Micro Focus Net Express[R]

移行ガイド

第 3g 版
2003 年 5 月

このマニュアルは、他の Micro Focus COBOL システムから Net Express へ移行する場合に使用します。このマニュアルでは、アプリケーション移行時のコード変更やビルド変更、以前のツールのかわりに使用するツールなどについて説明します。

第 1 章 : はじめに

このマニュアルは、Micro Focus の旧製品から Net Express のこのバージョンに移行するユーザを支援するものです。具体的には、アプリケーションの移行に必要なコード変更やビルド変更について説明します。

このマニュアルでは、次の製品から Net Express のこのバージョンに移行する方法について説明します。

- Object COBOL V4.0
- Workbench V4.0
- Object COBOL V3.4
- Workbench V3.4
- Visual Object COBOL V1.0
- Embedded SQL Toolkit for Microsoft SQL
- COBSQL
- Dialog System V2.5 (32 ビット版)
- Dialog System V2.5 (16 ビット版)
- Net Express 3.0
- Net Express 3.1

Object COBOL の上位セットである Workbench には、同じコンパイラと基本ツールが含まれていますが、その環境はさらに高度なものになっています。 V3.4 と V4.0 は機能的にはほとんど同じです。ただし、V3.4 は 16 ビット版、V4.0 は 32 ビット版です。

このマニュアルや他の Net Express のマニュアルを読む前に、Net Express の『*入門書*』を必ず読んでください。

ここの残りの部分では、Net Express と以前のシステムの間でアプリケーションの作成方法にどのような相違があるかについて説明します。次の章以降は、特定の機能とそれらの移行方法に関するリファレンスです。

原則的に、製品の相違点のうち変更する必要がないものについてはこのマニュアルで説明しません。Net Express のこのバージョンでの拡張機能の一覧、および以前のバージョンからこのバージョンにアプリケーションを移行するために必要な変更の一覧については、Readme ヘルプを参照してください。

1.1 移行元の製品について

1.1.1 Object COBOL V4.0 からの移行

Object COBOL V4.0 から移行する場合は、編集、コンパイル、デバッグには Animator Version 2 が使用されます。Net Express の統合開発環境 (IDE) は Animator Version 2 と似

ていますが、IDE にはより多くの機能があるため、メニューが異なります。 特に、IDE では、デバッグ中のブレークポイント設定機能とデータ問い合わせ機能が拡張されています。

Net Express では、主にプロジェクトを使用します。 プロジェクトは、アプリケーションの全ファイル、ファイル間の関連、ファイル間のリンク方法などを定義するため、アプリケーションをパッケージ化するときに最も影響します。 プロジェクトは、どの段階でも便利に使用できるので、アプリケーションの作成時にはまずプロジェクトを作成します。アプリケーションに対する作業時には、プロジェクトを常に読み込んだ状態にしておくことをお奨めします。

プロジェクトは、さまざまな方法で活用できます。たとえば、アプリケーションの全ファイルを作成するための正しいディレクトリを常に開いておくことができます。また、「プロジェクト」ウィンドウでソースファイル名をダブルクリックするのみですぐにソースファイルを読み込むことができます。プロジェクトを最初に作成すると、アプリケーションの設計、変更時のファイル名の追加や削除を簡略化できます。

個人的に使用する、1 つの小さなプログラムで構成された小さなアプリケーションでも、プロジェクトを作成する価値があります。プロジェクトを作成していない場合は、プログラムのアニメート時に Net Express によりプロジェクトが自動作成されます。これは、プロジェクトがアニメーションに必要なためです。

Net Express では、すべての機能 (特にソースレベルのデバッグ機能やアニメート機能) を **.exe** ファイルで使用できます。 マウスを 1 度クリックするのみでコンパイルとリンクをすべて処理できます。この処理は、ビルドと呼ばれます (Object COBOL でのビルドは、**.int** ファイルや **.gnt** ファイルをライブラリにビルドすることを指し、Net Express のビルドと異なります)。ビルド中には、ユーティリティ **cobol** と **cbllink** が自動的に呼び出されていることを示す情報メッセージが表示されることがあります。

COBOL V4.0 では、文字モードのツールである Animator と Editor が使用できましたが、Net Express にはこれらのツールはありません。 ただし、Animator V2 と同様に、IDE にはこの 2 つのツールの機能とその他の追加機能があります。 特に、グラフィカルインターフェイスには、再配置やサイズ変更を簡単に行える複数のウィンドウ、強力なデータ監視機能、デバッグ中のコード修正や再コンパイルを短時間で行うために密接に統合された編集機能とデバッグ機能などが備わっています。

1.1.2 Workbench V4.0 からの移行

Workbench V4.0 から移行する場合は、『*Object COBOL V4.0 からの移行*』で説明する方法のどれかが使用されていました。 また、プロジェクトも使用されていました。ただし、Net Express のプロジェクトは、Workbench V4.0 のプロジェクトとは異なります。詳細については、オンラインヘルプを参照してください。

また、Net Express には、Organizer や Advanced Organizer に相当するものはありません。すべての機能は IDE に統合されているので、メニューからアクセスできます。

1.1.3 Object COBOL V3.4 からの移行

Object COBOL V3.4 から移行する場合は、『*Object COBOL V4.0 からの移行*』で説明する内容がすべて適用されます。この場合は、16 ビットシステムから 32 ビットシステムへの移行となります。16 ビットシステムと 32 ビットシステムの相違については、『*16 ビット製品からの移行*』の章で説明します。

1.1.4 Workbench V3.4 からの移行

Workbench V3.4 から移行する場合は、『*Workbench V4.0 からの移行*』で説明する内容がすべて適用されます。この場合は、16 ビットシステムから 32 ビットシステムへの移行になります。16 ビットシステムと 32 ビットシステムの相違については、『*16 ビット製品からの移行*』の章で説明します。

1.1.5 Visual Object COBOL V1.0 からの移行

Visual Object COBOL V1.0 は Net Express の前身であり、その基本設計は同じです。IDE は非常によく似ています。それぞれのプロジェクトはわずかに異なります。Net Express の「プロジェクト」ウィンドウには、ソースファイルの一覧を表示する 2 番目のペインが含まれます。

大きな相違は、アプリケーションの表示用フォームを作成する方法です。リソースエディタと Resource Manager のかわりに、Form Designer と Dialog System を使用します。

Form Designer は、World Wide Web やイントラネット用のアプリケーションを作成します。Form Designer には、フォームを設計するスクリーンペインタがあり、フォームを定義する HTML 形式ファイルを生成し、フォームの入出力を処理する COBOL 原始プログラムを生成します。 ビジネスロジックは、このプログラムに挿入します。

Dialog System は、Windows スタイルの GUI を使用するアプリケーションを作成します。Dialog System には、ウィンドウやダイアログボックスを設計するスクリーンペインタと、ウィンドウやダイアログボックスのイベントを処理する手続き (ダイアログ) を作成する言語が用意されています。 作成したウィンドウ、ダイアログボックス、およびダイアログはスクリーンセットに保存されます。スクリーンセットは、ビジネスロジックを含む COBOL プログラムから呼び出します。

1.1.6 他の製品からの移行

Embedded SQL Toolkit for Microsoft SQL または Dialog System V2.5 (32 ビット版または 16 ビット版) から移行する場合の詳細については該当する章を参照してください。

1.2 コード互換性

Object COBOL または Workbench 用に作成されたソースコードは、Net Express で完全にコンパイルする必要があります。 ただし、Object COBOL や Workbench が生成する **.gnt** または **.obj** コードは Net Express との互換性がありません。このため、再コンパイルする必要があります。 **.int** コードは、オブジェクト指向のコードを含まない限り互換性があります。

1.3 ACCEPT と DISPLAY のリンクタイプ

アプリケーションで ANSI の ACCEPT 文と DISPLAY 文を使用して文字モードの入出力を行う場合は、プロジェクトのリンクタイプを「キャラクタ」に設定する必要があります。「プロジェクト」ウィンドウでプロジェクトの **.exe** ファイルを右クリックし、表示される「ビルド設定」ダイアログボックスで、[**リンク**] タブをクリックします。 [**キャラクタ**] を選択し、[**閉じる**] をクリックします。このように設定しない場合は、アプリケーションの実行時に出力が得られなくなります。

Micro Focus の全画面拡張 ACCEPT または DISPLAY では、このように設定する必要はありません。

1.4 未定義の結果

COBOL 操作の戻り値によっては、『***言語リファレンス***』で未定義として指定されているものがあります。たとえば、ゼロ除算や PICTURE 句に準拠しないデータを使用する場合などです。このような操作により実際に返される値を調べると、実行可能ファイル形式とプログラムがコンパイルされる COBOL システムによって戻り値が異なることがわかります。

たとえば、ゼロ除算では、中間コードプログラムの戻り値と生成コードプログラムの戻り値が異なることがあります。 同様に、16 ビットの COBOL システムでコンパイルされたプログラムの戻り値は、32 ビットの COBOL システムでコンパイルされた同じプログラムの戻り値とは異なることがあります。

プログラムの結果やロジックが、戻り値が未定義である演算の結果に依存しないことを確認してください。 たとえば、ゼロ除算の結果は予測できないため、この結果に基づいて特定のアクションまたは手続きを実行するには、常に ON SIZE ERROR を使用する必要があります。

1.5 Windows GUI から HTML への変換

Windows GUI アプリケーションを Web アプリケーションに変換する場合は、GUI アプリケーションと Web アプリケーションの構造上の相違を考慮する必要があります。

GUI アプリケーションでは、イベントはウィンドウやダイアログボックスの個々のコントロールにより発生します。 これらのイベントはプログラムに送信され、常に読み込まれた状態になります。 イベントは、送信されるたびに識別され、処理されます。これらの処理は、すべて 1 つのプログラムで行われるため (副プログラムを伴う場合があります)、プログラムデータは 1 つのイベントから次のイベントに引き継がれます。

対照的に、Web アプリケーションでは、ユーザが [送信] ボタンをクリックするとフォーム全体が Web サーバに送信されます。 その結果、Web サーバのソフトウェアがプログラムを起動します。 プログラムはフォームのデータを処理し、応答としてユーザにフォームを送信し、終了します。 通常は、フォームの種類ごとに別のプログラムがあります。 このように、フォームを受信するたびに新しいプログラムが起動されるため、プログラムのデータが次回のプログラム起動時に引き継がれることはありません。

通常の HTML のコントロールではなく ActiveX コントロールを使用し、JavaScript でイベントハンドラを作成すると、Web アプリケーションで GUI アプリケーションのスタイルを再現できます。ActiveX コントロールは、Web ブラウザで実行され、Web サーバにあるメインの処理プログラムに戻らずにコントロールのイベントを処理します。ただし、ActiveX と JavaScript は高度な機能であるため、GUI アプリケーションを変換するためにこの方法を使用することはお奨めしません。

