenas pela presença do Ollama ou de uma URL localhost.

Solução de problemas

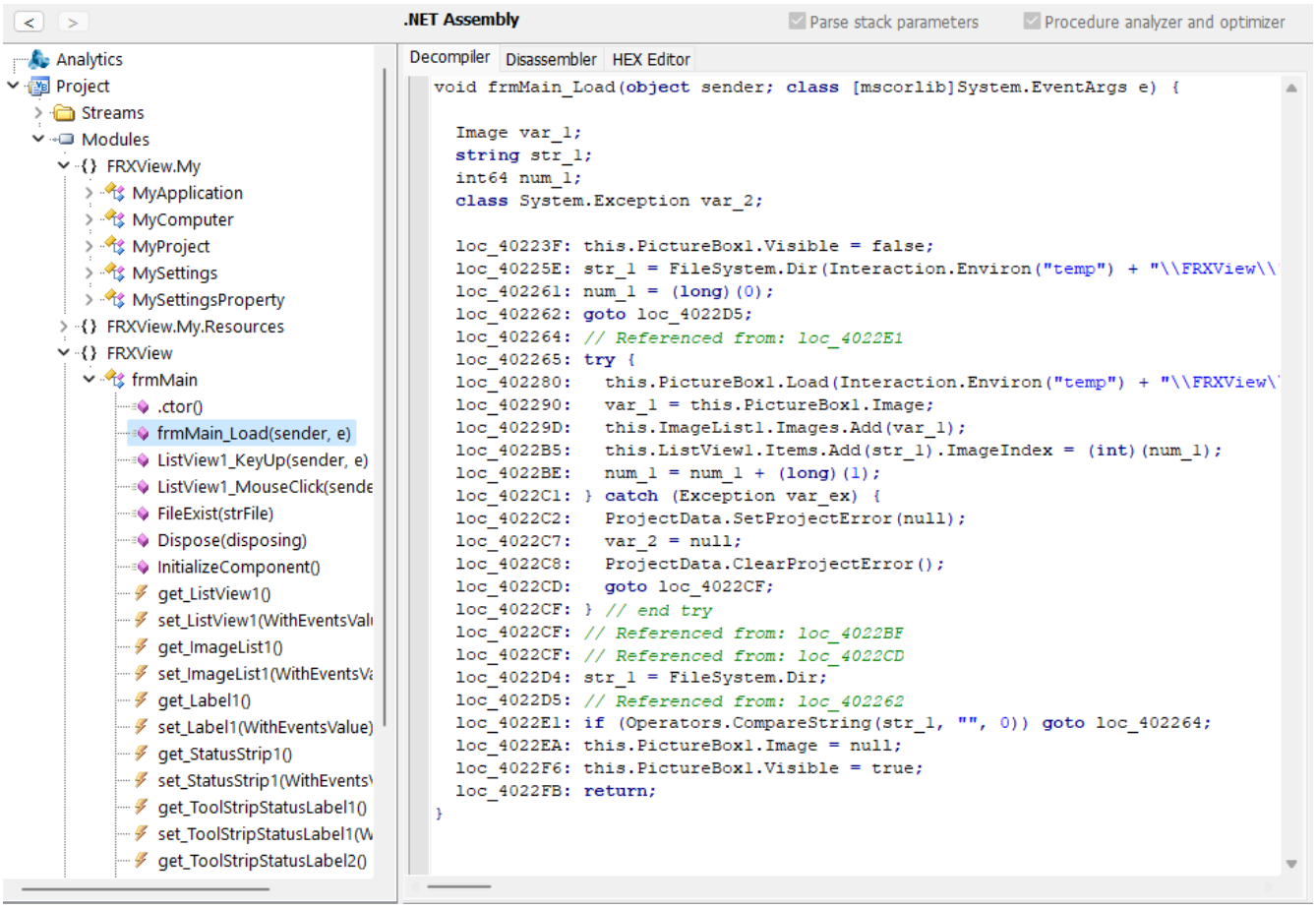
- Se um modelo local estiver ausente, confirme que ele está instalado na instância Ollama especificada pelo campo host.
- Se não houver resposta, verifique que o serviço selecionado está executando e acessível, e que o endpoint, credencial e identificador de modelo correspondem ao mesmo provedor.
- Se o processamento for muito lento, cancele-o. Ao usar outro modelo local, tente o modelo recomendado `qwen3:8b`.
- Se um modelo desconhecido deixar tags ou outros textos auxiliares em sua resposta, tente `qwen3:8b` ou remova esse texto manualmente enquanto revisa o resultado.
- Se o resultado alterar uma condição, chamada, constante ou tipo inesperadamente, compare-o com a listagem de baixo nível. A legibilidade por si só não estabelece comportamento equivalente.

O Ollama e os provedores de modelos externos são produtos de terceiros. Sua disponibilidade, comportamento dos modelos e políticas de serviço são independentes do VB Decompiler.

6 Abertura e salvamento

6.1 Abrir arquivo

Antes de iniciar o processo de descompilação, você precisa selecionar um arquivo para análise. Para isso, há uma seção **Abrir programa (Open program)** no menu **Arquivo (File)**. O **VB Decompiler** suporta a abertura de arquivos EXE/OCX/DLL criados em versões **Visual Basic** 5.0 ou 6.0, bem como aplicativos compilados em **Visual Studio .NET**. No último caso, a linguagem de origem não importa, pois o Visual Studio .NET compila aplicativos independentemente da linguagem original em um único formato interno - **MS IL**.



Após abrir o arquivo, o descompilador verifica se o programa pertence aos ambientes de desenvolvimento especificados. Se o programa aberto foi desenvolvido em um ambiente não suportado (por exemplo, **Delphi** ou **Visual C++**), uma mensagem aparece informando que tal ambiente não é suportado e que não é possível descompilar o programa. Caso essa mensagem apareça, você pode entrar em contato conosco, e podemos ajudar na descompilação do seu programa. Temos desenvolvimentos internos na área de descompilação de aplicativos **Delphi** e **FreePascal** e podemos oferecer serviços de descompilação para tais arquivos como serviço.

Se o programa aberto estiver compactado com um **compactador de EXE**, ofuscador ou protegido, o descompilador oferecerá a tentativa de descompactar este arquivo. Este recurso está disponível apenas em versões de Windows de 32 bits e é fornecido sem qualquer garantia. Se a descompactação for possível, o arquivo será executado, descompactado na memória, salvo no disco com um nome diferente e aberto para descompilação. **Atenção!** Nunca tente descompactar programas maliciosos ou qualquer outro código que possa prejudicar seu sistema usando o descompactador embutido, pois ao descompactar, o programa sendo aberto será executado e sua execução pode danificar o sistema ou os dados.

Observação importante: sob nenhuma circunstância você deve tentar descompilar ou descompactar software sem ter as autorizações necessárias, e se o titular dos direitos autorais proibir a descompressão e análise deste arquivo por licença. Isso pode violar as leis do seu país.

*Visual Studio .NET, Visual C++ e Visual Basic são marcas registradas da Microsoft Corporation. Delphi é uma marca registrada da Embarcadero.

Outras formas de abrir arquivos

Você pode arrastar um arquivo do Explorer para o VB Decompiler. Arquivos com extensão desconhecida ou sem extensão também podem ser submetidos dessa maneira; o reconhecimento ainda depende do conteúdo real do arquivo. Atalhos Windows **.lnk** suportados podem ser resolvidos em seus arquivos referenciados. Bancos de dados VDD salvos também podem ser abertos por arrastar e soltar ou a partir da linha de comando.

6.2 Trabalhar com bancos de dados de análise

Esta funcionalidade está disponível através da seção **Base de dados (Data Base)** no menu **Arquivo (File)** e serve para habilitar o salvamento dos resultados da descompilação em um arquivo comprimido separado. Isso pode ser útil para carregar os resultados da descompilação mais rapidamente no futuro, sem a necessidade de re-descompilar o código.

Caso você precise retornar à análise do programa ou compartilhar esses resultados com um colega, não há necessidade de realizar a descompilação novamente. No caso de uma descompilação completa (com a opção **Descompilação rápida (Fast decompilation)** desativada), esse processo pode exigir mais tempo e recursos. É muito mais conveniente pré-salvar o código descompilado no **Database**.

VB Decompiler utiliza um formato de banco de dados comprimido interno, que permite uma otimização significativa do espaço necessário para armazenar os resultados. Além disso, há compatibilidade com bancos de dados salvos pela maioria das versões anteriores do VB Decompiler, permitindo que você armazene informações sem correr o risco de não conseguir abrir tal banco de dados na nova versão do produto.

Abrindo e transferindo bancos de dados VDD

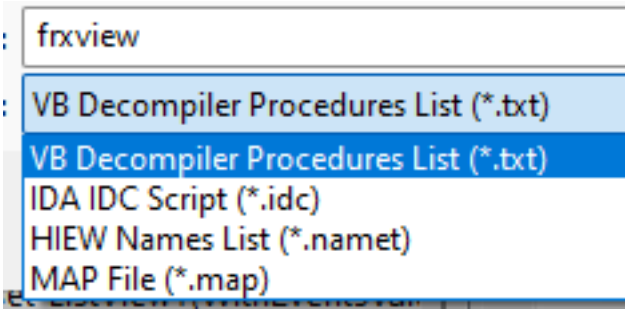
Bancos de dados de análise salvos utilizam a extensão **.vdd**. Você pode abrir um banco de dados VDD através do fluxo de trabalho de abertura de arquivos, arrastá-lo para a janela do descompilador ou passar seu caminho na linha de comando.

Quando localiza o binário original analisado, o VB Decompiler verifica tanto seu caminho salvo quanto o diretório que contém o banco de dados. Para colaboração, mantenha o binário correspondente ao lado do arquivo VDD quando seu caminho original não estiver disponível. Preserve a versão exata do arquivo analisado para que os endereços e bytes dos arquivos permaneçam significativos.

6.3 Salvar a lista de procedimentos

Esta operação é realizada chamando a opção **Salvar lista de procedimentos (Save procedures list)** no menu **Arquivo (File)** e é uma ferramenta poderosa para análise de programas, pois permite que utilitários externos complementem o **VB Decompiler**.

Atualmente, endereços, nomes de funções e procedimentos utilizados no programa processado podem ser salvos em vários formatos. Você pode escolher qualquer uma dessas opções especificando o tipo de arquivo na caixa de diálogo padrão de salvar.



Salvar em formato TXT permite salvar a lista de procedimentos em um arquivo de texto simples. Isso pode ser útil para comparar procedimentos entre diferentes versões de um programa, identificar alterações nos endereços e assim por diante.

Salvar como script IDC permite carregar os endereços das funções e procedimentos no desassemblador **IDA Pro**. Em casos onde o programa analisado é compilado em **Native Code**, este recurso pode ser muito útil.

Salvar em formato NAMET permite simplificar a análise usando a ferramenta **HIEW**. Copie o arquivo obtido para a pasta do programa analisado. Em seguida, abra o programa no HIEW e pressione a tecla F12. Uma janela com uma lista vazia de procedimentos aparecerá. Pressione a tecla F12 novamente se o endereço do arquivo NAMET corresponder ao criado – basta pressionar Enter. Os endereços dos procedimentos serão carregados no HIEW, permitindo navegar pelos métodos usando a lista pressionando a tecla F12.

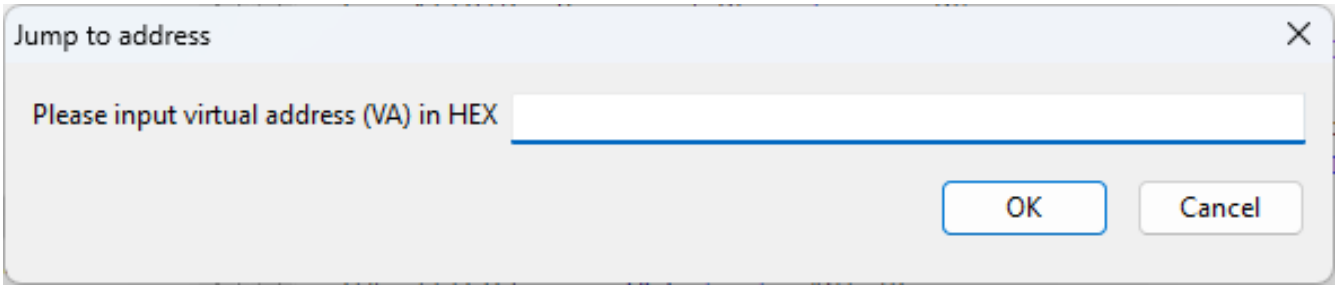
Salvar em formato MAP gera um arquivo de mapa padrão que geralmente é criado por compiladores. Este arquivo pode ser usado com muitos depuradores e desassembladores para marcar o início das funções do programa. Como programas escritos em Visual Basic são frequentemente desassemblados incorretamente por software de terceiros, o arquivo MAP gerado em **VB Decompiler** simplificará muito o processo. Este formato é suportado através de plugins em vários produtos como **OllyDbg**, **IDA Pro** e outros.

7 Funções do programa

7.1 Ir para um endereço

Pular para um endereço virtual

Selecione **Ferramentas > Ir para endereço virtual (Tools > Jump to virtual address)** ou pressione **Ctrl+G**. Digite o endereço virtual do código a ser inspecionado. Você também pode selecionar um endereço na lista e invocar Ctrl+G. O descompilador abre o código associado a esse endereço.



Um endereço virtual (VA) identifica uma localização no espaço de endereços da imagem executável e inclui sua base de imagem. Um deslocamento de arquivo identifica uma posição de byte no arquivo no disco. Não substitua um pelo outro: o Editor HEX suporta ambas as formas de navegação.

Retornar através do histórico de navegação

Use os botões de seta acima da árvore para mover-se para trás e para frente através dos procedimentos visualizados anteriormente. **Esc** retorna ao procedimento anterior. Os botões Back e Forward do mouse também usam esse histórico, mantendo a posição dentro do procedimento.

Compartilhar uma localização por seu caminho na árvore

No Explorador (Solution explorer), clique com o botão direito no objeto necessário e selecione **Copiar o nome da seção (Copy the name of section)**, ou pressione **Ctrl+C** enquanto a árvore tem o foco. Isso copia o caminho completo para o nó selecionado.

Para retornar a essa localização, selecione **Ir à seção (Jump to the section)** no menu de contexto da árvore, ou pressione **G** na árvore, e cole o caminho copiado no diálogo. Isso é útil ao passar uma investigação para outro analista ou retornar a um projeto grande. Use o caminho com o arquivo analisado correspondente; um caminho de outra compilação pode não identificar mais o mesmo código.

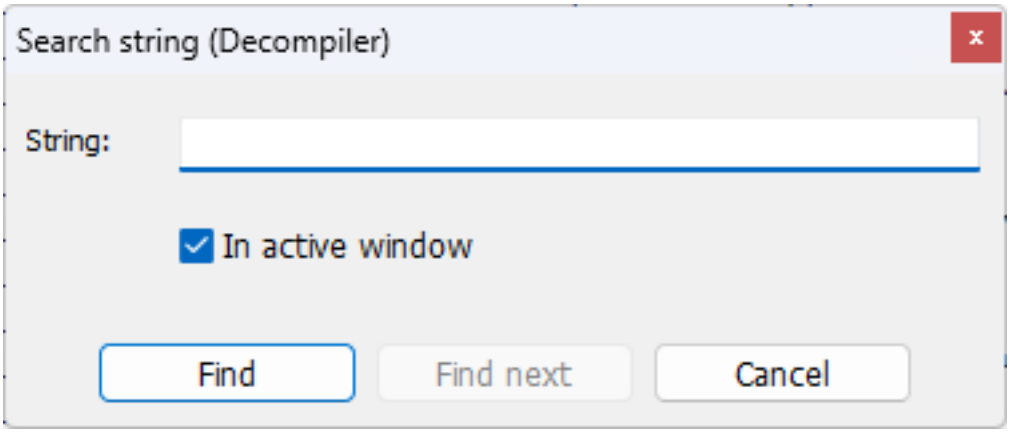
7.2 Pesquisa de strings

Encontrar um procedimento por nome

Selecione **Ferramentas > Buscar rotina (Tools > Find procedure)** ou pressione **Ctrl+T**. Digite o nome completo do procedimento ou uma parte distintiva dele. A pesquisa abrange a árvore do projeto, portanto você não precisa saber qual módulo ou classe contém o procedimento. Se vários nomes correspondem, a lista de resultados contém todas as entradas correspondentes; clique duas vezes na entrada necessária para abri-la.

