VB Decompiler

Manuale utente

Italiano

Visual Basic 5.0/6.0 · .NET

DotFix Software

Sommario

1 Descrizione 4

2 Guida rapida 5

3 Finestra principale 6

4 Visualizzazione del codice 8

5 Miglioramento del codice con IA 11

6 Apertura e salvataggio 15

6.1 Apertura di un file 15

6.2 Utilizzo dei database di analisi 16

6.3 Salvataggio dell'elenco delle procedure 17

7 Funzioni del programma 18

7.1 Passaggio a un indirizzo 18

7.2 Ricerca di stringhe 19

7.3 Riferimenti alle stringhe 20

7.4 Decompilazione da un indirizzo 22

7.5 Modifica dei dati 23

7.6 Offuscamento 25

8 Opzioni del programma 26

8.1 Opzioni del programma 26

8.2 Form 27

8.3 Decompilazione 28

8.4 Disassemblatore 30

8.5 Schema colori 31

8.6 Funzionalità avanzate 33

8.7 Selezione della lingua 37

9 Plugin 38

10 Ricostruzione visiva dei form 40

11 Tracciamento 44

12 Funzionalità di analisi 46

13 Limitazioni dell'offuscamento 48

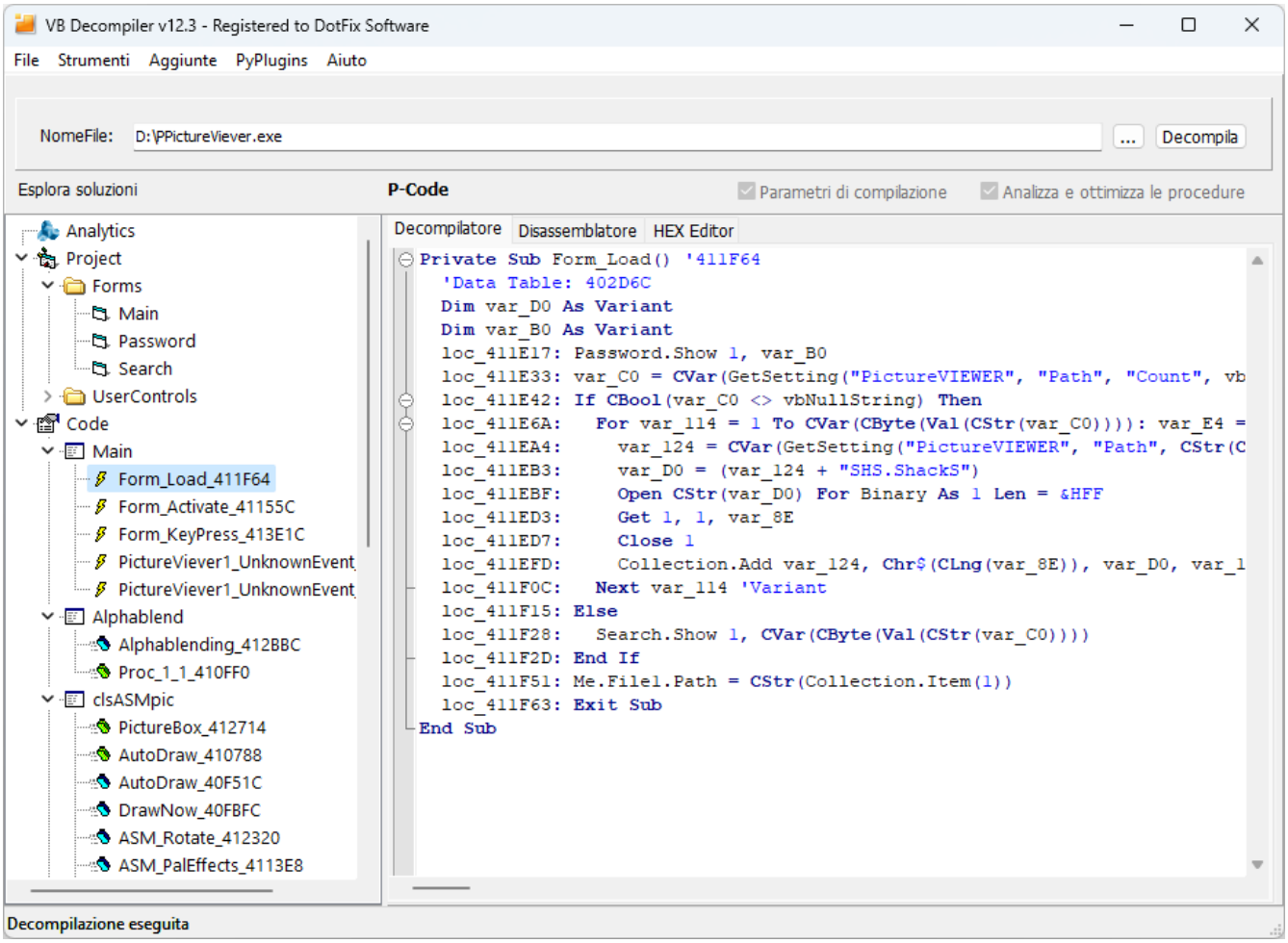
14 Supporto tecnico 50

15 Registrazione 50

16 Informazioni sullo sviluppatore 51

1 Descrizione

VB Decompiler è uno strumento di reverse engineering e recupero del codice sorgente per programmi Visual Basic 5.0/6.0 e assembly .NET. Combina decompilazione ad alto livello, disassemblaggio a basso livello, ispezione dei metadati, un editor HEX integrato e ricostruzione dell'interfaccia visiva. Gli utilizzi tipici includono la manutenzione di software legacy, risposta agli incidenti, analisi del malware e indagine sul comportamento non documentato delle applicazioni.



Rappresentazioni del codice supportate

- **VB5/6 P-Code:** ricostruisce le istruzioni Visual Basic dalle istruzioni P-Code del compilatore e dalle operazioni runtime.
- **VB5/6 Native Code:** disassembla le istruzioni x86 e utilizza un emulatore interno, firme runtime ed euristiche per ricostruire codice simile a Visual Basic.
- **.NET:** disassembla il Common Intermediate Language (CIL, anche chiamato MS IL) e decompila i pattern IL supportati in codice simile a C#, indipendentemente dal linguaggio sorgente .NET originale. Anche il codice nativo x86 e x64 negli assembly .NET può essere disassemblato.

Il motore di analisi .NET non richiede il runtime .NET o Reflection per l'analisi del codice o la ricostruzione visiva dei form.

Strumenti di analisi

Utilizzare ricerca di procedure e testo, riferimenti incrociati e rinomina delle variabili e Riferimenti Stringa per investigare il comportamento. Il visualizzatore ricostruisce i form classici VB, Windows Forms .NET e finestre WPF, pagine e controlli utente memorizzati nelle risorse BAML compilate.

Le licenze con funzionalità di Tracing forniscono emulazione passo-passo del Native Code VB5/6 e P-Code. Le licenze con funzionalità Analytic forniscono report categorizzati per applicazioni VB5/6 e .NET. Aiuto IA (AI Helper) può elaborare codice VB5/6 o C# recuperato utilizzando l'integrazione Ollama o OpenAI API.

Interpretazione del codice recuperato

La compilazione rimuove informazioni quali commenti, molti nomi di variabili locali e alcuni metadati sui tipi e sulle procedure. Ottimizzazione e offuscamento possono ulteriormente modificare la struttura del codice. Il listato ricostruito supporta l'analisi e il recupero del codice. Verificare le conclusioni importanti confrontandole con il disassemblaggio, i metadati, i riferimenti stringa e le procedure circostanti.

2 Guida rapida

1. Seleziona **File > Apri programma (File > Open program)** e apri il file EXE, DLL o OCX da analizzare. L'elaborazione inizia automaticamente. Attendi che l'albero del progetto si popoli e che l'elaborazione sia completata.
2. Apri una procedura in **Esplora soluzioni (Solution explorer)**. Confronta le viste **Decompilatore (Decompiler)** e **Disassemblatore (Disassembler)**; usa **HEX Editor** per ispezionare i byte corrispondenti.
3. Premi **Ctrl+T** per trovare una procedura in base al suo nome o a una parte del nome. Usa **Ctrl+F** per cercare il testo visualizzato, oppure **Strumenti > Riferimenti stringa (Tools > String references)** per investigare le stringhe e le procedure che le utilizzano.
4. Apri un form recuperato e seleziona **Designer (sola lettura) (Designer (read only))** per un'anteprima visiva. Per VB classico, Windows Forms e WPF, usa gli eventi recuperati per navigare verso il codice del gestore corrispondente.
5. Salva l'analisi tramite **File > Database (File > Data Base)** quando è necessario riprenderla in un momento successivo o condividere i risultati con un altro analista.

Scegliere una modalità di decompilazione

Decompilazione rapida elabora le procedure quando vengono aperte. Abilita **Analizza prototipi (Analyze prototypes)** per un'analisi preliminare aggiuntiva delle firme delle procedure private. Per cercare tutto il testo decompilato o esportare un progetto completo, disabilita la Decompilazione rapida ed elabora nuovamente il file.

Salvataggio del codice recuperato

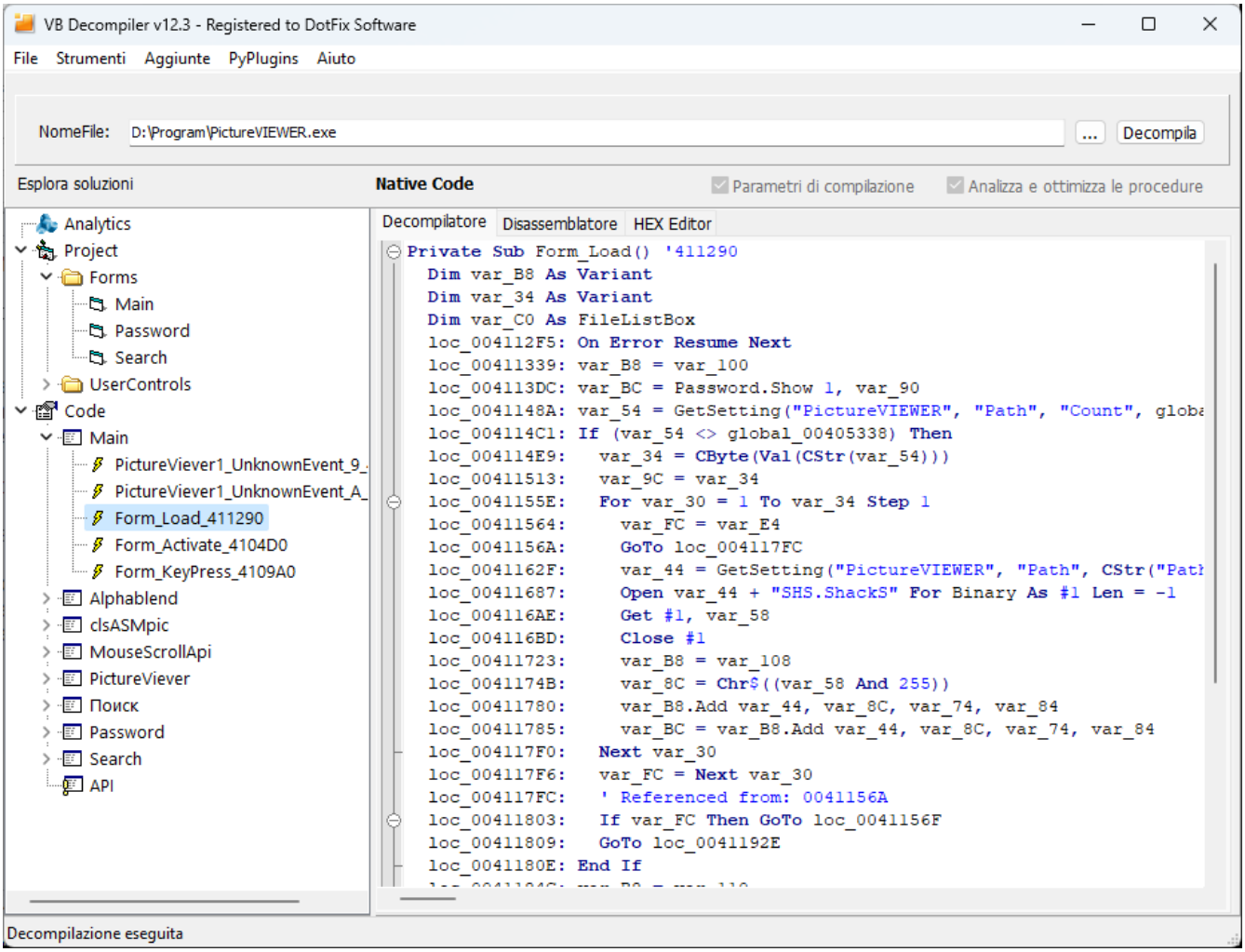
Per esportare i risultati della decompilazione VB5/6, seleziona **File > Salva progetto decompilato (File > Save decompiled project)** e scegli una cartella di output. L'esportazione include moduli e form recuperati; i file FRX contengono le immagini associate e altri dati binari dei form.

Prima di aprire il codice esportato in un IDE, correggere manualmente tutte le imprecisioni della decompilazione e le incompatibilità presenti nel codice sorgente recuperato.

Vedi Ricostruzione visiva dei form, Visualizzazione del codice e Miglioramento del codice con IA per i flussi di lavoro dettagliati.

3 Finestra principale

Esplora soluzioni (Solution explorer) organizza il progetto recuperato in form, moduli, classi e procedure. La vista adiacente visualizza il codice, il form o i metadati selezionati. L'opzione **Stile della come in Visual Studio (Window style as in Visual Studio)** colloca l'albero a destra e il codice a sinistra.



Viste del codice e dei dati

- **Decompilatore (Decompiler):** codice ricostruito ad alto livello.
- **Disassemblatore (Disassembler):** le istruzioni x86/x64 corrispondenti, il P-Code VB o l'IL .NET, con gli indirizzi.
- **HEX Editor:** i byte sottostanti del file, con navigazione tramite indirizzo virtuale o offset nel file.
- **Designer (sola lettura) (Designer (read only)):** un form ricostruito e le relative proprietà recuperate, ove supportato.
- **Aiuto IA (AI Helper):** elaborazione AI opzionale della procedura attualmente decompilata.

I tab disponibili dipendono dall'oggetto selezionato. L'indicatore del tipo di codice distingue **P-Code**, **Native Code** e **.NET Assembly**.

Analizza parametri stack (Parse stack parameters)

Per il P-Code, questo controllo combina le operazioni di pila e le chiamate in espressioni con argomenti. Disabilitarlo espone le operazioni intermedie e può essere utile quando un prototipo sconosciuto consuma valori di pila imprevisti. Per il codice nativo, abilitarlo utilizza l'emulatore per ricostruire espressioni di livello superiore; disabilitarlo espone l'output a livello di assembly. Per .NET, l'elaborazione della pila contribuisce alla ricostruzione dell'IL come espressioni simili a C#.

Analizza e ottimizza le procedure (Procedure analyzer and optimizer)

Questo controllo semplifica l'elenco ricostruito combinando espressioni e pattern di flusso di controllo riconosciuti e rimuovendo il codice di supporto del compilatore riconosciuto. Si applica al codice VB5/6 e .NET. L'ottimizzatore .NET può ricostruire i pattern supportati `while`, `do...while` e `foreach`, condizioni annidate, espressioni di operatori e pattern gestori di stringhe interpolate. Queste trasformazioni sono eseguite dal decompilatore stesso e non richiedono l'Aiuto IA.

Se un'espressione o una ramificazione è ambigua, confronta l'output con l'ottimizzazione disabilitata ed esamina il disassemblaggio. In modalità **Decompilazione rapida (Fast decompilation)** senza caching, questi controlli possono essere modificati mentre si ispeziona una procedura. Con decompilazione completa o output cached, elabora nuovamente il file dopo aver modificato le impostazioni.

Navigazione e accesso da tastiera

I pulsanti freccia sopra l'albero navigano tra le procedure precedentemente visualizzate; **Esc** ritorna alla procedura precedente. I pulsanti Mouse Indietro e Avanti utilizzano la stessa cronologia, inclusa la posizione memorizzata all'interno di una procedura.

Premi **Alt+1** per cambiare il focus tra Esplora soluzioni e la vista attiva. Premi **Enter** nell'albero per aprire l'oggetto selezionato. Nell'Editor HEX, **Tab** sposta tra i campi esadecimali e di testo e poi al controllo successivo. Le etichette dell'interfaccia sono associate ai loro campi di input per la navigazione tramite screen-reader.

Vedi Visualizzazione del codice per scorciatoie contestuali, riferimenti incrociati e rinomina, e Funzionalità di analisi per la vista Analitica (Analytics).

4 Visualizzazione del codice

Le viste del codice combinano l'evidenziazione della sintassi, blocchi comprimibili, la navigazione basata sugli indirizzi e i riferimenti incrociati. Utilizzare il listato ad alto livello per comprendere una procedura e il disassemblaggio e i byte del file per verificare come è stato ottenuto quel risultato.

Evidenziazione e navigazione degli identificatori

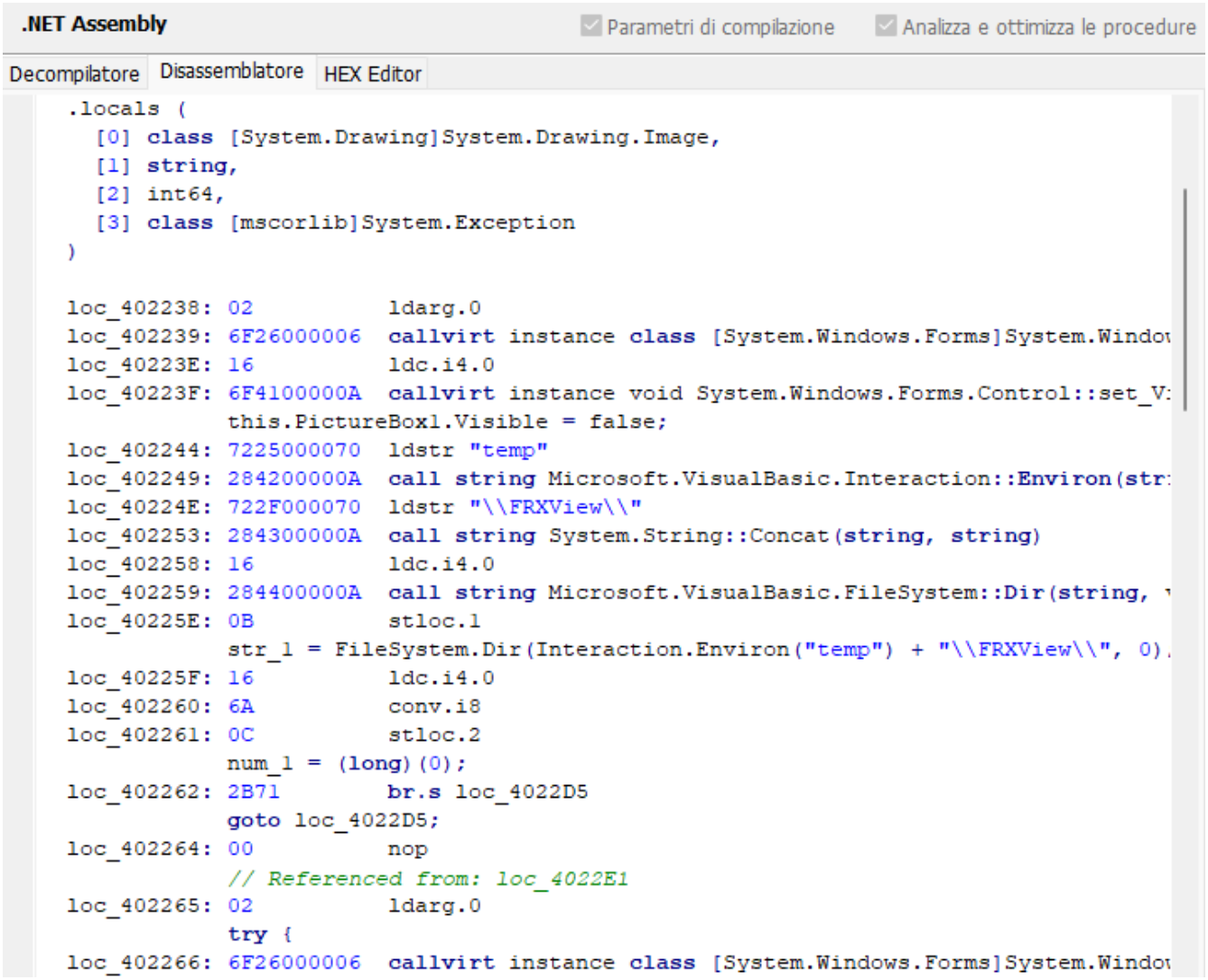
Posizionare il cursore su un registro, una variabile o un nome di funzione per evidenziare le occorrenze corrispondenti nell'elenco corrente. Le variabili locali come `var_XX` e `str_XX` sono inoltre evidenziate quando utilizzate come base di un'espressione membro, ad esempio `str_1.Value`. Selezionando un identificatore `global_XX` vengono evidenziate le sue occorrenze all'interno della procedura corrente; tale evidenziazione è separata da una ricerca di riferimenti su tutto il progetto.