前述のとおり、アプリケーションを再構築することをお奨めします。つまり、各ウィンドウやダイアログボックスを同等の Web フォームで単に置換しません。通常、アプリケーションを新しく設計すると、ウィンドウやダイアログボックスよりも単純な Web フォームを数多く使用できるようになります。

プログラムのデータがプログラムを起動するたびに引き継がれることはありませんが、ある起動時のデータを別の起動時に渡す方法はあります。たとえば、フォームの非表示フィールド、ディスクのファイル、cookie の 3 種類を使用すると便利です。Net Express には、この操作を簡略化できる機能があります。詳細については、マニュアル『***分散コンピューティング***』を参照してください。

第 2 章：移行前の機能

ここでは、既存の Micro Focus COBOL システムで使用できる機能を列挙し、Net Express でこれらに最も近い機能を使用するための変更について説明します。

また、開発ツールの変更点や代用するツールについても説明します。ランタイム機能を使用する既存のアプリケーションを Net Express でコンパイルして実行するために必要な変更について説明します。

ここでは、各章で説明されている機能は取り上げません。

2.1 削除された機能

Net Express の以前のバージョンで使用可能であった機能の一部は、このバージョンで削除されています。

削除された機能は次のとおりです。

- ファイルのストライプ化

  2 GB を超えるデータファイルがあり、そのファイルを大容量ファイルシステムに配置できない場合は、ファイルのストライプ化を使用できます。

2.2 除外された構成要素

Net Express では、次のような Object COBOL および Workbench で使用できる構成要素のカテゴリが削除されています。

- 文字モードのアプリケーションを処理するツール

  Net Express は、主にグラフィカルアプリケーションと Web アプリケーション用です。ただし、COBOL 言語の拡張 ACCEPT または DISPLAY 機能 (Adis) とウィンドウ化構文がサポートされており、また、Adis を構成するユーティリティも含まれています。Panels 機能も含まれています。ANSI の ACCEPT 文または DISPLAY 文もサポートされています。

- メインフレームからのオフロードを支援するツール

  Net Express は、分散型 Windows/UNIX 環境で実行するアプリケーション開発を目的としています。Mainframe Express は、メインフレームのオフロード用です。

- アニメータとエディタの全バージョンとアクセサリ

すべての編集機能とアニメート機能は、完全に IDE に統合されています。 重要な機能はすべて組み込まれ、その多くは拡張されています。

- グラフィックバージョンに置き換えられたツール (データファイル編集ツールなど)
- 広く使用されないツール

除外された構成要素は、次のとおりです。

- Application Configuration System、バナー、カラー、キー操作マクロ、Linein、メニューハンドラ、スクリーン

- Data File Filter、Data File Structure Catalog、File Transfer Aid、インターフェイスプログラム、Parameter Passer (文字)、Parameter Passer (グラフィック)、Setup (文字)、Source Converter

  Data File Structure Editor は、レコードレイアウトエディタに名前が変更されました。

- Analyzer、Animator、Animator V2、基本アニメータ、COBOL ソース情報 (CSI)、GNT Analyzer、Structure Animator、Xilerator
- 方言互換ツール (Convert3 と Convert5)、通信モジュール (MCS)

上記の構成要素については、特別言及する必要がない限り、ここでは説明しません。

2.3 新規の画面処理アプリケーション

ここでは、原則的に、新しいアプリケーションに対して使用する機能について説明します。 全体で繰り返し説明しないで、ここでまとめて説明します。新規の Web アプリケーションとイントラネットアプリケーションを作成するには、Form Designer を使用します。Windows 形式の GUI アプリケーションを作成するには、Dialog System を使用します。

2.4 機能説明

機能はアルファベット順に説明します。 説明されていない機能については、変更する必要はありません。 各項目で使用される副見出しは、次のとおりです。

- **この機能を含む製品** - 説明する機能を含む Net Express 以前の Micro Focus 製品を列挙します。
- **Net Express** - 説明する機能が Net Express に含まれているかどうか、および他の製品と比較して変更されているかどうかを説明します。
- **移行** - 説明する機能を使用するアプリケーションを Net Express へ移行するために必要な変更点を解説します。
- **Net Express で使用する機能** - 説明する機能が Net Express にない場合、または下位互換性を確保するためのみに装備されている場合に、新しく同じ効果を実現する機能を紹介します。

2.4.1 機能リスト

2.4.1.1 Advanced Organizer

Workbench デスクトップを作成するソフトウェア。Workbench の上位レベルウィンドウまたは開発環境です。

この機能を含む製品:

Workbench V4.0

Net Express:

使用できません。

移行:

Net Express では、Workbench プロジェクトを使用できません。最初から Net Express プロジェクトを作成する必要があります。

Net Express で使用する機能:

Net Express の開発環境は、統合開発環境 (IDE) と呼ばれます。 IDE では、プロジェクトの編集、アニメート、確認、作成などが可能です。 デスクトップはなく、アイコンは作成しません。 IDE へは Windows 95 または Windows NT の [**スタート**] メニューから進むことができます。 開発ツールは、IDE のプルダウンメニューに表示されます。

また、[**スタート**] メニューからコマンド行セッションを開始し、Net Express の環境設定を行うことができます。 開発ツールによっては、プロンプトでコマンドを入力すると、ここから実行できます。

2.4.1.2 アニメーション

ソースレベルのデバッガ機能

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

IDE に組み込まれています。以前の製品では、GUI バージョンは Animator V2 と呼ばれ、文字バージョンは Animator と呼ばれます。IDE には、GUI バージョンの重要な機能がすべて含まれています。これらの機能は、主に [**アニメート**] メニューにあります。

Net Express コマンドプロンプトからプログラムをコンパイルする場合は、1 つのわずかな相違が影響します。Net Express では、ANIM 指令を指定するのみでは、自動的に **.int** ファイルを作成できません。**.int** ファイルを作成するには、NOGNT 指令も指定する必要があります。IDE でコンパイルする場合 (通常の方法) には、作成されたファイルは選択したビルドタイプで定義されるため、影響はありません。

2.4.1.3 Application Configuration System (Mfconfig)

アプリケーションの文字モード表示のランタイムプロパティ (たとえば、色) を制御する構成ファイルにエントリするユーティリティ。アプリケーションからこのユーティリティを呼び出して、実行時にプロパティを変更することもできます。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

アプリケーションが Mfconfig を呼び出すことなく、単に Mfconfig が作成した構成ファイル (**.cfg**) を使用するのみの場合には、変更は必要ありません。Net Express は、下位互換性を確保するために **.cfg** ファイルをサポートします。**.cfg** ファイルを更新する必要がある場合は、IDE またはテキストエディタを使用してファイルを編集できます。この構成ファイルの形式については、Object COBOL または Workbench に添付されているマニュアル『***プログラマーズガイド - アプリケーション作成***』を参照してください。

アプリケーションが Mfconfig を呼び出す場合は、その呼び出し箇所を削除し、アプリケーションを再設計する必要があります。アプリケーションを再設計することで、実行時のプロパティ表示は変更されません。Net Express は、文字モード表示での実行時のプロパティ表示の変更をサポートしません。

2.4.1.4 Build

.lbr ライブラリファイルで構成されるアプリケーションを起動する起動プログラムを作成するユーティリティ。文字バージョンとグラフィックバージョンがあります。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

詳細:

以前の COBOL システムでは、アプリケーションを .int ファイルまたは .gnt ファイルとして配布できます。これらのファイルは、通常 .lbr ファイルにパッケージ化されます。 アプリケーションを起動するには、トリガと呼ばれる小さな .exe ファイルを含めます。 トリガを作成するには、Build ユーティリティを使用します。アプリケーションをこのようにパッケージ化することを「ビルド」と呼びます。 または、.obj ファイルにコンパイルし、これらのファイルを .dll ファイルと .exe ファイルにリンクして配布することもできます。

Net Express では、.exe ファイルと .dll ファイルにリンクされたアプリケーションを配布します。 .int ファイルと .gnt ファイルは、IDE でのデバッグのみに使用されるファイルです。 アプリケーションのプロジェクトでは、オプションにより .int ファイル、.gnt ファイル、または、.dll ファイルと .exe ファイルにリンクさせる .obj ファイルを作成するかを指定します。 .lbr ファイルも作成することができますが、このファイルはデータファイルを格納するためにのみ使用されます。

Net Express では、「ビルド」という用語が以前のシステムとは異なる意味で使用されています。以前のシステムでは、アプリケーションを .dll ファイルと .exe ファイルにリンクさせることに対して、アプリケーションを .lbr ファイルにパッケージ化することを「ビルド」と呼びますが、Net Express では、ソースファイルから配布可能な製品を作成する手順全体を「ビルド」と呼びます。この手順には、コンパイルとリンクも含まれます。

移行:

.dll ファイルと .exe ファイルとしてパッケージ化します。 プロジェクトでは、デフォルトのオプションがこの方法を使用するため、アプリケーションのプロジェクトを使用すると非常に簡単にこの操作を行うことができます。

メインの .int ファイルまたは .gnt ファイルを明示的に呼び出す起動プログラムを作成できます。 この場合は、起動プログラムを .exe ファイルとしてリンクするのみです。

Net Express で使用する機能:

アプリケーションを作成する場合は、プロジェクトを使用してください。

2.4.1.5 Call Name Resolution (Mfentmap)

エントリポイントを実行時に検索できるようにするマップファイルを作成するユーティリティ

この機能を含む製品:

Object COBOL V4.0、Workbench V4.0、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

このユーティリティを使用すると、**.gnt** ファイル内または **.lbr** ファイル内のエントリポイントを可視できます。このため、アプリケーションをアニメートするときにエントリポイントを見つけることができます。 かわりに、ファイル内のエントリポイントが参照される前に、ファイルを呼び出すことでエントリポイントを可視にすることもできます。Net Express で運用向けに作成する通常のリンク済みアプリケーションの場合は、参照がリンク時に解決されるため、この問題は発生しません。

2.4.1.6 Cbllink

アプリケーションをリンクさせるコマンド行ユーティリティ

この機能を含む製品:

Object COBOL V4.0、Workbench V4.0

Net Express:

変更点はありませんが、直接使用することはほとんどありません。

Net Express で使用する機能:

リンク処理は、通常、Net Express でプロジェクトを通じて行われます。この手順はほとんど自動化されており、Cbllink が自動的に呼び出されます。**cbllink** コマンドを使用して、Net Express コマンド行から Cbllink を呼び出すこともできます。

