Net Express

移行ガイド

Micro Focus NetExpress™

移行ガイド

Micro Focus®

第 3 版
1998 年 10 月

目次

第1章　はじめに

このマニュアルは、Micro Focus の旧製品から NetExpress 3.0 J に移行するユーザーを支援するためのものです。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。新規アプリケーションを作成する場合は、このマニュアルで説明する機能を使用しないでください。

このマニュアルでは、次の製品から NetExpress に移行する方法について説明します。

- COBOL V5.0 J
- COBOL V5.0 J with Host Emulation Support
- COBOL V4.0.00 J - V4.0.20 J
- COBOL V4.0.20 J with Host Emulation Support
- COBOL V3.1.10 J - V3.1.50 J 32ビット (Windows 95/98 および Windows NT)
- COBOL V3.1.00 J - V3.1.50 J 16ビット (DOS、OS/2 および Windows 3.1)
- Workbench V3.1.00 J - V3.1.50 J 16 ビット
- Embedded SQL Toolkit for Microsoft SQL V2.1
- Dialog System V2.5 J (16 および 32ビット)
- 日本語 COBOL/2 V1.1 16 ビット (DOS)
- LEVEL II COBOL V2.5.54 16 ビット (DOS)

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

この章の残りの部分では、NetExpress と旧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。

1.1 移行元の製品について

1.1.1 COBOL V4.0 J および COBOL V5.0 J

COBOL V4.0 J または COBOL V5.0 J から移行する場合、編集、コンパイル、デバッグには Animator Version 2 が使用されているはずです。Animator Version 2 は、NetExpress の統合開発環境 (IDE) 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高度化されています。

NetExpress 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、アプリケーションに対する作業時には常に読み込んだ状態にしておくことをお勧めします。

NetExpress には、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。

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だけですぐにソース 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 NetExpress 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。

NetExpress では、以前からあるすべての機能 (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 ファイルで使用することができます。マウスを 1 度クリックするだけでコンパイルとリンクをすべて処理することができます。この処理は、ビルドと呼ばれます (COBOL V4.0J と 5.0J 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、NetExpress のビルドと異なります)。ビルド中には、以前からあるユーティリティ 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 V4.0J と COBOL V5.0J では、文字モードのデバッガである Animatorが使用できましたが、NetExpress には Animator はありません。ただし、Animator Version 2 と同様に、IDE には Animator のほとんど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 COBOL V4.0.20 J with Host Emulation Support および COBOL V5.0 J with Host Emulation Support からの移行

「 COBOL V4.0 J と COBOL V5.0 J」の項で説明する内容は、すべてこれらの製品に適用されます。ただし、ホストエミュレーション サポート製品は、メインフレーム用のアプリケーション開発を目的としているため、開発支援用のツールやマニュアルは NetExpress には含まれていません。コンパイラは、特定のメインフレーム ダイアレクトをエミュレートすることができます。メインフレームの指令や wb*.dir ファイルの詳細については、付録「指令とダイアレクト」を参照してください。

1.1.3 COBOL V3.1 J 32 ビットからの移行

COBOL V3.1 J から移行する場合、以前使用していた PROCO や Animator のような文字モードのツールのかわりに、IDE のグラフィカル インターフェイスを使用することになります。

1.1.4 COBOL と Workbench V3.1 J 16 ビットからの移行

COBOL V3.1 J から移行する場合、以前使用していた PROCO や Animator のような文字モードのツールのかわりに、IDE のグラフィカル インターフェイスを使用することになります。詳細については、「16 ビット COBOL V3.1 J からの移行」の章を参照してください。

1.1.5 日本語 COBOL/2 V1.1 からの移行

COBOL/2 V1.1 から移行する場合、Animator などの文字モードのツールのかわりに、拡張された IDE を使用することになります。詳細は、「日本語 COBOL/2 V1.1 からの移行」の章を参照してください。

1.1.6 LEVEL II COBOL V2.5.54 からの移行

LEVEL II COBOL から移行する場合、Animator などの文字モードのツールのかわりに、拡張された IDE を使用することになります。詳細については、「LEVEL II COBOL V2.5.54 からの移行」の章を参照してください。

1.1.7 他の製品からの移行

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

1.2 コードの互換性

旧バージョンの Micro Focus COBOL 用に作成された有効なソース コードは、NetExpress でエラー無くコンパイルすることができます。

Micro Focus では、ソース コードの上位互換性を可能なかぎり保証していますが、ハードウェアやソフトウェアのプラットフォームが変更された場合には、古い機能をサポートできないことがあります。

次の表は、NetExpress と互換性のあるコードを示します。

製品	ソース	.INT	.IDY	.GNT	.OBJ
COBOL V5.0 J	あり	あり *	なし	なし	なし
COBOL V4.0 J	あり	あり *	なし	なし	なし

