

DPVC クイック リファレンス

このクイック リファレンスは、全ページまたは一部を印刷（できればカラー印刷）して、適宜参照できるように、手近なところに備えておいてください。

DevPartner の機能

DevPartner の機能に関する参照情報には、以下の表の右欄からリンクされています。

用途	使用する DevPartner 機能
ソース コードのランタイム エラーを診断する	エラー検出
アプリケーション内のパフォーマンス ボトルネックを特定する	カバレッジ分析とパフォーマンス分析
開発とテストのフェーズを通してコード ベース安定性を確保する	カバレッジ分析セッション データ

詳細情報

詳細については、DevPartner オンライン ヘルプまたは『DevPartner ユーザー ガイド』を参照してください。

共通要素






DevPartner のすべての機能で、以下の要素が提供されています。

- DevPartner ツールバー
- DevPartner メニュー
- DevPartner ファイル拡張子
- コマンド ライン インストゥルメンテーション オプション

DevPartner メニューおよびツールバー

DevPartner のメニューまたは Visual Studio のツールバーからアクセスします。

メモ：Visual Studio 6.0 では、オプションとアイコンが多少異なります。

メニュー項目またはツールバー ボタン	機能
 エラー検出	BoundsChecker テクノ ロジを使用した、ランタイム エラーの検出
 カバレッジ分析	ランタイム コード カバレッジの分析
 エラー検出とカバレッジ分析	ランタイム エラーの検出とコード カバレッジの分析
 パフォーマンス分析	ランタイム パフォーマンスの分析
エラー検出ルール	検出されたエラーのフィルタまたは抑制に使用されるエラー検出ルール管理へのアクセス
 ネイティブ C/C++ インストゥルメンテーション	エラー検出、エラー検出とカバレッジ分析、パフォーマンス分析、またはカバレッジ分析のコンパイル時インストゥルメンテーション
ネイティブ C/C++ インストゥルメンテーション マネージャ 関連付け	インストゥルメンテーション マネージャへのアクセス パフォーマンス ファイルまたはカバレッジ ファイルの関連付け
カバレッジ ファイルのマージ オプション	カバレッジ分析セッションのマージ DevPartner オプションへのアクセス オプションの内容：分析、コード レビュー、エラー検出

共通要素

DevPartner ファイル拡張子

セッション ファイルのファイル拡張子です。

実行するDevPartner機能	作成されるセッションファイル(拡張子)
コードカバレッジ	.dpcov
コードカバレッジマージファイル	.dpmrg
エラー検出	.dpbcl
パフォーマンス分析	.dpprf

コマンドラインインストールメンテーションオプション

NMCL オプション

以下の表に、コマンドラインからアンマネージ(ネイティブ) Visual C++コードをインストールするために使用できるNMCLオプションを示します。NMCL.EXEは、DevPartnerのパフォーマンス/カバレッジ分析、またはエラー検出がインストールされているアンマネージVisual C++コードのコンパイルだけに使用してください。マネージコードではNMCLは使用されず、実行時に共通言語ランタイムに渡される時にDevPartnerによってインストールされます。

NMCLオプションはすべて、以下の表に示されているように、スラッシュ (/) またはハイフン (-) に続くNMで始めてください。たとえば、/NMoptionまたは-NMoptionのように指定します。

オプション	機能
/NMbcpath:bc-path	パス上にNMCLを含むディレクトリがない場合、bcinterf.libのディレクトリ場所を指定します。
/NMclpath:cl-path	cl.exeのディレクトリ場所を指定します。このオプションは、DEVENVのインストール場所をバイパスするために、またはDEVENVがインストールされていないときに使用できます。
/NMhelpまたは/?	ヘルプテキストを表示します。
/NMignore:source-fileまたは /NMignore:source-file:method source-file	インストールしないソースファイルまたはソースファイル内のメソッドを指定します。

オプション	機能
/NMlog:log-file	NMCLメッセージのログファイルを指定します(デフォルト: stdout)。
/NMnogm	CL/Gm(最小ビルド)オプションが指定されている場合、これを無視します。このオプションは、すでに判明しているNMAKE/AとCL/Gmオプション間の競合を避けるために使用できます。
/NMonly:source-file	インストールするソースファイルを1つだけ指定します。
/NMopt:option-fileまたは /NM@option-file	オプションファイル(各コマンドラインオプションが別々の行に書かれたASCIIファイル)を指定します。
/NMpass	パススルーモードを指定します。パススルーモードでは、NMCLがユーザーの介入なしにCLを呼び出します。この場合、インストールメンテーションは行われません。
/NMstoponerror	インストールメンテーション中にエラーが発生した場合、NMCLを中止します。このオプションを指定しないと、デフォルトで標準CLコンパイルにフォールバックします。
/NMbcOn	DevPartnerのエラー検出インストールメンテーションを使用します。これはデフォルトの設定です。
/NMtxOn	パフォーマンス分析とカバレッジ分析のインストールメンテーションを指定します。
/NMtxInlines	/O1、/O2、/Ob1、または/Ob2オプションを使用してインライン最適化が有効になっている場合、インライン可能とマークされているメソッドをインストールします。
/NMtxNoLines	ライン情報を収集しないようにDevPartnerに指示します。このオプションを使用すると、[ソース]タブにラインデータが表示されなくなります。また、アプリケーションのインストールメンテーションと実行にかかる時間を短縮することもできます。
/NMtxpath:tx-path	パスにNMCLを含むディレクトリがない場合、パフォーマンス分析とカバレッジ分析のライブラリファイルのディレクトリ場所を指定します。