2.4.1.7 クラスブラウザ

オブジェクト指向の COBOL プログラムとクラスライブラリを表示し、編集するツール

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

参照機能は、Visual Object COBOL V1.0 の参照機能と同じです。

Net Express で使用する機能:

IDE で [検索] メニューの [**ブラウザ**] を使用します。このブラウザは、Visual Object COBOL V1.0 のクラスブラウザと同じで、Object COBOL と Workbench のブラウザに似ていますが、インターフェイスが改善されています。

2.4.1.8 クラスライブラリ

オブジェクト指向のアプリケーションで使用できる定義済みのクラスライブラリ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

機能が拡張されています。Net Express のクラスライブラリは、次の 3 種類に分けられます。基本クラスライブラリ、GUI クラスライブラリ、COM クラスライブラリです。Visual Object COBOL のすべてのクラスには、互換性があります。クラスによっては、変更されたため、Object COBOL や Workbench との互換性がないものがあります。

移行:

Visual Object COBOL から移行する場合は、変更は必要ありません。Object COBOL や Workbench から移行する場合は、『*クラスライブラリ*』の章で説明している互換性のない変更のリストを参照してください。

2.4.1.9 Cob

アプリケーションをリンクさせるコマンド行ユーティリティ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

このユーティリティは Cbllink で置き換えられますが、直接使用されることはほとんどありません。

Net Express で使用する機能:

リンク処理は、通常、Net Express でプロジェクトを通じて行われます。この手順はほとんど自動化されており、Cbllink が自動的に呼び出されます。**cbllink** コマンドを使用して、Net Express コマンド行から Cbllink を呼び出すこともできます。

2.4.1.10 COBOL システムライブラリルーチン

アプリケーションから呼び出せるルーチンのライブラリ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

この機能は Net Express にも含まれています。Visual Object COBOL のすべてのルーチンには、互換性があります。Object COBOL と Workbench のルーチンによっては、除外されたものもあります。

移行:

Visual Object COBOL から移行する場合は、変更は必要ありません。Object COBOL や Workbench から移行する場合は、『*COBOL ライブラリシステムルーチン*』の章で説明している互換性のない変更のリストを参照してください。

2.4.1.11 COBSQL

Micro Focus COBOL と特定のデータベースプリコンパイラとの間のインターフェイス。埋め込み SQL 文を含むプログラムの場合は、COBSQL を使用して元の原始プログラムをデバッグできます。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

下位互換性を確保するために、Net Express 2.0 以降に含まれています。Net Express 1.0 では使用できません。 詳細は、『*埋め込み SQL アプリケーション*』の章を参照してください。

移行:

サポートされている 2 つのプリコンパイラのどちらかを使用する以前の Micro Focus COBOL 製品でアプリケーションを開発した場合、または UNIX プラットフォームへ展開するアプリケーションを開発し、Oracle や Sybase リレーショナルデータベースのどちらかにアクセスする必要がある場合は、COBSQL を使用します。

それ以外の埋め込み SQL アプリケーションの開発には、OpenESQL を使用することをお奨めします。 OpenESQL を使用すると、埋め込み SQL を使用して ODBC データソースにアクセスできるアプリケーションを素早く、そして簡単に開発できます。 OpenESQL は UNIX でもサポートされています。

2.4.1.12 共通通信インターフェイス (CCI)

特定のプロトコルを使用したアプリケーション間通信ルーチンのライブラリ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0。Object COBOL V4.0、Workbench V4.0、および Visual Object COBOL V1.0 では、TCP/IP プロトコルと動的データ交換 (DDE) のみがサポートされています。

Net Express:

変更点がいくつかあります (次の『*詳細*』を参照)。インターフェイスに関する説明はありません。クライアント / サーバ結合と呼ばれる高度なインターフェイスを使用できます。 CCI は Fileshare で使用します。

詳細:

Net Express は NetBEUI、TCP/IP、DDE および Novell IPX プロトコルを装備しています。 他のプロトコル (名前付きパイプ、UNIX の名前付きパイプ) はサポートされていません。

Net Express では、タイムアウト期間を動的に設定できます。 以前の製品でアプリケーションコードに CCI-TIMEOUT 動詞を使用すると、それ以降の CCI 動詞すべてに 120 秒間のデフォルトのタイムアウト期間が設定されます。Net Express の CCI-TIMEOUT 動詞は、渡された値を受け付けます。

CCITCP2 は NT サービスとして実行できます。 -i オプションを使用して CCITCP2 をインストールし、サービスとして実行します (ただし、管理者権限をもつユーザとしてログオンする必要があります)。 デバッグコンソールが必要な場合は、-c オプションを使用してください。 アンインストールするには、-u オプションを使用します。 使用可能なオプションのリストを取得する

には、-? オプションを使用します。 CCITCP2 がサービスとしてインストールされると、通常の NT サービスの GUI を使用して (Windows の [コントロール パネル] から起動します)、動作を監視し、制御できます。

CCITCP2 は、NT サービスとしてインストールされている場合は、NT システムの再起動時に自動起動します。[**スタート**] メニューの [スタートアップ] グループに CCITCP2 サーバを追加している場合は、削除し、必要なときにのみ実行します。それ以外の場合には、Windows NT は CCITCP2 の 2 番目のインスタンスを起動しようとします。

移行:

アプリケーションで NetBEUI、TCP/IP、DDE または IPX を使用する場合には、変更は必要ありません (『*16 ビット製品からの移行*』の章を参照してください)。

Net Express で使用する機能:

クライアント / サーバ結合を使用することをお奨めします。

2.4.1.13 通信モジュール (MCS)

プログラム間でメッセージを送信する COBOL 構文

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

クライアント / サーバ結合を呼び出すことで通信構文を置き換えます。 この通信モジュールは、アプリケーション間の通信 (CCI と似ていますが、より高いレベル) に対するルーチンのセットです。

Net Express で使用する機能:

クライアント / サーバ結合を使用することをお奨めします。

2.4.1.14 コンパイラ

構文を確認し、実行可能形式に変換する開発ツール

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL

Net Express:

変更点はありません。ただし、コンパイルは、通常、プロジェクトを通じて Net Express で行われます。その際には、手順のほとんどが自動化され、コンパイラが自動的に呼び出されます。コンパイラは変更されていませんが、指令を使用して動作を詳細に設定できるようになっています。 なお、デフォルト設定は、変更されている場合があります。詳細は、『*指令*』の章を参照してください。

Net Express で使用する機能:

アプリケーションを作成する場合は、プロジェクトを使用してください。

コンパイラは、IDE の [プロジェクト] メニューとツールバーにある [**プログラムをコンパイル**] 機能を使用して実行することもできます。

また、Net Express コマンド行で **cobol** コマンドを使用してコンパイラを呼び出すこともできます。

2.4.1.15 コンパイラオプション選択エイド

コンパイラ指令を設定するグラフィカルインターフェイスを提供するユーティリティ

この機能を含む製品:

Workbench V3.4、Workbench V4.0

Net Express:

使用できません。

Net Express で使用する機能:

Net Express の「プロジェクト」ウィンドウで **.cbl** ファイルを右クリックすると、ポップアップメニューが表示されます。このメニューで [**プロジェクトのプロパティ**] 機能を選択すると、プロジェクトのコンパイラ指令をすべて指定できるダイアログボックスが表示されます。[**ビルド設定**] 機能を選択すると、個々の **.cbl** ファイルをコンパイルするための指令を指定できるダイアログボックスが表示されます。

2.4.1.16 Co-Writer

COBOL データファイルおよび製品レポートを処理して、クエリーを行うアプリケーションを生成するアプリケーションジェネレータ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

Co-Writer を使用して生成されたアプリケーションは自己格納式で、Co-Writer でのみ更新できます。 このため、このアプリケーションは Net Express に移行しないでください。

2.4.1.17 並行処理のサポート

COBOL プログラムを同じプロセスで並行して実行するための環境 (**cbl_create_run_unit** などの、並行処理用の COBOL システムライブラリルーチンと混同しないでください)。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

使用できません。

移行:

並行処理を使用しないように、アプリケーションを再設計する必要があります。

Net Express で使用する機能:

マルチスレッドを使用するようにアプリケーションを再作成してください。

2.4.1.18 並行 Workbench

さまざまな開発機能を同時に実行する機能

この機能を含む製品:

Workbench V3.4

Net Express:

使用できません。

Net Express で使用する機能:

IDE で複数のウィンドウを使用することで、多くの機能を同時に実行できます。 ただし、コンパイルとデバッグを同時に実行することはできません。

2.4.1.19 Directory Facility (Mfdir と Mfdir2)

ディレクトリを列挙し、ファイルの読み込みと保存を行うグラフィック機能を提供するプログラム。 アプリケーションから呼び出せます。 文字バージョンは、Directory Facility と呼ばれ、グラフィックバージョンは、Directory Facility Version 2 と呼ばれます。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

Windows 95 と Windows NT で FileOpen と FileSave に共通のダイアログボックスを使用します(例については、Net Express の Win32API デモを参照してください)。

2.4.1.20 外部ファイルマッパー (Mfextmap)

論理ファイル名から物理ファイル名へのマップや、COBOL プログラムからの環境変数の読み書きを実行時にサポートします。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

下位互換性を確保するために、変更されないまま Net Express に含まれています。説明はありません。

移行:

変更は必要ありません。

2.4.1.21 ファイル比較ユーティリティ (Diff)

2 つのファイル間で相違点を検索する文字モードユーティリティ。 文字バージョンとグラフィックバージョンがあります。

この機能を含む製品:

Workbench V3.4、Workbench V4.0

Net Express:

使用できません。

Net Express で使用する機能:

Net Express に同梱されている Microsoft 社の Win32 Software Development Kit の WinDiff を使用します。

2.4.1.22 File Finder (Mffinder)

指定されたファイル (複数可) の複数のディレクトリを検索するユーティリティ

この機能を含む製品:

Workbench V3.4、Workbench V4.0

Net Express:

使用できません。

Net Express で使用する機能:

Windows 95 や Windows NT V4.0 の **dir** コマンドは、Mffinder が実行するすべての機能 (**dir /s** でサブディレクトリを検索) を提供します。ただし、**.lbr** ファイルは検索しません。**.lbr** ファイルを検索する機能はありません。このファイルは、Net Express ではあまり使用されません。

2.4.1.23 ファイルのストライプ化

2 GB を超えるデータファイルがあり、そのファイルを大容量ファイルシステムに配置できない場合は、ファイルのストライプ化を使用できます。

Net Express:

Net Express 3.0、Net Express 3.1

詳細:

多くの古いオペレーティングシステムでは、ファイルサイズに制限があります (たとえば、Windows 95 では 2 GB)。これらのシステムで大容量ファイルを作成できるようにするには、IDXFORMAT(8) を使用するかわりに、ストライプ化オプションを使用します。ストライプ化は大容量論理ファイルを必要な数の物理的なファイルに分割します (最大数 256)。

ファイルストライプ化オプションを使用すると、最大 512 GB の論理ファイルを作成できます。これらの論理ファイルは、最大 256 の物理ファイルで構成できます。

これらの大容量の論理ファイルは複数の物理ファイルで構成されるため、ファイルストライプ化をグローバルに行うと、オペレーティングシステムでファイルハンドルが足りなくなる可能性があります。 そのため、ストライプ化を行うファイルを決定する必要があります。

ファイルをストライプ化するには、ファイルハンドラ構成ファイルでストライプ化する各ファイルに STRIPING=ON パラメータを設定します。

ストライプ化されたファイルでは、個々のストライプのヘッダーの長さは 128 バイトになります。ヘッダーには、次の形式の ASCII 文字列が含まれます。

ファイルのストライプ番号 *nnn* : *filename*

この *nnn* には、ストライプ番号を表す 3 桁の数値を指定します。*filename* は、関連付けられたファイルの名前です。

文字列は、128 バイトのヘッダーの末尾まで空白文字で埋められます。

移行:

2 GB を超える大容量ファイルをサポートしないシステムから大容量ファイルをサポートするシステムに移行した場合は、rebuild の /t:mf8 スイッチを指定して、ストライプ化されたファイルを大容量ファイル形式に変換できます。 たとえば、次のように入力します。

```
rebuild STRIPEDFILE.DAT,BIGFILE.DAT /t:mf8
```

Net Express で使用する機能:

IDXFORMAT(8) を使用して大容量の索引ファイルを作成します。

2.4.1.24 フォーム

文字モードの画面表示を作成する文字モードスクリーンペインタ

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

変更は必要ありません。フォームにより生成された画面処理コードを更新する必要がある場合は、手動で編集できます。これは標準 COBOL で、データ部に記述されています。

2.4.1.25 Header-to-Copy(H2cpy)

C ヘッダーファイルを COBOL コピーファイルに変換するユーティリティ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

変更点はありません。このユーティリティを使用するには、Net Express コマンド行で **h2cpy** コマンドを使用して呼び出します。

2.4.1.26 Hexedit

16 進でファイルをリストし、編集するユーティリティ

この機能を含む製品:

Workbench V3.4、Workbench V4.0

Net Express:

使用できません。

Net Express で使用する機能:

同等の機能はありません。

2.4.1.27 ライブラリ

ファイルを .lbr ファイルにパッケージ化するユーティリティ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

下位互換性を確保するために、変更されないまま Net Express に含まれています。

移行:

詳細は、『*Build*』を参照してください。

2.4.1.28 Memsnap

アプリケーションのメモリ使用量を監視するユーティリティ

このルーチンを含む製品:

Workbench V3.4

Net Express:

使用できません。

Net Express で使用する機能:

Windows 95 と Windows NT V4.0 には、メモリの監視に使用できるさまざまなユーティリティが用意されています。Net Express に同梱されている Microsoft 社の Win32 Software Development Kit の Pview95 を使用します。COBOL に限定された情報を取得する方法はありません。

2.4.1.29 MFSCCS

Workbench ツールからサードパーティのソースコントロールシステムへのインターフェイス

この機能を含む製品:

Workbench V3.4

Net Express:

使用できません。

移行:

すべてのソースファイルがローカルマシン上に存在することを確認してからビルドを開始してください。

2.4.1.30 オブジェクト指向 COBOL

オブジェクト指向のコードを作成する構文

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

この機能は、Net Express にも含まれています。ただし、若干、変更されています。

- CLASS-OBJECT 見出しと END CLASS-OBJECT 見出しは、Object COBOL クラスのクラスオブジェクト部分の前後で必要になります。
- CLASS-OBJECT 見出しと END CLASS-OBJECT 見出しで囲まれた範囲外の手続き部にクラスがある場合は、クラスの読み込み時に実行されなくなります。この場合は、「initializeClass」という名前のクラスメソッドを作成し、クラス初期化コードを転記してください。その結果、「initializeClass」メソッドがクラスの読み込み時に実行されるようになります。

移行:

以前の製品を使用して作成されたオブジェクト指向のプログラムは、Net Express で実行する前に再コンパイルする必要があります。

2.4.1.31 Organizer

Workbench デスクトップを作成するソフトウェア。Workbench の上位レベルウィンドウまたは開発環境です。

この機能を含む製品:

Workbench V3.4

Net Express:

使用できません。

Net Express で使用する機能:

詳細は、『*Advanced Organizer*』を参照してください。

2.4.1.32 Panels

文字モードの画面表示を作成し、操作するルーチンのライブラリ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

下位互換性を確保するために、変更されないまま Net Express に含まれています。 新しいアプリケーションには推奨できないため、下位互換性のヘルプに説明があります。

移行:

変更は必要ありません。

2.4.1.33 Panels Version 2 エミュレーションモード

Panels V2 と文字モード画面とをインターフェイスする実行時レイヤ。文字モードシンボルを使用してグラフィックオブジェクトと同じような形状を描いて、同じようなインタフェースを実現します。。

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

かわりに Panels V2 を使用してください。 インターフェイスは同じです。 初期化の呼び出しで、文字のフラグのかわりに、GUI のフラグを設定するのみです。

2.4.1.34 Probe

アプリケーションのメモリ使用量を監視するユーティリティ

このルーチンを含む製品:

Workbench V3.4

Net Express:

使用できません。

Net Express で使用する機能:

Windows 95 と Windows NT V4.0 には、メモリの監視に使用できるさまざまなユーティリティが用意されています。Net Express に同梱されている Microsoft 社の Win32 Software Development Kit の Pview95 を使用します。COBOL に限定された情報を取得する方法はありません。

2.4.1.35 プログラムファイルマッパー

異なるプログラムで環境変数から物理ファイル名を取得した際に、同じ EXTERNAL ファイルを異なる物理ファイルにマップさせる機能

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

下位互換性を確保するために、変更されないまま Net Express に含まれています。説明はありません。

移行:

変更は必要ありません。

2.4.1.36 プロジェクト (Visual Object COBOL)

アプリケーションのコンパイルとリンクを導く機能

この機能を含む製品:

Visual Object COBOL V1.0

Net Express:

機能が拡張されています。

移行:

変更は必要ありません。Net Express で Visual Object COBOL プロジェクトを使用できます。

2.4.1.37 プロジェクト (Workbench)

アプリケーションのコンパイルとリンクを導く機能

この機能を含む製品:

Workbench V3.4、Workbench V4.0

Net Express:

プロジェクトは、Net Express で主要なものですが、Workbench プロジェクトは異なるものです。

移行:

Net Express プロジェクトを作成してください。

2.4.1.38 ランタイムチューナ

ランタイムオプションを設定する機能

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

Net Express で構成できるランタイムオプションの標準設定は、IDE で使用可能です。 使用している他のオプションでは、適切なチューナが下位互換性を確保するために Net Express に含まれています。 ヘルプの索引『*ランタイムチューナ*』を参照してください。 ランタイムチューナ **program_environment_mapper** は変更されないまま Net Express に組み込まれていますが、説明はしていません。

移行:

可能な場合は、IDE でランタイムオプションを設定します。[**プロジェクト**] メニューで [**プロパティ**] をクリックしてから、[**アプリケーション**] をクリックします。設定できない場合は、設定されているランタイムチューナをそのまま使用してください。

Net Express で使用する機能:

IDE でランタイムオプションを設定します。

2.4.1.39 SCMF

ソースコードの中央記憶域を制御するユーティリティ。ロック機能とバージョンコントロール機能を備えているため、一連のリビジョンを再構成できます。

この機能を含む製品:

Workbench V3.4

Net Express:

使用できません。Net Express には、ソースコードコントロールを備えた PVCS が含まれています。

2.4.1.40 Screens

文字モードの画面表示を作成する文字モードスクリーンペインタ

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。

移行:

変更は必要ありません。Screens により生成された画面処理コードを更新する必要がある場合は、手動で編集できます。これは、画面節に記述される標準 COBOL です。

2.4.1.41 Setup (文字)

アプリケーションをインストールするユーティリティ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

使用できません。文字モードの Setup は、以前の製品のインストールに使用するユーティリティです。このユーティリティはスクリプト駆動型で、ユーザ作成のアプリケーションをインストールするスクリプトで、配布することもできます。Net Express は新しいグラフィック Setup でインストールします。グラフィック Setup は、ユーザ作成のアプリケーションで配布できません。

移行:

グラフィカルユーティリティを作成するツールは、InstallShield 社など、数社から入手できます。

2.4.1.42 ソートユーティリティ

ファイルをソートさせるユーティリティ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

IBM 社の Dfsort 製品をエミュレートする新しいバージョン Release 14 が Net Express に含まれています。

このユーティリティを使用するには、Net Express コマンド行で **mfsort** コマンドを使用して呼び出します。

移行:

新しいバージョンは、キータイプ FI が符号付きバイナリを意味すること (Dfsort の場合)、および後続符号を表示しないことを除けば、以前のバージョンと完全に互換性があります。「DISPLAY」フィールドに後続符号がある場合は、Dfsort に対して TI または ZD を指定する必要があります。

2.3.1.42 SQL 指令

IBM DB2 データベースのサポートを使用可能にする指令

Net Express:

この指令は DB2 指令で置き換えられました。

移行:

以前に SQLDB 指令または SQL 指令を使用していた箇所で DB2 指令を使用します。Workbench には、SQL 指令の新しい形式と古い形式があり、これらは同等です。 たとえば、SQLDB(SAMPLE) と SQL(DB=SAMPLE) は同等です。Net Express の DB2 指令では新しい形式を使用します。 たとえば、DB2(DB=SAMPLE) です。 DB2 コンパイラ指令の詳細については、オンラインマニュアル『***データベースアクセス***』にある『*DB2*』の章を参照してください。

2.4.1.43 ウィンドウ化サポート (テキストベース)

テキスト文字を使用して画面上のボックスを作成する COBOL 構文

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、Visual Object COBOL V1.0

Net Express:

下位互換性を確保するために、変更されないまま Net Express に含まれています。新しいアプリケーションには推奨できないため、下位互換性のヘルプに説明があります。

移行:

変更は必要ありません。

2.4.1.45 XM

640 KB を超えるメモリで DOS アプリケーションを実行するユーティリティ

この機能を含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

使用できません。

Net Express で使用する機能:

同等の機能はありません。DOS のみに適用されます。

第 3 章 : 16 ビット製品からの移行

16 ビットシステムの Object COBOL V3.4 または Workbench V3.4 から 32 ビットシステムの Net Express に移行する場合は、ここで説明する内容を考慮してください。

16 ビットの Dialog System V2.5 から移行する場合は、オンラインマニュアル『*Dialog System ユーザガイド*』を参照してください。

3.1 機能説明

機能はアルファベット順に説明します。 説明されていない機能については、変更する必要はありません。

3.1.1 機能リスト

3.1.1.1 PC_WIN... プリンタルーチン

プリンタを処理するライブラリルーチン

Net Express には、このルーチンはありません。 プリフィックス PC_PRINTER が付く同等の汎用ルーチンを呼び出すためには、コードを変更する必要があります。

3.1.1.2 番号による呼び出しルーチン - x"91" ファンクション 69

ディレクトリを検索するライブラリルーチン

32 ビットの Windows でパラメータ「*」を使用すると、拡張子のついた名前は検索されません。一方、16 ビットの Windows では検索されます。

3.1.1.3 番号による呼び出しルーチン - x"d0"

メモリを割り当てるライブラリルーチン

番号による呼び出しルーチン x"d0" に取ってかわる CBL_ALLOC_MEM は、優先して使用する必要があります。

番号による呼び出しルーチン x"d0" で割り当てたメモリは、この製品では必ず空白文字で初期化されます。 16 ビット COBOL システムの場合は、DEFAULT-BYTE 指令を使用してデフ

ォルトの FILLER バイトをデフォルト値 (空白文字) 以外の値に設定することで、x"d0" によるメモリ内容は、新しい値で初期化されます。この動作はマニュアルには記述されていませんが、プログラムはこの初期化処理を利用できます。ただし、ランタイムシステムの初期化処理に頼ることなく、x"d0" によるデータを明示的に初期化してください。

3.1.1.4 名前による呼び出しルーチン - CBL_CHANGE_DIR

カレントディレクトリを変更するライブラリルーチン

32 ビットの Windows の場合は、ディレクトリ指定でドライブ名を指定することで、現在のドライブも変更できます。16 ビットの Windows の場合には、現在のドライブは変更できません。

3.1.1.5 仮想カラーマップを処理する名前による呼び出しルーチン

16 ビット製品の場合:
次の 3 つの、名前による呼び出しルーチンは 16 ビット COBOL システムでのみ使用できます。

CBL_SCR_CREATE_VC	代替のカラーマップを作成する
CBL_SCR_ALLOCATE_VC_COLOR	色を仮想カラーマップに格納する
CBL_SCR_DESTROY_VC	システムカラーマップを元に戻す

3.1.1.6 CCI ファイル名

32 ビット製品では、ユーザレベルの CCI ファイル名が異なり、次のように表記されます。

cci*xx*32.lib
cci*xx*32.dll

xx は、サポートされているプロトコルを表し、次のどれかになります。

TC	TCP/IP (CCITCP)
NB	NetBEUI (CCINETB)
IX	Novell IPX (CCIIPX)
DE	動的データ交換 (CCIDDE)

CCITP プロセス登録デーモンが使用可能な環境では、このモジュールは、**ccitcp2.exe** として提供されます。

3.1.1.7 コンパイラ指令

16 ビットの COBOL システムで使用可能な次のコンパイラ指令は、Net Express を含む 32 ビットの Object COBOL システムでは必要ありません。 ただし、コンパイラは「Ignored」というメッセージを発行し、処理を続行するため、既存のコードからこれらの指令を削除する必要はありません。

64KPARA	MASM	SEGCROSS
64KSECT	MODEL	SEGSIZE
AUXOPT	OPTSIZE	SIGNCOMPARE
DATALIT	OPTSPEED	SMALLDD
EANIM	PARAS	TABLESEGCROSS
EXPANDDATA	PROTMODE	TRICKLECHECK
FIXING	REGPARM	

次の指令は、16 ビットと 32 ビットの両方で使用できますが、本来 16 ビットで使用するものです。そのため、Net Express のコードからこれらを削除する必要があります。

01SHUFFLE
CHIP
FLAG-CHIP

指令 GNT は .gnt コードを作成するために使用します。 16 ビットの指令 OMF"GNT" もサポートされており、同じ効果があります。

ASMLIST 指令はサポートされていますが、動作が異なります。

TARGET 指令はサポートされていますが、16 ビットに属する引数を受け付けません。

TRICKLE 指令はサポートされていますが、32 ビットコンパイラはプログラムが遅延している場合にも動作することができるため、通常は必要ありません。

3.1.1.8 LITLINK 呼び出し

「_」(二重アンダスコア) プリフィックスが記述された呼び出しは、呼び出し規約 8 を使用するために再コードする必要があります。LITLINK 呼び出しを強制する場合は、この方法を使用してください。 二重アンダスコアの規約を使用して LITLINK 呼び出しを行う場合は、プログラムのコンパイルには LITLINK(2) コンパイラ指令を使用する必要があります。

呼び出し規約 8 を使用すると、個々の呼び出しを静的にリンクできます。CALL *literal* 文にこの呼び出し規約を使用すると、コンパイラによりこの定数が外部シンボルとして宣言されるため、実行時ではなく、リンク時に処理されます。この呼び出し規約は、他の呼び出し規約と組み合わせることができます。たとえば、呼び出し規約 10 は、スタックからパラメータを削除する副プログラムに対して静的リンク呼び出しを行います。

3.1.1.9 Panels Version 2

Panels Version 2 を使用して描画された 3D オブジェクトを Net Express で表示した場合は、16 ビット製品で表示した場合よりも小さく見えます。Net Express では、16 ビット製品で使用されるパブリックドメイン DLL のかわりに、Windows 95 または Windows NT のネイティブの 3D 効果を使用します。16 ビット製品では、エントリフィールド、リストボックス、コンボボックスなどのコントロールの外側に 3D の窪んだ境界が描かれます。ネイティブの効果では、この境界がコントロールの内側に描かれるため、表示されるコントロールが小さく見えます。

3.1.1.10 実行時の構成

ランタイムスイッチに加えて、Net Express では、実行時に構成可能な次の要素を使用して特定のランタイム動作を構成することもできます。

- 環境変数
- ランタイムチューナ

3.1.1.11 ランタイムスイッチ

16 ビットの COBOL システムで使用可能な次のランタイムスイッチは、Net Express では使用不能であるか、または、別の意味をもっています。

A*	K1	L5	S6
B2*	K2	L6	/h
C*	K3	M*	/s
C1	L	M2	/m
C2	L1	P	/p
F1	L2	P1	
G	L3	S2	
I1	L4	S3	

* Net Express では、これらのスイッチの意味は異なります。

アプリケーションでこれらのスイッチを明示的に読み込む場合、または設定する場合は、従属関係にある部分も削除する必要があります。

第 4 章 : COBOL システムライブラリルーチン

ここでは、COBOL システムライブラリルーチンと Object COBOL や Workbench の相違について説明します。

16 ビットシステムと 32 ビットシステムの相違については、『*16 ビット製品からの移行*』の章を参照してください。

4.1 ルーチンの説明

ルーチンはアルファベット順に説明します。記述されていないルーチンについては、変更する必要はありません。各項目で使用される副見出しは、次のとおりです。

- **このルーチンを含む製品** - 説明するルーチンを含む Net Express 以前の Micro Focus 製品を列挙します。
- **Net Express** - 説明するルーチンが Net Express に含まれているかどうか、および他の製品と比較して変更されているかどうかを説明します。
- **移行** - 説明するルーチンを使用するアプリケーションを Net Express へ移行するために必要な変更について解説します。

4.1.1 ルーチンリスト

4.1.1.1 CBL_ALLOC_MEM

メモリを動的に割り当てるライブラリルーチン

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0

Net Express:

Net Express に組み込まれていますが、パラメータが変更されています。

移行:

Net Express ヘルプの索引 CBL_ALLOC_MEM を参照してください。

4.1.1.2 PC_TEST_PRINTER

プリンタの状態をテストするライブラリルーチン

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。 ただし、この名前をもつダミールーチンが含まれており、常にプリンタが使用不能であることを示す結果を返します。

移行:

プログラムを再コードします。 このルーチンの呼び出しを削除し、プリンタに WRITE 文を適用した後で、ファイル状態を確認します。

4.1.1.3 x"84" - DOS 割り込みの実行

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。

4.1.1.4 x"85" - メモリからのバイト読み取り

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。

4.1.1.5 x"86" - メモリへのバイト書き込み

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。

4.1.1.6 x"87" - ハードウェアポートからのバイト読み取り

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。ポートにアクセスするには、C ルーチンを作成するか、または、Win32API ルーチンを使用します。

4.1.1.7 x"88" - ハードウェアポートへのバイト書き込み

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。ポートにアクセスするには、C ルーチンを作成するか、または、Win32API ルーチンを使用します。

4.1.1.8 x"94" - メモリからのワード読み取り

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。

4.1.1.9 x"95" - メモリへのワード書き込み

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。

4.1.1.10 x"96" - ハードウェアポートからのワード読み取り

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。ポートにアクセスするには、C ルーチンを作成するか、または、Win32API ルーチンを使用します。

4.1.1.11 x"97" - ハードウェアポートへのワード書き込み

このルーチンを含む製品:

Object COBOL V3.4、Workbench V3.4

Net Express:

Net Express では、使用できません。このルーチンは DOS でしか機能しないため、Net Express では実装されていません。このルーチンを呼び出すと、RTS エラー 164 (「実行時サブプログラムが見つかりません」) が返されます。

移行:

Windows 95、Windows 98 および Windows NT V4.0 には、このルーチンによりアクセスできる機能と同等の機能がないため、アプリケーションを再設計する必要があります。ポートにアクセスするには、C ルーチンを作成するか、または、Win32API ルーチンを使用します。

第 5 章 : クラスライブラリ

ここでは、クラスライブラリルーチンと Object COBOL や Workbench の相違について説明します。

Visual Object COBOL V1.0 のクラスライブラリと同様に、Net Express のクラスライブラリでは Win32 API を使用します。Object COBOL と Workbench のクラスライブラリでは、Panels V2 を使用します。 いくつかの相違は、この変更によるものです。