製品	ソース	.INT	.IDY	.GNT	.OBJ
COBOL V3.1 J	あり	あり	なし	なし	なし
COBOL V4.0 J/5.0 J with Host Emulation Support	あり	あり *	なし	なし	なし
COBOL V3.1 J 16 ビット	あり	あり	なし	なし	なし
Workbench V3.1 J 16 ビット	あり	あり	なし	なし	なし
COBOL/2	あり	なし	なし	なし	なし
LEVEL II COBOL	あり	なし	なし	なし	なし

* .int コードは、オブジェクト指向のコードを含まないかぎり互換性があります。

注記

- バグのある旧製品を使用して作成されたアプリケーションでは、コードの互換性は保証されません。
- COBOL システム ライブラリ ルーチンの互換性については、「COBOL システム ライブラリ ルーチン」を参照してください。

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、プロジェクトのリンク タイプを「文字」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。[文字] 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、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 (ADIS) 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

ロで割る場合や、ピクチャ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されたコード 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プロシージャを実行するには、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、GUI アプリケーションと Web アプリケーションの構造上の違い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 (サブ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、ユーザーが [送信] ボタンをクリックするとフォーム全体が Web サーバーに送信されます。その結果、Web サーバー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、回答としてユーザー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することができます。ActiveX コントロールは、Web ブラウザで実行され、Web サーバー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勧めしません。

前述のとおり、アプリケーションを再構築することをお勧めします。通常、アプリケーションを新しく設計すると、各ウィンドウやダイアログ ボックスを Web フォームに置換するだけではなく、ウィンドウやダイアログ ボックスよりも簡単な 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。NetExpress 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、オンライン マニュアル『インターネット アプリケーション』を参照してください。

第2章　32 ビット COBOL からの移行

この章では、32 ビット Micro Focus COBOL の旧製品を使用して作成されたアプリケーションの維持方法について説明します。具体的には、問題の発生原因となりうる相違点やこれらの問題の解決方法について説明します。説明する 32 ビット COBOL 製品は、次のとおりです。

- COBOL V5.0 J
- COBOL V5.0 J with Host Emulation Support
- COBOL V4.0 J - V4.0.20 J
- COBOL V4.0.20 J with Host Emulation Support
- COBOL V3.1 J - V3.1.50 J

注記

コンピュータに COBOL V3.1J、V4.0 J または COBOL V5.0J などの 32 ビット製品がインストールされている場合、NetExpress を使用するには、NetExpress の使用前に次の環境変数から旧製品への参照を削除する必要があります。

COBDIR、COBLANG、COBCPY、INCLUDE、LIB および PATH

NetExpress は、上記の規則にしたがうかぎり、Micro Focus の旧製品がインストールされているコンピュータで使用できるため、ハードディスクから旧製品を削除する必要はありません。

2.1 上位互換性

2.1.1 ソース コードとファイル データ

32 ビット COBOL 旧製品で作成された有効なソース コードとファイル データは、NetExpress に対して上位互換性をもっています。そのため、32 ビット COBOL 旧製品で作成されたソース プログラムやファイルを NetExpress に移行することができます。

アプリケーションで COBOL システム ライブラリ ルーチンを使用している場合、NetExpress への移行に問題が生じることがあります。詳細については、「付録 B COBOL システム ライブラリ ルーチン」を参照してください。

NetExpress は、旧 COBOL システムと異なり、小文字と大文字の DBCS ローマ字 COBOL データ項目を同様に処理します。そのため、同じ DBCS ローマ字変数名を大文字と小文字で別々に表記しても、これらを 2 つの別個の

変数名として使用することはできません。

Windows 95 と Windows 98 で DBCS 文字の印刷を可能にするためには、COBCONFIG ファイルで *printer_redirection* 実行時調整可能変数を TRUE に設定する必要があります。*printer_redirection* は、プリント マネージャ経由でプリンタのリダイレクトを行います。*printer_redirection* 実行時調整可能変数の詳細については、「COBOL システム ライブラリ ルーチン」を参照してください。

2.1.2 INT、GNT および OBJ コード

INT 形式にコンパイルしたアプリケーションは、Net Express で実行することができます。ただし、オブジェクト指向のコードが使用されている場合は、アプリケーションを再コンパイルする必要があります。NetExpress により作成される GNT 形式と OBJ 形式のファイルと 32 ビット COBOL システムで作成される GNT 形式と OBJ 形式のファイルとの間では互換性がありません。そのため、すべての GNT ファイルと OBJ ファイルは、このリリースを使用して再コンパイルする必要があります。