Fare doppio clic su una chiamata di procedura riconosciuta per aprire la procedura chiamata. Costanti simili agli indirizzi possono invece portare ai byte corrispondenti nell'Editor HEX. Utilizzare **Esc**, le frecce di navigazione o i pulsanti Indietro/Avanti del mouse per spostarsi nella cronologia di visualizzazione.

Combinazione di codice decompilato e disassemblato

Fare clic con il pulsante destro sul listing decompilato e selezionare **Copia nel disassemblatore (Copy to disassembler)**, oppure premere **D** nella vista del codice. Questo fonde le istruzioni ricostruite con il listato a basso livello per la procedura corrente. Per mantenere la posizione corrente, posizionare il cursore sull'indirizzo pertinente `loc_` prima di invocare il comando.

Mantenere **Aggiungi Indirizzi (Add Addresses)** abilitato nelle impostazioni di decompilazione quando si utilizza questo flusso di lavoro. Il disassemblatore visualizza sempre gli indirizzi; Aggiungi Indirizzi controlla la vista del Decompilatore.



Rinomina delle variabili

Per VB5/6 P-Code e Native Code, fare clic con il pulsante destro su una variabile e selezionare **Rinomina variabile (Rename variable)**, oppure posizionare il cursore sulla variabile e premere **N**. Inserire un nome di analisi significativo. Il decompilatore controlla le collisioni di nomi e le parole chiave riservate. Per annullare una rinomina, utilizzare **Annulla rinomina (Undo renaming)** o **Ctrl+Shift+Z**. Questo comando è indipendente dall'annullamento dell'editing del testo ordinario.

Rinominare rende l'elenco recuperato più facile da analizzare; non recupera un nome che è stato rimosso dalla compilazione. Trattare i nomi assegnati dall'analista come annotazioni quando si comunicano le risultati.

Trovare chiamanti e riferimenti a variabili globali

Per trovare i chiamanti, fare clic con il pulsante destro su una procedura in Esplora soluzioni (Solution explorer) e selezionare **Trova riferimenti (Find references)**, oppure premere **X** con quella procedura selezionata nell'albero. I risultati elencano le procedure contenenti chiamate riconosciute ad essa. Fare doppio clic su un risultato per ispezionare il chiamante. Questo flusso di lavoro supporta VB5/6 e .NET; il supporto per .NET è stato aggiunto nella versione 12.4.

Per investigare una variabile globale, posizionare il cursore sul suo identificatore nella vista del codice e premere **X**. Lo stesso tasto cerca quindi i riferimenti alle procedure nell'albero e i riferimenti alle variabili globali nel codice. Un elenco di riferimenti riconosciuti non stabilisce che non ci siano ulteriori usi indiretti, dinamicamente risolti o offuscati.

Scorciatoie contestuali

Area attiva	Scorciatoia	Azione
Codice	N	Rinomina la variabile sotto il cursore (VB5/6).
Codice	X	Trova riferimenti alla variabile globale sotto il cursore.
Codice	D	Copia il codice decompilato corrente nel disassemblatore.
Codice	F4	Traccia all'indirizzo sulla riga corrente; richiede funzionalità di Tracing.
Codice	Ctrl+Shift+Z	Annulla la rinomina della variabile sotto il cursore.
Albero	X	Trova riferimenti alla procedura selezionata.
Albero	G	Apri la finestra di dialogo per passare a una sezione.
Albero	Ctrl+C	Copia il percorso completo dell'albero dell'oggetto selezionato.
Albero	Space or Insert	Seleziona o deseleziona l'elemento.

Le scorciatoie a lettera singola dipendono dalla vista focalizzata e non sostituiscono la digitazione normale nei campi di input. Utilizzare **Alt+1** per cambiare il focus. Vedere Ricerca di stringhe per Ctrl+T e Ctrl+F, Passaggio a un indirizzo per i percorsi dell'albero e gli indirizzi, e Tracciamento per le limitazioni dell'emulatore.

5 Miglioramento del codice con IA

Aiuto IA (AI Helper) elabora il codice già ricostruito da VB Decompiler. Supporta il C# decompilato da assembly .NET e il codice Visual Basic 5.0/6.0 da P-Code o Native Code. È possibile utilizzare un modello locale tramite Ollama o connettersi a un servizio tramite l'integrazione OpenAI API.

L'AI può suggerire flussi di controllo più chiari, nomi di variabili, dichiarazioni ed espressioni. Non può ripristinare in modo affidabile informazioni assenti dal binario o riparare un frammento interpretato in modo errato dal decompilatore. Confronta i suggerimenti con il listato decompilato originale, il disassemblaggio e le procedure circostanti prima di utilizzarli nel codice sorgente recuperato o in un rapporto di indagine.

Elaborazione di una procedura

1. Apri la procedura richiesta in Esplora soluzioni (Solution explorer) e esamina il suo listato **Decompilatore (Decompiler)**.
2. Passa a **Aiuto IA** e configura il provider, l'host e il modello come descritto di seguito.
3. Inizia l'elaborazione con il pulsante d'azione accanto al selettore del modello.
4. Revisiona il codice restituito in Aiuto IA e confrontalo con l'output del decompilatore.

L'elaborazione AI avviene in un thread separato. Usa **Cancella (Cancel)** se il modello impiega troppo tempo a rispondere. Le impostazioni aggiuntive di connessione possono essere compattate per lasciare più spazio al codice.

Utilizzo di un modello locale tramite Ollama

1. Installa Ollama dalla sua pagina di download ufficiale e avvia il servizio.
2. Scarica il modello consigliato eseguendo `ollama pull qwen3:8b` in un terminale.
3. Seleziona il provider LLM/Ollama locale in Aiuto IA. Imposta **Host Ollama (Ollama host)** sull'indirizzo del server; l'indirizzo locale predefinito è `http://localhost:11434`.
4. Seleziona il modello installato in **Mod. (Model)** e avvia l'elaborazione.

Consigliamo `qwen3:8b`. I prompt integrati di Aiuto IA sono ottimizzati per questo modello leggero. I modelli più recenti possono richiedere più tempo per elaborare il codice senza migliorare il risultato.

La scelta del modello non è limitata: è possibile usare qualsiasi modello installato nell'istanza di Ollama selezionata. Se il formato della risposta non è riconosciuto, il parser può lasciare nel risultato visualizzato tag specifici del modello o altro testo ausiliario.

Hardware per l’esecuzione locale con Ollama

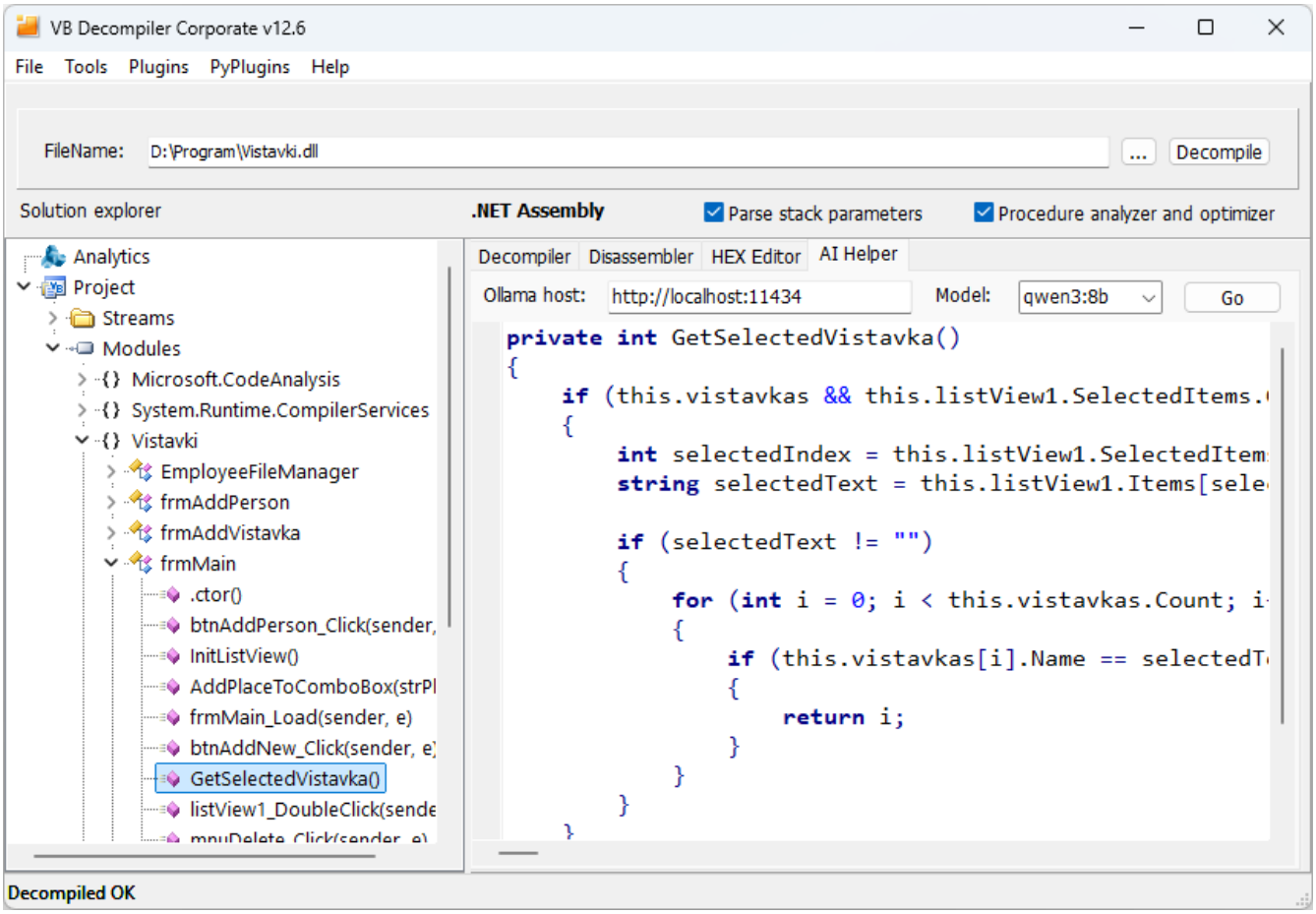
L’elaborazione IA locale viene eseguita sul computer che ospita Ollama e richiede molte più risorse della decompilazione. Le configurazioni pratiche seguenti si riferiscono a `qwen3:8b` con pesi `Q4_K_M`, cache del contesto standard `f16` e una richiesta alla volta. Il contesto si misura in token: 4K equivale a 4096 token e 128K a 131072 token. Il limite comprende istruzioni, codice sorgente e risposta generata.

Componente	Minimo pratico: contesto 4K	Consigliato: contesto 128K
Sistema operativo	Windows 10 (22H2) o Windows 11 a 64 bit	Windows 10 (22H2) o Windows 11 a 64 bit
Processore	CPU x64 moderna con AVX2 e almeno 4 core fisici	CPU x64 moderna con AVX2 e almeno 8 core fisici
Memoria di sistema (RAM)	16 GB	64 GB
Scheda grafica	GPU compatibile con Ollama con almeno 8 GB di VRAM dedicata	GPU compatibile con Ollama con almeno 32 GB di VRAM dedicata
Archiviazione	SSD con almeno 20 GB liberi per Ollama e questo modello	SSD con almeno 20 GB liberi per Ollama e questo modello

Impostare il contesto con il cursore nelle impostazioni di Ollama. Una scheda da 8 GB adatta a 4K non è sufficiente per la configurazione a 128K indicata sopra. Il modello scaricato occupa circa 5.2 GB, ma il fabbisogno totale di memoria è superiore. A 128K, la sola cache `f16` richiede circa 18 GiB, oltre ai pesi del modello e ai buffer di lavoro. Lasciare memoria disponibile per Windows e le altre applicazioni.

Con RAM sufficiente Ollama può usare la CPU, ma l’elaborazione del codice può essere molto lenta. I sistemi Pentium M e i computer di età analoga non sono adatti all’esecuzione locale di `qwen3:8b`. La memoria condivisa delle vecchie Intel HD Graphics non sostituisce la VRAM dedicata indicata sopra. Questi requisiti riguardano il computer con Ollama; un server remoto o un servizio API esterno usa il proprio hardware.

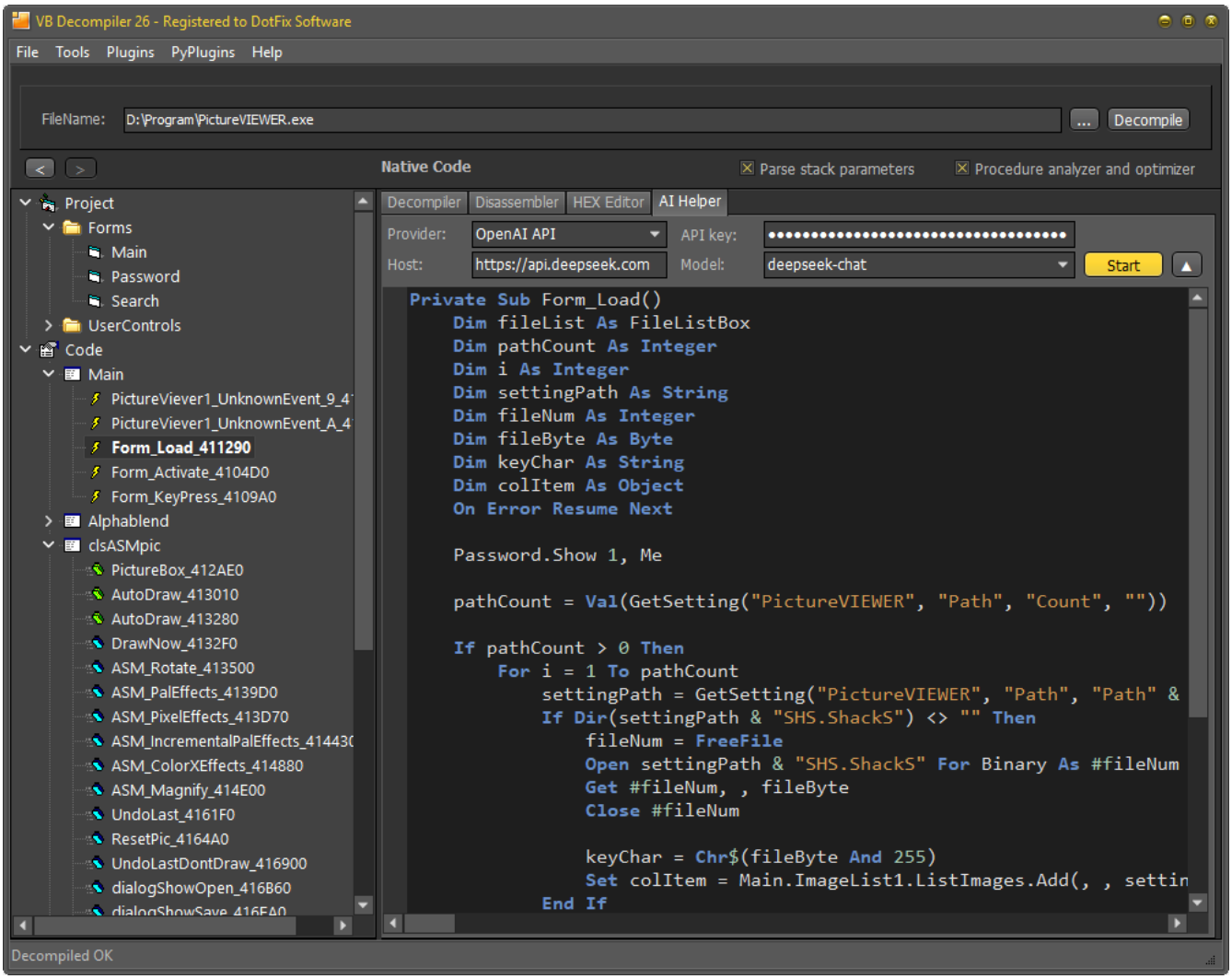
Per procedure brevi, iniziare da 4K. Un contesto più ampio aumenta il consumo di memoria e il tempo necessario per elaborare input lunghi. Se l’elaborazione è troppo lenta, annullarla, ridurre il contesto e controllare l’uso della GPU con `ollama ps`. Il consumo effettivo dipende dalla versione di Ollama, dal backend e dagli altri carichi in esecuzione.



Utilizzo dell'integrazione OpenAI API

Imposta **Servizio (Provider)** su **OpenAI API**. Configura **Host**, **API key** e **Mod.** per il servizio previsto. Inserisci una chiave API valida e un identificatore di modello disponibile per tale chiave. Mantieni le chiavi API fuori dagli screenshot e dai materiali di analisi condivisi.

Aiuto IA supporta anche DeepSeek tramite il provider **OpenAI API**. Inserisci l'host DeepSeek, la chiave API e l'identificatore del modello nei campi corrispondenti. Lo screenshot sottostante mostra questa configurazione.



Dove viene elaborato il codice

Aiuto IA invia il codice decompilato corrente al servizio configurato. Per un flusso di lavoro esclusivamente locale, utilizza un server Ollama in esecuzione localmente e un modello eseguito localmente. Un host Ollama su un'altra macchina, un modello Ollama cloud o un servizio API esterno possono trasferire il codice fuori dalla workstation di analisi.

Per il recupero di sorgenti riservate o indagini sulla sicurezza, scegli l'endpoint in base ai requisiti di gestione dei dati della tua organizzazione. Non dedurre il funzionamento offline solo dalla presenza di Ollama o da un URL localhost.