5.1 変更点のまとめ

相違点は、次のとおりです。

- GetWidth、GetHeight、および GetRectangle など、ウィンドウの寸法を返すメソッドには、寸法の装飾 (境界やタイトルバーなど) が含まれます。 Object COBOL と Workbench の場合は、クライアント領域のみが含まれます。 Net Express では、GetClientWidthHeight のような新しいメソッドを使用してクライアント領域の寸法を取得できます。
- クラス VirtualWindows の場合は、スクロールバーのページ増分は自動的に計算されます。 Object COBOL と Workbench の場合は、増分を渡す必要があります。 また、Net Express では、SetVirtualWidthHeight メソッドを使用すると、ウィンドウの仮想サイズを何度も変更することができます。 Object COBOL と Workbench の場合には、変更は 1 回のみです。
- GUI オブジェクトのイベントは、イベントオブジェクトにより識別されます。 Object COBOL と Workbench の場合は、イベントが Panels V2 と通信する集団項目により識別されます。 オブジェクトを含めるには、各イベントについて object reference を宣言する必要があります。
- サイドファイルはサポートされません。ビットマップやアイコンなどを指定する場合は、文字列を指定するのではなく、リソースから読み込むことが必要です。

  コードを次のように変更します。 まず、クラス Module のメソッド NewZ に .dll ファイルの名前を含む文字列を渡します。次に、返されたモジュールオブジェクトとリソース ID を IconData などのリソースクラスに渡します。 その後で、ソースクラスに返されたインスタンスを元のメソッドに渡します。

- GUI モードの実行中に例外メッセージがメッセージボックスに表示されます。

第 6 章 : 指令と方言

指令を使用すると、コンパイラの動作とコンパイラが受け付ける COBOL 言語の詳細情報を数多く指定できます。ここでは、以前の COBOL システムから変更されたデフォルト設定について説明します。

また、ここでは、DIALECT 指令を含む IBM メインフレームからの移行に役立つ指令についても説明します。DIALECT 指令は、Workbench の USE 指令と **wb*.dir** ファイルに取ってかわります。

SQL に関連した指令については、『*埋め込み SQL アプリケーション*』の章を参照してください。

6.1 デフォルト

Net Express のすべてのデフォルト指令設定は、Object COBOL V3.4、Workbench V3.4、Object COBOL V4.0、Workbench V4.0、および Visual Object COBOL V1.0 のデフォルト指令設定と同じです。

6.2 除外された指令

次の指令は Net Express にはありません。

AMODE BROWSE CONVERTPTR EXTINDEX OLDSTRSUB RDEFPTR STRUCT XNIM

次の指令は使用できますが、Net Express では意味がないため何も実行されません。

INCLUDE-FILLER MFOO

6.3 変更された指令

- Net Express でソースコードの $SET に NODBSPACE CONVSPACE を指定しても、ファイルハンドラは DBCS 空白文字 x"8140" を x"2020" には変換しません。この変換を実行させるには、プロジェクト設定またはコマンド行でこの指定を行う必要があります。

  ソースファイルを開く前に、COBSW=+Z を設定する必要があります。

- Micro Focus 製品の以前のバージョンには、CONNECT 機能を実行する SQLINIT モジュールまたは SQLINI2 モジュールが含まれています。IBM が EXEC SQL CONNECT 文をサポートするため、これらのルーチンは提供されなくなりました。EXEC SQL CONNECT 文には、SQLINIT モジュールで使用できるオプションよりも多くのオプションがあります。

INIT 指令には、アプリケーションが異常終了した場合でも、データベース接続を正しく終了させる追加オプションがあります。これにより、データベースの破損を防ぎます。アプリケーションが異常終了した場合には、最後に行った COMMIT 以降の変更はすべてロールバックされます。このデータベース保護は、DB2 コンパイラ指令で INIT=PROT オプションを指定することで選択できます。

INIT=PROT オプションは、1 つのアプリケーションに対して一度しか設定できません。他の SQL プログラムから呼び出された SQL プログラムには INIT=PROT オプションを設定できません。そのかわりに、実行単位で最初に実行される SQL プログラムに INIT=PROT オプションを指定できます。INIT=PROT オプションをもつアプリケーションで複数のモジュールをコンパイルすると、プログラムが異常終了する可能性があります。

6.4 メインフレーム指令

Net Express でコンパイラを IBM のメインフレームコンパイラと同様に動作させる指令は、次のとおりです。これらの指令は Visual Object COBOL V1.0 では使用できませんが、Workbench V3.4 と Workbench V4.0 で使用できます。詳細は、Workbench に添付されているマニュアルで説明されています。

ADV CMPR2 COBOL370 DIALECT DOSVS DYNAM DYNAMICFD FLAGMIG FP-ROUNDING HOST-NUMCOMPARE LIBRARIAN MAPNAME ODOOSVS OLDCOPY OSVS PANVALET PROGID-COMMENT PROTECT-LINKAGE RDW SAA TRACE VSC2 ZWB

6.5 DIALECT 指令

Workbench では、指令ファイル (**wb*.dir** ファイル) が提供されます。このファイルには、コンパイラに適合する特定の方言で COBOL が受け付けられるようにする指令セットが格納されます。使用するファイルは、USE(*filename*) 指令により指定できます。 そのため、元々 Micro Focus COBOL システム以外 (特に IBM メインフレーム) で使用するために作成されたプログラムを簡単に移行できます。

Net Express では、これらのファイルと USE 指令のかわりに DIALECT 指令を使用します。DIALECT 指令に方言を指定すると、必要な指令が自動的に設定されます。

6.5.1 定義

DIALECT 指令は、次のように定義されます。

方言

指定された方言がコンパイラで受け付けられるようにします。その方言に対する実行時とコンパイル時の動作が適切であるように他の指令を設定します。

構文:

>>----------DIALECT(*dialect*)-------><

パラメータ:

dialect 指定可能な値:ANS85 COBOL370 DOSVS ISO2000 MF OSVS SAA1 SAA2
VSC21 VSC22 VSC23 VSC24

プロパティ:

デフォルト:NODIALECT

$SET: initial

注釈

NODIALECT は、明示的に設定しません。DIALECT(*dialect*) は、他の Micro Focus COBOL システムの USE(WB*dialect.dir*) に相当します。次のようにパラメータをさまざまに指定し、指令を設定します。

- DIALECT(ISO2000)

  NOANS85 NOCOBOLDIR COMS85 COPYLBR NODBCHECK NODBCS NODBSPACE FLAG"ISO2000" FLAGCD"W" NOFLAGSTD FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" INTLEVEL"4" ISO2000 NOMF NOMFCOMMENT NESTCALL NOOPTIONAL-FILE NORESEQ WARNING"3" ZEROLENGTHFALSE

- DIALECT(ANS85)

  ANS85 NOCOBOLDIR COMS85 COPYLBR NODBCHECK NODBCS NODBSPACE NOFLAG FLAGCD"W" FLAGSTD"H C2 D2 S2 R O" FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" INTLEVEL"2" NOISO2000 NOMF NOMFCOMMENT NESTCALL NOOPTIONAL-FILE NORESEQ WARNING"3" ZEROLENGTHFALSE

- DIALECT(VSC24)

  APOST AREACHECK ARITHMETIC"VSC2" ASSIGN"EXTERNAL" NOBOUND BYTEMODEMOVE CHARSET"EBCDIC" CHECKDIV"VSC2" COBFSTATCONV NOCOBOLDIR COMS85 COPYEXT"CPY,CBL" COPYLBR DBCS"2" DBCSSOSI"14""15" DEFAULTBYTE"0" NODYNAM FDCLEAR FLAG"VSC2" NOFLAGAS"S" FLAGCD"W" FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" FP-ROUNDING"VSC2" HOST-NUMCOMPARE"1" IBMCOMP INDD"SYSIN 80 L A" MAPNAME NOMF NOMFCOMMENT NATIVE"EBCDIC" NESTCALL ODOSLIDE NOOPTIONAL-FILE OSEXT"CPY"

OUTDD"SYSOUT 132 L A" PERFORM-TYPE"VSC2" NOQUOTE RECMODE"VSC2" RTNCODE-SIZE"2" NOSEG SIGN"EBCDIC" STICKY-LINKAGE"2" NOTRUNC TRUNCCOPY"8" VSC2"4" WARNING"3" ZEROLENGTHFALSE ZWB

- **DIALECT(VSC23)**

  DIALECT(VSC24) と同じですが、次の点が異なります。VSC2"3"

- **DIALECT(VSC22)**

  DIALECT(VSC24) と同じですが、次の点が異なります。NOANS85 COMS85 COMP NESTCALL RECMODE"OSVS" STICKY-PERFORM NOTERMPAGE VSC2"2"

- **DIALECT(VSC21)**

  DIALECT(VSC24) と同じですが、次の点が異なります。NOANS85 COMS85 NESTCALL RECMODE"OSVS" STICKY-PERFORM NOTERMPAGE VSC2"1"

- **DIALECT(OSVS)**

  NOANS85 APOST AREACHECK ARITHMETIC"OSVS" ASSIGN"EXTERNAL" NOBOUND BYTEMODEMOVE CHARSET"EBCDIC" CHECKDIV"OSVS" COBFSTATCONV NOCOBOLDIR COPYEXT"CPY,CBL" COPYLBR NODBCHECK NODBCS NODBSPACE DEFAULTBYTE"0" NODYNAM FDCLEAR FLAG"OSVS" NOFLAGAS"S" FLAGCD"W" FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" FP-ROUNDING"OSVS" HOST-NUMCOMPARE"1" IBMCOMP INDD"SYSIN 80 L A" MAPNAME NOMF NOMFCOMMENT NATIVE"EBCDIC" ODOOSVS ODOSLIDE NOOPTIONAL-FILE OSEXT"CPY" OSVS OUTDD"SYSOUT 132 L A" PERFORM-TYPE"OSVS" NOQUOTE RDW RECMODE"OSVS" REPORT-LINE"132" RTNCODE-SIZE"2" SIGN"EBCDIC" STICKY-LINKAGE"2" STICKY-PERFORM TRACE NOTRUNC TRUNCCOPY"8" WARNING"3" ZWB

- **DIALECT(DOSVS)**

  DIALECT(OSVS) と同じですが、次の点が異なります。DOSVS FLAG"DOSVS"

- **DIALECT(COBOL370)**

  APOST AREACHECK ARITHMETIC"VSC2" ASSIGN"EXTERNAL" NOBOUND BYTEMODEMOVE CHARSET"EBCDIC" CHECKDIV"COBOL370" COBFSTATCONV COBOL370"2" NOCOBOLDIR COMS85 COPYEXT"CPY,CBL" COPYLBR DBCSSOSI"14""15" DEFAULTBYTE"0" NODYNAM FDCLEAR FLAG"COBOL370" NOFLAGAS"S" FLAGCD"W" FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" FP-ROUNDING"COB370" HOST-NUMCOMPARE"1"