メモ: NMCLを使用する場合、これらのユーティリティを含むディレクトリをパスに追加します。たとえば、製品をデフォルトディレクトリにインストールした場合、以下のディレクトリをパスに追加します。

C:\Program Files\Common Files\Compuware\NMShared

NMLINK オプション

以下の表に、コマンドラインからアンマネージ (ネイティブ コード) Visual C++ アプリケーションを DevPartner にリンクするために使用できる NMLINK オプションを示します。

メモ : NMLINK オプションはすべて、以下の表に示されているように、スラッシュ (/) またはハイフン (-) に続く NM で始めてください。たとえば、/NMoption または -NMoption のように指定します。

オプション	機能
/NMbcOn	DevPartner のエラー検出インストゥルメンテーションを使用します。これはデフォルトの設定です。
/NMbcpath:bc-path	パス上に NMCL を含むディレクトリがない場合、bcinterf.lib のディレクトリ場所を指定します。
/NMhelp または /?	ヘルプテキストを表示します。
/NMlinkpath:link-path	LINK.EXE のディレクトリ場所を指定します。このオプションは、DEVENV のインストール場所をバイパスするために、または DEVENV がインストールされていないときに使用できます。

オプション	機能
/NMpass	パススルー モードを指定します。パススルー モードでは、NMLINK がユーザーの介入なしに LINK を呼び出します。
/NMtxOn	パフォーマンス分析とカバレッジ分析のインストゥルメンテーションを指定します。
/NMtxpath:tx-path	パスに NMCL を含むディレクトリがない場合、パフォーマンス分析とカバレッジ分析のライブラリ ファイルのディレクトリ場所を指定します。

メモ : NMCL と NMLINK を使用する場合、これらのユーティリティを含むディレクトリをパスに追加します。たとえば、製品をデフォルト ディレクトリにインストールした場合、以下のディレクトリをパスに追加します。C:\Program Files\Common Files\Compuware\NMShared



カバレッジ分析とパフォーマンス分析

アプリケーションのテスト、カバレッジの確認、およびアプリケーションパフォーマンスのプロファイルを行います。

全般およびデータ収集のプロパティ

カバレッジ分析およびパフォーマンス分析では、以下のデータ収集プロパティを使用できます。

プロパティ	デフォルト設定
セッション ファイルを自動的にマージ	マージするかどうかを確認する
.NET アセンブリに関する情報を集める	True
COM 情報の収集	True
その他を除外	True
インライン関数をインストルメントする	True
インストルメンテーション レベル	行
システム オブジェクトの追跡	True

カバレッジおよびパフォーマンス用の DevPartner ツールバー ボタン

分析設定の選択
カバレッジ分析

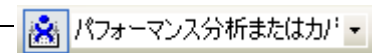


エラー検出とカバレッジ



DevPartner オプションの設定

ネイティブ C/C++
インストルメンテーション
インストルメンテーションのオン/
オフ



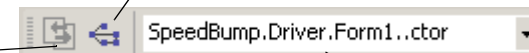
インストルメンテーションの種類
の選択

パフォーマンス分析とカバレッジ分析のセッション ツールバー

Visual Studio 6.0 では、コンテキストメニューを使用してセッションの比較やコールグラフの表示を行います。

パフォーマンス セッション ツール
バー

コールグラフの表示



セッションの比較

ソースコード内のメソッドの検索

カバレッジのセッション ツールバー



カバレッジ分析

カバレッジ分析セッション データ

データ ビューを
フィルタします

メソッドのカバレッジメトリクスを
表示します

カバレッジセッションをマージし、
マージ履歴を記録します

セッションファイルまたはマージ
ファイルの統計情報を表示します

ソースコードの各行の
実行データを表示します

結果のサマリ

DevPartnerはVisual Studioまたはカバレッジ分析ビューアにカバレッジ分析結果を表示します。セッションファイルのデータは、以下のタブに表示されます。

- メソッドリスト
- ソース
- マージ履歴
- セッション サマリまたはマージ サマリ

The screenshot displays the DevPartner Studio interface for coverage analysis. It is divided into several panes:

- File Explorer (Left):** Shows the project structure, including 'Driver_1.dpmre' and 'SpeedBump.cs'.
- Method List (Top Center):** A table showing coverage metrics for various methods.