Risoluzione dei problemi

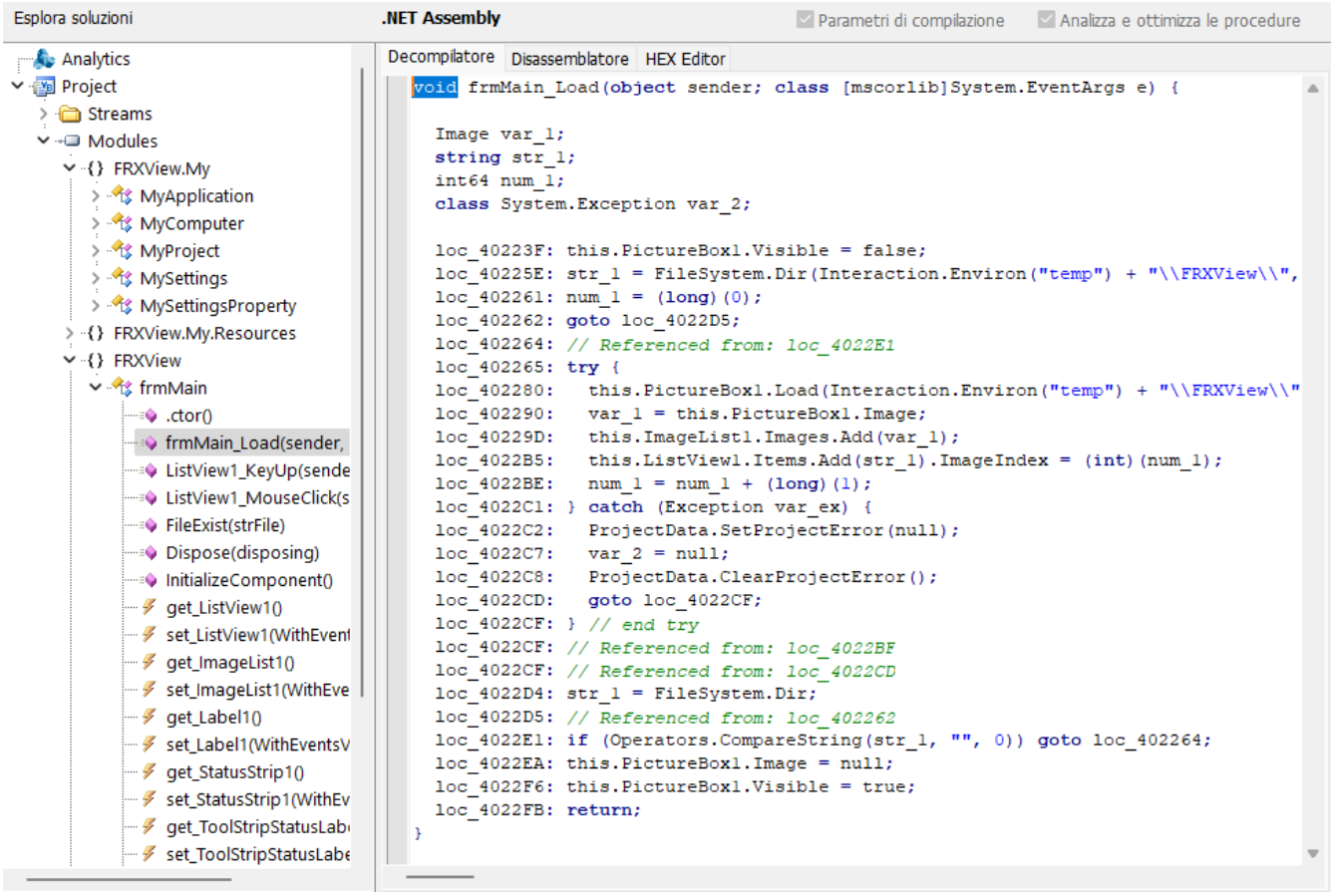
- Se manca un modello locale, verifica che sia installato nell'istanza Ollama specificata dal campo host.
- Se non c'è risposta, verifica che il servizio selezionato sia in esecuzione e raggiungibile, e che l'endpoint, le credenziali e l'identificatore del modello corrispondano allo stesso provider.
- Se l'elaborazione è troppo lenta, annullala. Quando si utilizza un altro modello locale, prova il modello consigliato `qwen3:8b`.
- Se un modello sconosciuto lascia tag o altri testi ausiliari nella sua risposta, prova `qwen3:8b` o rimuovi manualmente quel testo mentre revisioni il risultato.
- Se il risultato modifica una condizione, chiamata, costante o tipo in modo imprevisto, confrontalo con il listato a basso livello. La leggibilità da sola non stabilisce un comportamento equivalente.

Ollama e i provider di modelli esterni sono prodotti di terze parti. La loro disponibilità, il comportamento dei modelli e le politiche del servizio sono indipendenti da VB Decompiler.

6 Apertura e salvataggio

6.1 Apertura di un file

Prima di avviare il processo di decompilazione, è necessario selezionare un file per l'analisi. A tale scopo, esiste una sezione **Apri programma (Open program)** nel menu **File**. **VB Decompiler** supporta l'apertura di file EXE/OCX/DLL creati nelle versioni **Visual Basic** 5.0 o 6.0 nonché applicazioni compilate in **Visual Studio .NET**. In quest'ultimo caso, il linguaggio sorgente non ha importanza poiché Visual Studio .NET compila le applicazioni indipendentemente dal linguaggio originale in un unico formato interno - **MS IL**.



Dopo aver aperto il file, il decompilatore verifica se il programma appartiene agli ambienti di sviluppo specificati. Se il programma aperto è stato sviluppato in un ambiente non supportato (ad esempio, **Delphi** o **Visual C++**), appare un messaggio che indica che tale ambiente non è supportato ed è impossibile decompilare il programma. In caso di comparsa di questo messaggio, è possibile contattarci e potremmo essere in grado di assistervi nella decompilazione del vostro programma. Disponiamo di strumenti interni per la decompilazione di applicazioni **Delphi** e **FreePascal** e possiamo offrire questa attività come servizio separato.

Se il programma aperto è compresso con un **packer EXE**, obfuscatore o protetto - il decompilatore offrirà di tentare di decomprimere questo file. Questa funzionalità è disponibile solo nelle versioni a 32-bit di Windows e viene fornita senza alcuna garanzia. Se la decompressione è possibile - il file verrà avviato, decompresso nella memoria, salvato su disco con un nome diverso e aperto per la decompilazione. **Attenzione!** Non utilizzare mai il decompressore integrato per decomprimere malware o altro codice che potrebbe danneggiare il sistema: durante la decompressione, il programma aperto viene eseguito e può causare gravi danni al PC.

Nota importante: sotto nessuna circostanza dovete tentare di decompilare o decomprimere software senza disporre delle autorizzazioni necessarie, e se il titolare del copyright vieta la decompressione e l'analisi di questo file tramite licenza. Questo può violare le leggi del vostro paese.

*Visual Studio .NET, Visual C++ e Visual Basic sono marchi registrati della Microsoft Corporation. Delphi è un marchio registrato di Embarcadero.

Altri modi per aprire un file

È possibile trascinare un file dall'Esplora Risorse su VB Decompiler. File con estensione sconosciuta o senza estensione possono essere inviati anche in questo modo; il riconoscimento dipende comunque dal contenuto effettivo del file. I collegamenti Windows **.lnk** supportati possono essere risolti nei loro file di riferimento. I database VDD salvati possono essere aperti anche tramite trascinamento o da riga di comando.

6.2 Utilizzo dei database di analisi

Questa funzionalità è disponibile tramite la sezione **Database (Data Base)** nel menu **File** e serve ad abilitare il salvataggio dei risultati della decompilazione in un file compresso separato. Questo può essere utile per un caricamento più rapido dei risultati della decompilazione in futuro senza dover ripetere la decompilazione del codice.

Nel caso in cui sia necessario tornare all'analisi del programma o condividere questi risultati con un collega, non è necessario eseguire nuovamente la decompilazione. Nel caso di una decompilazione completa (con l'opzione **Decompilazione rapida (Fast decompilation)** disabilitata), questo processo può richiedere più tempo e risorse. È molto più conveniente pre-salvare il codice decompilato nel **Database**.

VB Decompiler utilizza un formato di database compresso interno, che consente una significativa ottimizzazione dello spazio necessario per memorizzare i risultati. Inoltre, esiste compatibilità con i database salvati dalla maggior parte delle versioni precedenti di VB Decompiler, consentendo di memorizzare le informazioni senza rischiare la possibilità di non poter aprire un tale database nella nuova versione del prodotto.

Apertura e trasferimento dei database VDD

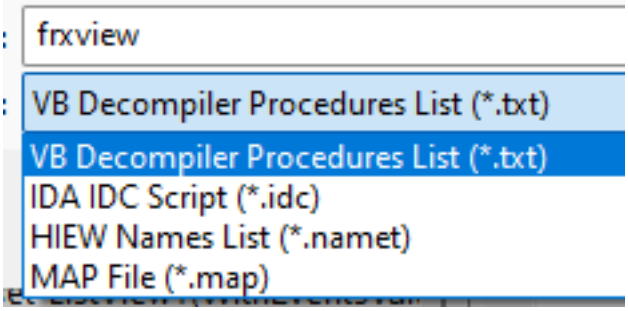
I database di analisi salvati utilizzano l'estensione **.vdd**. È possibile aprire un database VDD tramite il flusso di lavoro di apertura dei file, trascinarlo sulla finestra del decompilatore o passare il suo percorso nella riga di comando.

Nella localizzazione dell'eseguibile originale analizzato, VB Decompiler controlla sia il suo percorso salvato che la directory contenente il database. Per la collaborazione, mantieni l'eseguibile corrispondente accanto al file VDD quando il suo percorso originale non è disponibile. Conserva l'edizione analizzata esatta in modo che gli indirizzi e i byte del file rimangano significativi.

6.3 Salvataggio dell’elenco delle procedure

Questa operazione viene eseguita chiamando l'opzione **Salva lista procedure (Save procedures list)** nel menu **File** ed è uno strumento potente per l'analisi dei programmi in quanto consente a utility esterne di assistere il **VB Decompiler**.

Attualmente, indirizzi, nomi di funzioni e procedure nel programma elaborato possono essere salvati in diversi formati. È possibile scegliere tra queste opzioni specificando il tipo di file nella finestra di dialogo standard di salvataggio.



Salvataggio in formato TXT consente di salvare l'elenco delle procedure in un semplice file di testo. Questo può essere utile per confrontare le procedure tra diverse versioni di un programma, rilevare cambiamenti negli indirizzi e così via.

Salvataggio come script IDC consente di caricare gli indirizzi delle funzioni e delle procedure nel disassemblatore **IDA Pro**. Nei casi in cui il programma analizzato viene compilato in **Native Code**, questa funzionalità può essere molto utile.

Salvataggio in formato NAMET consente di semplificare l'analisi utilizzando l'utility **HIEW**. Copiare il file ottenuto nella cartella contenente il programma analizzato. Quindi aprire il programma in HIEW e premere il tasto F12. Apparirà una finestra con un elenco vuoto delle procedure. Premere di nuovo il tasto F12 se l'indirizzo del file NAMET corrisponde a quello creato - semplicemente premere Invio. Gli indirizzi delle procedure verranno caricati in HIEW, consentendo di navigare tra i metodi utilizzando l'elenco premendo il tasto F12.

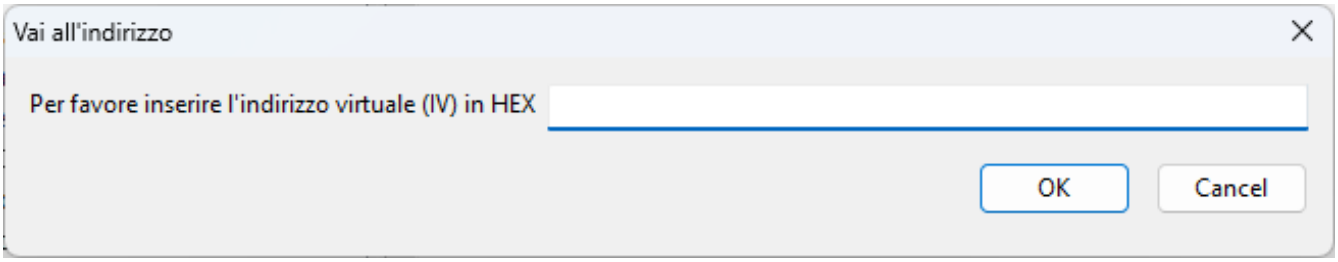
Salvataggio in formato MAP genera un file mappa standard che viene solitamente creato dai compilatori. Questo file può essere utilizzato con molti debugger e disassemblatori per segnare l'inizio delle funzioni del programma. Poiché i programmi scritti in Visual Basic sono spesso disassemblati in modo errato da software di terze parti, il file MAP generato in **VB Decompiler** semplificherà notevolmente il processo. Questo formato è supportato tramite plugin in vari prodotti come **OllyDbg**, **IDA Pro** e altri.

7 Funzioni del programma

7.1 Passaggio a un indirizzo

Salto a un indirizzo virtuale

Seleziona **Strumenti > Vai a Indirizzo Virtuale (Tools > Jump to virtual address)** oppure premi **Ctrl+G**. Inserisci l'indirizzo virtuale del codice da ispezionare. Puoi anche selezionare un indirizzo nell'elenco e invocare Ctrl+G. Il decompilatore apre il codice associato a quell'indirizzo.



Un indirizzo virtuale (VA) identifica una posizione nello spazio degli indirizzi dell'immagine eseguibile e include la sua base immagine. Un offset nel file identifica una posizione di byte nel file su disco. Non sostituire l'uno con l'altro: l'Editor HEX supporta entrambe le forme di navigazione.

Ritorno tramite cronologia di navigazione

Usa i pulsanti freccia sopra l'albero per muoverti indietro e avanti attraverso le procedure precedentemente visualizzate. **Esc** ritorna alla procedura precedente. I pulsanti Indietro e Avanti del mouse usano anche questa cronologia, mantenendo la posizione all'interno della procedura.

Condivisione di una posizione tramite il suo percorso dell'albero

In Esplora soluzioni (Solution explorer), fai clic con il pulsante destro sull'oggetto richiesto e seleziona **Copia il nome della sezione (Copy the name of section)**, oppure premi **Ctrl+C** mentre l'albero ha il focus. Questo copia il percorso completo del nodo selezionato.

Per rivedere quella posizione, seleziona **Vai alla sezione (Jump to the section)** nel menu contestuale dell'albero, oppure premi **G** nell'albero, e incolla il percorso copiato nel dialogo. Questo è utile quando si consegna un'indagine a un altro analista o si ritorna su un progetto di grandi dimensioni. Usa il percorso con il file analizzato corrispondente; un percorso da un'altra compilazione potrebbe non identificare più lo stesso codice.

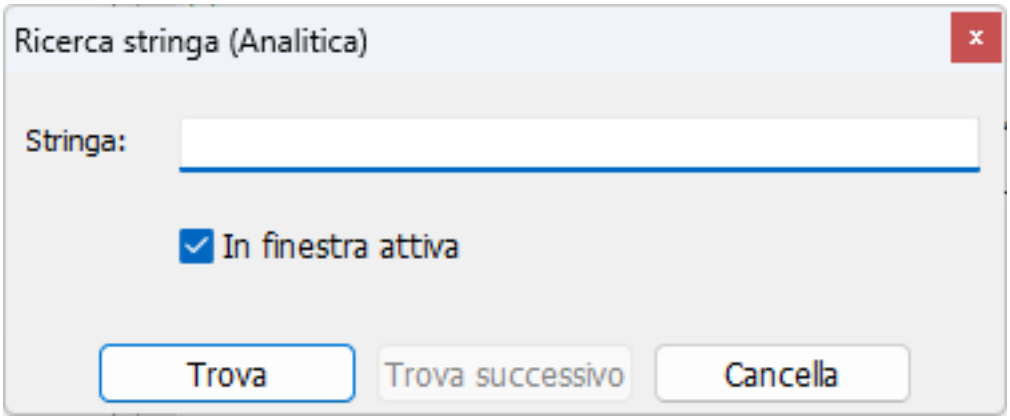
7.2 Ricerca di stringhe

Trovare una procedura per nome

Selezionare **Strumenti > Cerca routine (Tools > Find procedure)** o premere **Ctrl+T**. Inserire il nome completo della procedura o una parte distintiva dello stesso. La ricerca copre l'albero del progetto, quindi non è necessario sapere in quale modulo o classe si trova la procedura. Se corrispondono più nomi, l'elenco dei risultati contiene tutte le voci corrispondenti; fare doppio clic sulla voce richiesta per aprirla.

Ricerca testo visualizzato o intero progetto

Selezionare **Strumenti > Find string (Tools > Find string)** o premere **Ctrl+F** per aprire la finestra di dialogo della ricerca del testo. Con **In finestra attiva (In active window)** selezionato, la ricerca si applica al codice o ai dati del form attualmente visualizzati. Deselezionare questa casella di controllo per cercare in tutto il progetto decompilato.



Inserire il testo in **Stringa (String)** e selezionare **Trova (Find)**. Utilizzare **Trova successivo (Find next)** per continuare la ricerca dello stesso testo. **Cancella (Cancel)** chiude la finestra di dialogo.

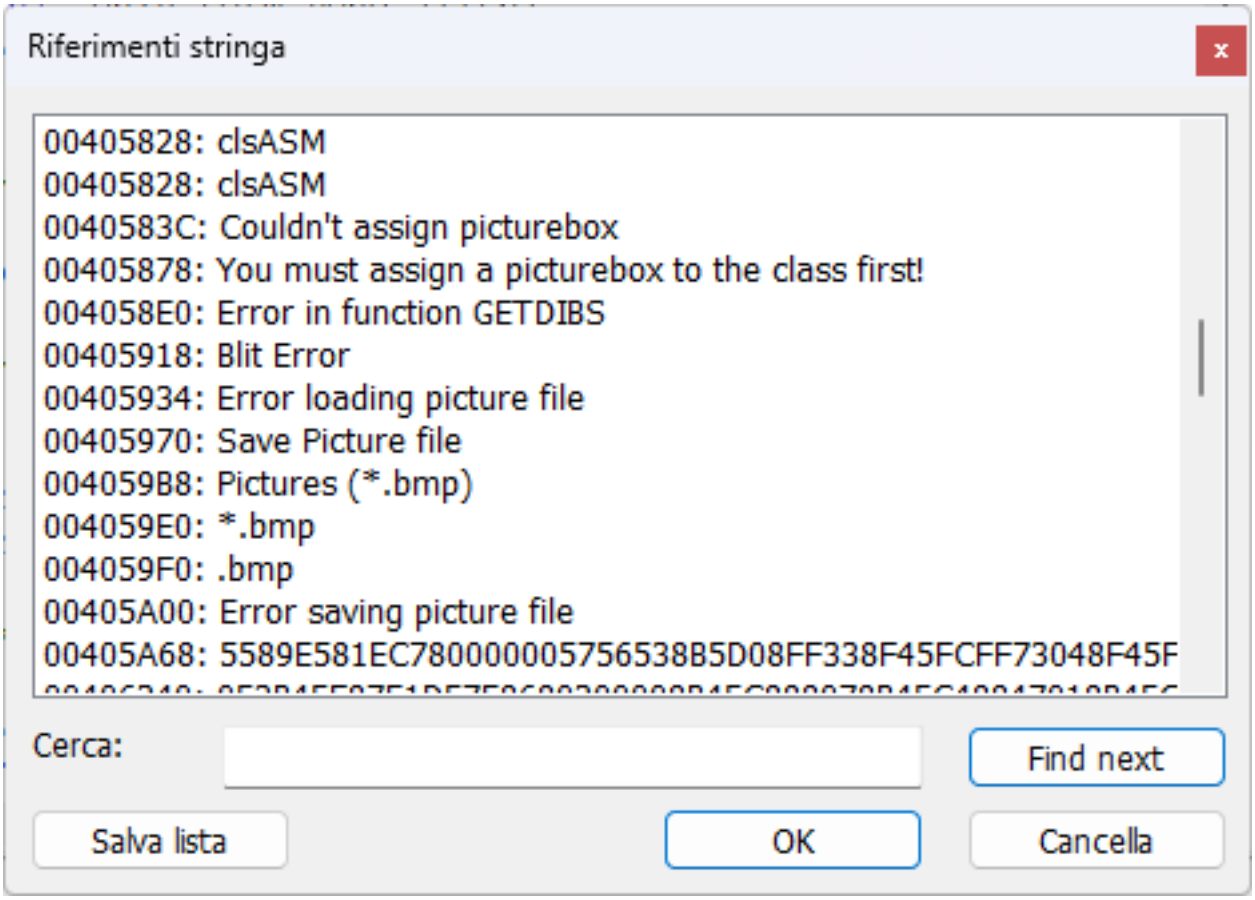
La ricerca di tutto il testo decompilato richiede una decompilazione completa. Disabilitare **Decompilazione rapida (Fast decompilation)** in Funzionalità avanzate ed elaborare nuovamente il file prima. La ricerca di procedure per nome è un'operazione separata sulla struttura del progetto.