Pesquisar texto exibido ou todo o projeto

Selecione **Ferramentas > Find string (Tools > Find string)** ou pressione **Ctrl+F** para abrir a caixa de diálogo de pesquisa de texto. Com **Na janela ativa (In active window)** selecionado, a pesquisa aplica-se ao código ou dados do formulário atualmente exibidos. Desmarque esta caixa de seleção para pesquisar em todo o projeto descompilado.



Digite o texto em **Texto (String)** e selecione **Localizar (Find)**. Use **Localizar próximo (Find next)** para continuar a pesquisa do mesmo texto. **Cancelar (Cancel)** fecha o diálogo.

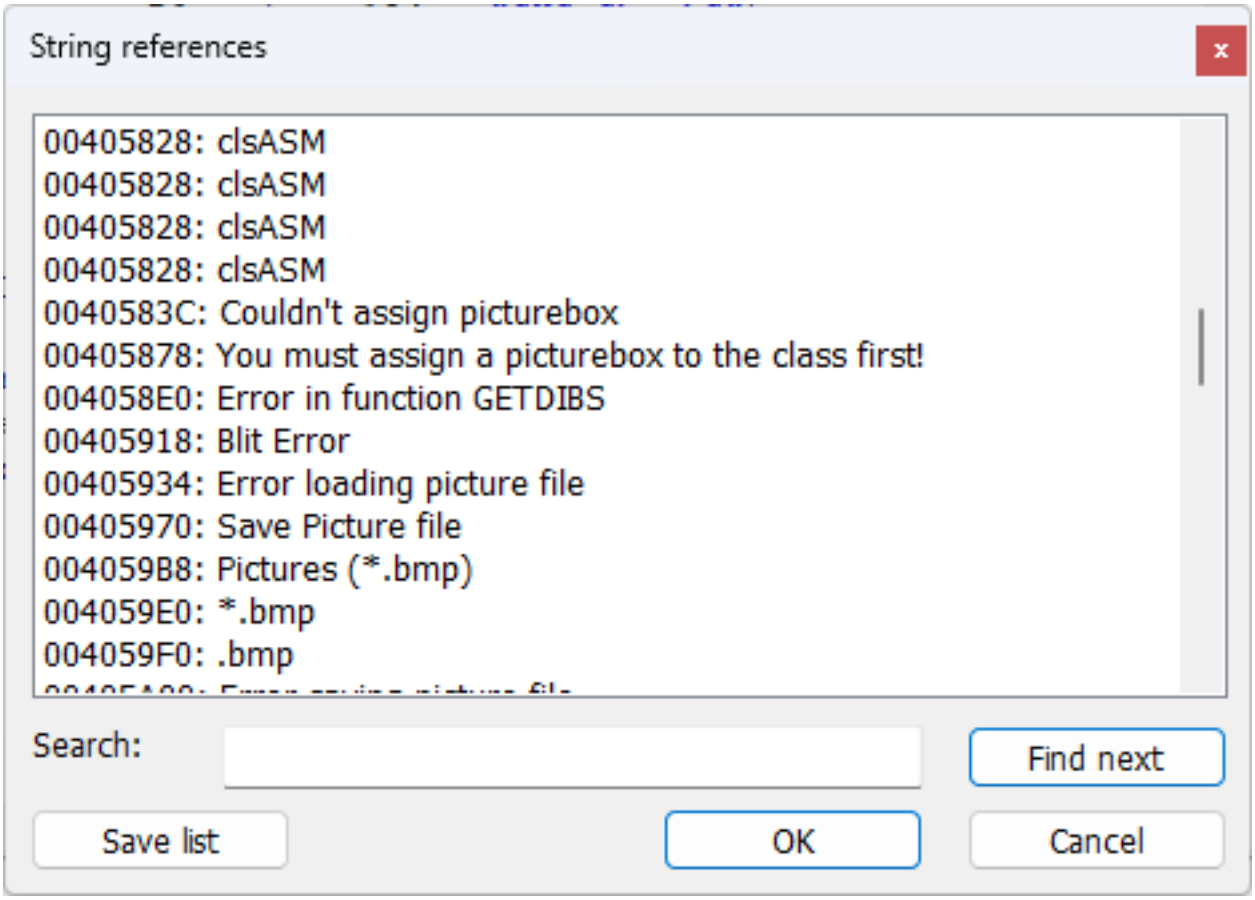
A pesquisa de todo o texto descompilado requer descompilação completa. Desabilize **Descompilação rápida (Fast decompilation)** em Recursos avançados e processe o arquivo novamente primeiro. A busca de procedimentos por nome é uma operação separada na estrutura do projeto.

Pesquisar referências de string

Se você precisa de strings literais e do código que as utiliza, abra Referências a strings. Seu campo **Procurar (Search)** e botão **Localizar próximo** pesquisam a lista de strings. As Referências a strings podem ser coletadas por desassemblagem preliminar no modo Descompilação Rápida; elas não exigem uma pesquisa de cada procedimento totalmente descompilado.

7.3 Referências a strings

No código de qualquer programa, você frequentemente encontra dados de string (chaves do registro, nomes de arquivos, mensagens de erro, etc.). Pode ser bastante útil ver a lista de strings utilizadas por procedimentos e funções em aplicativos descompilados. É exatamente para isso que nossa função **Referências de texto (String references)** foi projetada, acessível através do menu **Ferramentas (Tools)**. Para usá-la, basta navegar até **Ferramentas** -> **Referências de texto**, e se seu programa contiver dados de string, eles serão exibidos conforme mostrado abaixo:

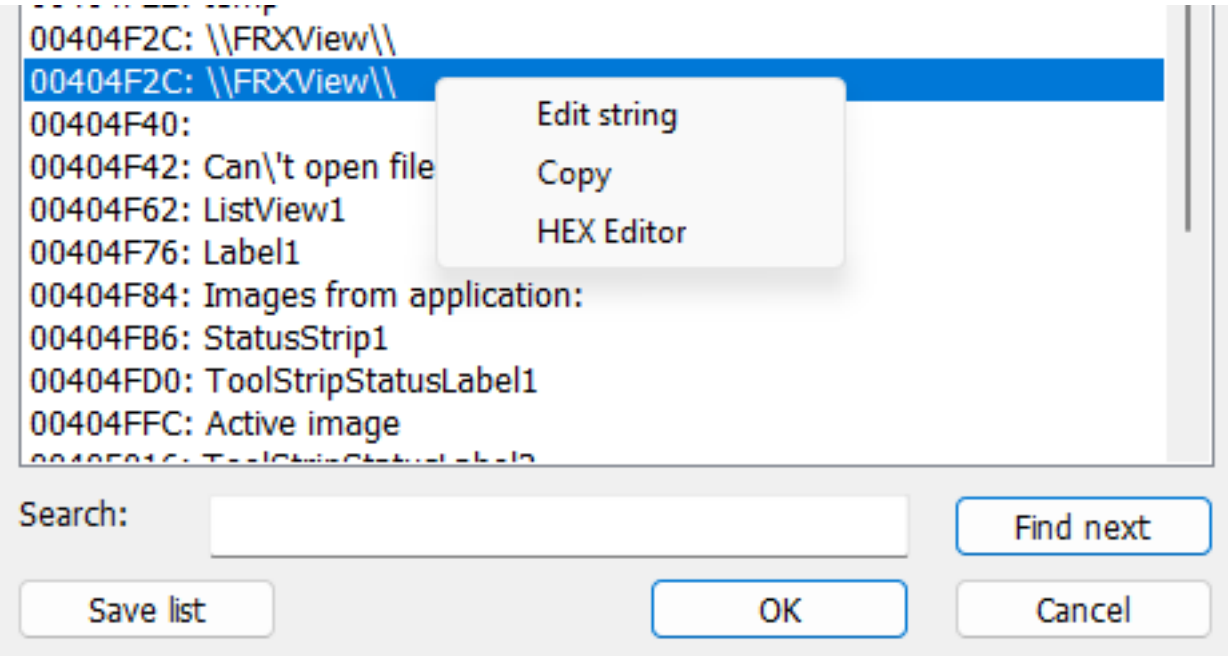


Encontre a linha desejada na lista e clique duas vezes nela. O segmento de código contendo aquela linha será localizado, e o procedimento correspondente será aberto na janela de código. Clicar duas vezes novamente na linha continua a busca por essa linha dentro do código do programa. Clicar em 'OK' encontrará a linha destacada no código e fechará o diálogo de seleção da linha. Cancelar simplesmente fecha a caixa de diálogo.

A função para exibir **Referências de texto** está disponível apenas na versão licenciada do software e não é acessível para a versão **Lite**.

Observação importante: as strings são exibidas não apenas para o módulo ativo, como nas versões 1.x do programa, mas para todo o programa. Essa mudança foi feita para facilitar a análise de programas com um grande número de formulários e módulos.

VB Decompiler também suporta a edição de strings. Para chamar o diálogo de edição de string, clique com o botão direito na string que deseja alterar e selecione **Editar texto (Edit string)** no menu de contexto que aparece. Uma caixa de diálogo para inserir uma nova string aparecerá. Após alterar a string, clique em **Ok** e a string será modificada diretamente no arquivo editado. Recomendamos criar um backup do arquivo sendo analisado previamente, porque nenhum backup é feito durante esta ação e cancelar essa operação não é possível.



Observação importante: ao editar strings, lembre-se de que o comprimento da nova string não deve exceder o comprimento da antiga. Isso está relacionado ao formato dos arquivos binários de aplicativos onde nenhum espaço é reservado para inserir uma string mais longa.

Você também pode visualizar e editar parcialmente a string no editor HEX. Para fazer isso, clique com o botão direito na string e selecione Editor HEX do menu que aparece. No editor HEX, existem funções para desfazer e refazer alterações, o que simplifica a experimentação ao modificar dados de string de aplicativos.

A janela com a lista de strings também possui um campo de texto chamado **Procurar (Search)** e um botão **Localizar próximo (Find next)**. Essa funcionalidade será útil para encontrar strings dentro da lista, especialmente ao lidar com projetos grandes onde a busca visual manual através da lista é extremamente difícil.

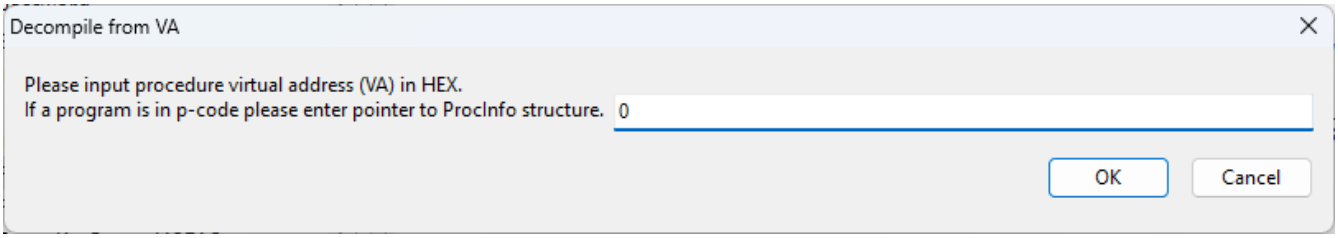
Descompilação Rápida e listas de strings grandes

As referências de string podem ser coletadas através de desassemblagem preliminar, incluindo no modo **Descompilação rápida (Fast decompilation)**. Isso difere da busca em todo o texto totalmente descompilado. A versão 26.2 expande o diálogo além de 32 000 referências e melhora o gerenciamento de memória para listas grandes.

Para assemblies .NET, strings do usuário do heap `#US` podem ser inspecionadas com suas localizações de arquivo. A lista representa strings disponíveis para análise estática; strings criptografadas ou geradas dinamicamente podem exigir investigação separada.

7.4 Descompilação a partir de um endereço

No menu **Ferramentas (Tools)**, há uma função muito útil - **Descompilar do VA (Decompile from VA)** (descompilar qualquer código a partir do endereço virtual fornecido). A principal característica desta função é que ela não exige que o arquivo sendo descompilado tenha uma estrutura correta e seja processado da maneira padrão. Naturalmente, neste caso, o código será descompilado com um número de limitações. Por exemplo, se funções importadas ou objetos OLE não foram pré-processados, o código será pouco informativo nesses pontos.



Para descompilar uma parte específica de código dentro de um arquivo, você simplesmente precisa especificar o endereço virtual desse código (se estivermos falando de **Native Code** ou **assembly .NET**), ou fornecer uma referência à estrutura **ProcInfo** (se estivermos falando do clássico **Visual Basic P-Code**). A singularidade de trabalhar com tais **P-Code** é que a estrutura **ProcInfo** contém um array de dados usados pelo código. Sem este array, é impossível determinar com quais funções o código trabalha e a quais strings ele se refere e assim por diante. No caso de trabalhar com **assemblies .NET**, após inserir o endereço, o descompilador oferecerá a você uma escolha entre descompilar o código como código intermediário **MS IL** ou desassemblar este código. Esta opção deve-se ao fato de que os assemblies .NET podem conter tanto código intermediário MS IL (**código gerenciado**) quanto código máquina (o chamado **código não gerenciado**).