メソッド名	カバーされた比率	呼び出し回数	未実行の行数
<Global>_CxxCallUnwindD...	0.0	0	4
SpeedBump.CSharp.Form1.Di...	40.0	1	6
<Global>.<CrImplementation...	44.4	1	10
<Global>.<CrImplementation...	50.0	1	3
<Global>_atexit_helper(void...	52.4	1	10
SpeedBump.ManagedCPP.Fo...	57.1	1	1
SpeedBump.VBdotNet.Form1...	60.0	1	1
<Global>_exit_callback(void)	62.5	1	1
SpeedBump.Driver.Form1.Dis...	62.5	1	1
<Global>.<CrImplementation...	70.0	4	1
SpeedBump.CSharp.Form1.S...	76.9	1	1
- Bar Chart (Center):** A bar chart comparing coverage across different sessions (driver_1.dpcov, driver_2.dpcov, driver_3.dpcov, driver_4.dpcov). The Y-axis represents the percentage of coverage from 0 to 100.
- Source Code View (Bottom Left):** Shows the source code for 'SpeedBump.cs' with execution counts next to each line. For example, line 44 has 848 executions.


```

1 private void BubbleSortBtn_Click(object sender, System.EventArgs e)
1 {
1     StartTiming("Timing...");
1     QuickSortBtn.Enabled = false;
1     BubbleSortBtn.Enabled = false;
1     int i, j;
1     for(i = numElems - 2; i>=1; i--)
1     {
1         for(j = 0; j<i; j++)
1         {
1             if (Elements[j] > Elements[j + 1])
1                 SwapEm(j, j + 1);
1         }
1     }
1     if (bNeedUpdate) UpdateAll();
1     RandomizeBtn.Enabled = true;
1     EndTiming("Bubble Sort:");
1 }
1 private DateTime dt;
1 private void StartTiming(string sPlaceholder)
1 {
1     dt = DateTime.Now;
            
```
- Summary Window (Bottom Right):** A window titled 'DevPartner - カバレッジ分析セッションのサマリ' showing session statistics.

項目	値
開始日	2007/11/19 17:48:57
終了日	2007/11/19 17:50:47
実行可能ファイル	C:\SpeedBump.NET\Bin\Driver.exe
コマンド引数	
終了コード	0
プロセッサのクロック	2208 Mhz
プロセッサ数	1
OSのバージョン	Microsoft Windows XP
実行済み行数の比率	34.1
行数	572
実行済みの行数	195

パフォーマンス分析

パフォーマンス分析セッション データ

データ ビューを
フィルタします

メソッドのパフォーマンス
メトリクスを表示します

ソース コード内でメソッドを
検索します

セッションの統計情報を
表示します

The screenshot displays the Performance Analysis tool interface. On the left, a tree view shows the session structure with filters for '呼び出されたメソッドの上位20' and '呼び出されたソース・メソッドの上位20'. The main area is divided into several panes:

- Method List:** A table showing method names, their percentage of time spent, and call counts. For example, 'SpeedBump.ManagedCPP.Form1.BtnB...' has a 0.90% ratio and 60.93 calls.
- Call Graph:** A hierarchical diagram showing the flow of calls between methods. 'SpeedBump.CSharp...' is the root, with 'SpeedBump.CSharp...' as a child, and further down, 'System.Windows.Forms...' and 'RtlAllocateHeap'.
- Source Code:** A snippet of C# code showing a 'Dispose' method with comments and a 'private: void Initialize()' method.
- Session Comparison:** A bar chart and table comparing the current session with a baseline. The table shows metrics like '現在の値' (Current Value) and '変化率' (Change Rate) for various methods.
- Summary:** A window titled 'DevPartner - パフォーマンス分析セッションのサマリ' (DevPartner - Performance Analysis Session Summary) providing overall statistics: start/end times, command line, processor clock (3895 Mhz), command count (2), OS version (Microsoft Windows XP), call counts (3,013), and total timing (29,708,856.89 milliseconds).

結果のサマリ

DevPartnerはVisual Studioまたはパフォーマンス分析ビューアにパフォーマンス分析結果を表示します。セッション ファイルのデータは、以下のタブに表示されます。

- メソッド リスト
- ソース
- セッション サマリ

コード変更の影響を
評価するための
セッション データを
比較します

メソッドのコール
シーケンスを調べて、
クリティカルパスを
特定します

DPAnalysis.exeの使用

DPAnalysis.exeを使用して、コマンドラインからカバレッジ分析セッションまたはパフォーマンス分析セッションを実行します。DPAnalysis.exeにはコマンドラインスイッチまたはXML構成ファイルを指定できます。

コマンドライン操作

コマンドラインからカバレッジまたはパフォーマンスの各セッションを実行するには、以下の構文を使用します。

```
DPAnalysis.exe [a] {b} {c} {d} [e] target {target args}
```

DPAnalysis.exeでは、分析とターゲットのタイプを指示するスイッチは必須です。その他のスイッチはオプションです。

以下の表に、DPAnalysis.exeで使用するスイッチをリストします。

カテゴリ	スイッチ
[a] 分析タイプ	/Cov[erage] - DevPartner カバレッジ分析に分析のタイプを設定します /Perf[ormance] - DevPartner パフォーマンス分析に分析のタイプを設定します
[b] データ収集	/E[nable] - 特定のプロセスまたはサービスのデータ収集を有効にします /D[isable] - 特定のプロセスまたはサービスのデータ収集を無効にします /R[epeat] - /D スwitchを使用してプロファイリングを無効にしないかぎり、指定プロセスを実行するたびにプロファイリングが実行されます