Ricerca riferimenti a stringhe

Se sono necessarie stringhe letterali e il codice che le utilizza, aprire Riferimenti alle stringhe. Il campo **Cerca (Search)** e il pulsante **Trova successivo** eseguono la ricerca nell'elenco delle stringhe. I Riferimenti alle stringhe possono essere raccolti tramite una disassemblaggio preliminare in modalità Decompilazione Rapida; non richiedono la ricerca di ogni procedura completamente decompilata.

7.3 Riferimenti alle stringhe

Nel codice di qualsiasi programma, si incontrano spesso dati a stringa (chiavi di registro, nomi file, messaggi di errore, ecc.). Può essere molto utile visualizzare l'elenco delle stringhe utilizzate da procedure e funzioni in applicazioni decompilate. È esattamente a questo scopo che è progettata la nostra funzione **Riferimenti stringa (String references)**, accessibile tramite il menu **Strumenti (Tools)**. Per utilizzarla, basta navigare su **Strumenti** -> **Riferimenti stringa** e, se il programma contiene dati a stringa, questi verranno visualizzati come mostrato di seguito:

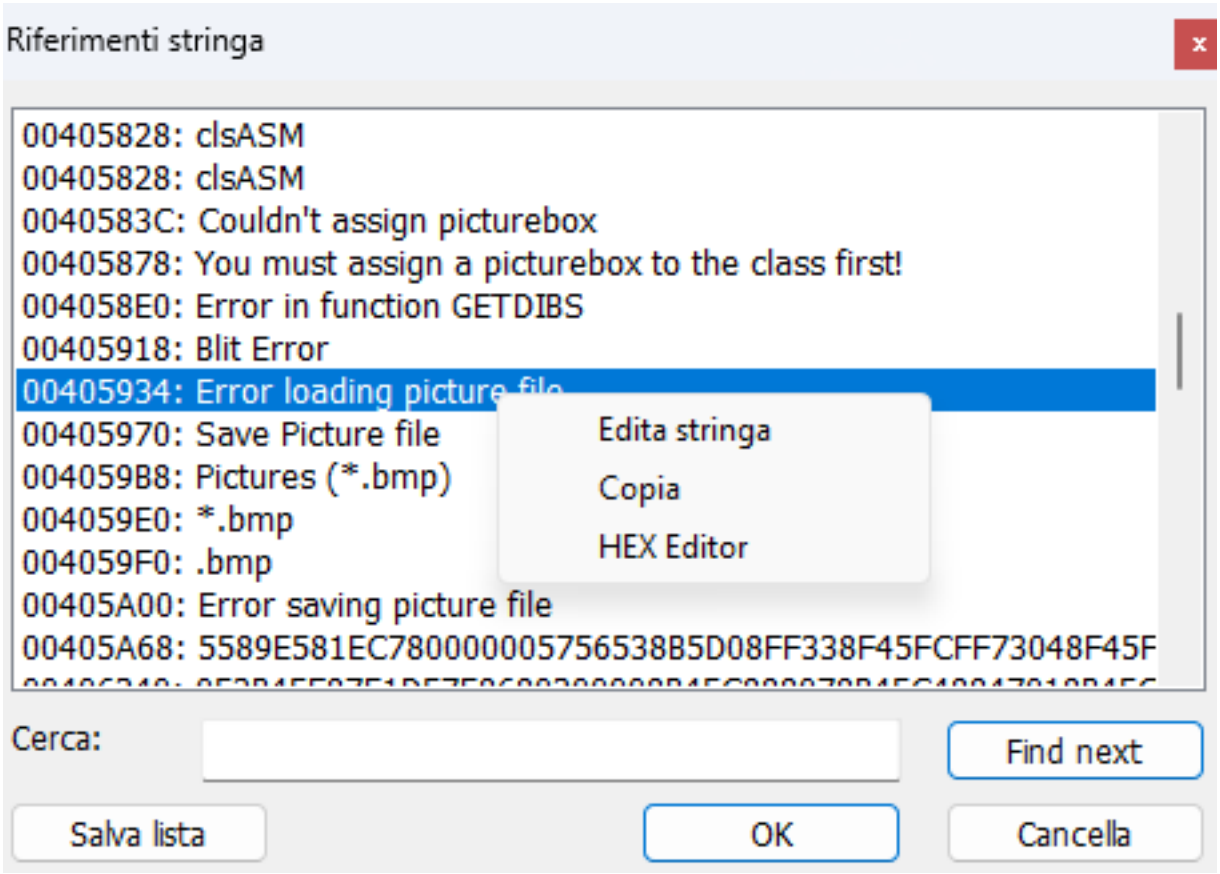


Trova la riga desiderata nell'elenco e fai doppio clic su di essa. Verrà localizzato il segmento di codice contenente quella riga e la procedura corrispondente verrà aperta nella finestra del codice. Fare doppio clic nuovamente sulla riga continua la ricerca di questa riga all'interno del codice del programma. Cliccando su 'OK', viene trovata la riga evidenziata nel codice e viene chiusa la finestra di selezione della riga. Annullare chiude semplicemente la finestra di dialogo.

La funzione per visualizzare **Riferimenti stringa** è disponibile solo nella versione licenziata del software e non è accessibile per la versione **Lite**.

Nota importante: le stringhe vengono rivelate non solo per il modulo attivo come nelle versioni 1.x del programma, ma per l'intero programma. Questa modifica è stata apportata per facilitare l'analisi di programmi con un gran numero di moduli e form.

VB Decompiler supporta anche la modifica delle stringhe. Per chiamare la finestra di dialogo per la modifica della stringa, fai clic con il pulsante destro del mouse sulla stringa che desideri modificare e seleziona **Edita stringa (Edit string)** dal menu contestuale che appare. Apparirà una finestra di dialogo per inserire una nuova stringa. Dopo aver modificato la stringa, clicca su **Ok** e la stringa verrà modificata direttamente nel file editato. Si consiglia di creare un backup del file in esame prima, poiché non viene effettuato alcun backup durante questa azione e l'annullamento di questa operazione non è possibile.



Nota importante: quando si modificano le stringhe, ricorda che la lunghezza della nuova stringa non deve superare quella della vecchia. Questo è legato al formato dei file binari delle applicazioni in cui non viene riservato spazio per inserire una stringa più lunga.

È possibile anche visualizzare e modificare parzialmente la stringa nell'editor HEX. Per fare questo, fai clic con il pulsante destro del mouse sulla stringa e seleziona Editor HEX dal menu che appare. Nell'editor HEX, ci sono funzioni per annullare e ripristinare le modifiche, il che semplifica la sperimentazione quando si modificano i dati a stringa delle applicazioni.

La finestra con l'elenco delle stringhe ha anche un campo di testo chiamato **Cerca (Search)** e un pulsante **Trova successivo (Find next)**. Questa funzionalità sarà utile per trovare le stringhe all'interno dell'elenco, specialmente quando si trattano progetti grandi in cui la ricerca visiva manuale attraverso l'elenco è estremamente difficile.

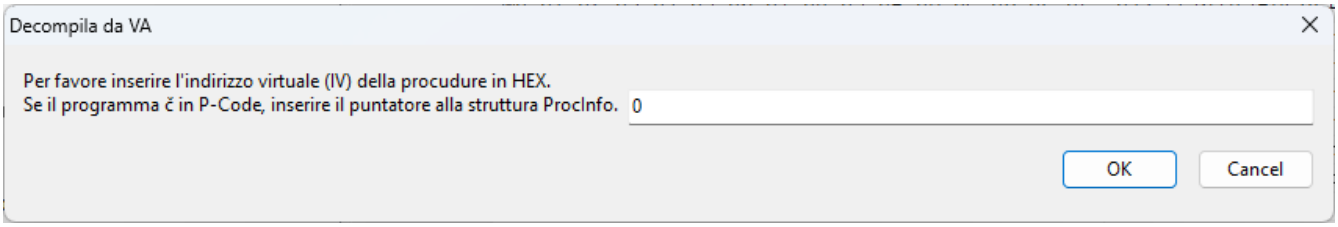
Decompilazione rapida e elenchi di stringhe grandi

I riferimenti alle stringhe possono essere raccolti tramite una disassemblaggio preliminare, incluso il modo **Decompilazione rapida (Fast decompilation)**. Questo differisce dalla ricerca di tutto il testo completamente decompilato. La versione 26.2 espande la finestra oltre 32,000 riferimenti e migliora la gestione della memoria per elenchi grandi.

Per gli assembly .NET, le stringhe utente dall'heap #US possono essere ispezionate con le loro posizioni file. L'elenco rappresenta le stringhe disponibili per l'analisi statica; le stringhe crittografate o generate dinamicamente potrebbero richiedere un'indagine separata.

7.4 Decompilazione da un indirizzo

Nel menu **Strumenti (Tools)**, esiste una funzione molto utile - **Decompila da VA (Decompile from VA)** (decompila qualsiasi codice dall'indirizzo virtuale fornito). La caratteristica principale di questa funzione è che non richiede che il file da decompilare abbia una struttura corretta e venga elaborato in modo standard. Naturalmente, in questo caso, il codice verrà decompilato con un certo numero di limitazioni. Ad esempio, se le funzioni importate o gli oggetti OLE non sono stati pre-elaborati, il codice risulterà poco informativo in quei punti.



Per decompilare una specifica porzione di codice all'interno di un file, è sufficiente specificare l'indirizzo virtuale di tale codice (se si parla di **Native Code** o **assembly .NET**), oppure fornire un riferimento alla struttura **ProcInfo** (se si parla del classico **P-Code** di **Visual Basic**). L'unicità del lavoro con tali **P-Code** è che la struttura **ProcInfo** contiene un array di dati utilizzati dal codice. Senza questo array, non è possibile determinare con quali funzioni il codice lavora, a quali stringhe fa riferimento e così via. Nel caso del lavoro con **assembly .NET**, dopo aver inserito l'indirizzo, il decompilatore vi offrirà la scelta tra decompilare il codice come codice intermedio **MS IL** o disassemblare questo codice. Questa opzione è dovuta al fatto che gli assembly .NET possono contenere sia codice intermedio MS IL (**codice gestito**) sia codice macchina (cosiddetto **codice non gestito**).

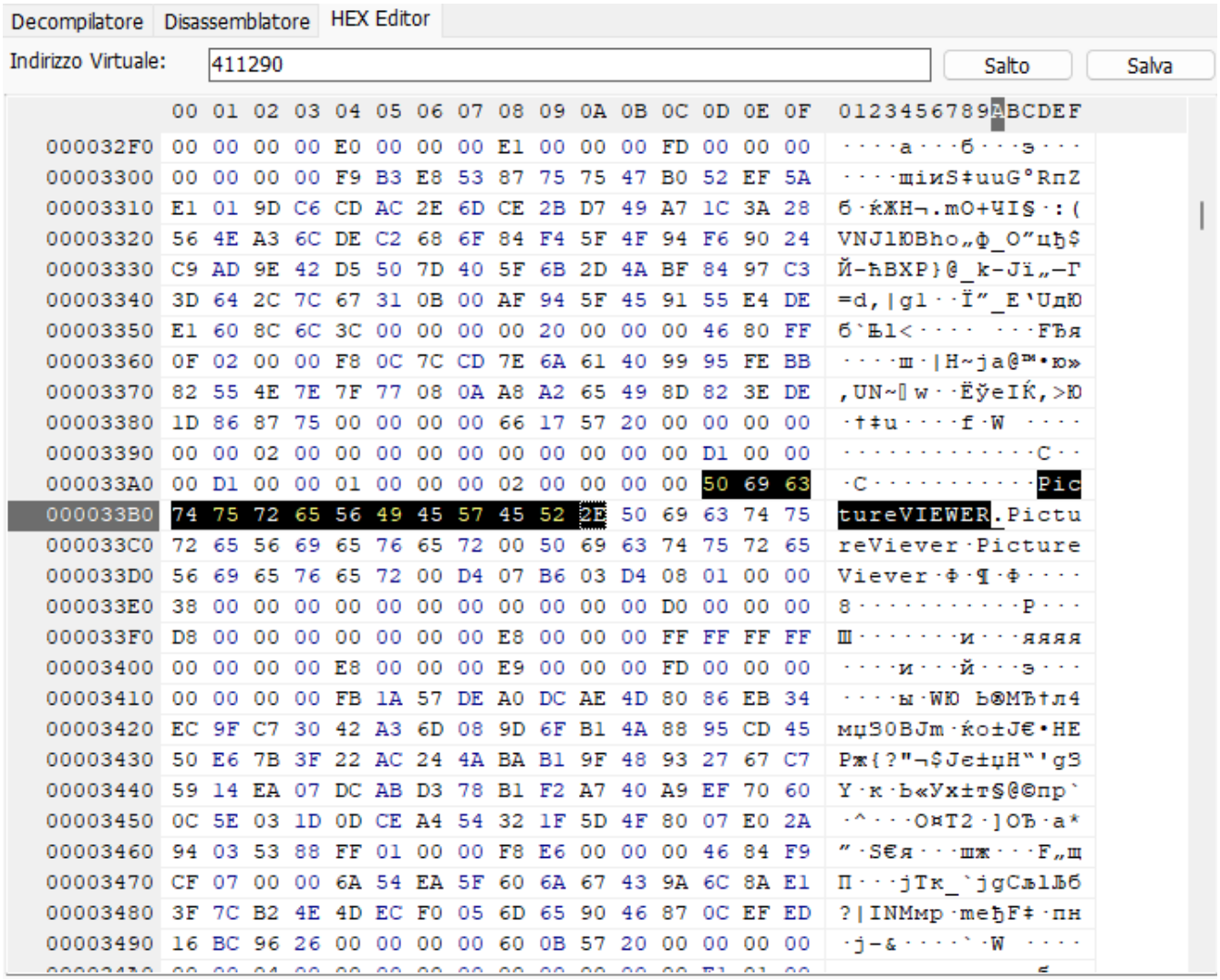
Se il file è già aperto nel decompilatore, è sufficiente chiamare la funzione **Decompila da VA** e specificare l'indirizzo da elaborare. Se il file non è ancora stato aperto, questa operazione verrà suggerita automaticamente e il file verrà aperto in modalità semplificata.

La funzione **Decompila da VA** è particolarmente utile quando il codice desiderato non viene decompilato automaticamente o se il file è parzialmente danneggiato, impedendone l'analisi con la procedura standard. Naturalmente, dovete comprendere che il risultato della decompilazione in questo caso sarà peggiore rispetto a una decompilazione standard.

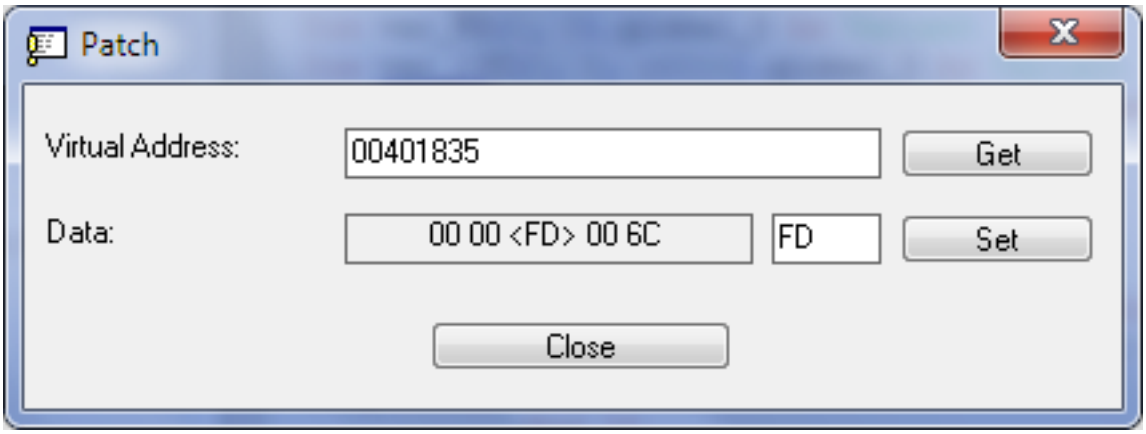
7.5 Modifica dei dati

A partire dalla versione 9.4, **VB Decompiler** include uno strumento moderno e comodo per modificare i dati binari all'interno dei file: un editor completo **HEX Editor**. È accessibile dalla finestra principale del programma insieme al codice decompilato nella scheda **HEX Editor** per ciascuna procedura o modulo.

L'editor **HEX Editor** consente di modificare i byte del file uno alla volta e anche in blocchi. Per navigare all'indirizzo desiderato all'interno del file esaminato, è possibile utilizzare sia gli indirizzi virtuali (inclusi **Image Base**) che gli offset (un indirizzo regolare che indica l'offset dei dati necessari rispetto all'inizio del file).