IBMCOMP INDD"SYSIN 80 L A" MAPNAME NOMF NOMFCOMMENT NATIVE"EBCDIC" NESTCALL ODOSLIDE NOOPTIONAL-FILE OSEXT"CPY" OUTDD"SYSOUT 132 L A" PERFORM-TYPE"COB370" NOQUOTE RTNCODE-SIZE"2" NOSEG SIGN"EBCDIC" STICKY-LINKAGE"2" NOTRUNC TRUNCCOPY"8" WARNING"3" ZEROLENGTHFALSE ZWB

- DIALECT(SAA2)

  AREACHECK NOCOBOLDIR COMS85 COPYLBR NODATE DBCS"2" DEFAULTBYTE"0" FLAG"SAA" FLAGCD"W" FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" IBMCOMP NOMF NOMFCOMMENT NESTCALL ODOOSVS ODOSLIDE NOOPTIONAL-FILE RTNCODE-SIZE"2" SAA"2" NOSEG WARNING"3" ZEROLENGTHFALSE

- DIALECT(SAA1)

  DIALECT(SAA2) と同じですが、次の点が異なります。NOANS85 NODATE SAA"1"

- DIALECT(MF)

  ANS85 NOAMODE NOAPOST NOAREACHECK ARITHMETIC"MF" ASSIGN"DYNAMIC" BOUND NOBYTEMODEMOVE CHARSET"ASCII" CHECKDIV"ANSI" NOCOBFSTATCONV NOCOBOL370 NOCOMP NOCOMS85 NOCOPYLBR DBCHECK DBCS"3" NODBCSSOSI DBSPACE DEFAULTBYTE"32" NODG NODOSVS DYNAM NOFDCLEAR NOFLAG NOFLAGAS NOFLAGCD NOFLAGSTD NOFOLDCALLNAME NOFOLDCOPYNAME NOFP-ROUNDING NOHOST-NUMCOMPARE NOIBMCOMP NOINDD NOMAPNAME MF"11" MFCOMMENT NOMS NATIVE"ASCII" NONESTCALL NOODOOSVS NOODOSLIDE OPTIONAL-FILE NOOSVS NOOUTDD PERFORM-TYPE"MF" QUOTE NORDW RECMODE"F" REPORT-LINE"256" RESEQ NORM RTNCODE-SIZE"4" NOSAA SEG SIGN"ASCII" NOSTICKY-LINKAGE NOSTICKY-PERFORM TERMPAGE NOTRACE TRUNC"ANSI" NOTRUNCCOPY NOVSC2 WARNING"1" NOXOPEN NOZEROLENGTHFALSE NOZWB

6.6 DB2 PASS コンパイラ指令

ユーザ ID とパスワードを指定する PASS 指令は、Net Express DB2 ECM インターフェイスには使用されません。

アプリケーションを DB2 ECM を含む DB2 データベースに接続する必要がある場合には、コンパイラはユーザ ID とパスワードを共通モジュール **MFDB2CON** から取得します。ユーザ ID とパスワードが必要な場合には、このモジュールはグラフィカルウィンドウを表示してユーザ ID とパスワードの入力を要求します。

ユーザ ID とパスワードを保存するオプションがあるため、同じデータベースを使用してプログラムを次回コンパイルする際にユーザ ID とパスワードの入力が要求されることはありません。クライアントマシンをリブートすると、保存されたこれらの情報は失われます。保存されたこれらの情報は、Net Express のコマンドプロンプトで次のコマンドを入力して削除することもできます。

```
MFDAEMON CLOSE
```

DB2 PASS 指令が指定されない場合には、DB2 INIT 指令は空白の LOGON ID と PASSWORD ホスト変数を生成しなくなりました。これは、CONNECT 文で空白または LOW-VALUE が指定されたホスト変数が渡されても DB2 UDB が SQL エラーを生成しないためです。これらの指令設定でコンパイルされたプログラムは動作しません。プログラムではこれらの指令を使用しないで、かわりに SQL CONNECT 文をプログラムに追加することを強くお奨めします。この動作の変更は、FixPack 8 以降の DB2 UDB V7.2 に影響します。

第 7 章 : 埋め込み SQL アプリケーション

ここでは、OpenESQL と以前の埋め込み SQL システムとの相違点を説明します。

7.1 COBSQL

COBSQL は、他の Micro Focus 製品での使用方法と同じく、Oracle、Sybase、および Informix で提供される埋め込み SQL プリプロセッサで使用できます。さらに、ODBC を使用して埋め込み SQL をサポートする OpenESQL を提供します。アプリケーションの要件によっては、OpenESQL へ移行する利点があります。

ここでは、OpenESQL の実装と他の埋め込み SQL の実装との相違を説明します。

7.1.1 変更のまとめ

Oracle の ESQL でサポートされている PIC x VARYING 構文が OpenESQL ではサポートされません。

7.2 Embedded SQL Toolkit

Embedded SQL Toolkit for Microsoft SQL Server は Net Express では使用できません。かわりに、Net Express で ODBC ベースの埋め込み SQL をサポートする OpenESQL を使用します。

ここでは、OpenESQL と Embedded SQL Toolkit for Microsoft SQL Server の相違について説明します。

7.2.1 変更のまとめ

SQL のコンパイラ指令には、新しい形式を使用する必要があります。ただし、Embedded SQL Toolkit の個別の指令と同じオプションを使用する指令 SQL が 1 つあります。次に例を示します。

```
SQL(MSSQL) NOSQLDB NOSQLPASS NOSQLACCESS
```

この場合は、次のようになります。

```
SQL(DBMAN=ODBC,TARGETDB=MSSQLSERVER,NOACCESS)
```

(NODB と NOPASS はデフォルトにより設定されます)

たとえば、

```
SQL(MSSQL) SQLDB(server.database)
    SQLPASS(user.pwd) SQLINIT
```

この場合は、次のようになります。

```
SQL(DBMAN=ODBC,TARGETDB=MSSQLSERVR, DB=datasourcename.db,
    PASS=user.pwd,INIT=PROT)
```

たとえば、

```
SQL(MSSQL)SQLPROT
```

この場合は、次のようになります。

```
SQL(DBMAN=ODBC,TARGETDB=MSSQLSERVER, INIT=PROT)
```

その他の相違は、次のとおりです。

- デフォルトでは、OpenESQL は ANSI SQL 92 規格に従ってトランザクションを管理します。トランザクションを開始するために、BEGIN TRAN 文は必要ありません。新しいトランザクションは、CONNECT、COMMIT、または ROLLBACK 文の直後に開始します。Toolkit との互換性を提供するには、ESQLVERSION 指令または AUTOCOMMIT 指令のどちらかを使用します。
    - AUTOCOMMIT は手動トランザクションモードを無効にします。
    - ESQLVERSION は AUTOCOMMIT を設定し、Toolkit バージョンに基づいて追加の互換性オプションを設定します。

- SQL Server の bit データ型はサポートされていません。
- BROWSE モードのカーソルはサポートされていません。このカーソルは、上位レベルでサポートされます。BROWSE モードの構文は、可能な限り新しいカーソルサポートにマップされます。
- ストアドプロシージャでは、EXECUTE IMMEDIATE を実行できません。そのため、EXEC SQL EXEC *stored_procedure* END-EXEC を使用してください。
- 一部のエラーメッセージのテキストが変更されています。SQLCODE と SQLSTATE は以前と同様です。
- SQL Server CHAR 列を埋め込み SQL INTEGER フィールドに返すと、別の結果が得られます。
- カーソルは、開いた (OPEN) ままではなく、COMMIT の実行時に閉じます。新しい動作は ANSI 実装と同じです。
- 空トランザクションに COMMIT を実行した場合には、エラー状態は返されません。
- COMPUTE 文はカーソル宣言では使用できません。
- DATETIME フィールドは別の形式で返されます。Embedded SQL Toolkit と同じ形式で挿入することもできますが、データは YYYY-MM-DD 形式で返されます。ODBC ドライバでは、この形式しかサポートされていません。
- カーソル宣言で DISTINCT を使用すると、カーソルが読み取り専用になります。
- このリリースでは、サイズ 1 の FETCHBUFFER しかサポートされていません。

- 存在していない行に DELETE を指定すると、SQLCODE = 100 (「結果の末尾」) ではなくエラー状態になります。
- SCROLLOPTION MIXED*n* はサポートされていません。
- バイナリデータでマーカーとして「??」を使用すると、下位互換性が確保されます。 新しいアプリケーションでは、このマーカーを使用しないでください。

第 8 章 : Dialog System V2.5

ここでは、Net Express の Dialog System に移行する場合の注意点について、Dialog System V2.5 と比較しながら説明します。

Windows 95 または Windows NT の 32 ビットのグラフィック Dialog System V2.5 から移行する場合には、移行に関する問題はありません。互換性のない小さな変更が 1 つあります。次の『*DSRUN*』を参照してください。非グラフィックバージョンおよびほとんど使用しない一部のツールは削除されています。ソースコードテンプレートが用意されおり、スピンボタン、ステータスバー、OCX コントロール、およびツリービューコントロールの新しいクラスライブラリサポートを使用できます。

定義ソフトウェアへは、IDE の [**ツール**] メニューからアクセスします。

16 ビット製品から 32 ビット製品に移行する場合と OS/2 などの他のオペレーティングシステムから移行する場合は、オンラインマニュアル『*Dialog System ユーザガイド*』にある『*異なるプラットフォームへの移行*』の章を参照してください。

8.1 DSRUN

グラフィック Dialog System ランタイムモジュールは、DSGRUN と呼ばれます。下位互換性のために提供されていますが、古い DSRUN モジュールは全面的にはサポートされなくなりました。アプリケーション内で DSRUN モジュールを呼び出す箇所では、DSGRUN モジュールを呼び出すように変更する必要があります。その他の変更は必要ありません。

DSRUN モジュールが Dialog System の文字バージョン用にのみ提供されているため、DSRUN モジュールを呼び出したときにエラーが発生する場合があります。Dialog System の文字バージョンは、Net Express の設定で UNIX オプションを選択するとインストールされます。

8.2 注意点

- 文字モード Dialog System は含まれていません。

  Net Express の Dialog System は、グラフィック Dialog System V2.5 で使用するものと同様のスクリーンセット形式を使用します。このスクリーンセット形式は、文字 Dialog System V2.5 で使用するスクリーンセット形式とは異なります。文字スクリーンセットをグラフィック Dialog System 用に作成し直す必要があります。

  文字 Dialog System は、Net Express の UNIX オプションに含まれていますが、UNIX に展開するアプリケーションを作成するのみです。