カテゴリ	スイッチ
{c} その他のオプション	/O[utput] - セッションファイルの出力ディレクトリとファイル名のいずれかまたは両方を指定します /W[orkingDir] - プロセスまたはサービスの作業ディレクトリを指定します /H[ost] - ターゲットのホストマシンを指定します /NOWAIT - プロセスの終了は待機せず、起動のみ待機します /N[ewconsole] - 新しいコマンドウィンドウでプロセスを実行します /F[orce] - マネージコードまたはCTIを使用せずに記述したアプリケーションのカバレッジまたはパフォーマンスのプロファイリングを強制します
{d} 分析オプション	/NO_MACH5 - 他のスレッドで費やされた時間の除外を無効にします /NM_METHOD_GRANULARITY - データ収集の精度をメソッドレベルに設定します（デフォルトは行レベル） /EXCLUDE_SYSTEM_DLLS - システムDLLに対するデータ収集を除外します（パフォーマンス分析のみ） /NM_ALLOW_INLINEING - インラインメソッドの実行時インストゥルメンテーションを有効にします（カバレッジ分析とパフォーマンス分析のみ） /NO_OLEHOOKS - COMの収集を無効にします /NM_TRACK_SYSTEM_OBJECTS - 追跡システムオブジェクトの割り当ての収集を無効にします（メモリ分析のみ）
[e] ターゲットのタイプ	プロセスまたはサービスとして、ターゲットを指定します。1つだけ選択します。ターゲットの名前/パスのあとに指定するすべてのステートメントは、引数としてターゲットに渡されます。 /P[rocess] - ターゲットプロセスを指定します（プロセスに渡される引数が続きます） /S[ervice] - ターゲットサービスを指定します（サービスに渡される引数が続きます） /C[onfig] - 構成ファイルへのパスを指定します



構成ファイル

構成ファイルからカバレッジまたはパフォーマンスの分析セッションを実行するには、以下の構文を使用します。

```
DPAnalysis.exe /config c:\temp\config.xml
```

以下の表で、XML 要素について簡単に説明します。詳細については、DevPartner オンライン ヘルプまたは『DevPartner ユーザー ガイド』を参照してください。

要素	説明
AnalysisOptions	(オプション) プロセスまたはサービスごとに、0 または 1 を指定します。特定のターゲット プロセスまたはターゲット サービスにランタイム属性を定義します。DevPartner プロパティに対応する属性には、Visual Studio のプロパティ ウィンドウからアクセスできます。 属性：SESSION_DIR、SESSION_FILENAME、NM_METHOD_GRANULARITY、EXCLUDE_SYSTEM_DLLS、NM_ALLOW_INLINING、NO_OLEHOOKS、NM_TRACK_SYSTEM_OBJECTS、NO_MACH5
Arguments	(オプション) プロセスまたはサービスごとに、0 または 1 を指定します。特定のターゲット プロセスまたはターゲット サービスにランタイム属性を定義します。DevPartner のカバレッジ分析、メモリ分析、パフォーマンス分析の各プロパティに対応する属性には、Visual Studio のプロパティ ウィンドウからアクセスできます。 属性：SESSION_DIR、SESSION_FILENAME、NM_METHOD_GRANULARITY、EXCLUDE_SYSTEM_DLLS、NM_ALLOW_INLINING、NO_OLEHOOKS、NM_TRACK_SYSTEM_OBJECTS、NO_MACH5
ExcludeImages	(オプション) プロセスまたはサービスごとに、0 または 1 を指定します。省略した場合のデフォルトはありません。ターゲット プロセスまたはターゲット サービスでロードされ、プロファイルされない場合に、イメージ (1 つ以上、上限なし) を定義します。属性はありません。

要素	説明
Host	(オプション) プロセスまたはサービスごとに、0 または 1 を指定します。省略した場合のデフォルトはありません。ターゲット プロセスまたはターゲット サービスのホスト マシンを設定します。属性はありません。
Name	サービスごとに 1 つ指定します。サービス コントロール マネージャに登録されているサービスの名前を指定します。これは、システムの NET START コマンドを使用するときと同じ名前です。属性はありません。
Path	プロセスごとに 1 つ指定します。実行可能ファイルの完全修飾パスまたは相対パスを指定します。実行可能ファイルが現在のディレクトリにある場合は、パスを指定せずに実行可能ファイル名を指定できます。属性はありません。
Process	構成ファイルには、少なくとも 1 つの Process 要素または Service 要素を指定する必要があります。ターゲットの実行可能ファイルを指定します。 属性：CollectData、Spawn、NoWaitForCompletion、NewConsole
RuntimeAnalysis	必須の要素です。1 つだけ指定します。分析のタイプと最長のセッション時間を定義します。
Service	構成ファイルには、少なくとも 1 つの Process 要素または Service 要素を指定する必要があります。ターゲット サービスを指定します。 属性：CollectData、Start、RestartIfRunning、RestartAtEndOfRun
Targets	必須の要素です。1 つだけ指定します。1 つ以上の Process エントリまたは Service エントリのブロックを開始します。ターゲットのプロセスとサービスは、構成ファイルに指定されている順に開始されます。 属性：RunInParallel