Nelle versioni precedenti del decompilatore (prima della versione 9.4), la funzione per modificare i file era applicata tramite uno strumento più primitivo: Data Patcher. Funzionava come segue. Nel menu **Strumenti (Tools)** -> **Patch dati (Patch data)**, c'era uno strumento chiamato patcher che poteva essere utilizzato nei casi in cui, dopo aver analizzato il codice del programma, era necessario modificare una parte. Per modificare i byte necessari senza utilizzare strumenti esterni, **VB Decompiler** disponeva di funzionalità integrate. La finestra del patcher aveva questo aspetto:



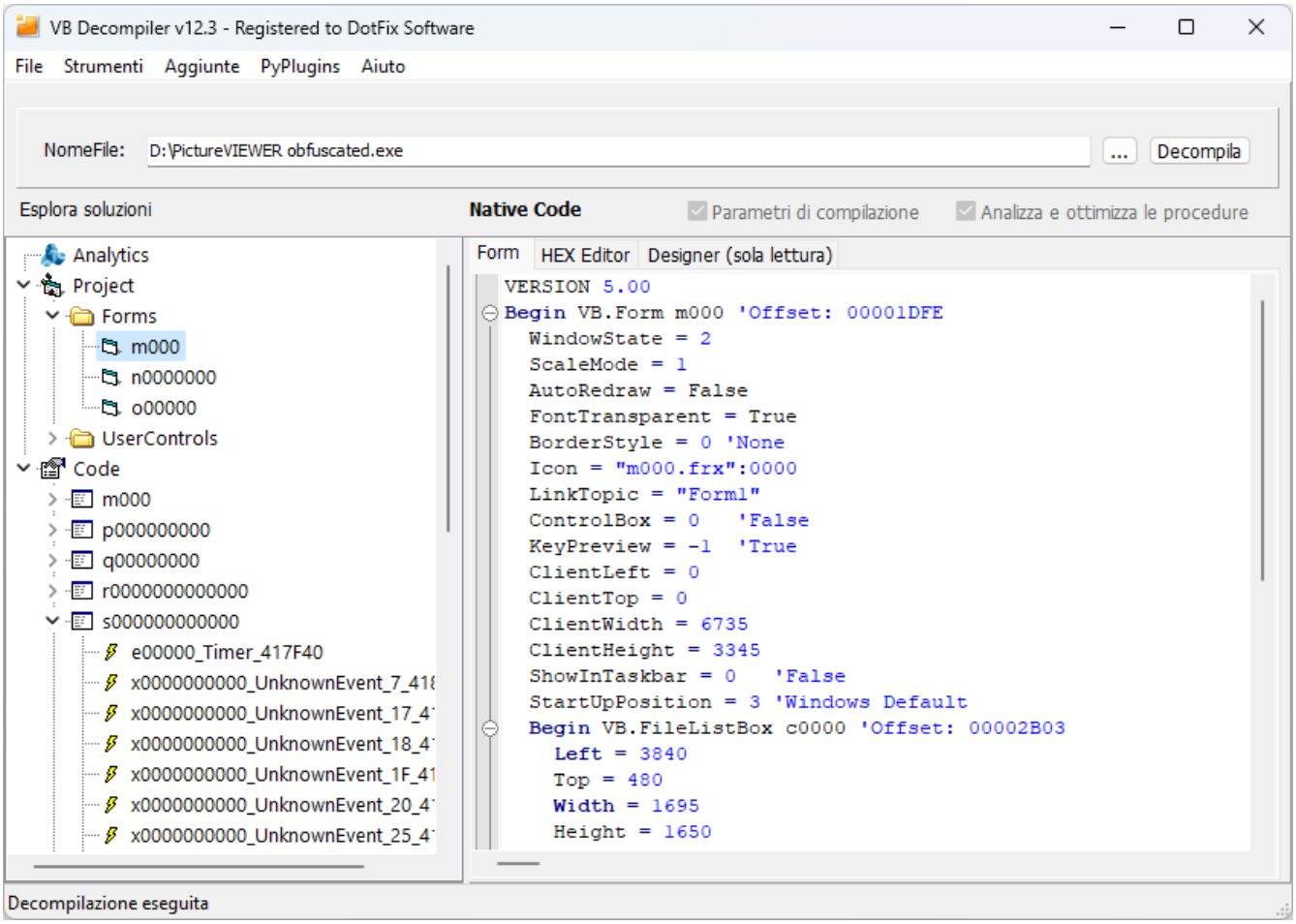
Il lavoro con il patcher era estremamente semplice. Nel campo **Indirizzo Virtuale (Virtual Address)**, si inseriva l'indirizzo per la modifica di un byte, quindi si premeva **Enter** oppure si cliccava su **Get**. I byte dei dati situati nelle vicinanze dell'indirizzo selezionato venivano visualizzati nel campo Dati e il byte selezionato poteva essere modificato, dopodiché era necessario cliccare su **Set**, e il byte inserito veniva modificato.

Se nell'Opzioni del programma c'era un'opzione per **Decompilazione rapida (Fast decompilation)** abilitata e l'opzione **Codice decompilato cache (Cache decompiled code)** disabilitata, allora tutte le modifiche venivano riflesse in quella funzione, dove i byte erano stati modificati. Per fare questo, era sufficiente riaprire la stessa funzione facendo doppio clic sul suo nome nell'albero.

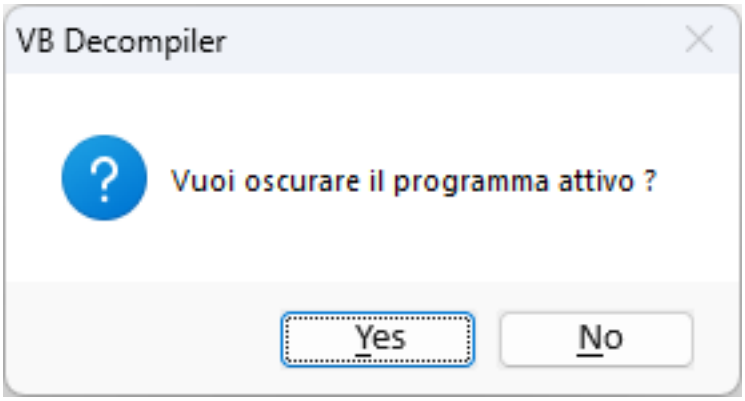
Nota importante: questa funzionalità è progettata per modificare solo i file del proprio programma o quelli per i quali si hanno i diritti appropriati, nonché per analizzare codice malevolo in laboratori antivirus. Prima di modificare il codice di qualcun altro - assicurarsi di rivedere l'accordo di licenza riguardo alla presenza di tale diritto.

7.6 Offuscamento

Se si stanno sviluppando programmi in **Visual Basic 5.0/6.0**, è possibile aumentarne la protezione contro la decompilazione. Per ottenere questo risultato, viene incorporato all'interno del decompilatore l'**Obfuscator**, capace di rimuovere tutti i nomi di funzioni, form, moduli e oggetti dal file binario del programma, nonché varie firme. Ciò renderà più difficile l'analisi del programma sia utilizzando il decompilatore fornito che altre soluzioni. Inoltre, la maggior parte dei prodotti di terze parti rifiuterà di decompilare completamente questo file.



Per applicare questa protezione, aprire il progetto nel decompilatore e selezionare **Offuscare progetto (Obfuscate project)** dal menu **Strumenti (Tools)**. Questo processo non richiede molto tempo, quindi si consiglia di offuscare i propri progetti prima della pubblicazione.



Questa funzionalità è anche integrata nella nostra soluzione completa per la protezione del software - **DotFix NiceProtect**, che può essere scaricata da www.niceprotect.com.

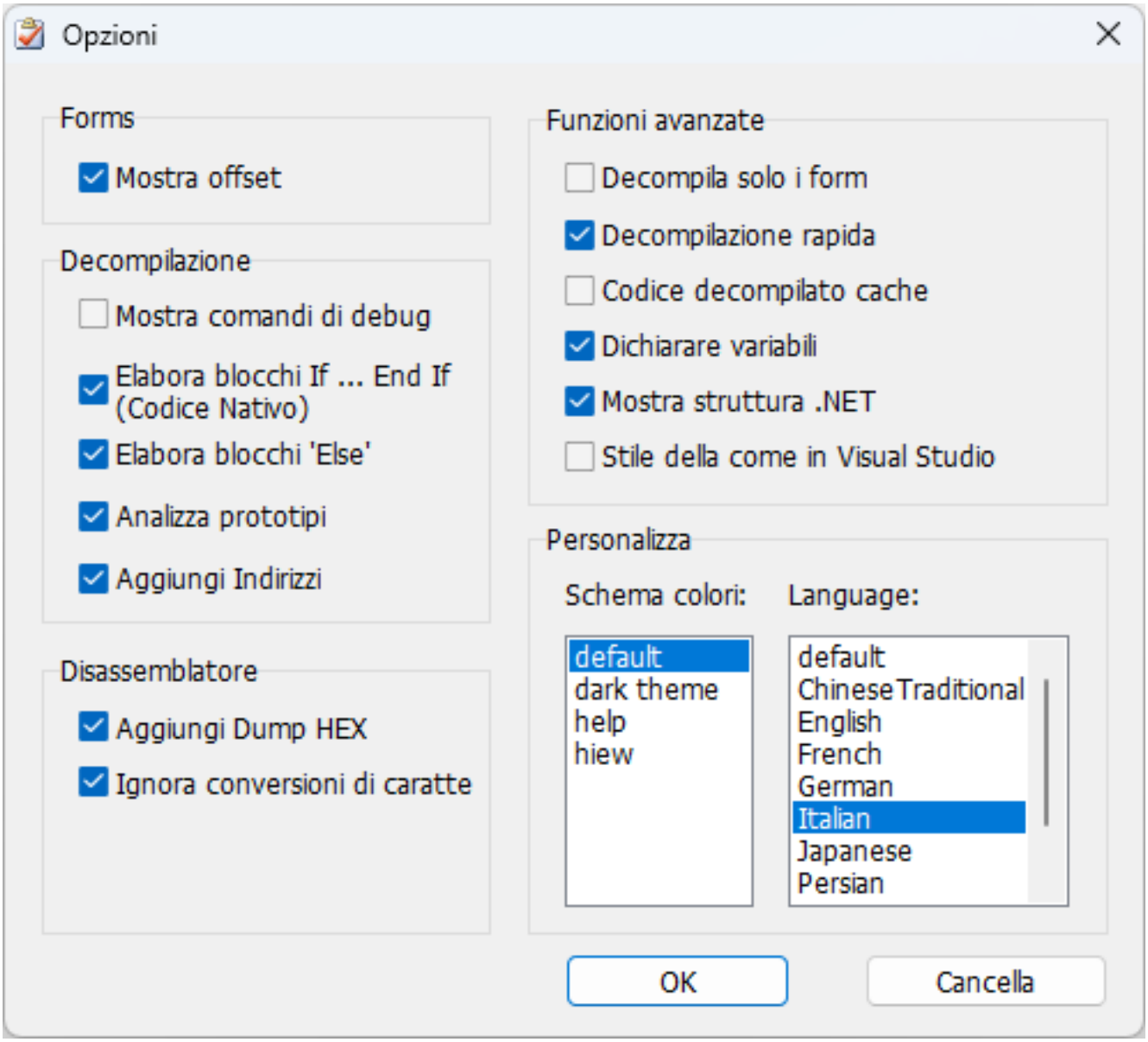
L'offuscamento è disponibile solo per gli utenti con una licenza Business del **VB Decompiler**.

Restrizioni di offuscamento

8 Opzioni del programma

8.1 Opzioni del programma

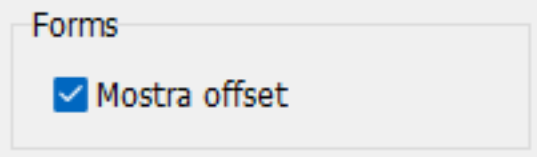
Per modificare le impostazioni del programma, vai al menu **Strumenti (Tools)** > **Opzioni (Options)**. Nella finestra di dialogo risultante, vedrai le seguenti opzioni:



Form
Decompilazione
Disassemblatore
Schema colori Funzionalità avanzate Selezione della lingua

8.2 Form

È possibile modificare le seguenti impostazioni di decompilazione per i form:



In questa sezione delle impostazioni, è possibile abilitare la visualizzazione degli offset accanto agli oggetti form. Questo può essere necessario, ad esempio, quando si localizza un programma in un'altra lingua senza accesso al codice sorgente. **Mostra offset (Show offsets)** consentirà di trovare la posizione all'interno del programma dove sono situati questi oggetti, i loro nomi e le loro proprietà.

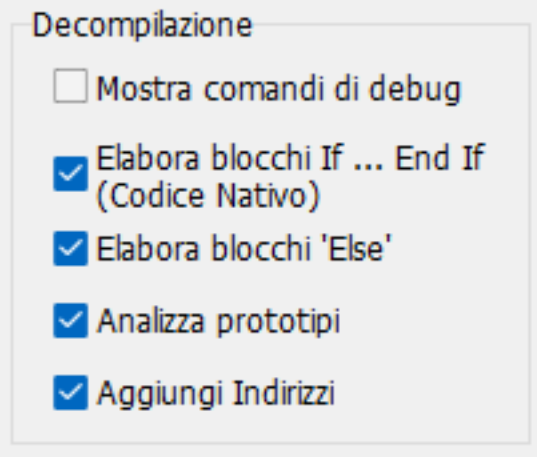
Successivamente, utilizzando un Editor HEX, è possibile modificare questi valori. Si noti che se le proprie conoscenze si limitano al livello utente di base e non si sono familiarizzati con il funzionamento interno dei file binari - si raccomanda di non abilitare questa opzione. Qualsiasi modifica errata apportata al file lo renderà non funzionante, e gli indirizzi non necessari semplicemente ostacoleranno l'utente.

Anteprima visiva e navigazione degli eventi

Le opzioni form sono separate dal visualizzatore **Designer (sola lettura) (Designer (read only))**. Vedere Ricostruzione visiva del form per i form VB classici, Windows Forms, supporto WPF/BAML, navigazione degli handler di eventi e le limitazioni di ciascuna tecnologia dell'interfaccia.

8.3 Decompilazione

Apri **Strumenti > Opzioni (Tools > Options)** per configurare la decompilazione. Quando si confrontano le impostazioni in modo interattivo, usa Decompilazione rapida senza cache; altrimenti rielabora il file dopo aver modificato un'opzione.



Mostra comandi di debug (Show debug commands)

Per il P-Code VB classico, include le operazioni del compilatore e del debugger che altrimenti verrebbero omesse dal listato ad alto livello. Attivalo quando si ispeziona la sequenza di istruzioni sottostante anziché solo la logica del programma ricostruita.

Elabora blocchi If ... End If (Codice Nativo) (Process If ... End If blocks (Native Code))

Ricostruisce i salti condizionali riconosciuti del Native Code come blocchi `If...End If`. Disattivalo quando è necessario ispezionare il pattern di branching originale o investigare una procedura la cui struttura appare errata.

Elabora blocchi 'Else' (Process 'Else' blocks)

Utilizza euristiche aggiuntive per distinguere i rami Else dalle condizioni annidate e dai salti GoTo. Per il Native Code, ciò richiede l'elaborazione dei blocchi If. Mantienilo abilitato per la lettura normale; confronta i risultati con questa opzione disabilitata quando un ramo ricostruito è ambiguo.

Analizza prototipi (Analyze prototypes)

Esegue un'analisi euristica preliminare degli argomenti e dei valori di ritorno delle procedure. Questo è particolarmente utile con la Decompilazione rapida: le funzioni e le procedure private nei moduli standard possono mancare di metadati della firma espliciti. L'analisi preliminare richiede tempo aggiuntivo, ma evita di trattare una chiamata esclusivamente in base ai contenuti dello stack attualmente noti. Una firma euristica è comunque un'inferenza, specialmente quando i metadati, i tipi esterni o il codice sono incompleti.

Aggiungi Indirizzi (Add Addresses)

Visualizza gli indirizzi virtuali accanto alle istruzioni ricostruite. Mantienilo abilitato per correlare il codice con il disassemblaggio, i byte del file e le risultati di altri strumenti. È anche necessario per **Copia nel disassemblatore (Copy to disassembler)** per fondere le istruzioni ad alto livello con le loro controparti a basso livello. Lo Disassemblatore (Disassembler) visualizza sempre gli indirizzi indipendentemente da questa impostazione.

Comprensione della ricostruzione del Native Code

Il motore Native Code riconosce i pattern `Select Case` supportati, l'accesso ad array multidimensionali, le chiamate COM e le operazioni runtime VB. La gestione degli errori può essere ricostruita come `On Error Resume Next`, `On Error GoTo` e `On Error GoTo 0`. L'ottimizzazione del compilatore e le informazioni sui tipi mancanti possono comunque limitare il risultato; ispeziona il disassemblaggio quando l'output ad alto livello è incompleto.

La versione 26.2 ha espanso la tipizzazione Variant, la lettura e scrittura di SafeArray multidimensionali e la ricostruzione `VarPtr` per gli elementi dell'array. L'output può identificare un riferimento a un array come `SafeArray`, riflettendo la struttura runtime riconosciuta dal motore di analisi. L'isolamento dello stato del ramo e l'analisi delle variabili temporanee migliorano anche la ricostruzione di codice If/Else complesso. La versione 26.5 migliora ulteriormente la gestione di Select Case e le funzioni runtime VB5/VB6.

Informazioni sui tipi COM e ActiveX

VB Decompiler utilizza prima la sua base di conoscenza interna per le interfacce note. Per un oggetto COM esterno sconosciuto, può risolvere gli identificatori di libreria, classe e interfaccia tramite il registro di sistema e analizzare la corrispondente libreria di tipi DLL/OCX. Questo fornisce firme di metodo e proprietà, inclusi i tipi di interfaccia restituiti, per chiamate early-bound e late-bound supportate.

Se una libreria di tipi richiesta è assente o non può essere risolta tramite la registrazione, alcune chiamate possono rimanere non identificate, ad esempio come `UnkVCall`. Le dipendenze mancanti sono quindi rilevanti quando si confronta l'output tra macchine di analisi. La base di conoscenza IID espansa include automazione Office, ADO/DAO, XML e HTTP, scripting, shell, messaggistica e interfacce di servizio directory; non implica che ogni possibile chiamata COM possa essere risolta.

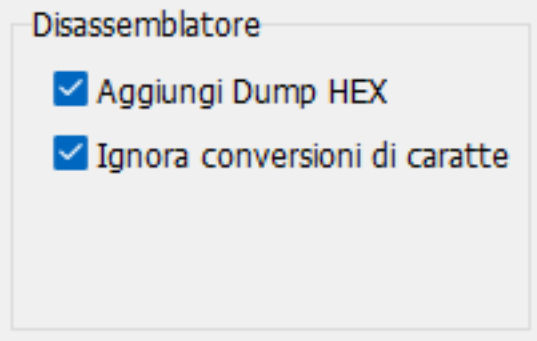
In alcuni file binari VB rimane solo una quantità limitata di metadati UDT. Il decompilatore può recuperare informazioni come la dimensione di una struttura e il conteggio dei campi stringa senza recuperare la disposizione originale completa dei campi. Non interpretare una dichiarazione parziale come una definizione a livello di sorgente completa.

Ricostruzione e ottimizzazione .NET

L'IL supportato è ricostruito come espressioni simili a C# e flusso di controllo. Il controllo **Analizza e ottimizza le procedure (Procedure analyzer and optimizer)** nella finestra principale può semplificare pattern supportati di loop, condizionali, operatori e variabili temporanee. Ispeziona l'IL e i metadati dove rimane un frammento non supportato o offuscato. Questa elaborazione è indipendente da Aiuto IA (AI Helper).

8.4 Disassemblatore

Questo blocco di opzioni è responsabile della disassemblaggio del codice macchina:



Aggiungi Dump HEX (Add HEX dump) visualizza i codici macchina prima di ogni comando disassemblato, il che può essere necessario solo quando si modifica il file binario dopo l'analisi in un Editor HEX.

Ignora conversioni di caratteri (Ignore chars convertions) viene utilizzato per disabilitare la conversione dei dati delle stringhe non inglesi in codici numerici Unicode. Abilitando questa opzione, le stringhe visualizzate diventano più facili da leggere. Disabilitala quando è necessaria la rappresentazione Unicode numerica.

Mostra classi utente (Show user classes) include l'analisi delle classi utente all'interno delle applicazioni **Delphi** / **Free Pascal**. Questa opzione non viene utilizzata per le applicazioni **Visual Basic** e **C#**.

Mostra classi VCL (Show VCL classes) include la decompilazione di tutte le classi interne della libreria **VCL**. Si consiglia di disabilitare questa opzione in modo che il disassemblatore non analizzi migliaia di classi di servizio, che sono solitamente non necessarie quando si analizzano i programmi. Questa opzione viene utilizzata esclusivamente per la decompilazione delle applicazioni **Delphi** a scopo interno ed è stata creata per l'unificazione con altri prodotti, ma non è implementata in **VB Decompiler**.

La decompilazione parziale delle applicazioni **Delphi** e **Free Pascal** è attualmente offerta esclusivamente come servizio separato (non come software).

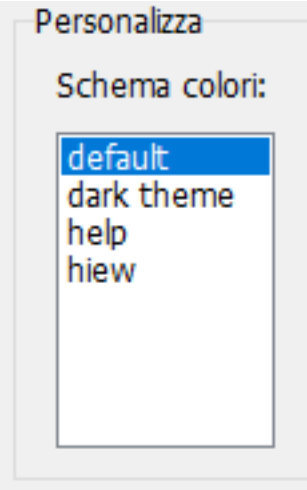
P-Code VB e .NET IL

Aggiungi Dump HEX supporta anche il listato P-Code a basso livello. Usare **Copia nel disassemblatore (Copy to disassembler)** dalla vista Decompilatore (Decompiler) per confrontare le istruzioni recuperate con il P-Code o le istruzioni native nello stesso listato.