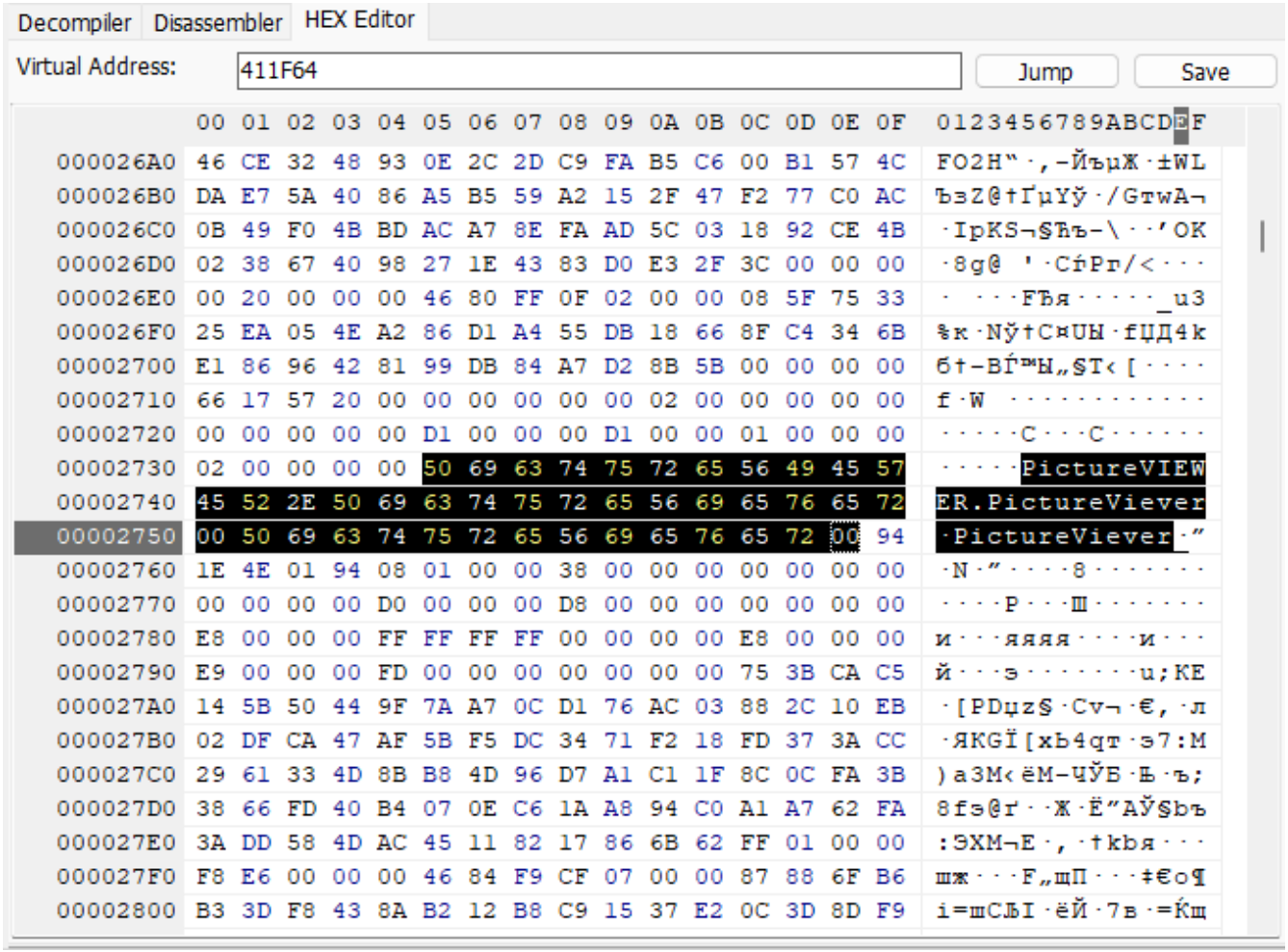
Se o arquivo já estiver aberto no descompilador, você só precisa chamar a função **Descompilar do VA** e especificar o endereço para processamento. Se o arquivo ainda não tiver sido aberto, isso será sugerido automaticamente, e o arquivo será aberto em modo simplificado.

A função **Descompilar do VA** é particularmente útil quando o código desejado não é descompilado automaticamente ou se o arquivo está parcialmente danificado, impedindo sua análise pelo procedimento padrão. Naturalmente, você deve entender que o resultado da descompilação neste caso será pior do que com uma descompilação padrão.

7.5 Alteração de dados

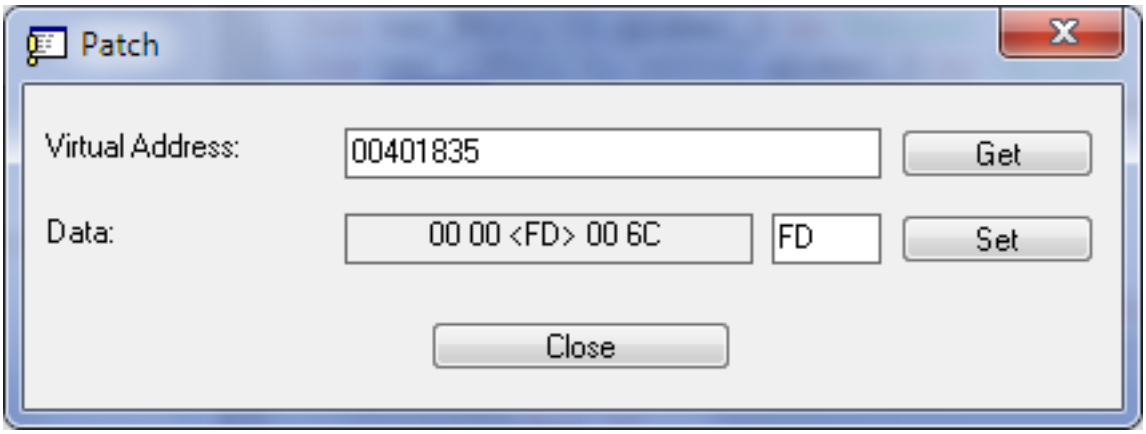
A partir da versão 9.4, o **VB Decompiler** vem com uma ferramenta moderna e conveniente para edição de dados binários dentro de arquivos - um editor hexadecimal completo **Editor HEX (HEX Editor)**. Ele é acessível a partir da janela principal do programa, ao lado do código descompilado na aba **Editor HEX** para cada procedimento ou formulário.

O **Editor HEX** permite editar os bytes do arquivo byte a byte, bem como em blocos. Para navegar até o endereço desejado dentro do arquivo analisado, você pode usar endereços virtuais (incluindo **Image Base**) ou deslocamentos (um endereço regular indicando o deslocamento dos dados necessários em relação ao início do arquivo).



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
000026A0	46	CE	32	48	93	0E	2C	2D	C9	FA	B5	C6	00	B1	57	4C	FO2H“ ·,-ЙъµЖ·±WL
000026B0	DA	E7	5A	40	86	A5	B5	59	A2	15	2F	47	F2	77	C0	AC	ЪзZ@†Ґ µYў·/GтwA¬
000026C0	0B	49	F0	4B	BD	AC	A7	8E	FA	AD	5C	03	18	92	CE	4B	·IpKS¬§Ћъ-\··’OK
000026D0	02	38	67	40	98	27	1E	43	83	D0	E3	2F	3C	00	00	00	·8g@ '·Cѓрг/<···
000026E0	00	20	00	00	00	46	80	FF	0F	02	00	00	08	5F	75	33	· ···FЂя·····_u3
000026F0	25	EA	05	4E	A2	86	D1	A4	55	DB	18	66	8F	C4	34	6B	%к·NўtCЊUЊ·fЏД4k
00002700	E1	86	96	42	81	99	DB	84	A7	D2	8B	5B	00	00	00	00	б†–BЃ™Њ„§Т‹[····
00002710	66	17	57	20	00	00	00	00	00	00	02	00	00	00	00	00	f·W ············
00002720	00	00	00	00	00	D1	00	00	00	D1	00	00	01	00	00	00	·····С···С·····
00002730	02	00	00	00	00	50	69	63	74	75	72	65	56	49	45	57	·····PictureVIEW
00002740	45	52	2E	50	69	63	74	75	72	65	56	69	65	76	65	72	ER.PictureViever
00002750	00	50	69	63	74	75	72	65	56	69	65	76	65	72	00	94	·PictureViever·”
00002760	1E	4E	01	94	08	01	00	00	38	00	00	00	00	00	00	00	·N·”····8·····
00002770	00	00	00	00	D0	00	00	00	D8	00	00	00	00	00	00	00	····Р···Ш·····
00002780	E8	00	00	00	FF	FF	FF	FF	00	00	00	00	E8	00	00	00	и···яяяя····и···
00002790	E9	00	00	00	FD	00	00	00	00	00	00	00	75	3B	CA	C5	й···э·····u;КЕ
000027A0	14	5B	50	44	9F	7A	A7	0C	D1	76	AC	03	88	2C	10	EB	·[PDџz§·Cv¬·€,·л
000027B0	02	DF	CA	47	AF	5B	F5	DC	34	71	F2	18	FD	37	3A	CC	·ЯКGЇ[xb4qт·э7:М
000027C0	29	61	33	4D	8B	B8	4D	96	D7	A1	C1	1F	8C	0C	FA	3B	)a3M‹ёМ–ЧЎБ·Ь·ъ;
000027D0	38	66	FD	40	B4	07	0E	C6	1A	A8	94	C0	A1	A7	62	FA	8fэ@ґ··Ж·Ё”АЎ§bъ
000027E0	3A	DD	58	4D	AC	45	11	82	17	86	6B	62	FF	01	00	00	:ЭXM¬E·,·†kbя···
000027F0	F8	E6	00	00	00	46	84	F9	CF	07	00	00	87	88	6F	B6	шж···F„щП···‡€o¶
00002800	B3	3D	F8	43	8A	B2	12	B8	C9	15	37	E2	0C	3D	8D	F9	і=шСЉІ·ёЙ·7в·=Ќщ

Em versões anteriores do descompilador (antes da versão 9.4), a função de edição de arquivos era aplicada usando uma ferramenta mais primitiva: Data Patcher. Ele funcionava da seguinte maneira. No menu **Ferramentas (Tools)** -> **Ajuste em dados (Patch data)**, havia um instrumento chamado patcher que podia ser usado em casos onde, após analisar o código do programa, era necessário alterar alguma parte. Para alterar os bytes necessários sem utilizar ferramentas externas, o **VB Decompiler** tinha funcionalidade embutida. A janela do patcher parecia-se com isto:



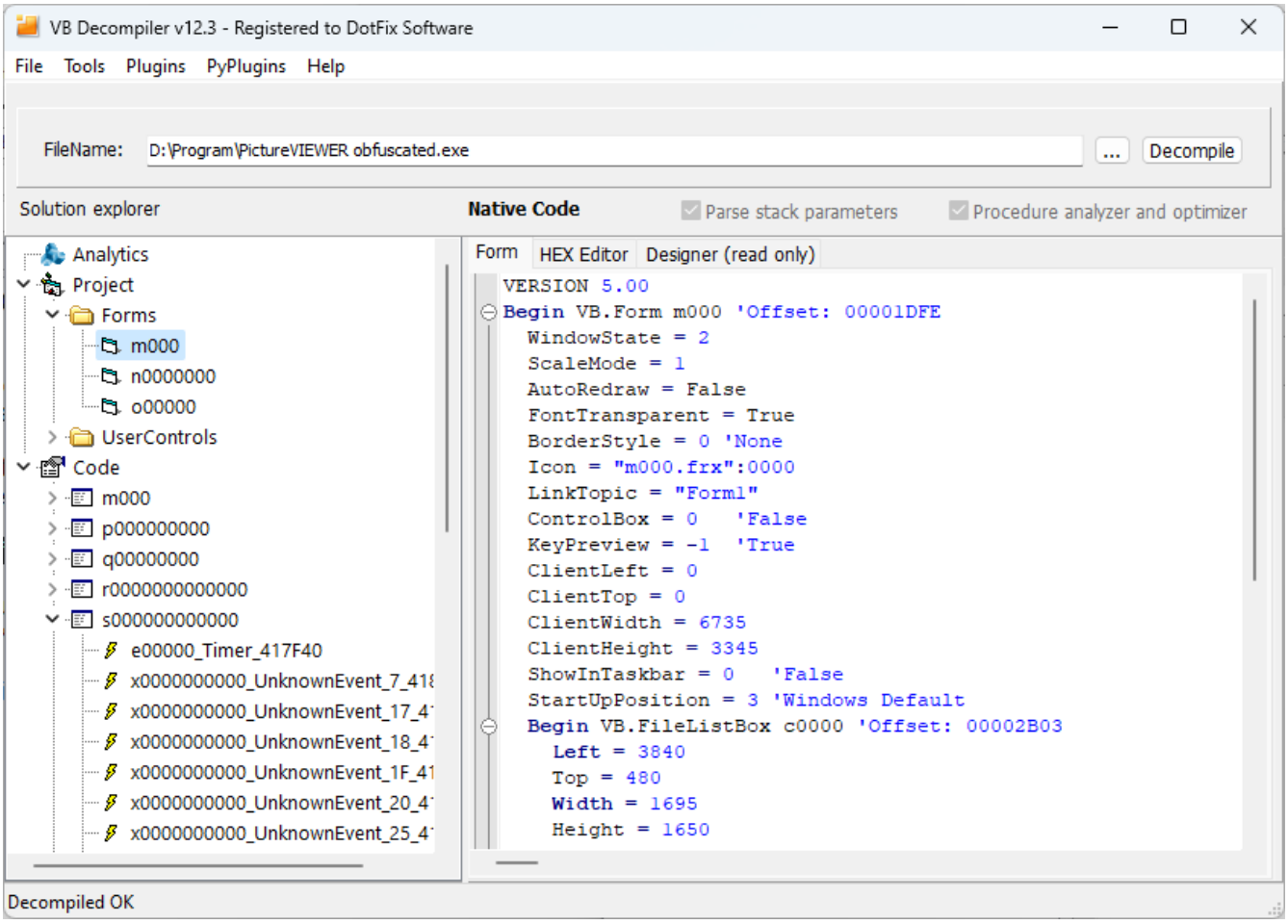
Trabalhar com o patcher era extremamente simples. No campo **Endereço Virtual (Virtual Address)**, você inseria o endereço para edição de um byte, em seguida pressionava **Enter** ou clicava em **Pegar (Get)**. Os bytes de dados localizados nas proximidades do endereço selecionado eram exibidos no campo Data, e o byte selecionado podia ser alterado, após o qual era necessário clicar em **Configurar (Set)**, e o byte inserido era alterado.

Se na opção Opções do Programa havia uma opção para **Descompilação rápida (Fast decompilation)** ativada e a opção **Código decompilado do Cache (Cache decompiled code)** desativada, então todas as alterações eram refletidas naquela função, onde os bytes foram modificados. Para fazer isso, era suficiente reabrir aquela mesma função clicando duas vezes no seu nome na árvore.

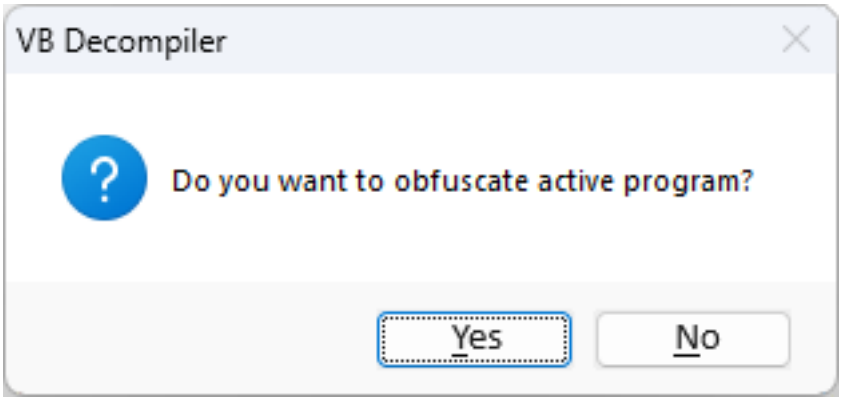
Nota importante: esta funcionalidade é projetada para edição apenas dos seus arquivos de programa ou daqueles para os quais você tem os direitos apropriados, bem como pesquisa de código malicioso em laboratórios de antivírus. Antes de alterar o código de outra pessoa - certifique-se de revisar o acordo de licença sobre a presença de tal direito.

7.6 Ofuscação

Se você estiver desenvolvendo programas em **Visual Basic 5.0/6.0**, pode aumentar sua proteção contra descompilação. Para isso, o **Obfuscator** é incorporado ao descompilador, capaz de remover todos os nomes de funções, formulários, módulos e objetos do arquivo binário do seu programa, bem como várias assinaturas. Isso tornará a análise do programa mais difícil, seja usando o descompilador fornecido ou outras soluções. Além disso, a maioria dos produtos de terceiros recusará a descompilação deste arquivo por completo.