エラー検出

エラー検出

エラー検出で使用されるファイル拡張子

拡張子	ファイルの種類	説明
.dpbcl	エラー検出セッション ファイル	ユーザーのプログラム実行に関するエラー検出ログです。
.dpbcc .dpbcd	エラー検出設定ファイル	エラー検出に関するさまざまな設定を格納するファイルです。 .dpbcd 拡張子のファイルは、作成されたデフォルト設定ファイルを参照します。 .dpbcc 拡張子のファイルは、別に保存されているカスタム設定ファイルを参照します。
.dpsup	エラー検出抑制ファイル	ユーザーのプログラムに関するさまざまな抑制情報を格納するファイルです。
.dpfit	エラー検出フィルタ ファイル	ユーザーのプログラムに関するさまざまなフィルタ情報を格納するファイルです。
.dprul	エラー検出ルール ファイル	ユーザーの抑制とフィルタに関するデータベースです。

デフォルトのオプション (Visual Studio) または設定 (Visual C++)

Visual Studio 6.0、Visual Studio .NET 2003、Visual Studio .NET 2005 の間で、デフォルト値は多少異なります。

カテゴリ	設定
全般	オン イベントをログに記録
	オン エラーを表示して一時停止
	オフ プラグラム検証結果の保存を確認する
	オフ アプリケーションを終了したときに、メモリおよびリソース ビューアを表示する
	オン ソース ファイルの検索パス - .EXE (スタンドアロン)、.DSW (Visual C++)、または .SLN (Visual Studio) の場所に応じる
	- シンボルパスの上書き - デフォルト: 空白
	- 作業ディレクトリ (スタンドアロンのみ) - .EXE の場所に応じる
	- コマンドライン引数 (スタンドアロンのみ) - デフォルト: 空白
	オン コールパラメータのデータ表示の深さ = 1
	オン メモリ割り当ての最大コールスタック数 = 5
オン エラーの最大コールスタックの深さ = 20	
オン NLB ファイル ディレクトリー .EXE (スタンドアロン)、.DSW (Visual C++)、または .SLN (Visual Studio) の場所に応じる	
オン NLB ファイルを動的に生成する	

カテゴリ	設定
API コール レポートイング	オフ API コール レポートイングを有効にする。この項目を選択しないと、その他の項目は選択できません。 - ウィンドウメッセージを収集する - アクティブなときのデフォルト: オフ - API メソッドのコールとリターンを収集 - アクティブなときのデフォルト: オン - このアプリケーションに必要なモジュールだけを表示 - アクティブなときのデフォルト: オン - すべてのモジュール (ツリー ビュー) - アクティブなときのデフォルト: 選択したものすべて
コール バリデーション	オフ コールバリデーションを有効にする。この項目を選択しないと、その他の項目は選択できません。 - メモリ ブロック チェックを有効にする - アクティブなときのデフォルト: オフ - コール前に出力情報を入力する - アクティブなときのデフォルト: オフ - COM 失敗コード - アクティブなときのデフォルト: オン - COM の "実装されていません" リターン コードをチェックする - アクティブなときのデフォルト: オン - API 失敗コード - アクティブなときのデフォルト: オン - 無効なパラメータ エラーのチェック: API、COM - アクティブなときのデフォルト: どちらもオン - カテゴリ: ハンドルとポインタの引数 - アクティブなときのデフォルト: オン - カテゴリ: フラグ、範囲、および列挙の引数 - アクティブなときのデフォルト: オン - C ランタイムの静的ライブラリ API をチェックする - アクティブなときのデフォルト: オン - API エラーをチェックする DLL (失敗または無効な引数) - アクティブなときのデフォルト: 選択したものすべて
COM コール レポートイング	オフ 選択したモジュールに実装されたオブジェクト上での COM メソッド コールのレポートを有効にする - リストされたモジュール外で実装されたオブジェクトの COM メソッド コールをレポートする - アクティブなときのデフォルト: オン - すべてのコンポーネント ツリー ビュー - アクティブなときのデフォルト: 選択したものすべて
COM オブジェクトの追跡	オフ COM オブジェクトの追跡を有効にする - すべての COM クラス (ツリー ビュー) - アクティブなときのデフォルト: 選択したものすべて
デッドロック分析	オフ デッドロック分析を有効にする - シングル プロセスと仮定する - アクティブなときのデフォルト: オン - ウォッチャー スレッドを有効にする - アクティブなときのデフォルト: オフ

エラー検出

カテゴリ	設定
	<ul style="list-style-type: none"> - エラーを生成するとき：クリティカル セクションが再入力されたとき - アクティブなときのデフォルト：オフ - エラーを生成するとき：所有するミューテックスに待機が要求されたとき - アクティブなときのデフォルト：オフ - リソースごとの過去のイベント数 - アクティブなときのデフォルト：10 - 同期API タイムアウトをレポート - アクティブなときのデフォルト：オフ - 待機制限または実際の超過時間（秒）をレポート - アクティブなときのデフォルト：60 - 同期ネーミングルール - アクティブなときのデフォルト：リソースのネーミングは警告しない
メモリの追跡	<ul style="list-style-type: none"> オン メモリの追跡を有効にする オフ リーク分析のみを有効にする オフ リークしたアロケータブロックを表示する オフ 厳密な再割り当てセマンティクスを適用する オン FinalCheckを有効にする オン 保護バイトを有効にする；パターン = FC; カウント = 4バイト <ul style="list-style-type: none"> - 実行時のヒープ ブロックをチェックする：解放時 オン 確保時にフィルする；パターン = FB オン 初期化されていないメモリをチェックする；サイズ = 2バイト オン 解放時に無効データでフィルする；パターン = FD
.NET 分析	<ul style="list-style-type: none"> オフ .NET 分析を有効にする <ul style="list-style-type: none"> - 例外の監視 - アクティブなときのデフォルト：オン - ファイナライザの監視 - アクティブなときのデフォルト：オン - COM 相互運用性の監視 - アクティブなときのデフォルト：オン - PInvoke 相互運用性の監視 - アクティブなときのデフォルト：オン - 相互運用性レポートのしきい値 - アクティブなときのデフォルト：1
.NET コール レポート	<ul style="list-style-type: none"> オフ .NET メソッドコール レポートを有効にする <ul style="list-style-type: none"> - すべてのタイプ（ツリー ビュー） - アクティブなときのデフォルト：選択されている - .NET ユーザー アセンブリ（ツリー ビュー ノード） - アクティブなときのデフォルト：選択されている - .NET システム アセンブリ（ツリー ビュー ノード） - アクティブなときのデフォルト：選択されていない
リソースの追跡	<ul style="list-style-type: none"> オン リソースの追跡を有効にする オン リソース（ツリー ビュー）リストにあるすべてのリソースがデフォルトで選択される