Per gli assembly .NET, il Disassemblatore può mostrare i metodi IL o nativi x86/x64 a seconda del metodo selezionato. Le firme delle variabili locali recuperate possono includere array, generici, tipi di valore e riferimenti TypeSpec. Abilitare **Mostra struttura .NET (Show .NET structures)** quando è necessario ispezionare le tabelle dei metadati e gli heap insieme al codice. Le versioni successive espandono il supporto per i metadati offuscati, inclusa la tabella FieldPtr; ciò non garantisce il recupero riuscito di ogni assembly danneggiato o offuscato.

8.5 Schema colori

Questa sezione definisce lo schema dei colori e l'evidenziazione della sintassi quando si visualizza il codice in un decompilatore:



I preset sono memorizzati nella cartella "colors" all'interno dei file .ini situati nella directory in cui è stato installato VB Decompiler. I nomi dei file determinano i nomi dei preset nella sezione impostazioni del programma. Questi file hanno un formato semplice e puoi crearli facilmente da solo o modificare quelli esistenti.

Il formato di configurazione è il seguente:

```
[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)
```

Il codice colore è specificato nel formato true-color a 32-bit. Decimale nella sezione Colors (per la compatibilità con le versioni precedenti di VB Decompiler) e esadecimale nella sezione Colors2.

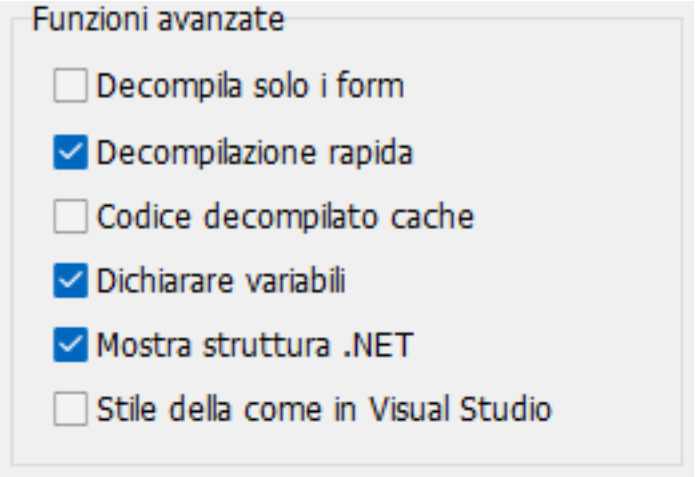
Cambiamento dei temi durante l'analisi del codice

Il sistema dei temi supporta il tema scuro e può applicare un tema selezionato senza riavviare VB Decompiler, incluso dopo aver modificato il suo file INI. Le impostazioni del font nella configurazione estesa possono contenere nomi di fallback separati da virgola, il che aiuta quando i font equivalenti differiscono tra Windows e ambienti di compatibilità.

I colori sotto il cursore controllano l'evidenziazione degli identificatori. Vedi Visualizzazione del codice per l'evidenziazione delle variabili locali, dei membri e globali.

8.6 Funzionalità avanzate

Queste impostazioni controllano come viene elaborato il progetto e come vengono presentate le informazioni recuperate.



Decompila solo i form (Decompile only forms)

Disabilita la decompilazione del codice quando sono necessarie solo le definizioni dei form e degli UserControl recuperati. Per il visualizzatore e le sue capacità specifiche della tecnologia, vedi Ricostruzione visiva dei form.

Decompilazione rapida (Fast decompilation)

Quando abilitata, VB Decompiler elabora prima la struttura del progetto e decompila le singole procedure man mano che vengono aperte. Questo riduce il tempo di attesa iniziale durante l'indagine sulle procedure selezionate. Con **Codice decompilato cache (Cache decompiled code)** disabilitato, è possibile modificare le impostazioni di analisi e riaprire una procedura per ottenere un nuovo listato.

Abilita **Analizza prototipi (Analyze prototypes)** nelle Impostazioni di decompilazione per eseguire un'analisi euristica preliminare delle firme delle procedure private. Questo aggiunge tempo di elaborazione iniziale ma può migliorare la ricostruzione degli argomenti senza richiedere una decompilazione completa.

La decompilazione rapida non fornisce un listato ad alto livello completo e ricercabile del progetto. Disabilitala ed elabora nuovamente il file prima di utilizzare la ricerca di testo su tutto il progetto o l'esportazione di un progetto decompilato completo. I Riferimenti a stringhe sono diversi: possono essere raccolti tramite disassemblaggio preliminare e sono disponibili in modalità Decompilazione rapida.

Decompilazione completa

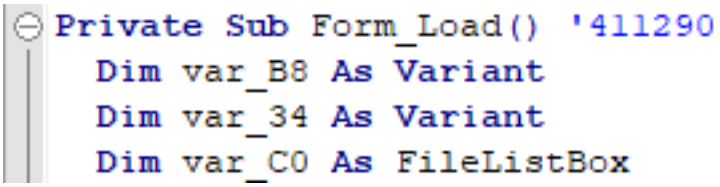
Con la decompilazione rapida disabilitata, il decompilatore elabora le procedure del progetto in anticipo. Questo richiede più tempo, ma rende disponibile il listato completo generato per la ricerca di testo e l'esportazione. Le modifiche alle impostazioni di decompilazione richiedono di elaborare nuovamente il file.

Codice decompilato cache

Mantiene i listati già generati in modalità Decompilazione rapida. Questo evita di decompilare ripetutamente una procedura aperta, ma i risultati memorizzati non riflettono le successive modifiche alle impostazioni di analisi. Quando si confrontano le impostazioni **Analizza parametri stack (Parse stack parameters)** o **Analizza e ottimizza le procedure (Procedure analyzer and optimizer)** in modo interattivo, disabilita la memorizzazione.

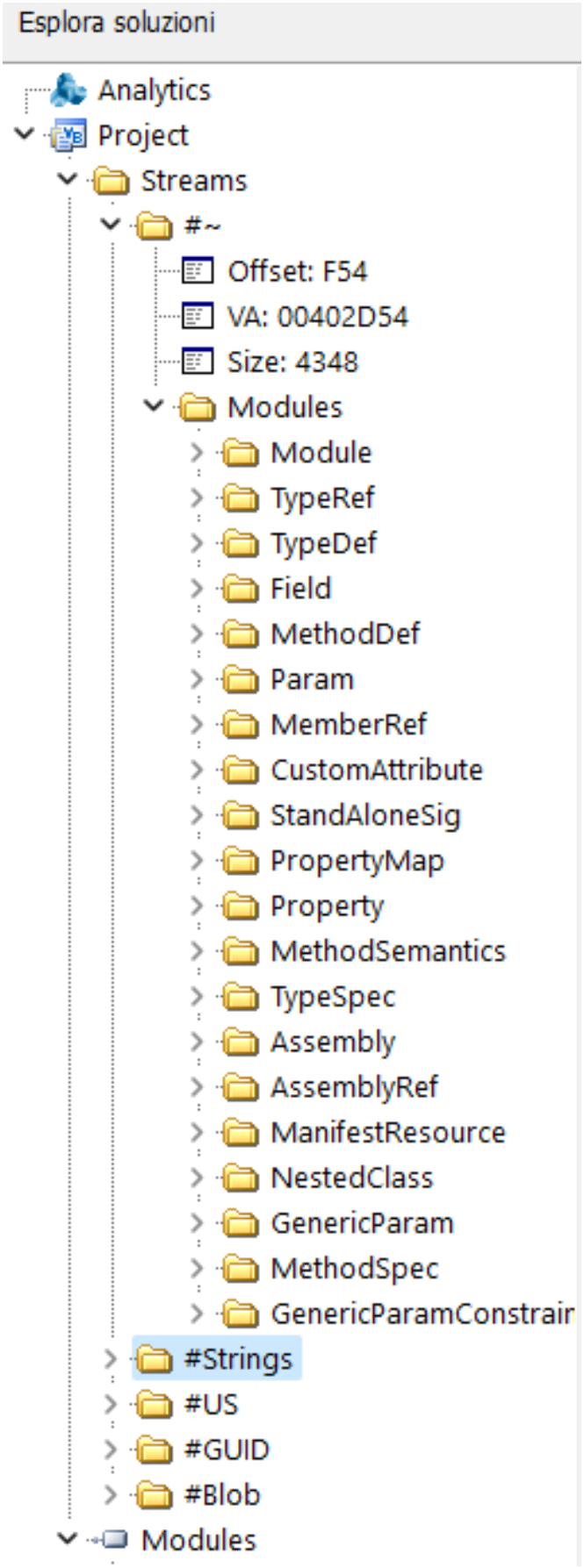
Dichiarare variabili (Declare variables)

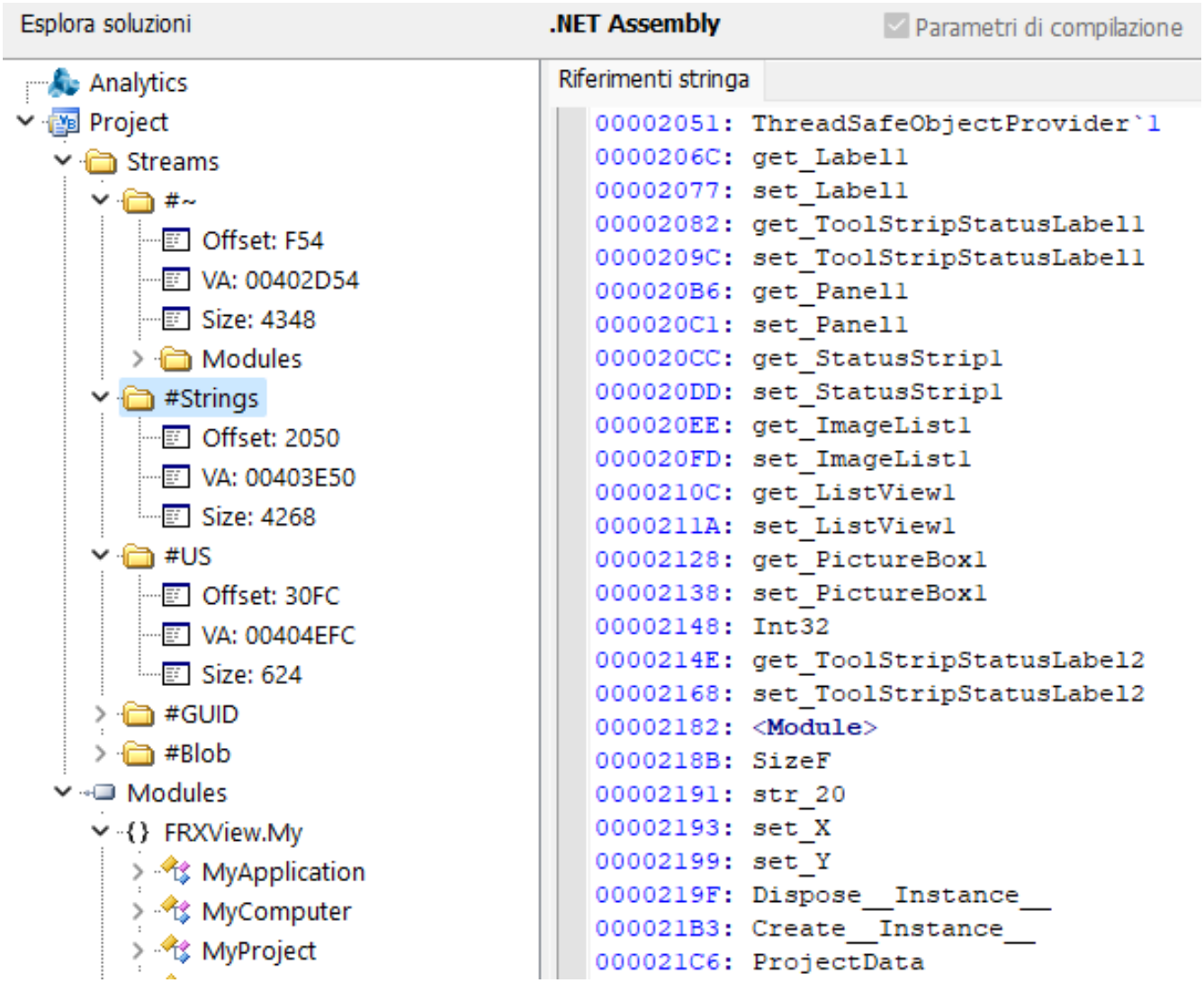
Aggiunge un blocco di dichiarazione per le variabili recuperate all'inizio di ciascuna procedura decompilata. Questo è utile per il recupero del codice sorgente e l'ispezione dei tipi. Per una rapida lettura della logica della procedura, è possibile nascondere le dichiarazioni. I tipi recuperati sono soggetti ai metadati e all'analisi disponibili al decompilatore.



Mostra struttura .NET (Show .NET structures)

Rivela le tabelle e gli heap dei metadati .NET in Esplora soluzioni (Solution explorer). Utilizzalo per ispezionare strutture e firme insieme al codice. Selezionando una tabella vengono visualizzati i suoi contenuti; l'heap #US contiene le stringhe utente. Disabilita l'opzione quando non è necessaria la navigazione dei metadati.


Esplora soluzioni
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
GenericParamConstrair
#Strings
#US
#GUID
#Blob
Modules



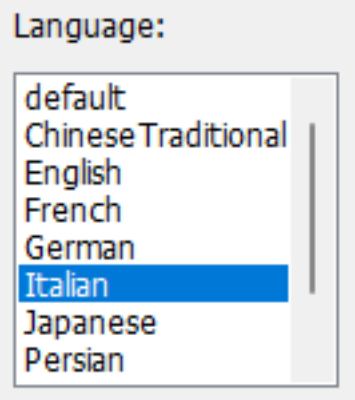
Stile della come in Visual Studio (Window style as in Visual Studio)

Posiziona il codice a sinistra e l'Esplora soluzioni a destra. Questo modifica la disposizione dello workspace senza modificare il codice recuperato.

8.7 Selezione della lingua

VB Decompiler fornisce localizzazioni dell'interfaccia per oltre 20 lingue utilizzate in tutto il mondo. La documentazione è disponibile anche in più lingue.

È possibile scegliere l'interfaccia nella lingua preferita per il programma nel menu **Strumenti (Tools)** -> **Opzioni (Options)**, sotto le impostazioni **Personalizza (Personalize)**. Dopo aver scelto la lingua, è necessario riavviare **VB Decompiler** per applicare correttamente le impostazioni della lingua in tutto il programma. Se il riavvio del programma non è desiderabile, è possibile continuare a utilizzare **VB Decompiler**; tuttavia, alcuni testi potrebbero essere ancora visualizzati nella lingua precedente.



La lingua dell'assistenza viene selezionata automaticamente in base alla lingua di sistema del computer. Per leggere il manuale in un'altra lingua, aprire il file di assistenza corrispondente dalla cartella in cui è installato **VB Decompiler**. Il PDF è il formato principale per i manuali localizzati; i file di assistenza CHM rimangono disponibili per alcune lingue.

Il manuale inglese è la fonte per tutte le traduzioni. Se si nota un errore o un'incoerenza in un manuale localizzato, si prega di segnalarlo al supporto tecnico.

Accesso al manuale

Se il file di assistenza locale o il visualizzatore PDF richiesto non sono disponibili, VB Decompiler può aprire la documentazione online. Per un ambiente di indagine offline, tenere i file di assistenza locali richiesti disponibili in anticipo.

9 Plugin

I plugin estendono VB Decompiler tramite DLL native. I plugin installati sono disponibili dal menu **Aggiunte (Plugins)**. L'API callback fornisce accesso alle informazioni e al codice del progetto recuperato, consentendo a un plugin di ispezionare o modificare le strutture interne supportate.

FRX Image Viewer

FRX Image Viewer 1.2.0 visualizza ed estrae le immagini registrate da VB Decompiler o DE Decompilatore (Decompiler). Usare VB Decompiler per la grafica dei form VB5/6 e .NET e DE Decompilatore per quella dei DFM binari Delphi. Sono supportati BMP, PNG, JPEG, GIF, ICO e CUR.

Installazione e apertura

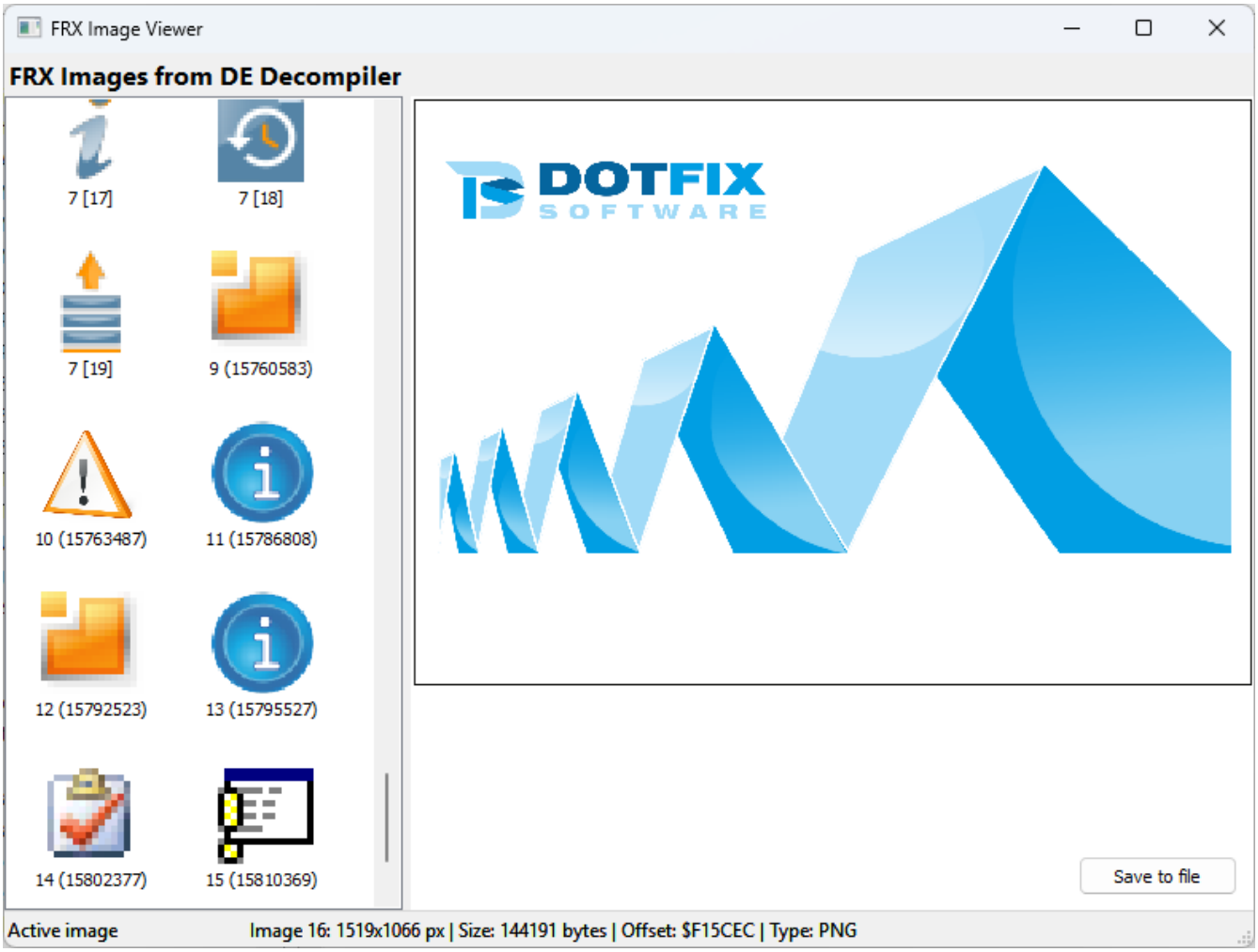
1. Copiare `FRXViewer.dll` nella sottocartella `plugins` dell'installazione host e riavviare il decompilatore. L'architettura della DLL deve corrispondere all'applicazione host, indipendentemente dall'architettura del file analizzato.
2. Aprire il file EXE o DLL e decodificarne i form, affinché l'host registri le immagini disponibili.
3. Selezionare **Aggiunte** > **FRX Image Viewer**, quindi scegliere una miniatura.