2.1.3 IDY ファイル

コンパイラ辞書構造体が修正されています。つまり、旧リリースで作成された IDY ファイルとこのシステムの IDE には、互換性がありません。そのため、このリリースのコンパイラを使用して、アニメートするプログラムをすべて再コンパイルする必要があります。

2.1.4 LBR ファイル

このリリースにより作成されたアプリケーションで使用する Micro Focus ライブラリ形式の (.LBR) ファイルについては、すべてリビルドすることをお勧めします。以前の .LBR ファイルには、上位互換性を持つものもありますが、必ずしも保証されているわけではありません。NetExpress 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 でのデバッグだけに使用されるファイルです。ライブラリ ファイルには、データ ファイルだけが格納されます。アプリケーションの再パッケージ化については、「付録 A 開発ツール」に記載された「Build」の項目を参照してください。

2.1.5 実行コード

Micro Focus COBOL の旧製品を使用して作成された実行可能ファイルは、すべてこのリリースを使用してリビルドする必要があります。

2.2 バッチ ファイル

2.2.1 コンパイル

2.2.1.1 ANIM 指令

ANIM 指令を指定するだけでは、自動的に .int ファイルを作成することはできません。.int ファイルを作成するためには、NOGNT 指令も指定する必要があります。

2.2.1.2 PROCO

32 ビット製品では、PROCO システムを通してプログラムをコンパイルすることができます。NetExpress には、PROCO システムがないため、次の表にしたがってバッチ ファイルを変更する必要があります。

どのバージョンの PROCO でも (PROCO.EXE、PROCOC.EXE または PROCOW.EXE)、コマンド行は同じであることに注意してください。

PROCO のコマンド行	NetExpress のコマンド行
PROCO CHECK filename.cbl [指令]	COBOL filename.cbl ANIM NOGNT [指令] ;
PROCO C filename.cbl [指令]	COBOL filename.cbl ANIM NOGNT [指令] ;
PROCO ANIMATE filename.int	MFNETX /debug:filename.int
PROCO A filename.int	MFNETX /debug:filename.int
PROCO ZOOM filename.int	MFNETX /debug:filename.int *
PROCO Z filename.int	MFNETX /debug:filename.int *
PROCO COMPILE filename.int [指令]	COBOL filename.int OBJ [指令] ;
PROCO CO filename.int [指令]	COBOL filename.int OBJ [指令] ;
PROCO GENERATE filename.int [指令]	COBOL filename.int GNT [指令] ;
PROCO G filename.int [指令]	COBOL filename.int GNT [指令] ;
PROCO RUN filename.int	RUN filename.int
PROCO R filename.int	RUN filename.int
PROCO ADISCF	RUN ADISCF

この場合、

[指令] には、オプションで 1 つのコンパイラ指令を指定するか、複数のコンパイラ指令を連続して指定します。

* デバッガが起動し、最初の実行文が実行されますが、プログラム全体はズームされません。

2.2.1.3 実行

NetExpress では、グラフィックの RUN トリガにより新しいプロセスが生成されます。そのため、バッチファイルが次のようなファイルで構成されている場合、これらのファイルは連続して呼び出されません。

```
RUN A.GNT
RUN B.GNT
```

この場合、バッチファイルを次のように修正する必要があります。

```
RUNC A.GNT
RUNC B.GNT
```

2.3 コンパイル

2.3.1 構文

2.3.1.1 オブジェクト指向プログラミング

Object-Storage Section は、データを継承するオブジェクト COBOL クラスで必要になります。

2.4 クラス ライブラリ

クラス ライブラリは、Micro Focus COBOL V4.0 J および 5.0 J の全バージョンで提供されます。ただし、Micro Focus COBOL V3.1 J では提供されません。

NetExpress のクラス ライブラリでは、Win32 API を使用します。一方、旧製品のクラス ライブラリでは、Panels V2 を使用します。この変更により生じる相違点は、次のとおりです。

- `GetWidth`、 `GetHeight`、および `GetRectangle` のようなウィンドウの寸法を返すメソッドには、寸法に関する修飾 (境界線やタイトル バーなど) も含まれます。旧製品の場合は、クライアント領域しか含まれません。NetExpress 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することができます。
- クラス `VirtualWindows` を使用すると、スクロール バーのページ増分を自動的に計算することができます。旧製品では、増分を渡す必要があります。また、NetExpress では、`SetVirtualWidthHeight` 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。旧製品では、仮想サイズは一度しか変更できません。

- 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。旧製品では、イベントが Panels V2 と通信するグループ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、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。