Visual Studio のエラー検出ツールバー

エラー検出を選択して開始
カバレッジ分析の開始
エラー検出とカバレッジ分析を選択して開始
デバッグを実行せずにパフォーマンス分析を選択して開始
メモ：各ボタンの横にある矢印を使用すると、デバッグのあり/なしを指定して分析を開始できます。デバッグのあり/なしは、ボタンのデフォルトのアクションによって決まります。

ネイティブ C/C++ インストールメンテーション
インストールメンテーションのオン/オフ
インストールメンテーションタイプの選択

DevPartner オプションの設定

Visual C++ 6.0 のエラー検出ツールバー

DevPartner エラー検出

イベントのログ
エラーを表示して一時停止
フィルタされたメッセージの表示

パフォーマンスでビルド
カバレッジでビルド
エラー検出でビルド表示

エラー検出ウィンドウ

検証結果ペイン

[サマリ]、[メモリ リーク]、[その他のリーク]、[エラー]、[.NET パフォーマンス]、[モジュール]、[通知情報]の各タブに、検出されたエラーに関する情報が表示されます。

詳細ペイン

検出されたエラー、コール スタック、参照回数グラフなどについて、詳細な説明が表示されます（下の別図を参照）。

ソース ペイン

検出されたエラーのソースコードがあれば、表示されます。

詳細ペイン—参照回数グラフ

検証結果ペインで[インターフェイス リーク]を選択すると、[参照カウントビュー]タブと[オブジェクト ID ビュー]タブが表示されます。

検証結果ペインで使用されるアイコン

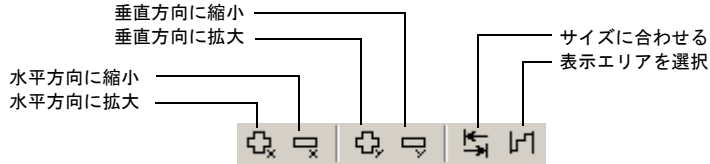
アイコン	説明	アイコンが表示されるタブ
	メモリ リーク	サマリ、メモリ リーク、通知情報
	その他のリーク	サマリ、その他のリーク、通知情報
	エラー	サマリ、エラー、通知情報
	.NET パフォーマンス	サマリ、.NET パフォーマンス
	モジュールのロード イベント	サマリ、モジュール、通知情報
	サブルーチン コール	通知情報
	ガベージ コレクション イベント	通知情報
	イベントの開始	通知情報
	イベントの再開	通知情報
	イベントの終了	通知情報

詳細ペインで使用されるアイコン

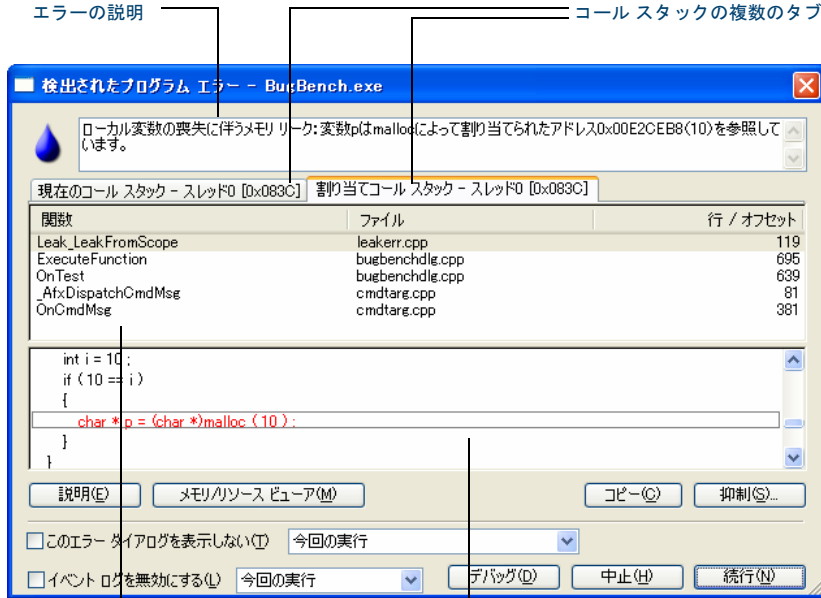
アイコン	説明
	サブルーチン コール
	開始パラメータ
	終了パラメータ
	戻り値
	データ型のプロパティ (デフォルト)
	データ型のプロパティ