Il download della versione 1.2.0 contiene una DLL per host x86. Per un host x64 occorre una build corrispondente. Il plugin richiede Windows 10/11; le istruzioni di compilazione sono nel repository del codice sorgente.

Anteprima ed esportazione

La galleria e l'anteprima mostrano le immagini disponibili, inclusi i file ICO/CUR composti. La barra di stato indica elemento, dimensioni, quantità di dati o offset della bitmap sorgente e formato. Il visualizzatore segue il tema dell'host e supporta il tema scuro.

Scegliere **Save to file** per esportare l'elemento selezionato. Le immagini indipendenti mantengono byte e formato originali. Le singole celle ImageList vengono esportate in ICO con maschere e trasparenza alfa; questi file ICO sono generati dalle bitmap sorgenti.



FRX Image Viewer in DE Decompilatore: grafica DFM Delphi e singole icone ImageList.

ImageList native Delphi e compatibilità

Con un host compatibile, le ImageList native Delphi riconosciute compaiono come icone individuali. Le immagini codificate separatamente negli elementi delle collection restano normali immagini indipendenti. Per i flussi non supportati rimane la visualizzazione dell'intera immagine, ove siano disponibili i dati originali registrati.

La revisione condivisa SDK 26.6, supportata da VB Decompiler 26.6 e DE Decompilatore 26.5, aggiunge i comandi facoltativi `GetFrxImageListInfo` (63) e `GetProductName` (64). Gli host precedenti mantengono l'API originale e la visualizzazione dell'intera immagine. Per il rilevamento delle funzionalità e le regole sugli offset, vedere il riferimento SDK.

Visualizzazione ed estrazione non eseguono il file analizzato né ne caricano i componenti. Per le GIF animate viene mostrato il primo fotogramma. Le risorse disponibili dipendono dalle immagini decodificate e registrate correttamente dall'host.

Sviluppo di un plugin

L'SDK include esempi per Visual C++, Rust, Delphi e Free Pascal. Consulta il riferimento VB Decompiler Plugin API per i punti di ingresso esportati, la firma del callback, gli identificatori dei comandi, le rappresentazioni dei dati e i link ai progetti di esempio.

Un plugin DLL fornisce le informazioni sul nome e sull'autore e riceve il callback dell'host tramite `VBDecompilerPluginLoad`. Il callback viene utilizzato per richiedere o aggiornare i dati del progetto supportati. Segui la convenzione di chiamata, i requisiti su puntatori, codifica delle stringhe e buffer dell'SDK quando implementi questa interfaccia; quando utilizzi un linguaggio gestito, assicurati che il livello di interoperabilità rispetti l'interfaccia nativa.

VB Decompiler inizializza i plugin DLL disponibili all'avvio. Costruisci i plugin per l'architettura dell'host e l'ambiente Windows previsti e verifica la DLL risultante con tale ambiente. L'SDK espone sia il codice della vista attiva che l'accesso al codice a livello di progetto; le operazioni che richiedono tutte le procedure decompilate necessitano di una decompilazione completa piuttosto che solo un elenco on-demand.

10 Ricostruzione visiva dei form

La scheda **Designer (sola lettura) (Designer (read only))** visualizza una ricostruzione visiva delle interfacce supportate estratte dal binario analizzato. Aiuta a collegare gli elementi dell'interfaccia al comportamento dell'applicazione senza dover interpretare manualmente ogni proprietà recuperata o dichiarazione di inizializzazione.

Apertura di un form recuperato

1. Apri l'applicazione e lascia che VB Decompiler recuperi la sua struttura del progetto.
2. Seleziona un form recuperato in Esplora soluzioni (Solution explorer) ed apri **Designer (sola lettura)**.
3. Seleziona un controllo per ispezionarne le proprietà disponibili e le informazioni sugli eventi recuperati. Utilizza gli eventi per navigare verso il codice del gestore corrispondente.

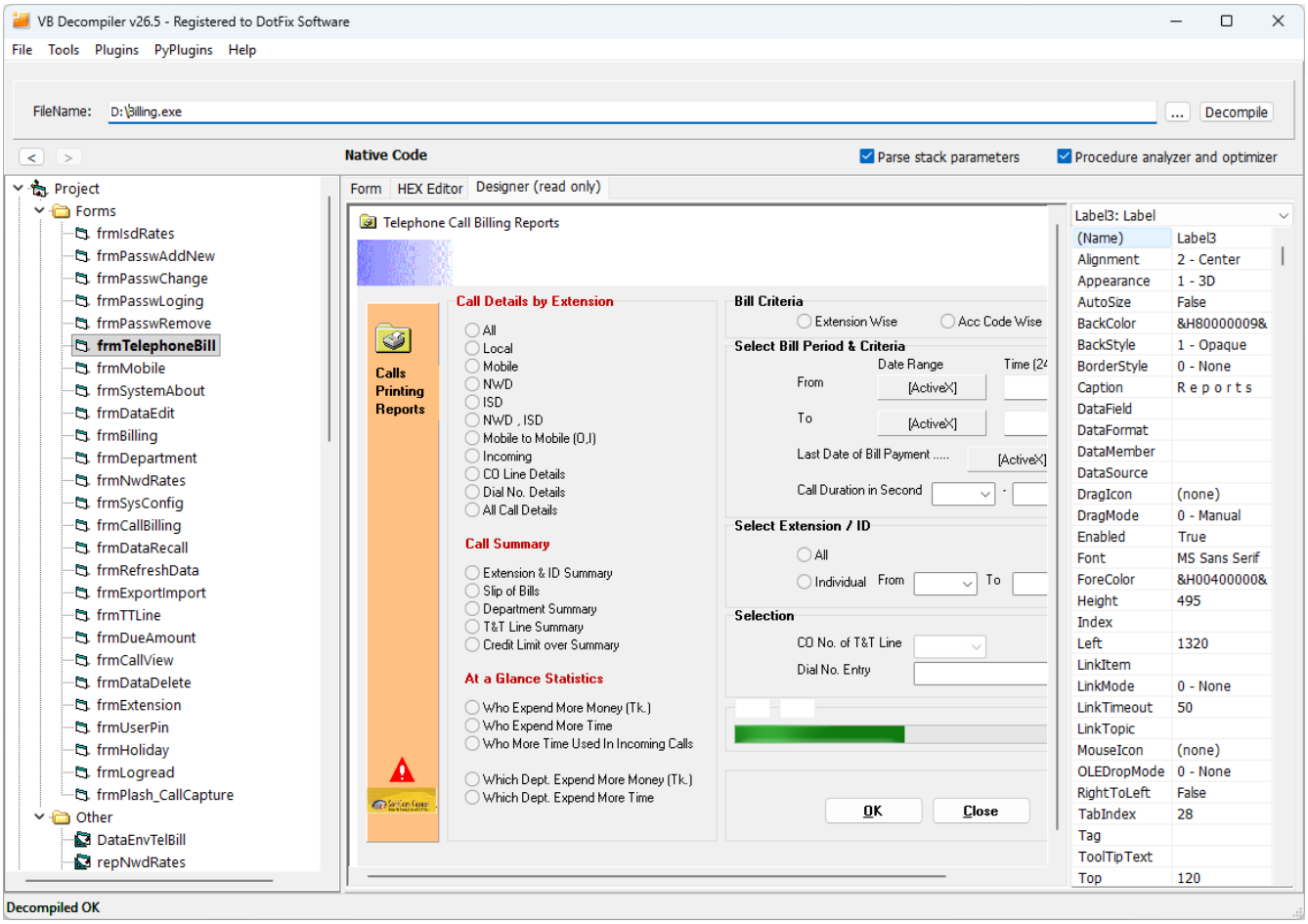
La anteprima viene ricostruita da informazioni memorizzate. Non si tratta di un'istanza in esecuzione dell'interfaccia originale. Controlli non supportati, contenuti creati in runtime, risorse mancanti e logiche di layout personalizzate possono produrre risultati incompleti o approssimativi.

Visual Basic 5.0/6.0

Il visualizzatore ricostruisce i form e gli UserControl, incluse le proprietà dei controlli supportate, i font, i colori e le immagini. Legge i dati binari dei form utilizzati da elenchi, caselle di combinazione e altri controlli. Le immagini possono includere icone, cursori, immagini bitmap, JPEG, PNG e GIF.

La ricostruzione del menu visivo supporta sia le applicazioni P-Code che Native Code. I controlli supportati includono `SSTab`, `MSComctlLib.StatusBar`, `MSComctlLib.ListView`, `MSComctlLib.ProgressBar` e `MSComctlLib.TreeView`.

Per ispezionare il gestore di un controllo VB classico, seleziona il controllo e fai doppio clic sul suo campo **_Event** nell'elenco delle proprietà. Scegli l'evento richiesto dall'elenco risultante per aprire il gestore.

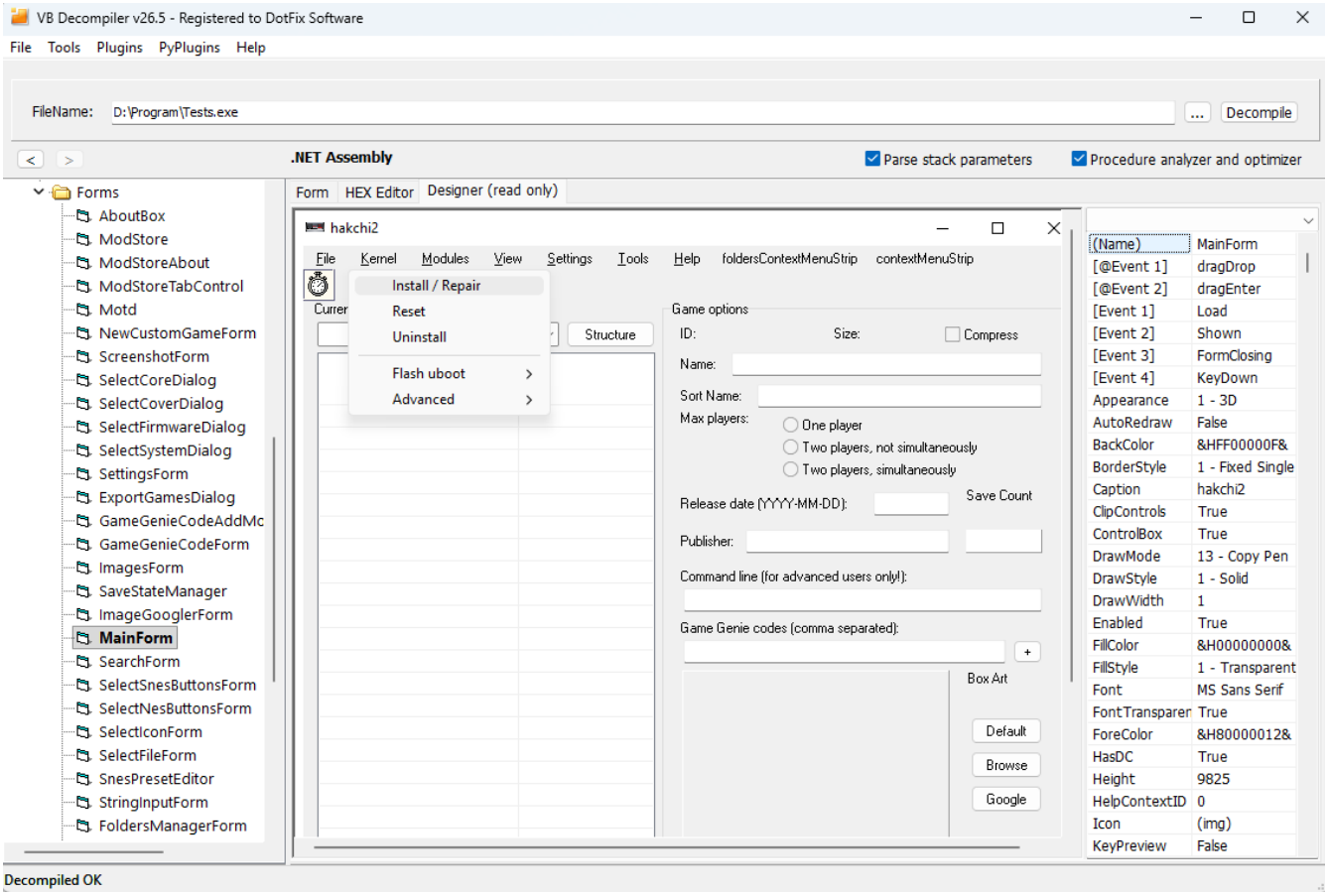


.NET Windows Forms

VB Decompiler analizza staticamente `InitializeComponent` e ricostruisce i controlli Windows Forms supportati e il loro layout. Il parser e il visualizzatore non richiedono il runtime .NET o l'esecuzione dell'applicazione target. Le immagini recuperate supportate possono essere visualizzate anche sui controlli e sullo sfondo dei form.

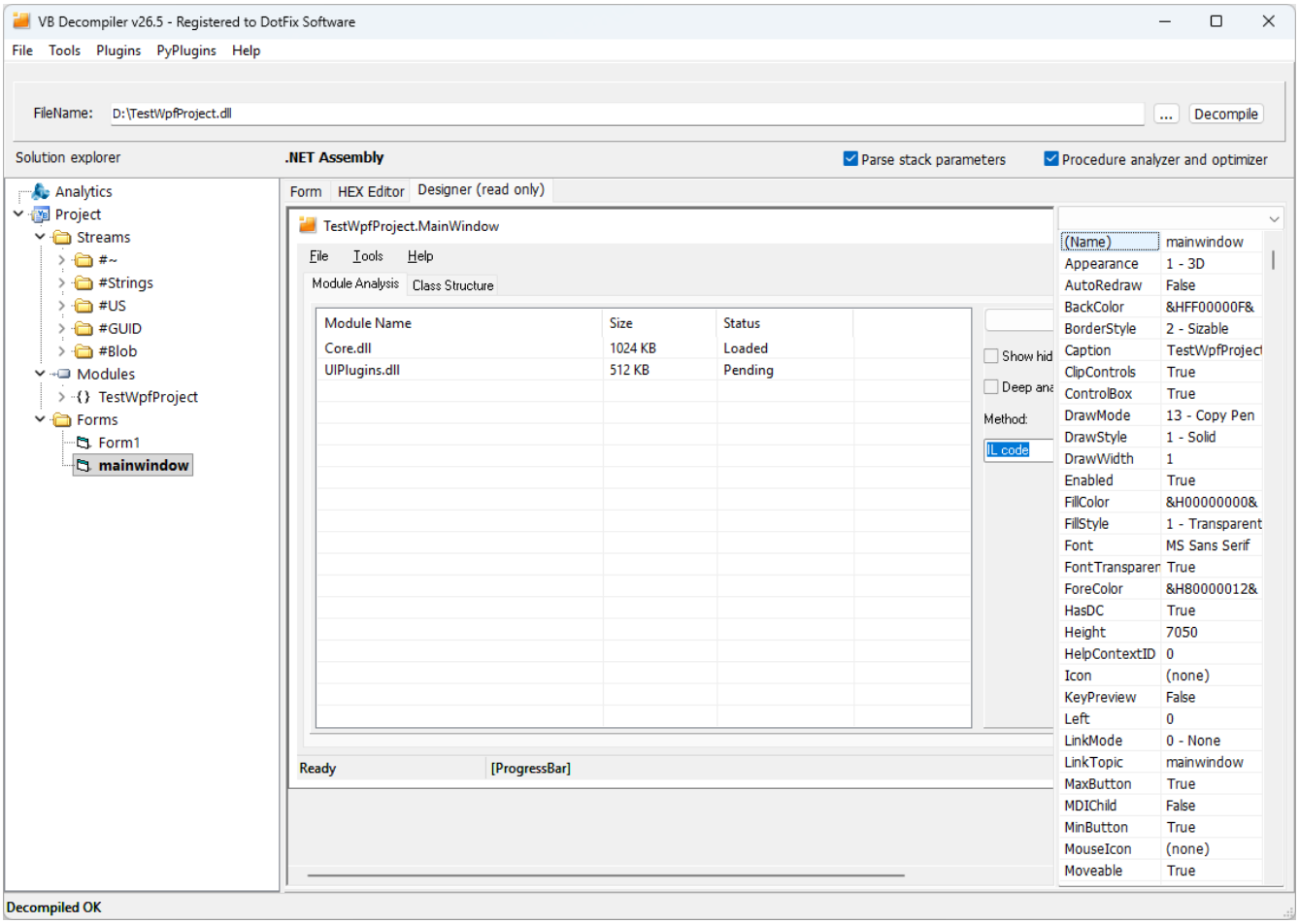
Seleziona un controllo per ispezionare gli eventi recuperati, quindi fai doppio clic su un evento per aprire il gestore corrispondente. Il visualizzatore ricostruisce le gerarchie dei menu e dei menu contestuali; facendo clic su un elemento di menu recuperato si apre l'evento Click corrispondente. Gli eventi Windows Forms personalizzati sono mostrati con il prefisso **@**.

Il visualizzatore gestisce l'ordine Z dei controlli, i colori di primo piano e sfondo ereditati, valori `Color.FromArgb`, flag numerici `FontStyle`, proprietà delle risorse Booleane e `MinimumSize` insieme a `ClientSize`.



Finestre, pagine e controlli utente WPF

Il visualizzatore ricostruisce finestre WPF supportate, pagine e controlli utente dalle risorse XAML compilate memorizzate come **BAML**.



I contenitori di layout supportati includono **Grid** con dimensioni di riga e colonna fisse, Auto e proporzionali (*), **StackPanel**, **DockPanel** e **Canvas**, inclusi layout annidati. La ricostruzione tiene conto delle dimensioni supportate, margini, padding, allineamento, orientamento, aggancio e estensioni sulle righe e sulle colonne di Grid.

Oltre ai controlli di input comuni, il visualizzatore può ricostruire schede, caselle di gruppo, espanditori, gerarchie dei menu e separatori, barre di stato, colonne ListView e dati statici supportati, nodi TreeView statici ed elementi di casella di combinazione statici. Le interfacce Ribbon, i pannelli di navigazione e i pannelli agganciati possono avere rappresentazioni semplificate.

Dove sono disponibili i dati delle risorse, il visualizzatore recupera nomi, testo visibile, tooltip, font, colori di sfondo, stati di abilitazione e visibilità e immagini incorporate, inclusa la trasparenza PNG supportata e l'opacità dello sfondo. I controlli personalizzati non supportati possono apparire come segnaposto che identificano il tipo WPF originale.

Il visualizzatore WPF collega gli eventi recuperati dei controlli ai rispettivi gestori. Utilizzare le informazioni sugli eventi nel visualizzatore per aprire il codice decompilato corrispondente.

Limiti della ricostruzione: l'anteprima visiva è approssimativa. Template, stili, associazioni dati, controlli personalizzati e contenuti generati dinamicamente possono essere rappresentati parzialmente.

Ispezione ed estrazione dei dati dei form

Vedi Form per l'impostazione Mostra offset (Show offsets) e l'ispezione del binario collegata. Utilizza il plugin FRX Image Viewer per ispezionare ed esportare le risorse immagine recuperate supportate.