Para aplicar essa proteção, abra seu projeto no descompilador e selecione **Ofuscar projeto (Obfuscate project)** no menu **Ferramentas (Tools)**. Este processo não leva muito tempo, por isso recomendamos ofuscar seus projetos antes de publicá-los.



Esta funcionalidade também está integrada à nossa solução abrangente para proteção de software - **DotFix NiceProtect**, que pode ser baixada de www.niceprotect.com.

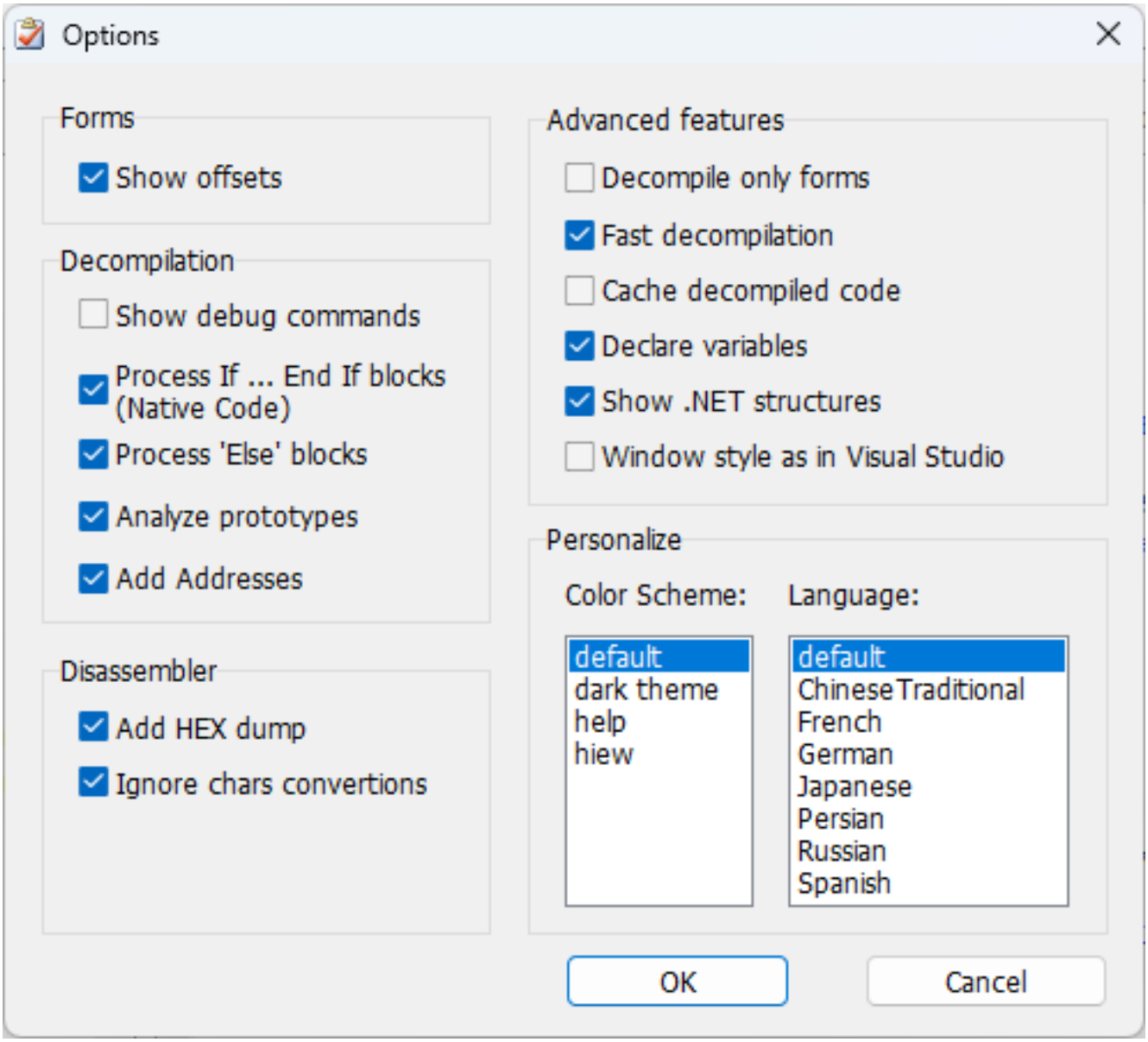
A ofuscação está disponível apenas para usuários com uma licença Business do **VB Decompiler**.

Restrições da ofuscação

8 Opções do programa

8.1 Opções do programa

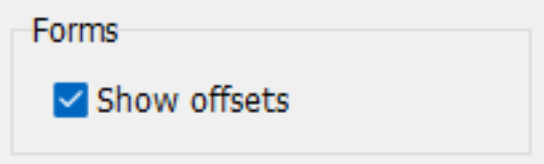
Para alterar as configurações do programa, vá para o menu **Ferramentas (Tools)** > **Opções (Options)**. No diálogo resultante, você verá as seguintes opções:



Formulários
Descompilação
Desassemblador
Esquema de cores Recursos avançados Seleção de idioma

8.2 Formulários

Você pode alterar as seguintes configurações de descompilação para formulários:



Nesta seção de configurações, você pode ativar a exibição de deslocamentos ao lado dos objetos de formulário. Isso pode ser necessário, por exemplo, ao localizar um programa em outro idioma sem acesso aos códigos-fonte. **Mostrar offsets (Show offsets)** permitirá que você encontre a localização dentro do programa onde esses objetos estão situados, seus nomes e propriedades.

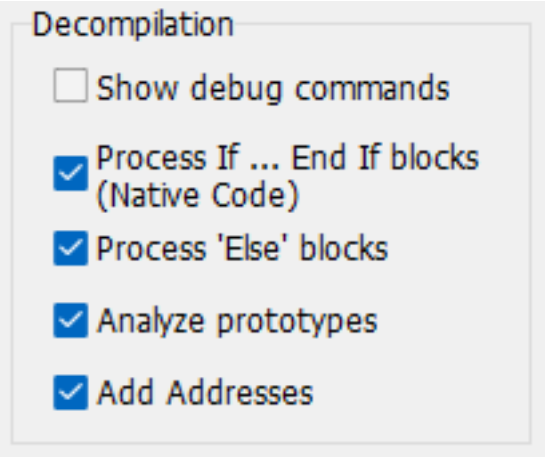
Em seguida, usando um Editor HEX, você pode modificar esses valores. Observe que se seu conhecimento for limitado ao nível básico de usuário e você não estiver familiarizado com o funcionamento interno de arquivos binários - recomenda-se não ativar esta opção. Modificações incorretas podem impedir o funcionamento do arquivo. Se não precisar dos endereços, mantenha a opção desativada.

Pré-visualização e navegação por eventos

As opções de formulário são separadas do visualizador **Editor (só leitura) (Designer (read only))**. Consulte Reconstrução visual de formulários para formulários VB clássicos, suporte a Windows Forms, WPF/BAML, navegação por manipuladores de eventos e as limitações de cada tecnologia de interface.

8.3 Descompilação

Abra **Ferramentas > Opções (Tools > Options)** para configurar a descompilação. Ao comparar configurações interativamente, use Descompilação Rápida sem cacheamento; caso contrário, processe o arquivo novamente após alterar uma opção.



Mostrar comandos de debug (Show debug commands)

Para P-Code do VB clássico, inclui operações do compilador e do depurador que seriam omitidas da lista de alto nível. Ative-o ao inspecionar a sequência de instruções subjacente em vez de apenas a lógica do programa reconstruída.

Processar blocos If ... End If (Código Nativo) (Process If ... End If blocks (Native Code))

Reconstrói saltos condicionais reconhecidos de Código Nativo como blocos `If...End If`. Desative-o quando precisar inspecionar o padrão de ramificação original ou investigar um procedimento cuja estrutura parece incorreta.

Blocos 'Else' (Process 'Else' blocks)

Utiliza heurísticas adicionais para distinguir ramos Else de condições aninhadas e transições GoTo. Para Código Nativo, isso requer processamento de blocos If. Mantenha-o ativado para leitura normal; compare com ele desativado quando um ramo reconstruído for ambíguo.

Análise protótipos (Analyze prototypes)

Realiza análise heurística preliminar dos argumentos e valores de retorno de procedimentos. Isso é particularmente útil na Descompilação Rápida: funções e procedimentos privados em módulos padrão podem carecer de metadados de assinatura explícitos. A análise preliminar leva tempo adicional, mas evita tratar uma chamada apenas de acordo com o conteúdo da pilha atualmente conhecido. Uma assinatura heurística ainda é uma inferência, especialmente quando metadados, tipos externos ou código estão incompletos.

Adicionar Endereços (Add Addresses)

Exibe endereços virtuais ao lado das instruções reconstruídas. Mantenha-o ativado para correlacionar código com desassemblagem, bytes do arquivo e achados de outras ferramentas. Também é necessário para **Copiar para disassembler (Copy to disassembler)** para mesclar instruções de alto nível com seus equivalentes de baixo nível. O Desassemblador sempre exibe endereços independentemente desta configuração.

Entendendo a reconstrução de Código Nativo

O motor de Código Nativo reconhece padrões `Select Case` suportados, acesso a arrays multidimensionais, chamadas COM e operações de runtime do VB. A gestão de erros pode ser reconstruída como `On Error Resume Next`, `On Error GoTo` e `On Error GoTo 0`. Otimização do compilador e falta de informações de tipo ainda podem limitar o resultado; inspecione a desassemblagem quando a saída de alto nível estiver incompleta.

A versão 26.2 expandiu a tipagem Variant, leituras e escritas de SafeArray multidimensionais e reconstrução `VarPtr` para elementos de array. A saída pode identificar uma referência a um array como `SafeArray`, refletindo a estrutura de runtime reconhecida pelo motor de análise. Isolamento do estado de ramificação e análise de variáveis temporárias também melhoram a reconstrução de código If/Else complexo. A versão 26.5 melhora ainda mais o tratamento de Select Case e funções de runtime VB5/VB6.

Informações de tipo COM e ActiveX

O VB Decompiler primeiro utiliza sua base de conhecimento interna para interfaces conhecidas. Para um objeto COM externo desconhecido, ele pode resolver identificadores de biblioteca, classe e interface através do registro do sistema e analisar a biblioteca de tipos DLL/OCX correspondente. Isso fornece assinaturas de método e propriedade, incluindo tipos de interface retornados, para chamadas early-bound e late-bound suportadas.

Se uma biblioteca de tipos necessária estiver ausente ou seu registro não puder ser localizado, algumas chamadas podem permanecer não identificadas, por exemplo como `UnkVCall`. Dependências faltantes são, portanto, relevantes ao comparar saídas entre máquinas de análise. A base de conhecimento IID expandida inclui automação do Office, ADO/DAO, XML e HTTP, script, shell, mensagens e interfaces de serviço de diretório; isso não implica que toda chamada COM possível possa ser resolvida.

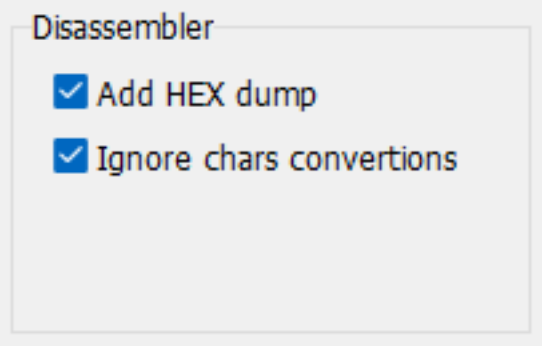
Apenas metadados limitados de UDT sobrevivem em alguns binários VB. O descompilador pode recuperar informações como o tamanho de uma estrutura e a contagem de campos de string sem recuperar o layout completo dos campos originais. Não interprete uma declaração parcial como uma definição completa no nível da fonte.

Reconstrução e otimização .NET

O IL suportado é reconstruído como expressões semelhantes a C# e fluxo de controle. O controle **Analisador e otimizador de procedimento (Procedure analyzer and optimizer)** no painel principal pode simplificar padrões de loop, condicional, operador e variável temporária suportados. Inspecione o IL e metadados onde um fragmento não suportado ou ofuscado permanece. Este processamento é independente do Ajuda IA (AI Helper).

8.4 Desassemblador

Este bloco de opções é responsável pela desassemblagem de código máquina:



Adicionar dump em HEX (Add HEX dump) exibe os códigos de máquina antes de cada comando desassemblado, o que pode ser necessário apenas ao modificar o arquivo binário após a análise em um Editor HEX.

Ignorar conv. de carac. (Ignore chars convertions) é usado para desativar a conversão de dados de string não-ingleses em Unicode numérico. Habilitar esta opção torna as strings exibidas mais fáceis de ler. Desabilize-a quando precisar da representação Unicode numérica.

Mostrar classes de usuário (Show user classes) inclui a análise de classes do usuário dentro das aplicações **Delphi** / **Free Pascal**. Esta opção não é utilizada para as aplicações **Visual Basic** e **C#**.

Mostrar classes VCL (Show VCL classes) inclui a descompilação de todas as classes internas da biblioteca **VCL**. Recomenda-se desativar esta opção para que o desassemblador não analise milhares de classes de serviço, que geralmente são desnecessárias ao analisar programas. Esta opção é utilizada apenas para a descompilação de aplicações **Delphi** para fins internos e foi criada para unificação com outros produtos, mas não é implementada em **VB Decompiler**.

A descompilação parcial de aplicações **Delphi** e **Free Pascal** é atualmente oferecida exclusivamente como um serviço separado (não como software).

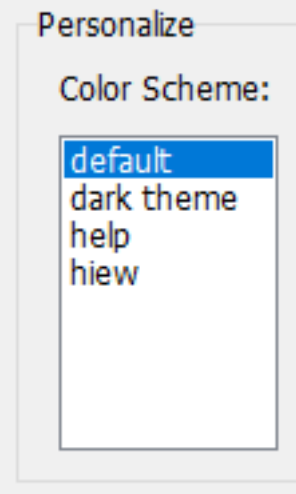
P-Code VB e .NET IL

Adicionar dump em HEX também oferece suporte à listagem P-Code de baixo nível. Use **Copiar para disassembler (Copy to disassembler)** na visualização Descompilador (Decompiler) para comparar as instruções recuperadas com P-Code ou instruções nativas em uma única listagem.

Para assemblies .NET, o Desassemblador pode mostrar métodos IL ou nativos x86/x64 conforme aplicável. As assinaturas de variáveis locais recuperadas podem incluir arrays, genéricos, tipos de valor e referências TypeSpec. Habilite **Mostrar estruturas .NET (Show .NET structures)** quando precisar inspecionar tabelas de metadados e heaps juntamente com o código. Versões posteriores expandem o suporte para metadados ofuscados, incluindo a tabela FieldPtr; isso não garante a recuperação bem-sucedida de todos os assemblies danificados ou ofuscados.

8.5 Esquema de cores

Esta seção define o esquema de cores e o realce de sintaxe ao visualizar código em um descompilador:



Os presets são armazenados na pasta "colors" dentro dos arquivos .ini localizados no diretório onde o VB Decompiler foi instalado. Os nomes dos arquivos determinam os nomes do preset na seção de configurações do programa. Estes arquivos possuem um formato simples, e você pode facilmente criá-los ou editar os existentes.

O formato de configuração é o seguinte:

```
[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)
```

O código de cor é especificado no formato true-color de 32-bits. Decimal na seção Colors (para compatibilidade com versões antigas do VB Decompiler) e hexadecimal na seção Colors2.