エラー検出

参照回数グラフのツールバー



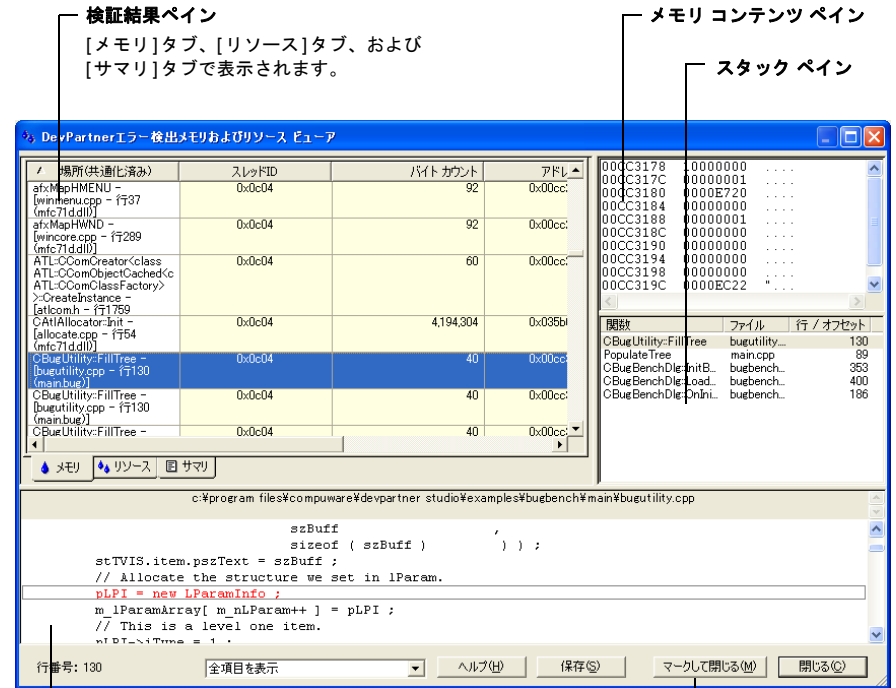
[検出されたプログラムエラー]ダイアログボックス



コールスタック情報

検出されたエラーのソースコード

[メモリおよびリソースビューア]ダイアログボックス



ソースペイン

検出されたエラーのソースコードがあれば、表示されます。

マークして閉じる

既存の割り当てをマークしたあとダイアログボックスを閉じる場合に、クリックします。[メモリおよびリソースビューア]を次に開いたとき、マークされた項目は表示されません。

ActiveCheck と FinalCheck によるエラー検出

ActiveCheck

ActiveCheck™ はプログラムを分析し、プログラム実行ファイル、およびプログラムで使用されているダイナミック リンク ライブラリ (DLL)、他社製モジュール、COM コンポーネント内のエラーを検索します。以下の表に、ActiveCheck エラー検出機能によって検出されるエラーの種類を示します。

デッドロック関連エラー	APIエラーとCOMエラー
デッドロック	COM インターフェイス メソッドの失敗
潜在的なデッドロック	不正な引数
スレッドのデッドロック	パラメータ範囲エラー
クリティカル セクションのエラー	スレッドの不正使用
セマフォ エラー	Windows 関数が失敗した場合
リソースの使用とネーミング エラー	Windows 関数が実装されていない場合
問題のある可能性が高いリソース使用状況	不正な COM インターフェイス メソッドの引数
ハンドル エラー	
イベント エラー	
ミュートックス エラー	
Windows イベント エラー	

.NET エラー	ポインタ エラーとリーク エラー
ファイナライザ エラー	インターフェイス リーク
GC.Suppress finalize が呼び出されていない場合	メモリ リーク
Dispose 属性 エラー	リソース リーク
処理されていないネイティブの例外がマネージ コードに渡された場合	

メモリ エラー
ダイナミック メモリ オーバーラン
開放したハンドルがまだロックされている場合
ハンドルがすでにアンロックされている場合
メモリ割り当ての競合
アンロックされたメモリ ブロックをポインタが参照する場合
スタック メモリ オーバーラン
スタティック メモリ オーバーラン

FinalCheck のコンパイル時インストゥルメンテーション 徹底したエラー検出

FinalCheck™ コンパイル時インストゥルメンテーション (CTI) を使用すると、メモリ リーク、ポインタ エラー、データ破壊エラーなどのエラーも、発生するたびにリアルタイムで検出されます。FinalCheck では、ActiveCheck で検出されるすべてのエラーのほか、以下のエラーが検出されます。

メモリ エラー	ポインタ エラーとリーク エラー
バッファ読み込みオーバーフロー	範囲を超えた配列の読み込み
未初期化メモリからの読み込み	有効範囲外を示すポインタのコピー
バッファ書き込みオーバーフロー	ダングリング ポインタの演算
	非関連ポインタの演算
	関数を示していない関数ポインタ
	リークによるリーク
	モジュール アンロードによるリーク
	アンワインドによるリーク
	メモリ領域の解放に伴うメモリ リーク
	メモリの再割り当てに伴うメモリ リーク
	ローカル変数の喪失に伴うメモリ リーク
	ローカル変数を指すポインタを返している場合