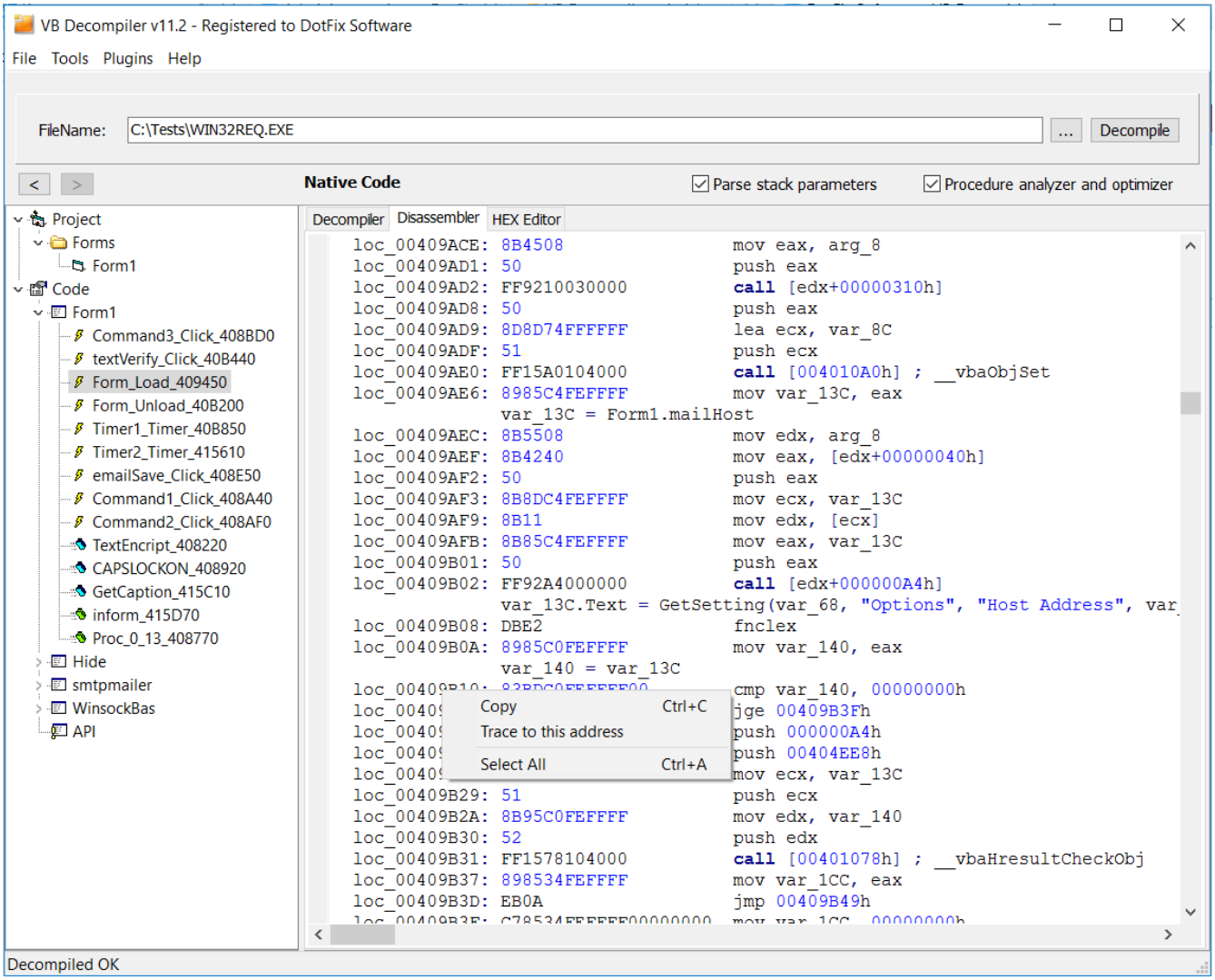
11 Tracciamento

Tracing features consente l'emulazione passo passo del Native Code e del P-Code di Visual Basic 5.0/6.0. È necessaria una licenza con funzionalità di tracciamento. Il tracciatore opera sulla rappresentazione interna della procedura nell'emulatore; non esegue la procedura sulla CPU della workstation.

Inizio di un tracciamento

1. Apri una procedura VB5/6 e lascia attiva la visualizzazione degli indirizzi nel listato.
2. Sulla riga da investigare, invoca **Traccia a questo indirizzo (Trace to this address)** dal menu contestuale del codice, oppure premi **F4**.
3. Ispeziona le viste del codice e dello stato del tracciatore e procedi usando **Prossima (Next)**. Annulla il tracciamento quando è completato.

Dalla versione 26.4, il comando può risolvere l'indirizzo iniziale `loc_` dopo aver rimosso gli spazi bianchi all'inizio della riga. Puoi posizionare il cursore altrove su quella riga invece che precisamente sulla sua etichetta di indirizzo.



Stato del codice nativo

Il tracciatore espone registri CPU e FPU, la pila, le variabili e il codice ricostruito mentre l'emulazione avanza. Usa queste viste per esaminare come un valore o un'espressione è stato derivato. La ricerca nella lista delle variabili e il display del tipo recuperato dell'oggetto corrente possono aiutare con procedure più grandi.

Tracciamento P-Code

Il tracciamento P-Code, introdotto nella versione 11, mostra come cambiano le informazioni sulle variabili e si formano le istruzioni decompilate durante l'elaborazione. Può aiutare a spiegare le differenze tra le operazioni intermedie P-Code e il listato ad alto livello ottimizzato.

Ambito e limitazioni

- Il tracciamento riguarda una singola procedura. Parte da uno stato vuoto dei registri emulati, dello stack e delle variabili, anziché dallo stato di un processo in esecuzione.
- Istruzioni del codice nativo non supportate possono essere saltate. Le chiamate a funzioni esterne non vengono eseguite; la gestione delle convenzioni di chiamata supportate può regolare la pila modellata senza riprodurre il comportamento della funzione esterna.
- Il tracciatore non fornisce una visualizzazione completa dei flag del processore. Non interpretare il suo stato come una traccia completa dell'esecuzione hardware.
- Il modello documentato è un'emulazione passo-passo in avanti, senza tracciamento inverso o esecuzione ripetuta di iterazioni di loop.
- Questo flusso di lavoro descrive il tracciamento VB5/6. La presenza del disassemblaggio .NET non implica un debugger del runtime .NET.

Usa il tracciamento per investigare la ricostruzione del decompilatore e i valori intermedi. Un valore emulato da solo non stabilisce il valore prodotto da un processo effettivo con globali inizializzati, dipendenze esterne e dati di input.

12 Funzionalità di analisi

Le licenze con **Analytic features** generano rapporti categorizzati delle operazioni riconosciute nel programma analizzato. Questi rapporti aiutano gli analisti della sicurezza a identificare le procedure da investigare prima di leggere l'intero codice sorgente. Supportano le applicazioni Visual Basic 5.0/6.0 e, dalla versione 11.7, gli assembly .NET.

Generazione di un rapporto

1 Abilita **Analizza prototipi (Analyze prototypes)** in **Strumenti > Opzioni (Tools > Options)**.

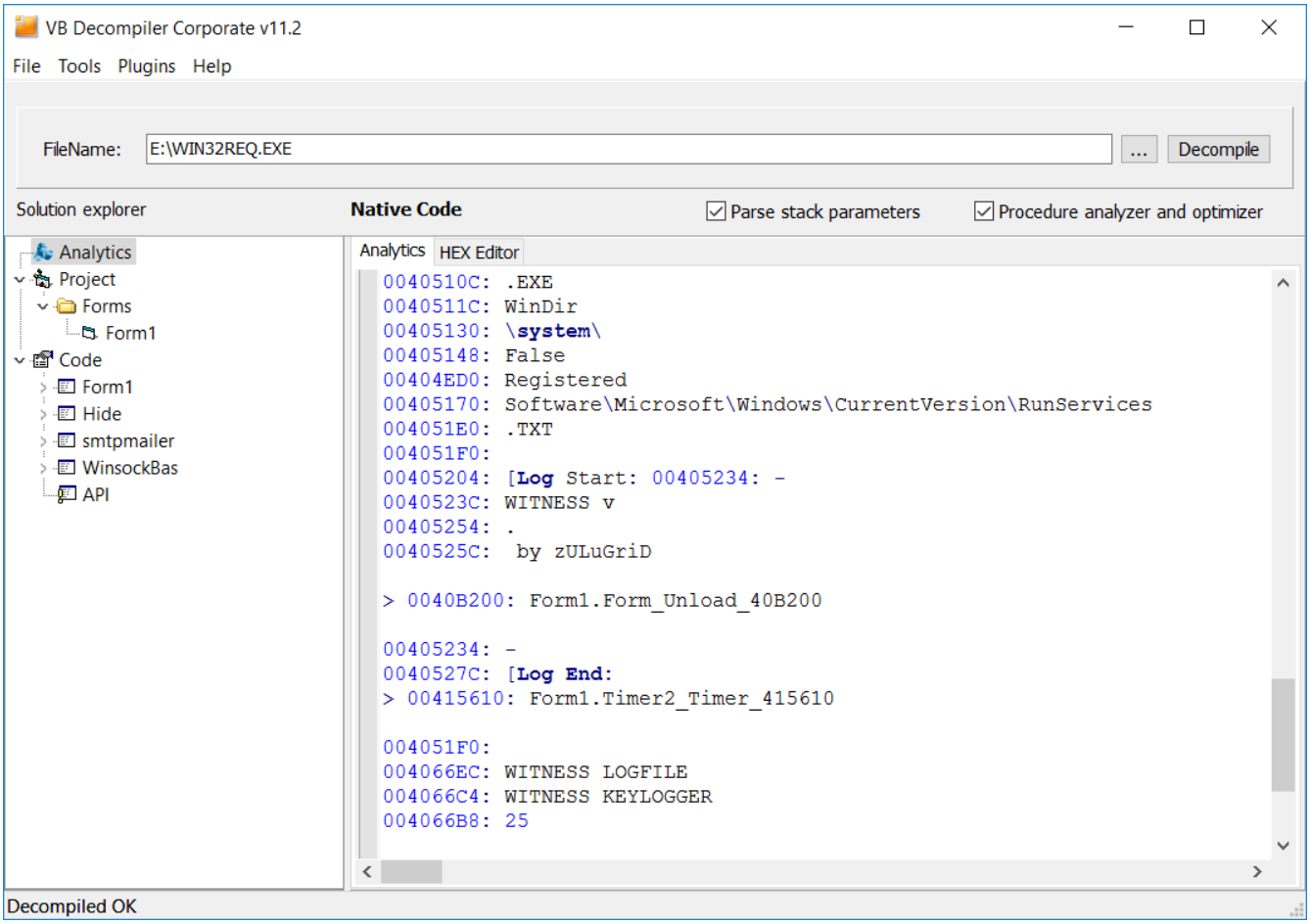
2 Apri il file e attendi il completamento dell'analisi. Con la licenza e le impostazioni richieste, il rapporto viene generato e visualizzato automaticamente.

3 Rivedi le categorie, quindi apri le procedure pertinenti ed ispeziona il loro codice e i riferimenti alle stringhe.

Le categorie coprono operazioni riconosciute come accesso a file e volumi, accesso al registro di sistema, attività di rete, creazione di processi, operazioni sulle finestre e funzioni interne VB selezionate come `CallByName` e `VarPtr`.

Rapporto rapido e contesto delle stringhe

Il rapporto rapido elenca gli indirizzi e i nomi delle procedure contenenti operazioni riconosciute. Il rapporto esteso aggiunge i riferimenti alle stringhe disponibili da quelle procedure. Questo contesto può rivelare percorsi candidati, chiavi del registro di sistema, nomi host o altri valori che meritano esame.



Le stringhe crittografate, oscurate o generate in tempo di esecuzione possono essere assenti. Una stringa associata a una procedura è un contesto utile, ma non dimostra da sola che venga passata a una specifica chiamata API o che un ramo venga eseguito.

Navigazione e salvataggio dei risultati

Fai doppio clic su un nome di procedura nel rapporto per aprire il suo codice. Fai doppio clic su un indirizzo precedente a una procedura o a una stringa per ispezionarne la posizione nell'Editor HEX. Usa il pulsante **A** sopra l'albero degli oggetti per tornare al rapporto alla posizione precedente, e le frecce di navigazione o **Esc** per rivedere le procedure ispezionate.

Seleziona e copia il testo del rapporto come necessario, oppure scegli **File > Salva rapporto analytics (File > Save analytics report)** per salvarlo per un'analisi successiva.

Interpretazione delle risultati

Il rapporto identifica modelli di codice e riferimenti riconosciuti staticamente. È uno strumento di triage, non una registrazione dell'esecuzione osservata o un verdetto automatico che un file sia dannoso. Le chiamate indirette, le interfacce esterne non risolte, i metadati danneggiati, l'offuscamento e i valori generati in tempo di esecuzione possono limitare la copertura. Conferma le risultati significative nelle procedure corrispondenti, nel disassemblaggio e nei dati del file.

Per approfondire la ricostruzione del codice VB5/6, vedi Tracciamento. Per le stringhe letterali e le loro posizioni nel codice, vedi Riferimenti alle stringhe.

13 Limitazioni dell’offuscamento

VB Decompiler, dotato di una tecnologia di offuscamento unica, nasconde efficacemente i dati lasciati dal compilatore nella nostra ricerca sulla decompilazione di **applicazioni Visual Basic 5.0 e 6.0**. I moderni linguaggi di programmazione spesso memorizzano grandi quantità di metadati e informazioni di debug all'interno dei file compilati, rendendo più facile per gli hacker individuare le funzioni desiderate all'interno del tuo programma. Per contrastare questa minaccia, abbiamo sviluppato un offuscatore specificamente progettato per le applicazioni **Visual Basic**.

Il processo di offuscamento in dettaglio

L'offuscatore funziona scomponendo l'intero programma nelle sue parti costituenti, inclusi form, moduli, classi e controlli come pulsanti e campi di testo. Ogni form viene quindi analizzato per individuare i singoli controlli con gli eventi associati. Successivamente, vengono raccolte informazioni sulle funzioni e le procedure, nonché sugli eventi legati a ciascun oggetto nel programma. Durante questo processo, l'offuscatore verifica le dipendenze tra gli oggetti e modifica in modo coerente i loro nomi per rendere difficile determinare a quale modulo o classe appartiene una determinata funzione.

Di conseguenza, trovare form specifici come **frmTrial** o **frmRegistration** all'interno del programma diventa significativamente più difficile.

Limitazioni dell’offuscamento delle applicazioni Visual Basic 5.0/6.0

L'algoritmo utilizzato dall'offuscatore è complesso e progettato per analizzare le dipendenze degli oggetti, ma non supporta ancora tutto. Di conseguenza, in alcuni casi, dovrai preparare il tuo programma in anticipo per il processo di offuscamento. I punti discussi di seguito si applicano esclusivamente all'offuscamento delle applicazioni Visual Basic 5.0 / 6.0.

1. L'offuscatore non supporta i riferimenti in fase di progettazione ad altri oggetti, il che può portare a problemi durante l'esecuzione.

Ad esempio, considera uno scenario in cui hai un controllo **TreeView** che si basa su un associato **ImageList**. Se il nome di questo oggetto **ImageList** è memorizzato direttamente nelle proprietà del tuo **TreeView**, qualsiasi tentativo di offuscamento causerà errori, poiché il riferimento originale rimane intatto. Questa limitazione si applica anche ai componenti relativi al database, come **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
```

Questo problema può essere facilmente risolto spostando le assegnazioni dalla fase di progettazione alla fase di esecuzione, semplicemente spostando queste assegnazioni all'evento **Form_Load**:

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

La stessa correzione deve essere applicata per ogni istanza. Fortunatamente, nella maggior parte dei progetti ci sono solitamente solo pochi casi del genere.

2. Non sono supportate le dichiarazioni di oggetti tramite As New (chiamate LateCall)

Questa limitazione di solito non influisce sulla maggior parte dei progetti VB6, ma è essenziale tenerne conto. Se stai lavorando con form come classi e crei istanze di form, potresti essere familiare con questo scenario:

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

In tali casi, il compilatore VB6 inserisce direttamente il nome **frmMain** nel codice della funzione **Form_Load**. Poiché il nostro offuscatore modifica solo i dati e non il codice, il nome incorporato nel codice rimane invariato dopo l'offuscamento. Sfortunatamente, questo porta a un arresto del programma con errore.

Sfortunatamente, non è possibile per noi supportare tali chiamate nell'offuscatore di **VB Decompiler**, poiché dobbiamo disassemblare e decompilare l'intero codice di tutte le funzioni. Nonostante questa capacità, la decompilazione è un processo estremamente dispendioso in termini di tempo. Per i grandi progetti, può richiedere oltre mezz'ora solo per analizzare l'intera base di codice. Inoltre, dobbiamo anche analizzare i riferimenti agli oggetti e rinominarli, il che aggiunge ancora più tempo di elaborazione.

Date queste limitazioni tecniche, abbiamo deciso di non implementare la decompilazione completa e l'analisi del codice nell'offuscatore di **VB Decompiler**, specialmente poiché questo risolverebbe il problema solo in casi rari nelle applicazioni Visual Basic 5.0 / 6.0 e non fornirebbe alcun beneficio per la maggior parte dei clienti.

14 Supporto tecnico

Se hai domande sul programma e non riesci a trovare le risposte in questa guida di aiuto, contatta il nostro team di supporto tecnico. Questa opzione è disponibile solo per gli utenti registrati. Per gli utenti non registrati, siamo felici di assisterti con le richieste relative all'acquisto della licenza per **VB Decompiler**.

Inoltre, se incontri qualsiasi bug all'interno del programma o hai suggerimenti su come migliorarlo, sentiti libero di contattare il nostro servizio di supporto tecnico con il tuo feedback.

15 Registrazione

Per utilizzare le funzionalità complete del **VB Decompiler**, è necessario acquistare una **Business License**. Acquistando una licenza Business per singolo sviluppatore o una licenza Enterprise, riceverete la possibilità di:

1. Recuperare circa l'85% del codice delle applicazioni tipiche supportate in **Visual Basic 5.0/6.0 P-Code**;
2. Recuperare fino all'85% del codice delle applicazioni tipiche supportate in **Visual Basic 5.0/6.0 Native Code**;
3. Decompilare gli assembly **.NET** tipici supportati e non offuscati in codice simile a **C#**, con un recupero fino al 95%;
4. Avere sempre l'ultima versione del programma: gli aggiornamenti sono assolutamente gratuiti per un anno dopo l'acquisto della licenza;
5. Ricevere supporto tecnico via email per un anno dopo l'acquisto della licenza;
6. Sostenere lo sviluppo di questo decompilatore.

Inoltre, consigliamo vivamente di valutare l'acquisto di licenze speciali per gli Analisti Malware e la Forensics:

1. licenza con funzionalità di tracciamento
2. licenza con funzionalità analitiche

Per acquistare ora una **licenza** del decompilatore, selezionate l'opzione **Ordina versione PRO (Order Pro version)** nel menu **Aiuto (Help)** del programma. In questo momento, dovete essere connessi a Internet. Si aprirà una finestra del browser sul sito web del programma dove è possibile acquistare la licenza.

La **Business License** del decompilatore è progettata per l'uso su un solo computer e solo da un **Single Developer** (proprietario della licenza). Se avete bisogno di utilizzare il programma su più computer, dovete acquistare una licenza separata per ciascuno o valutare l'acquisto della **Business Enterprise license** (utilizzabile su un massimo di 20 computer).

16 Informazioni sullo sviluppatore

Il decompilatore è sviluppato da **DotFix Software**.

Sito del prodotto: www.vb-decompiler.org
Email del supporto tecnico: admin@vb-decompiler.org