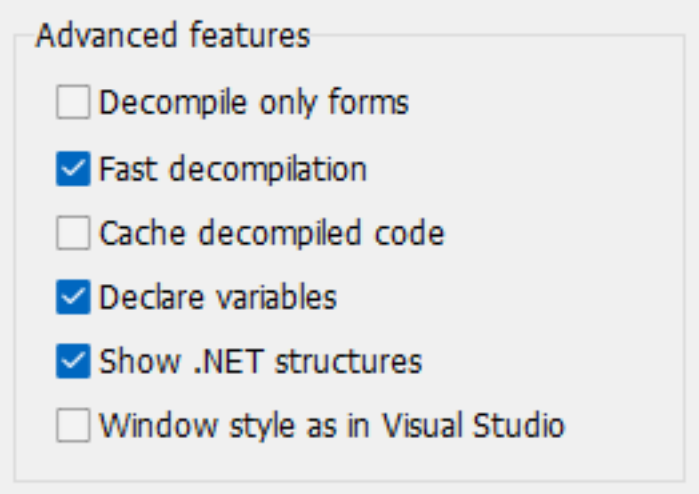
Mudança de temas durante a análise de código

O sistema de temas suporta modo escuro e pode aplicar um tema selecionado sem reiniciar o VB Decompiler, incluindo após editar seu arquivo INI. As configurações de fonte na configuração estendida podem conter nomes de fallback separados por vírgula, o que ajuda quando fontes equivalentes diferem entre Windows e ambientes de compatibilidade.

As cores sob o cursor controlam o realce de identificadores. Consulte Visualização de código para realce de variáveis locais, membros e globais.

8.6 Recursos avançados

Essas configurações controlam como o projeto é processado e como as informações recuperadas são apresentadas.



Descompilar apenas formulários (Decompile only forms)

Desativa a descompilação de código quando você precisa apenas das definições recuperadas de formulário e UserControl. Para o visualizador e suas capacidades específicas da tecnologia, consulte Reconstrução de formulários visuais.

Descompilação rápida (Fast decompilation)

Quando ativado, o VB Decompiler processa primeiro a estrutura do projeto e descompila procedimentos individuais conforme você os abre. Isso reduz o tempo de espera inicial ao investigar procedimentos selecionados. Com **Código decompilado do Cache (Cache decompiled code)** desativado, você pode alterar as configurações de análise e reabrir um procedimento para obter uma nova listagem.

Ative **Análise protótipos (Analyze prototypes)** nas Configurações de descompilação para realizar uma análise heurística preliminar das assinaturas de procedimentos privados. Isso adiciona tempo de processamento inicial, mas pode melhorar a reconstrução de argumentos sem exigir descompilação completa.

A Descompilação rápida não fornece uma lista de alto nível pesquisável completa do projeto. Desative-a e processe o arquivo novamente antes de usar a busca de texto de todo o projeto ou exportar um projeto descompilado completo. As Referências a strings são diferentes: elas podem ser coletadas através da desassemblagem preliminar e estão disponíveis no modo de Descompilação rápida.

Descompilação completa

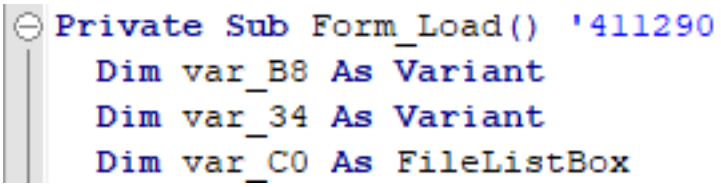
Com a Descompilação rápida desativada, o descompilador processa os procedimentos do projeto com antecedência. Isso leva mais tempo, mas torna a listagem completa gerada disponível para busca de texto e exportação. Alterações nas configurações de descompilação exigem processar o arquivo novamente.

Código decompilado do Cache

Mantém as listagens já geradas no modo de Descompilação rápida. Isso evita descompilar repetidamente um procedimento aberto, mas os resultados em cache não refletem alterações posteriores nas configurações de análise. Ao comparar as configurações **Analisar parâmetros do stack (Parse stack parameters)** ou **Analisador e otimizador de procedimento (Procedure analyzer and optimizer)** interativamente, desative o cache.

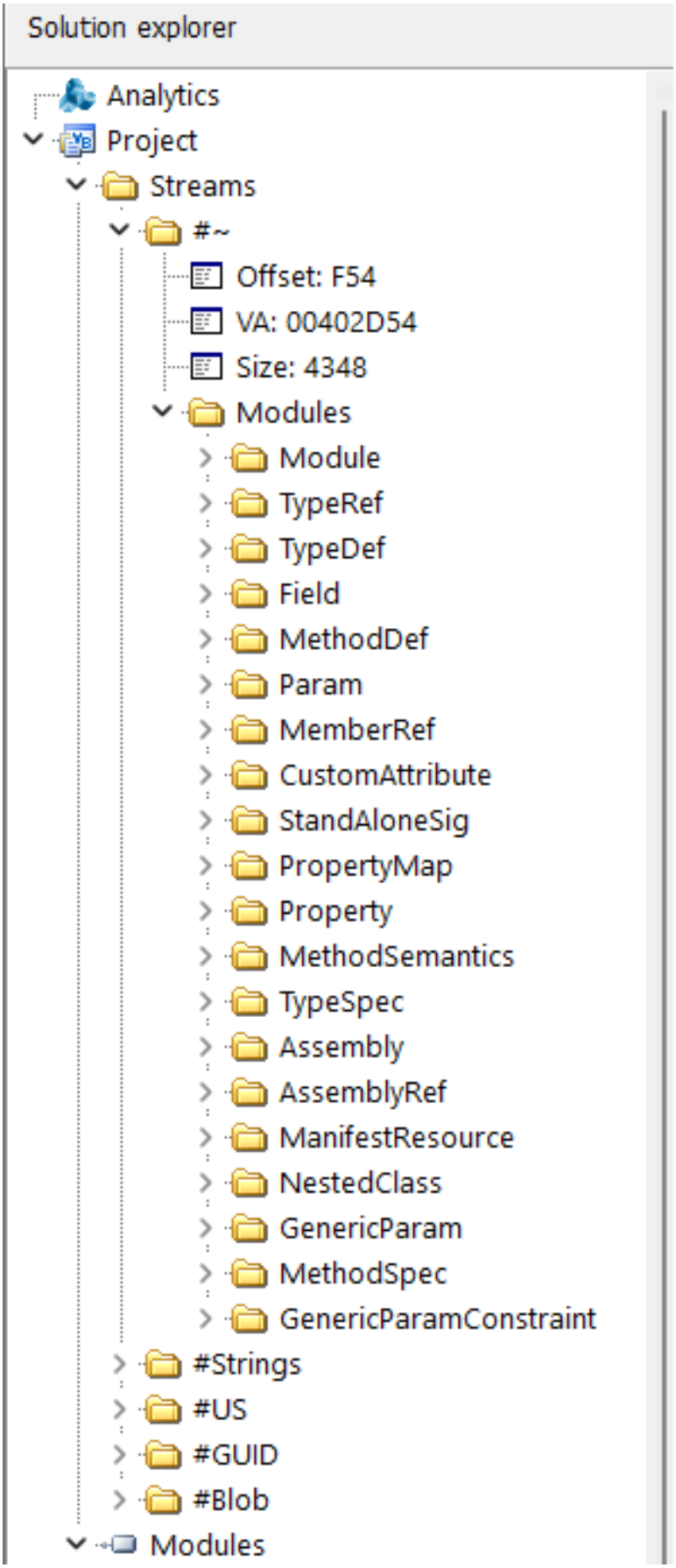
Declarar variáveis (Declare variables)

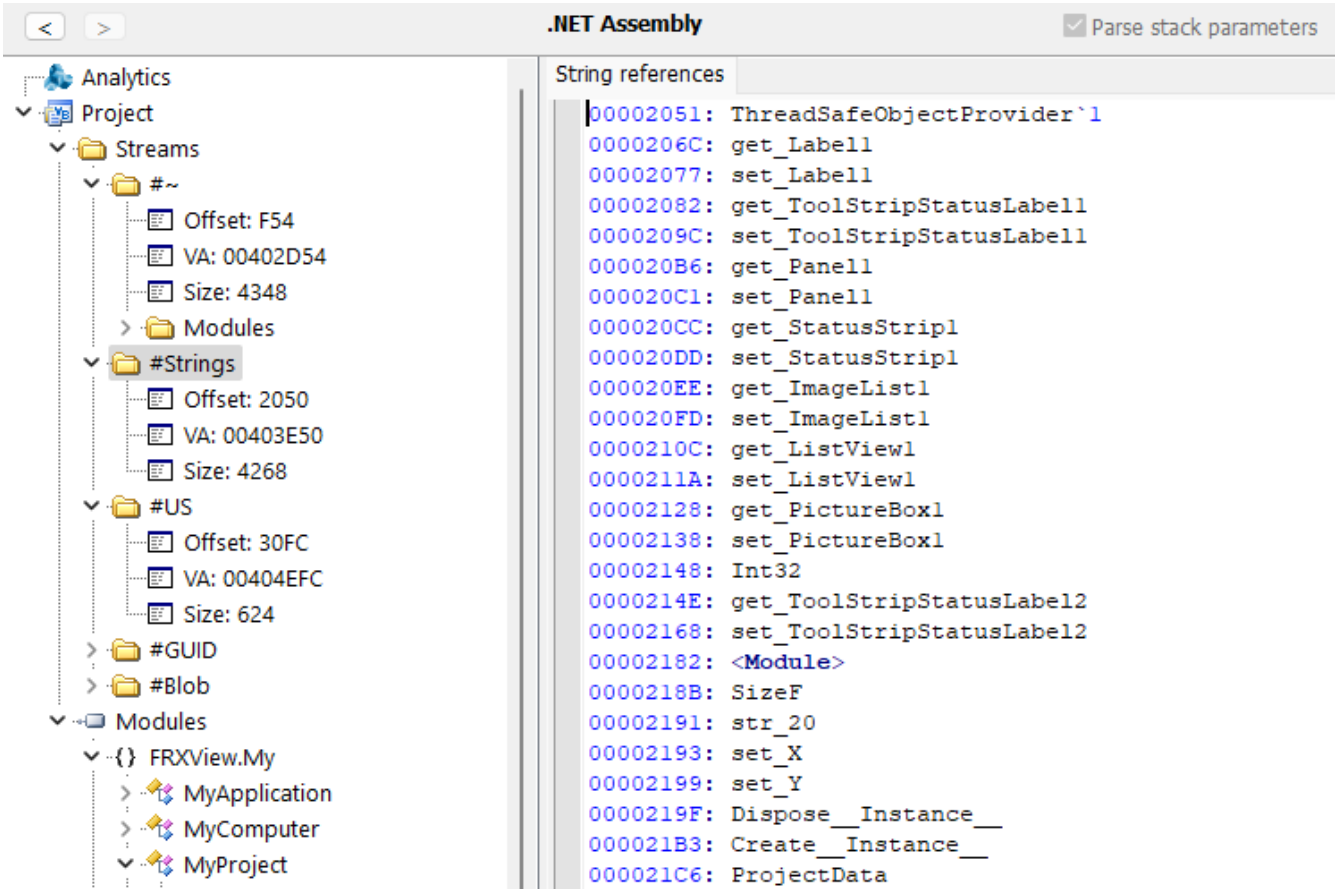
Adiciona um bloco de declaração para variáveis recuperadas no início de cada procedimento descompilado. Isso é útil para recuperação de código-fonte e inspeção de tipos. Para uma revisão comportamental concisa, você pode preferir ocultar as declarações. Os tipos recuperados estão sujeitos aos metadados e à análise disponíveis ao descompilador.



Mostrar estruturas .NET (Show .NET structures)

Exibe tabelas e heaps de metadados .NET no Explorador (Solution explorer). Use-o para inspecionar estruturas e assinaturas juntamente com o código. Selecionar uma tabela exibe seu conteúdo; o heap #US contém strings do usuário. Desative a opção quando você não precisa de navegação de metadados.


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



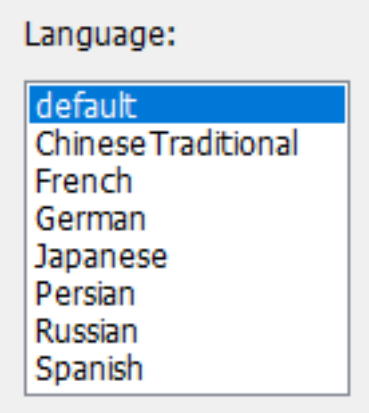
Estilo de janela Visual Studio (Window style as in Visual Studio)

Coloca o código à esquerda e o Explorador à direita. Isso altera a disposição da área de trabalho sem alterar o código recuperado.

8.7 Seleção de idioma

VB Decompiler fornece localizações de interface para mais de 20 idiomas utilizados mundialmente. A documentação também está disponível em múltiplos idiomas.

Você pode escolher a interface de idioma preferida para o programa no menu **Ferramentas (Tools)** -> **Opções (Options)**, sob as configurações **Personalizar (Personalize)**. Após escolher o idioma, é necessário reiniciar **VB Decompiler** para a aplicação adequada das configurações de idioma em todo o programa. Se reiniciar o programa não for desejável, você pode continuar usando **VB Decompiler**; no entanto, alguns textos ainda podem ser exibidos no idioma anterior.



O idioma da ajuda é selecionado automaticamente com base no idioma do sistema do seu computador. Para ler o manual em outro idioma, abra o arquivo de ajuda correspondente a partir da pasta onde **VB Decompiler** está instalado. O PDF é o formato principal para manuais localizados; os arquivos de ajuda CHM permanecem disponíveis para alguns idiomas.

O manual em inglês é a fonte para todas as traduções. Se você notar um erro ou inconsistência em um manual localizado, por favor, relate-o ao suporte técnico.

Acessando o manual

Se um arquivo de ajuda local ou o visualizador de PDF necessário não estiver disponível, o VB Decompiler pode abrir a documentação online. Para um ambiente de investigação offline, mantenha os arquivos de ajuda locais necessários disponíveis com antecedência.

9 Plugins

Os plugins estendem o VB Decompiler por meio de DLLs nativas. Os plugins instalados estão disponíveis no menu **Plugins**. A API de callback fornece acesso às informações e ao código recuperados do projeto, permitindo que um plugin inspecione ou modifique estruturas internas suportadas.

VB/DE Decompiler MCP Server 2.1.0

Conecte um cliente MCP a um projeto já aberto. Você pode consultar módulos, formulários, procedimentos e código recuperado sem executar o programa analisado.