エラー検出

使用可能なコマンド キーのリスト - Visual Studio

コマンド	動作
Ctrl+Shift+O	[ファイル]>[開く]>[プロジェクト]
Ctrl+Shift+N	[ファイル]>[新規作成]>[プロジェクト]
Ctrl+S	[ファイル]>[プロジェクトの保存]
Ctrl+Shift+S	[ファイル]>[すべて保存]
Ctrl+Shift+F	[編集]>[ファイル内の検索]
Ctrl+Shift+H	[編集]>[ファイル内の置換]
Alt+F12	[編集]>[シンボルの検索]
Ctrl+Alt+L	[表示]>[ソリューション エクスプローラ]
Ctrl+Shift+C	[表示]>[クラス ビュー]
Ctrl+Alt+S	[表示]>[サーバー エクスプローラ]
Ctrl+Shift+E	[表示]>[リソース ビュー]
F4	[表示]>[プロパティ ウィンドウ]
Ctrl+Alt+X	[表示]>[ツールボックス]
Shift+Alt+Enter	[表示]>[全画面表示]
Shift+F4	[表示]>[プロパティ ページ]
Ctrl+Shift+B	[ビルド]>[ソリューションのビルド]
F5	[デバッグ]>[開始]
Ctrl+F5	[デバッグ]>[デバッグなしで開始]
Ctrl+Alt+E	[デバッグ]>[例外]
F11	[デバッグ]>[ステップイン]
F10	[デバッグ]>[ステップ オーバー]
Ctrl+B	[デバッグ]>[ブレークポイントの作成]
Ctrl+F1	[ヘルプ]>[ダイナミック ヘルプ]
Ctrl+Alt+F1	[ヘルプ]>[目次]
Ctrl+Alt+F2	[ヘルプ]>[インデックス]
Ctrl+Alt+F3	[ヘルプ]>[検索]
Shift+Alt+F2	[ヘルプ]>[キーワード検索の結果]
Shift+Alt+F3	[ヘルプ]>[検索結果]

使用可能なコマンド キーのリスト - Visual C++ 6.0

コマンド	動作
Ctrl+F	[編集]>[検索]
Ctrl+H	[編集]>[置換]
Ctrl+G	[編集]>[ジャンプ]
Alt+F2	[編集]>[ブックマーク]
Alt+F9	[編集]>[ブレークポイント]
Ctrl+Alt+T	[編集]>[メンバーリスト]
Ctrl+Shift+space	[編集]>[パラメータ情報]
Ctrl+Space	[編集]>[完全に一致する単語の検索]
Ctrl+W	[表示]>[クラス ウィザード]
Alt+0	[表示]>[ワークスペース]
Alt+2	[表示]>[出力]
Alt+Enter	[表示]>[プロパティ]
Ctrl+F7	[ビルド]>[(ファイル名)のコンパイル]
F7	[ビルド]>[(アプリケーション名)のビルド]
F5	[ビルド]>[デバッグの開始]>[実行]
F11	[ビルド]>[デバッグの開始]>[ステップイン]
Ctrl+F10	[ビルド]>[デバッグの開始]>[カーソル行の前まで実行]
Alt+F12	[ツール]>[ソース ブラウザ]
Ctrl+Shift+R	[ツール]>[クイック マクロの記録]
Ctrl+Shift+S	[ツール]>[クイック マクロの実行]

DevPartner データのエクスポート： コマンド ラインの使用

DevPartner データのエクスポート：コマンド ラインの使用

コマンド ラインから DevPartner.Analysis.DataExport.exe を使用して、DevPartner カバレッジ分析 (.dpcov)、カバレッジ分析マージ (.dpmrg)、およびパフォーマンス分析 (.dpprf) のセッション ファイル データを XML ファイルに変換することができます。

セッション データを XML にエクスポートするには、以下の構文を使用します。

```
DevPartner.Analysis.DataExport.exe [セッション ファイル名 | ディレクトリへのパス] {オプション}
```

オプション

以下の表に、DevPartner.Analysis.DataExport.exe のコマンド ライン オプションの一覧を示します。
指定するオプションとオプション値を区切るには、等号、コロン、スペースのいずれかを使用します。

スイッチ	説明
/out[put]=<String>	エクスポートされる XML ファイルの出力ディレクトリを指定します。ディレクトリが存在しない場合、ディレクトリが作成されます。
/r[ecurse]	DevPartner セッション ファイルのサブディレクトリを検索します。
/f[ilename]=<String>	XML 出力ファイルの名前を指定します。指定した名前に .xml が付加されます。
/showAll	パフォーマンス分析またはカバレッジ分析のセッション ファイルで利用可能な、すべてのパフォーマンス分析およびカバレッジ分析のセッション ファイル データが表示されます。 たとえば、このオプションを指定してパフォーマンス セッション ファイルをエクスポートすると、結果の XML ファイルにはパフォーマンスとカバレッジの両方のデータ フィールドが含まれます。
/w[ait]	ユーザーの入力を待機して、コンソール ウィンドウを閉じます。
/nologo	ロゴや著作権情報を表示しません。
/help または /?	コンソール ウィンドウにヘルプを表示します。