- エミュレーションモード Dialog System は含まれていません。

Net Express の Dialog System は、エミュレーションモード Dialog System V2.5 で作成されたスクリーンセットをロードできます。スクリーンセットは、変更することなく継続して使用できます。 表示はグラフィック表示となります。

- Panels Version 2 については、原則的に、Net Express では説明しません。

  ただし、Net Express では、Panel Version 2 がサポートされており、これを使用したアプリケーションを実行できます。新規アプリケーションで Panel Version 2 と同様の機能が必要な場合は、クラス ライブラリを使用することをお奨めします。詳細については、『*Panels V2*』の章を参照してください。

- 下位互換性を確保するために、ファイル **dsclnk32.bat** が含まれています。

  ただし、Dialog System アプリケーションのプロジェクトの基礎としては、Net Express プロジェクトテンプレートが使用されます。 このため、将来に備えてこのテンプレートに移行する必要があります。16 ビットの **dslink.bat** から 32 ビットに移行する場合は、『*Dialog System ユーザガイド*』を参照してください。

第 9 章 : Panels V2

Net Express には、下位互換性を確保するために Panels V2 がそのまま組み込まれていますが、新規アプリケーションにはお奨めできないので説明を記載していません。 Dialog System を通じて明示的にアクセスできる特定の機能については、Dialog System のマニュアルで説明されています。

Panels V2 を使用して作成された既存のアプリケーションをさらに開発する場合は、Panels V2 による既存のコードを維持しながら、オブジェクト指向 (OO) クラスライブラリを使用することをお奨めします。 ここでは、この方法について説明します。

9.1 方法

ここでは、サンプルプログラム **Net Express\Base\Demo\Pan2oo\Pan2oo.cbl** を参照してください。 このサンプルプログラムでは、Panels V2 とクラスライブラリをともに使用する方法を説明します。 ウィンドウの作成には Panels V2 が使用され、そのウィンドウにボタンを作成するために、クラスライブラリを使用するコードが追加されています。 ボタンを追加するには、次のコードを使用します。

- Panels2 の追加呼び出し
- 追加構文
- Object COBOL コード

次では、**Pan2oo.cbl** のコードについて説明します。

9.1.1 Panels V2 の初期化

クラスライブラリを使用するために Panels V2 に次の初期化コードが追加されています。

```
* 新規呼び出し - クラスライブラリを初期化します。
     move Pf-Load-Class-Library to P2-Function
     call PANELS2 using P2-Parameter-Block.
```

この呼び出しにより、Panels V2 に GUI クラスライブラリが読み込まれます。 Panels V2 とクラスライブラリをともに使用するには、この呼び出しを行う必要があります。 その結果、クラスライブラリは、Windows イベント処理を制御できるようになります。

9.1.2 OO オブジェクトの作成

次のコードは、GUI クラスライブラリで使用するオブジェクトのクラスとファイル名を宣言するコードです。

```
 class-control.
     window is class "awindow"
```

```
     pushbutton is class "pushbutt".
```

次のコードでは、使用する各オブジェクトの変数を宣言します。

```
 working-storage section.
 01 aWindow object reference.
 01 aButton object reference.
```

次のコードでは、ウィンドウのプッシュボタンを作成します。

```
* このハンドルのクラスライブラリオブジェクト参照を取得します。
     invoke window "fromhandle"
         using window-handle returning aWindow
* ウィンドウに新しいプッシュボタンを作成します。
     invoke pushbutton "new"
         using aWindow returning aButton
     move 10 to button-x
     move 10 to button-y
     move 300 to button-w
     move 100 to button-h
     invoke aButton "setRectangle"
         using button-x button-y button-w button-h
     invoke aButton "setLabelZ" using z"Test"
     invoke aButton "Show"
```

9.1.3 イベント処理

次のコードでは、エントリポイントを宣言し、特定のイベントが発生するたびに GUI クラスライブラリがエントリポイントを呼び出すように設定します。

```
 78 BC-Entry "buttonClicked".
*  「button clicked」イベントに関する情報を検索します。
     move p2ce-clicked to event-index
     invoke aButton "setEventToEntry"
         using event-index BC-Entry & x"00".
*  「button clicked」イベントに対するコールバックです。
* lnkEvent は、クラス「event」のオブジェクトです。
     ButtonClicked section.
     entry BC-Entry using lnkEvent.
     display "The button was clicked!"
     exit program.
```

第 10 章 : データツール

ここでは、COBOL Workbench から Net Express に移行する際に使い慣れているメソッドの変更が必要になるデータファイルの表示と編集との相違について説明します。

10.1 概要

データファイルエディタ、データファイルコンバータ、および Structure Editor は、Net Express に含まれています。これらは、機能拡張のために広範囲にわたって大幅に書き換えられています。

構造体ファイルは*レコードレイアウトファイル*と呼ばれ、Structure Editor は*レコードレイアウトエディタ*と呼ばれます。ユーザインターフェイスは設計し直され、有用性が改善しています。Workbench などの場合は、データファイルを開くたびにレコードレイアウトをロードできます。

そのかわりに、レコードレイアウトをデータファイルに関連付けることができます。このため、データファイルを使用するたびに、データファイルエディタは正しいレコードレイアウトを見つけます。このように関連付けるには、レコードレイアウトファイルをデータファイルと同じディレクトリに保存します。レコードレイアウトファイルの基本名をデータファイルの基本名と同じにし、拡張子は **.str** にします。もはや Structure Catalog は存在しません。

データファイルエディタにはコマンド行オプションはありません。索引ファイルと相対ファイルのデフォルトの作業方法がクイックエディットモードのため、このモードは明示的に設定する必要はありません。プルダウンメニューとツールバーボタンに加えて、編集したいレコードを右クリックするとポップアップメニューが表示されます。これらのコンテキストメニューは、ファイルタイプに関連するオプションのみを表示します。

「データファイルエディタ」ウィンドウに加えられた変更は次のとおりです。

- ウィンドウの下部にはボタンバーがありません。これらのボタンの機能は、サブメニューとコンテキストポップアップメニューに組み込まれました。
- ウィンドウの上部のツールバーは、従来の Windows 95 タイプのツールバーと同じになりました。
- フォーマット表示と非フォーマット表示は、2 つの個々のペインで利用できます。このため、レコードレイアウトファイルを作成している場合には、フォーマット表示と非フォーマット表示を同時に表示できます。
- 現在のカーソル位置などの情報を表示するステータス行は、Net Express のウィンドウの下部に表示されるようになりました。新しいステータス行には、他の追加情報も表示されます。この情報は、フォーマットペインまたは非フォーマットペインのどちらがハイライト表示されたかによって異なります。

10.2 詳細

ここでは、データツール機能が Net Express にない場合やデータツール機能が変更されている場合について説明します。機能を変更しないでそのまま使用できる場合については説明しません。このマニュアルでは拡張機能については説明しません。

10.2.1 レコードレイアウトのデータファイルとの関連付け

レコードレイアウトファイルとデータファイルを利用者定義のテキストラベルを使用して関連付けるデータツールの機能

Net Express:

Net Express では、レコードレイアウトファイルをデータファイルと同じディレクトリに保存します。レコードレイアウトファイルの基本名はデータファイルの基本名と同じにし、拡張子は **.str** にします。

10.2.2 データツールインターフェイス

データファイルエディタとデータファイルコンバータへのインターフェイス

Net Express:

Net Express は、データファイルの編集機能 (データファイルエディタ) へのインターフェイスとデータファイルの変換機能 (データファイルコンバータ) へのインターフェイスを引き続き提供します。

Net Express には、データファイルコンバータへの Workbench バッチインターフェイス **DFCONV** が含まれます。Net Express のコマンドプロンプトで **run dfconv** を使用します。Load、Save、または Convert プロファイルを指定するダイアログボックスは表示されなくなりました。このため、プロファイルをコマンド行のパラメータとして指定する必要があります。Load オプションと Save オプションは含まれていません。

10.2.3 Fileshare サポート

リモート Fileshare サーバを介してデータファイルを編集するデータファイルエディタの機能

Net Express:

データツールの Fileshare サポートが含まれていません。このサポートは、将来のリリースで検討されます。

10.2.4 IMS データベース

IMS データベース内のデータを表示および編集するデータファイルエディタの旧リリース機能

Net Express:

この機能は含まれていません。

10.2.5 プロファイルファイル

ファイルヘッダーの詳細を格納するファイル。このファイルは、データファイルエディタとデータファイルコンバータで使用します。

Net Express:

下位互換性を確保するために使用できます。

アプリケーションを移行する方法:

ファイルにヘッダーがない場合には、データファイルエディタはプロファイルファイル (**.pro**) が存在する場合はそのファイルを検索して該当する情報を使用します。

10.2.6 コピーブックからのレコードレイアウトの作成

コピーブックからレコードレイアウトを作成するデータファイルエディタの機能

Net Express:

コピーブックは含まれていません。レコードレイアウトはレコードレイアウトファイルに保存されます。プロジェクトビューで、データファイルエディタで使用するレコードレイアウトを保存している COBOL ソースファイルを選択するとレコードレイアウトファイルが作成されます。

10.2.7 データファイルの検索

データファイル内の検索を行うデータファイルエディタの機能

Net Express:

データファイルの検索には、4 つの方法があります。

- すべての種類のファイルでの文字列の検索と置換
- 索引ファイルのみのキー検索
- すべての種類のファイルでの指定した位置へのジャンプ
- フィールドへの移動

10.3 FAQ (よくある問題)

ここでは、移行する際に発生するよくある問題の解決策を説明します。

10.3.1 データファイルエディタの起動方法

メニュー項目からデータファイルエディタを起動できません。Net Express では、ファイルを開いたり作成したりすると、そのファイルの種類に適したエディタが自動的に起動されます。次のどれかの操作を実行すると、データファイルエディタが起動されます。

- 拡張子が **.dat** の PC ファイルを開く
- [**用途**] で [**データ**] を選択してファイルを開く
- [**ファイル**] メニューで [**新規作成**] をクリックしてから、「**データファイル**」を選択してデータファイルを新規に作成する

10.3.2 新規データファイルの作成方法

[**ファイル**] メニューで [**新規作成**] をクリックしてから、「**データファイル**」を選択します。

10.3.3 レコードレイアウトファイルの作成方法

プロジェクトビューで、適用したいレイアウトを保存している COBOL ソースファイルを右クリックします。次に、ポップアップメニューで [**レコードレイアウトの生成**] をクリックします。

10.3.4 編集するレコードレイアウトファイルを開く方法

[**ファイル**] メニューで [**開く**] を選択し、「**ファイルの種類**」フィールドに**レコードレイアウトファイル (.str)** を設定します。