Baixar MCP Server 2.1.0 para hosts x86 · Código-fonte em Delphi

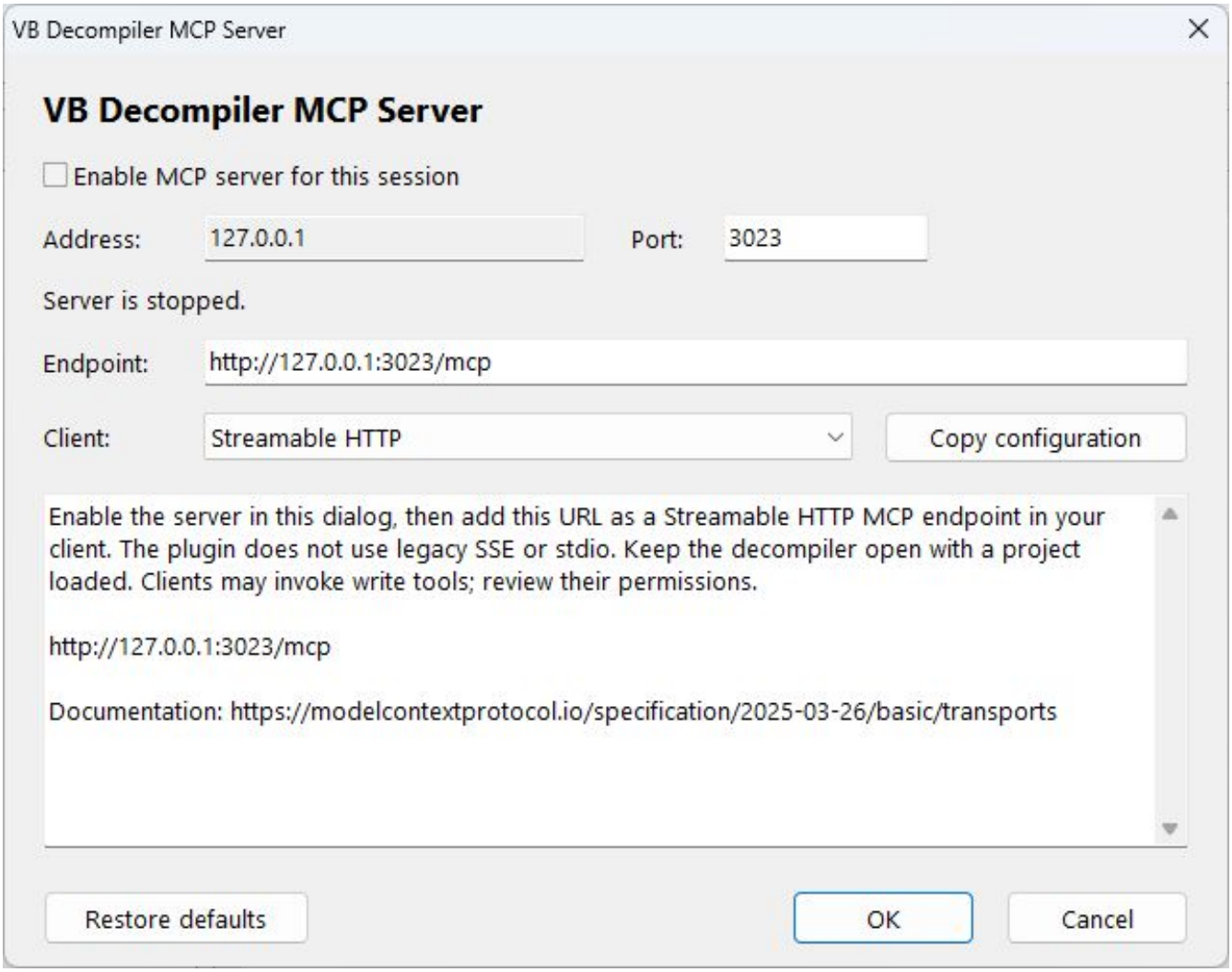
`vb_get_form` devolve o índice do formulário `form_index`, o nome `form_name` e o texto das propriedades `form_text`. Se o nome não estiver disponível, `form_name` é uma cadeia vazia. Procure unidades e classes com `vb_list_modules` e `query`, e texto nas funções com `vb_search_module_code`. Referência de comandos MCP e exemplos

Requisitos

O pacote contém o plug-in x86 para o descompilador Windows de 32 bits. As ferramentas dependem da versão, da edição e do projeto aberto. Em versões mais antigas algumas leituras usam apenas o código já disponível em cache.

Instalação e início

1. Feche o descompilador. Extraia o pacote e copie `vbmcp.dll` para a pasta `plugins` da instalação. Substitua a DLL anterior, se houver.
2. Abra o descompilador e carregue um projeto. Escolha **Plugins** → **VB/DE Decompiler MCP Server**.
3. Marque **Enable MCP server for this session**, clique em **OK** e use o endereço exibido.



Configurações do servidor MCP no VB Decompiler. O plug-in identifica qual descompilador o carregou e exibe o nome do produto no título da janela.

Conexão do cliente MCP

Use Streamable HTTP. O endereço padrão é `http://127.0.0.1:3023/mcp`. Atribua portas diferentes quando usar os dois produtos ao mesmo tempo.

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

No VS Code, use **MCP: Add Server** e escolha HTTP.

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

Documentação dos clientes · Codex · Claude Code · VS Code

Verificação da conexão

Abra um exemplo com formulários e chame `vb_list_forms` com `limit` igual a `5`. Os nomes das ferramentas desta versão começam com `vb_`.

Acesso local e alterações no projeto

O servidor aceita conexões apenas em `127.0.0.1` e não exige autenticação. Os clientes locais podem ler o projeto ou alterar listagens, nomes e marcações. Mantenha a confirmação das gravações ativada. Para parar o servidor, desmarque a opção e clique em **OK**. O servidor fica desativado a cada inicialização.

O plug-in opera localmente. O cliente MCP pode enviar resultados ao provedor do modelo conforme a configuração do cliente.

Desenvolvido em Delphi por Sergey Chubchenko (DotFix Software). O plug-in MCP original serviu de ponto de partida. Distribuído sob a licença MIT.

FRX Image Viewer

O FRX Image Viewer 1.2.0 permite visualizar e extrair imagens registradas pelo VB Decompiler ou DE Descompilador (Decompiler). Use o VB Decompiler para gráficos de formulários VB5/6 e .NET e o DE Descompilador para gráficos de DFM binário do Delphi. São aceitos BMP, PNG, JPEG, GIF, ICO e CUR.

Instalação e abertura

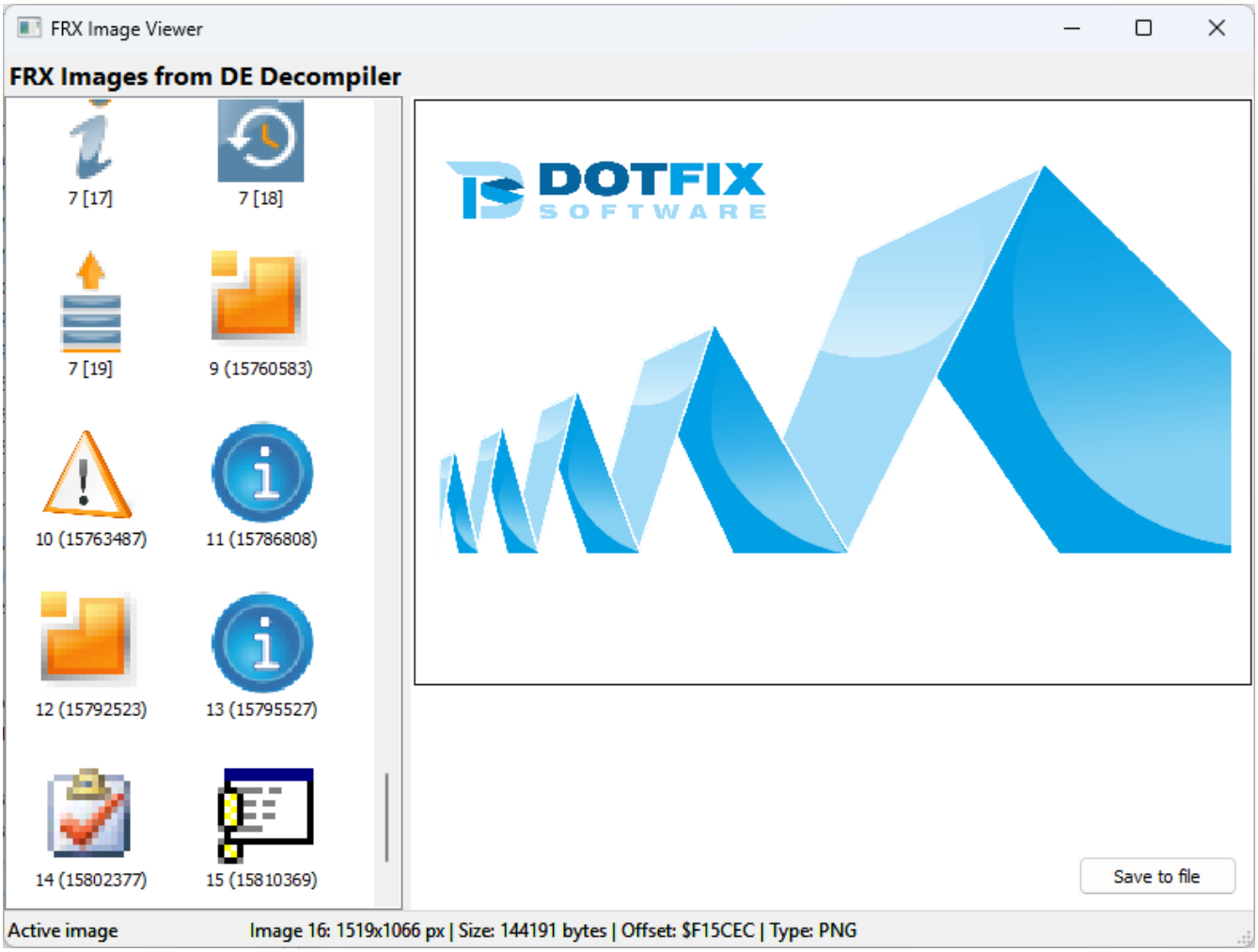
1. Copie `FRXViewer.dll` para a subpasta `plugins` da instalação do host e reinicie o decompilador. A arquitetura da DLL deve corresponder à aplicação host, independentemente da arquitetura do arquivo analisado.
2. Abra o EXE ou DLL e decodifique os formulários para que o host registre as imagens disponíveis.
3. Selecione **Plugins** > **FRX Image Viewer** e escolha uma miniatura.

O download da versão 1.2.0 contém uma DLL para host x86. Um host x64 exige uma compilação correspondente. O plugin requer Windows 10/11; as instruções de compilação estão no repositório do código-fonte.

Visualização e exportação

A galeria e a visualização mostram as imagens disponíveis, incluindo arquivos ICO/CUR compostos. A área de status indica o item, dimensões, tamanho ou deslocamento do bitmap de origem e formato. O visualizador acompanha o tema do host e oferece modo escuro.

Clique em **Save to file** para exportar o item selecionado. Imagens independentes preservam seus bytes e formato originais. Células individuais de ImageList são exportadas como ICO com suas máscaras e transparência alfa; esses ICO são gerados a partir dos bitmaps de origem.



FRX Image Viewer no DE Descompilador, exibindo gráficos DFM do Delphi e ícones individuais de ImageList.

ImageLists nativas do Delphi e compatibilidade

Em um host compatível, ImageLists nativas do Delphi reconhecidas aparecem como ícones individuais. Imagens codificadas separadamente em itens de coleções permanecem entradas de imagem independentes. Fluxos não suportados mantêm a visualização da imagem completa quando há entradas originais disponíveis.

A revisão compartilhada SDK 26.6, suportada pelo VB Decompiler 26.6 e DE Descompilador 26.5, adiciona os comandos opcionais `GetFrxImageListInfo` (63) e `GetProductName` (64). Hosts anteriores preservam a API original e a visualização de imagens completas. Consulte a referência do SDK para detectar recursos e aplicar as regras de deslocamento.

A visualização e a extração não executam o arquivo analisado nem carregam seus componentes. GIFs animados mostram o primeiro quadro. Os recursos disponíveis dependem das imagens decodificadas e registradas corretamente pelo host.

Desenvolvendo um plugin

O SDK inclui exemplos para Visual C++, Rust, Delphi e Free Pascal. Consulte a referência da API de Plugin do VB Decompiler para os pontos de entrada exportados, assinatura de callback, identificadores de comando, representações de dados e links para os projetos de exemplo.

Um plugin DLL fornece suas informações de nome e autor e recebe o callback do hospedeiro por meio de `VBDecompilerPluginLoad`. O callback é usado para solicitar ou atualizar dados de projeto suportados. Siga as convenções de chamada, ponteiros, codificação de string e requisitos de buffer do SDK ao implementar esta interface; ao usar uma linguagem gerenciada, garanta que a camada de interoperabilidade respeite a interface nativa.

O VB Decompiler inicializa os plugins DLL disponíveis na inicialização. Compile plugins para a arquitetura de hospedeiro e ambiente Windows pretendidos e verifique a DLL resultante nesse ambiente. O SDK expõe tanto o acesso ao código da visualização ativa quanto ao nível do projeto; operações que exigem todos os procedimentos descompilados requerem descompilação completa, não apenas uma lista sob demanda.

10 Reconstrução visual de formulários

A aba **Editor (só leitura) (Designer (read only))** exibe uma reconstrução visual das interfaces suportadas a partir do binário analisado. Ela auxilia na conexão dos elementos da interface com o comportamento da aplicação sem interpretar manualmente cada propriedade recuperada ou declaração de inicialização.

Abrindo um formulário recuperado

1. Abra a aplicação e permita que o VB Decompiler recupere sua estrutura de projeto.
2. Selecione um formulário recuperado no Explorador (Solution explorer) e abra **Editor (só leitura)**.
3. Selecione um controle para inspecionar suas propriedades disponíveis e informações de eventos recuperados. Use os eventos para navegar até o código do manipulador correspondente.

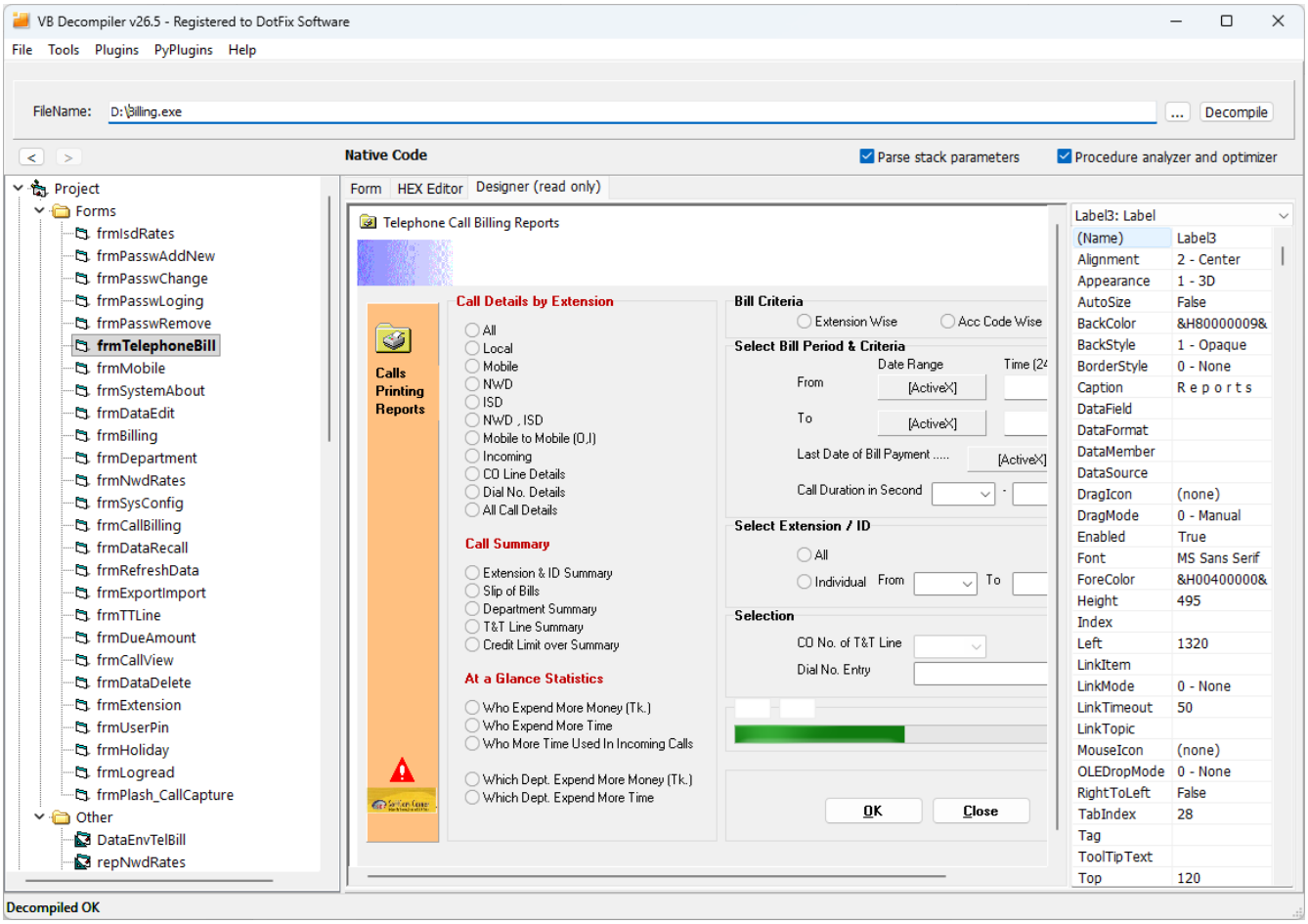
A pré-visualização é reconstruída a partir das informações armazenadas. Não se trata de uma instância em execução da interface original. Controles não suportados, conteúdo criado em tempo de execução, recursos ausentes e lógica de layout personalizada podem produzir resultados incompletos ou aproximados.

Visual Basic 5.0/6.0

O visualizador reconstrói formulários e UserControls, incluindo propriedades de controle suportadas, fontes, cores e imagens. Ele lê dados binários de formulário usados por listas, caixas de combinação e outros controles. As imagens podem incluir ícones, ponteiros, mapas de bits, JPEG, PNG e GIF.

A reconstrução do menu visual suporta tanto aplicações P-Code quanto Native Code. Os controles suportados incluem `SSTab`, `MSComctlLib.StatusBar`, `MSComctlLib.ListView`, `MSComctlLib.ProgressBar` e `MSComctlLib.TreeView`.

Para inspecionar o manipulador de um controle clássico do VB, selecione o controle e clique duas vezes no campo **_Event** na lista de propriedades. Escolha o evento necessário da lista resultante para abrir o manipulador.

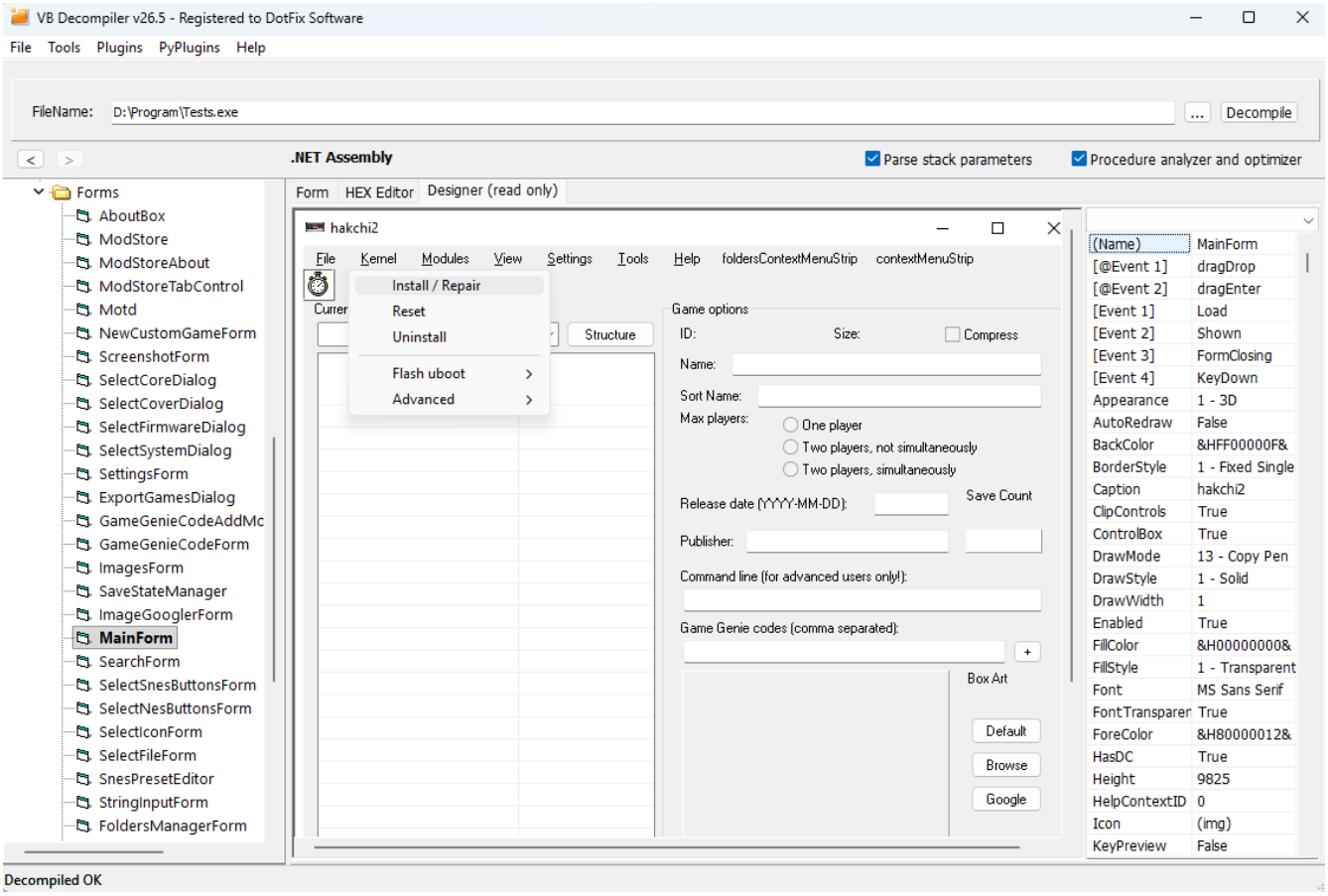


.NET Windows Forms

O VB Decompiler analisa estaticamente `InitializeComponent` e reconstrói controles de Windows Forms suportados e seu layout. O parser e o renderizador não exigem o runtime do .NET ou a execução da aplicação alvo. Imagens recuperadas suportadas também podem ser exibidas em controles e fundos de formulário.

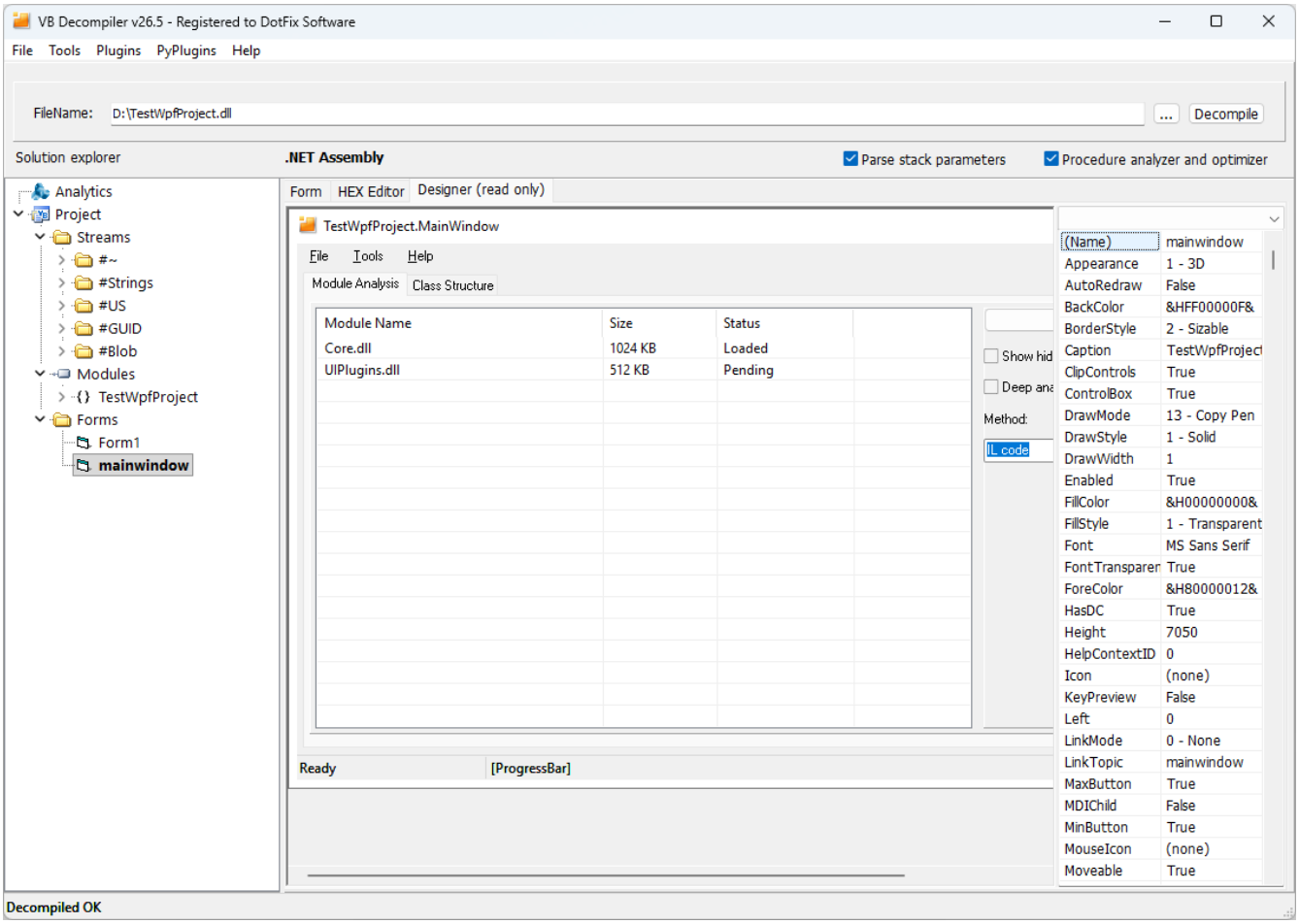
Selecione um controle para inspecionar seus eventos recuperados, em seguida, clique duas vezes em um evento para abrir o manipulador correspondente. O visualizador reconstrói hierarquias de menu e menus contextuais; clicar em um item de menu recuperado abre seu evento Click correspondente. Eventos personalizados de Windows Forms são exibidos com o prefixo **@**.

O visualizador lida com a ordem Z dos controles, cores de primeiro plano e fundo herdadas, valores `Color.FromArgb`, flags numéricas `FontStyle`, propriedades de recursos Booleanas e `MinimumSize` juntamente com `ClientSize`.



Janelas, páginas e controles de usuário WPF

O visualizador reconstrói janelas WPF suportadas, páginas e controles de usuário a partir de recursos XAML compilados armazenados como **BAML**.



Os contêineres de layout suportados incluem **Grid** com tamanhos de linha e coluna fixos, Auto e proporcionais (*), **StackPanel**, **DockPanel** e **Canvas**, incluindo layouts aninhados. A reconstrução leva em consideração tamanhos, margens, preenchimento, alinhamento, orientação, ancoragem e extensões dos controles sobre linhas e colunas de Grid suportados.

Além dos controles de entrada comuns, o visualizador pode reconstruir abas, caixas de grupo, expansores, hierarquias de menu e separadores, barras de status, colunas ListView e dados estáticos suportados, nós estáticos TreeView e itens estáticos de caixa de combinação. Interfaces Ribbon, painéis de navegação e painéis ancorados podem ter representações simplificadas.

Onde os dados de recurso estão disponíveis, o visualizador recupera nomes, texto visível, balões de ajuda, fontes, cores de fundo, estados de habilitação e visibilidade e imagens incorporadas, incluindo transparência PNG suportada e opacidade de fundo. Controles personalizados não suportados podem aparecer como marcadores identificando o tipo WPF original.

O visualizador WPF associa os eventos recuperados dos controles aos respectivos manipuladores. Use as informações de eventos do visualizador para abrir o código descompilado correspondente.

Limites da reconstrução: a pré-visualização visual é aproximada. Templates, estilos, vinculações, controles personalizados e conteúdo gerado dinamicamente podem ser representados parcialmente.

Inspecionando e extraindo dados de formulário

Veja Formulários para a configuração de offsets de exibição e inspeção de binário vinculada. Use o plugin FRX Image Viewer para inspecionar e exportar recursos gráficos recuperados suportados.

11 Rastreamento

Tracing features permite a emulação passo a passo de Native Code e P-Code do Visual Basic 5.0/6.0. É necessária uma licença com recursos de rastreamento. O rastreador trabalha com a representação interna do procedimento no emulador; ele não executa o procedimento na CPU da estação de trabalho.

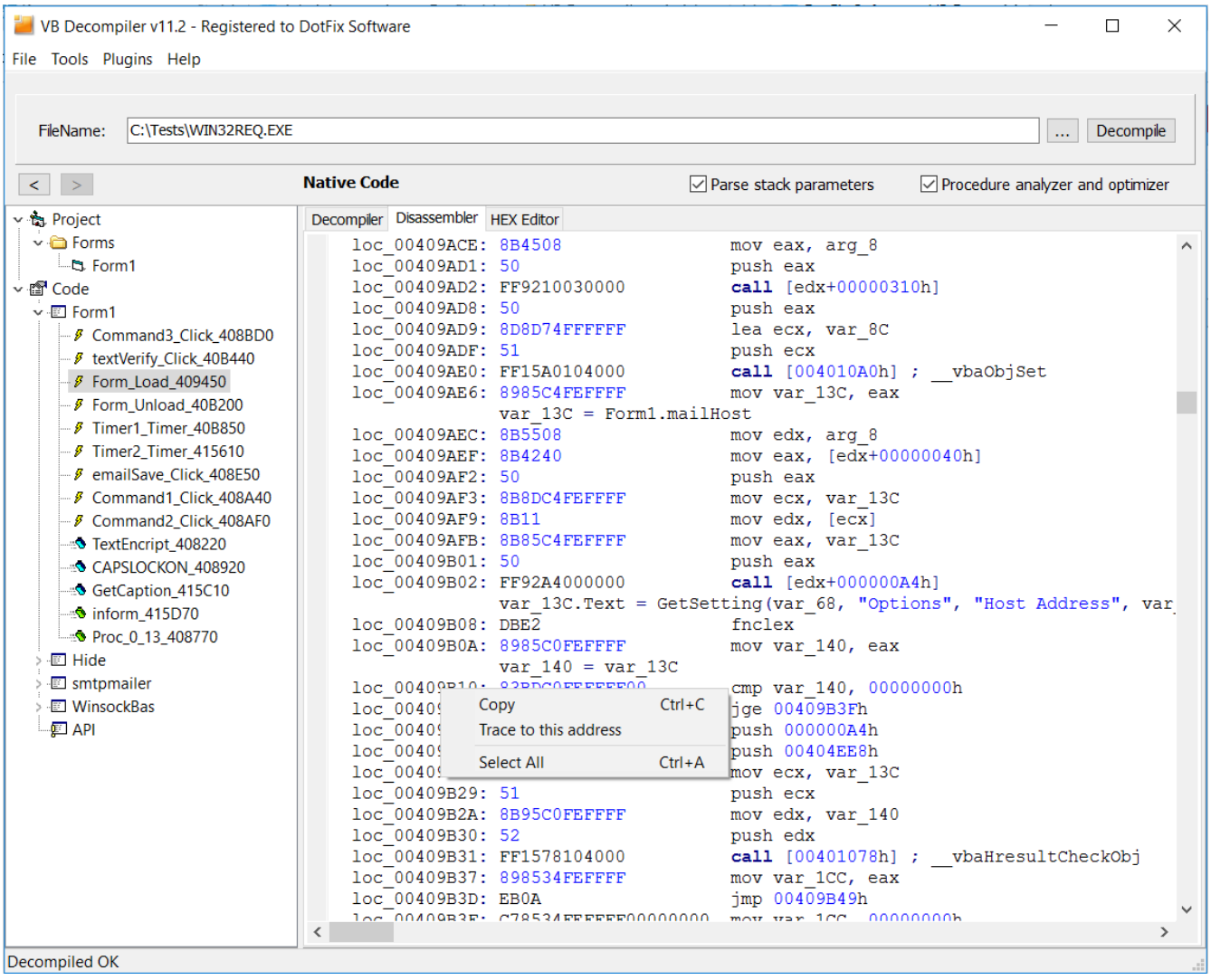
Iniciar um rastreamento

1 Abra um procedimento VB5/6 e mantenha os endereços visíveis na lista.

2 No comando a ser investigado, invoque **Rastrear para este endereço (Trace to this address)** no menu de contexto do código, ou pressione **F4**.

3 Inspecione as visualizações de código e estado do rastreador e avance usando **Next**. Encerre o rastreamento ao concluir a análise.

A partir da versão 26.4, o comando pode resolver o endereço inicial `loc_` após remover espaços em branco no início da linha. Você pode posicionar o cursor em outro lugar naquela linha, em vez de precisamente no rótulo do seu endereço.



Estado do Código Nativo

O rastreador expõe registradores de CPU e FPU, a pilha, variáveis e o código reconstruído conforme a emulação avança. Use essas visualizações para examinar como um valor ou expressão foi derivado. A busca na lista de variáveis e exibição do tipo recuperado do objeto atual podem ajudar com procedimentos maiores.

Rastreamento de P-Code

O rastreamento de P-Code, introduzido na versão 11, mostra como as informações de variáveis e instruções descompiladas evoluem durante o processamento. Esse recurso pode ajudar a explicar diferenças entre as operações intermediárias de P-Code e a lista de alto nível otimizada.

Escopo e limitações

- O rastreamento se aplica a um procedimento individual. Ele começa com um estado vazio dos registradores emulados, da pilha e das variáveis, em vez do estado de um processo em execução.
- Instruções de Código Nativo não suportadas podem ser ignoradas. Chamadas de funções externas não são executadas; o tratamento de convenções de chamada suportado pode ajustar a pilha modelada sem reproduzir o comportamento da função externa.
- O rastreador não fornece uma visualização completa de flags do processador. Não interprete seu estado como um rastreamento completo de execução de hardware.
- O modelo documentado é emulação passo a passo para frente, sem rastreamento reverso ou execução repetida de iterações de loops.
- Este fluxo de trabalho descreve o rastreamento VB5/6. A presença da desassemblagem .NET não implica um depurador de tempo de execução .NET.

Use o rastreamento para investigar a reconstrução do descompilador e valores intermediários. Um valor emulado por si só não estabelece o valor produzido por um processo real com globais inicializados, dependências externas e dados de entrada.

12 Recursos analíticos

Licenças com **Analytic features** geram relatórios categorizados de operações reconhecidas no programa analisado. Esses relatórios ajudam analistas de segurança a identificar procedimentos dignos de investigação antes de ler todo o código-fonte. Eles suportam aplicativos Visual Basic 5.0/6.0 e, a partir da versão 11.7, assemblies .NET.

Gerando um relatório

1 Habilite **Análise protótipos (Analyze prototypes)** em **Ferramentas > Opções (Tools > Options)**.

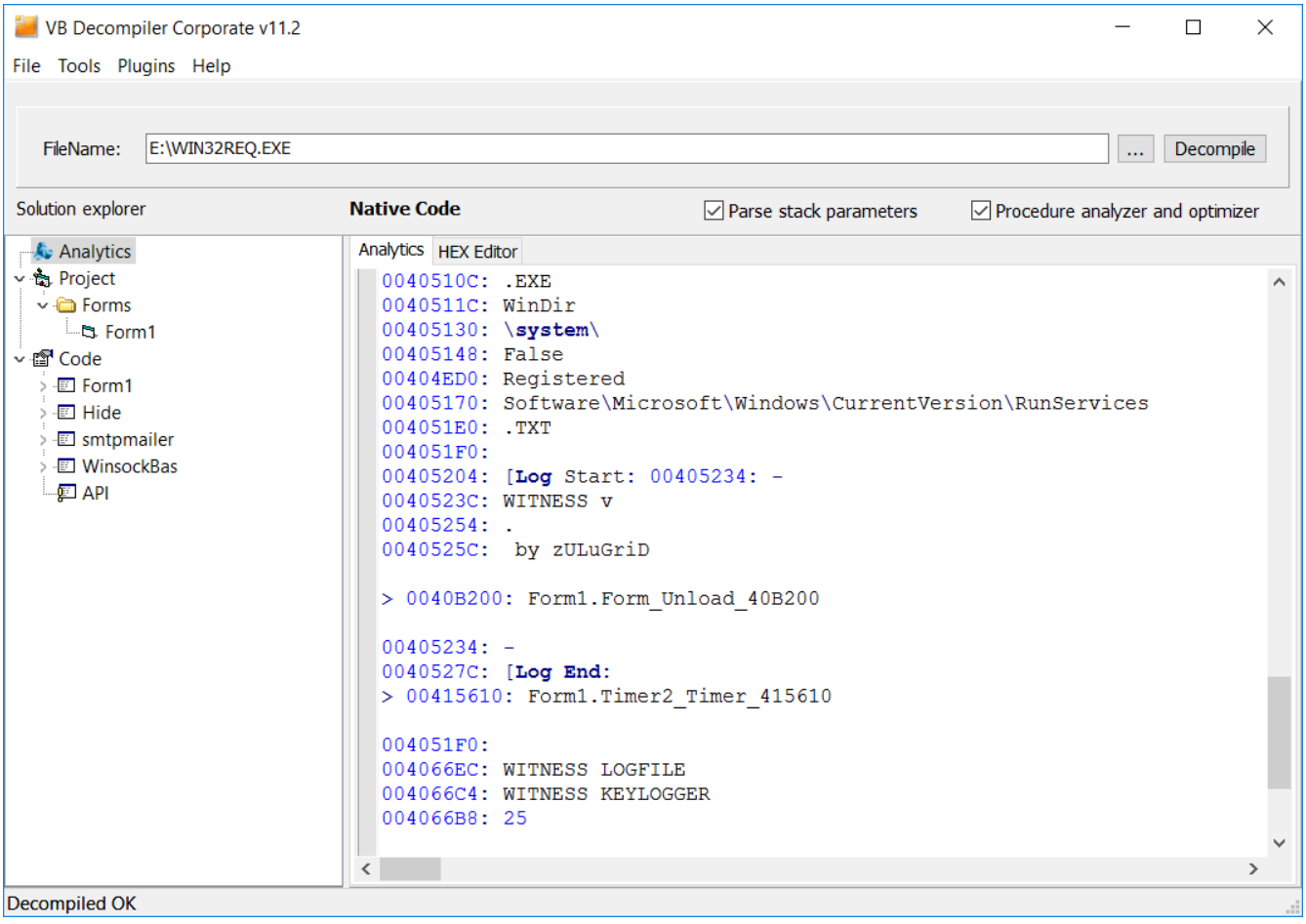
2 Abra o arquivo e aguarde a conclusão da análise. Com a licença e as configurações necessárias, o relatório é gerado e exibido automaticamente.

3 Revise as categorias, em seguida, abra os procedimentos relevantes e examine seu código e referências de strings.

Categorias abrangem operações reconhecidas, como acesso a arquivos e volumes, acesso ao registro, atividade de rede, criação de processos, operações de janela e funções internas selecionadas do VB, como `CallByName` e `VarPtr`.

Relatório rápido e contexto de string

O relatório rápido lista endereços e nomes de procedimentos contendo operações reconhecidas. O relatório estendido adiciona as referências de strings disponíveis desses procedimentos. Esse contexto pode revelar caminhos candidatos, chaves do registro, nomes de host ou outros valores que merecem exame.



Strings criptografadas, ofuscadas ou geradas em tempo de execução podem estar ausentes. Uma string associada a um procedimento é um contexto útil, mas não prova por si só que ela é passada para uma chamada de API específica ou que um ramo é executado.

Navegando e salvando resultados

Clique duas vezes em um nome de procedimento no relatório para abrir seu código. Clique duas vezes em um endereço anterior a um procedimento ou string para examinar sua localização no Editor HEX. Use o botão **A** acima da árvore de objetos para retornar ao relatório na posição anterior e as setas de navegação ou **Esc** para rever procedimentos examinados.

Selecione e copie o texto do relatório conforme necessário, ou escolha **Arquivo > Salvar relatório (File > Save analytics report)** para salvá-lo para análise futura.

Interpretando achados

O relatório identifica padrões de código reconhecidos estaticamente e referências. É uma ferramenta de triagem, não um registro de execução observada ou um veredito automático de que um arquivo é malicioso. Chamadas indiretas, interfaces externas não resolvidas, metadados danificados, ofuscação e valores gerados em tempo de execução podem limitar a cobertura. Confirme achados significativos nos procedimentos correspondentes, desassemblagem e dados do arquivo.

Para inspeção mais profunda da reconstrução VB5/6, veja Rastreamento. Para strings literais e suas localizações de código, veja Referências a strings.

13 Restrições da ofuscação

VB Decompiler, equipado com uma tecnologia única de ofuscação, esconde efetivamente os dados deixados pelo compilador em nossa pesquisa sobre a descompilação de **aplicativos Visual Basic 5.0 e 6.0**. Linguagens de programação modernas frequentemente armazenam quantidades significativas de metadados e informações de depuração dentro de arquivos compilados, facilitando que crackers localizem funções desejadas dentro do seu programa. Para combater essa ameaça, desenvolvemos um ofuscador especificamente projetado para aplicações **Visual Basic**.

O processo de ofuscação em detalhes

O ofuscador funciona decompondo o programa inteiro em suas partes constituintes, incluindo formulários, módulos, classes e controles como botões e campos de texto. Cada formulário é então dissecado em controles individuais com eventos associados. Em seguida, informações sobre funções e procedimentos, bem como eventos vinculados a cada objeto no programa, são coletadas. Durante esse processo, o ofuscador verifica dependências entre objetos e altera simultaneamente seus nomes para tornar difícil determinar qual função pertence a um módulo ou classe específico.

Como resultado, encontrar formulários específicos como **frmTrial** ou **frmRegistration** dentro do programa torna-se significativamente mais difícil.

Limitações da ofuscação de aplicativos Visual Basic 5.0/6.0

O algoritmo empregado pelo ofuscador é complexo e projetado para analisar dependências de objetos, mas ainda não suporta tudo. Consequentemente, em alguns casos, você precisará preparar seu programa previamente para o processo de ofuscação. Os pontos discutidos abaixo aplicam-se exclusivamente à ofuscação de aplicações Visual Basic 5.0 / 6.0.

1. O ofuscador não suporta referências de tempo de projeto a outros objetos, o que pode levar a problemas durante a execução.

Por exemplo, considere um cenário onde você possui um controle **TreeView** que depende de um associado **ImageList**. Se o nome desse objeto **ImageList** estiver codificado diretamente nas propriedades do seu **TreeView**, qualquer tentativa de ofuscação resultará em erros, pois a referência original permanece intacta. Essa limitação também se aplica a componentes relacionados a banco de dados, 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 pode ser facilmente resolvido movendo as atribuições do tempo de projeto para o tempo de execução, simplesmente movendo essas atribuições para o evento **Form_Load**:

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

A mesma correção precisa ser aplicada para cada instância idêntica. Felizmente, geralmente há apenas alguns casos assim na maioria dos projetos.

2. Declarações de objetos com As New não são suportadas (chamadas LateCall)

Essa limitação tipicamente não afeta a maioria dos projetos VB6, mas é essencial mantê-la em mente. Se você estiver trabalhando com formulários como classes e criando instâncias de formulários, pode estar familiarizado com esse cenário:

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

Nesses casos, o compilador VB6 insere o nome **frmMain** diretamente no código da função **Form_Load**. Como nosso ofuscador apenas modifica dados e não código, o nome incorporado ao código permanece inalterado após a ofuscação. Infelizmente, isso leva à terminação do programa com erro.

Infelizmente, é impossível para nós suportar tais chamadas no ofuscador de **VB Decompiler**, pois precisamos desassemblar e descompilar todo o código de todas as funções. Apesar de possuir essa capacidade, a descompilação é um processo extremamente demorado. Para projetos grandes, pode levar mais de meia hora apenas para analisar toda a base de código. Além disso, também precisamos analisar referências a objetos e renomeá-los, o que adiciona ainda mais tempo de processamento.

Dadas essas restrições técnicas, decidimos não implementar a descompilação completa e análise de código no ofuscador de **VB Decompiler**, especialmente já que isso abordaria o problema apenas em casos muito raros em aplicações Visual Basic 5.0 / 6.0 e não proporcionaria benefícios para a maioria dos clientes.

14 Suporte técnico

Se você tiver dúvidas sobre o programa e não encontrar respostas neste guia de ajuda, entre em contato com nossa equipe de suporte técnico. Esta opção está disponível apenas para usuários registrados. Para usuários não registrados, ficamos felizes em ajudar com consultas relacionadas à aquisição de licença para **VB Decompiler**.

Além disso, se encontrar algum bug no programa ou tiver sugestões sobre como melhorá-lo, sinta-se à vontade para entrar em contato com nosso serviço de suporte técnico com seu feedback.

15 Registro

Para utilizar as capacidades completas do **VB Decompiler**, você precisa adquirir uma **Business License**. Ao adquirir uma Licença Comercial para Desenvolvedor Único ou uma Licença Empresarial, você receberá a capacidade de:

1. Recuperar cerca de 85% do código de aplicativos típicos compatíveis em **Visual Basic 5.0/6.0 P-Code**;
2. Recuperar até 85% do código de aplicativos típicos compatíveis em **Visual Basic 5.0/6.0 Native Code**;
3. Descompilar assemblies **.NET** típicos compatíveis e não ofuscados em código semelhante a **C#**, com recuperação de até 95%;
4. Sempre ter a versão mais recente do programa: atualizações são absolutamente gratuitas por um ano após a compra da licença;
5. Receber suporte técnico via e-mail por um ano após a compra da licença;
6. Apoiar o desenvolvimento deste descompilador.

Além disso, recomendamos considerar a aquisição de licenças especiais para Analistas de Malware e Forense:

1. licença com funcionalidades de rastreamento
2. licença com funcionalidades analíticas

Para adquirir uma **licença** do descompilador agora, selecione a opção **Solicite versão Pro (Order Pro version)** no menu **Ajuda (Help)** do programa. Neste momento, você deve estar conectado à internet. Uma janela de navegador será aberta para o site do programa onde você pode comprar a licença.

A **Business License** do descompilador é projetada para uso em apenas um computador e apenas por um **Single Developer** (proprietário da licença). Se você precisar usar o programa em múltiplos computadores, você precisará adquirir uma licença separada para cada um ou considerar a aquisição do **Business Enterprise license** (para até 20 computadores).

16 Sobre o desenvolvedor

O descompilador é desenvolvido pela **DotFix Software**.

Site do produto: www.vb-decompiler.org
Email do suporte técnico: admin@vb-decompiler.org