2.5 構成可能な属性バイト

NetExpress のデフォルトの属性バイトは、旧 32 ビット COBOL システムと同じで、次のようになります。

ビット

7	6	5	4	3	2	1	0
オーバーライン	反転表示	レフトライン	下線	輝度	赤の前景色	緑の前景色	青の前景色

旧 32 ビット COBOL システムの構成可能な属性バイト サポートを利用して作成したアプリケーションは、NetExpress で実行することができます。ただし、構成可能な属性バイト サポートの使用方法は、NetExpress では変更されています。旧製品では、ランタイム システムの初期化時に、実行されたプログラムと同じ名前をもつ .INI ファイルが検索されます。たとえば、COBOL.EXE を実行した場合、COBOL.INI が検索されます。また、PROCO.EXE を実行すると PROCO.INI が検索されます。検索されたファイルで属性節が正しく定義されている場合、デフォルトのかわりにこれらの設定が使用されます。一方、ランタイム システムは、.INI ファイルの内容が不十分であると判断した場合に MFRTS32.INI という名前のファイルを検索し、同様に内容を確認します。このファイルも不十分である場合は、デフォルトが適用されます。

次の構造体は、32 ビットの旧製品で .INI ファイルに属性情報を格納する方法について表したものです (指定されている値は、デフォルトの構成です)。

```
 [jpnatrconfig]
highlight=8
underline=16
reversevideo=64
blink=255
```

```
overline=128
leftline=32
```

上記のとおり、ビット位置を表すために 10 進値が使用され、バイト形式に適合しない属性を表すために 255 が使用されます。ビット位置の 0 から 2 までを再定義することはできません。

NetExpress の構成可能な属性バイトは、互換性を確保するためだけに設けられたものなので、新規開発では構成しないことをお勧めします。

2.5.1 NetExpress の構成可能な属性バイト

NetExpress では、ファイル COBOPT.CFG を使用して、次のように属性バイトを構成します。

```
set screen_attribute(index) = value
```

index には、定義する文字属性を指定します。

値	属性 ?
0	強調表示
1	下線
2	反転
3	点滅
4	オーバーライン
5	レフトライン
6	非強調表示 (淡色表示)

alue は、8 ビットの属性で、文字属性を表示するビットを定義します。

値	ビット
0	この属性は、8 ビット バイトでは使用できません。
8	ビット 3
16	ビット 4
32	ビット 5
64	ビット 6
128	ビット 7

例

```
set screen_attribute(0) = 8
set screen_attribute(1) = 16
set screen_attribute(5) = 32
set screen_attribute(2) = 64
set screen_attribute(4) = 128
```

第3章　16ビット COBOL V3.1 J からの移行

この章では、16 ビット Micro Focus COBOL V3.1 J 旧製品を使用して作成されたアプリケーションの維持方法について説明します。具体的には、問題の発生原因となりうる相違点やこれらの問題の解決方法について説明します。説明する 16 ビット COBOL 製品は、次のとおりです。

- COBOL V3.1 J - V3.1.50 J
- Workbench V3.1 J - V3.1.50 J
- Toolkit V1.2 J - V1.2.50 J

3.1 上位互換性

3.1.1 ソース コード

16ビット COBOL 旧製品で作成された有効なソース コードとファイル データは、NetExpress に対して上位互換性をもっています。そのため、16 ビット COBOL 旧製品で作成されたソース プログラムやファイルを NetExpress に移行することができます。

アプリケーションで COBOL システム ライブラリ ルーチンを使用している場合、NetExpress への移行に問題が生じることがあります。詳細については、「付録 B COBOL システム ライブラリ ルーチン」を参照してください。

NetExpress は、旧 COBOL システムと異なり、小文字と大文字の DBCS ローマ字 COBOL データ項目を同様に処理します。そのため、同じ DBCS ローマ字変数名を大文字と小文字で別々に表記しても、これらを 2 つの別個の変数名として使用することはできません。

DOS、OS/2 および Windows V3.1 の製品では、COBOL 構文 ASSIGN TO LPT1: を使用すると DBCS 文字を印刷することができます。この場合、CONFIG.SYS で定義されたプリンタ ドライバが使用されます。Windows 95 と Windows 98 では、プリンタ ドライバが使用されないため、この構文により DBCS 文字が作成されることはありません。プリンタ ドライバを使用して DBCS 文字を印刷できるようにするには、COBCONFIG ファイルで *printer_redirection* 実行時調整可能変数を TRUE に設定する必要があります。*printer_redirection* 実行時調整可能変数の詳細については、「COBOL システム リファレンス」を参照してください。

プリンタ ドライバでは、制御コードがサポートされていません。そのため、プログラムにより制御コードをプリンタ出力に挿入すると、出力結果が異なります。

3.1.2 INT、GNT および OBJコード

INT 形式にコンパイルしたアプリケーションは、Net Express で実行することができます。このリリースで作成される GNT 形式ファイルや OBJ 形式ファイルと、旧リリースで作成される GNT 形式ファイルや OBJ 形式ファイル

との間では互換性がない場合があります。そのため、すべての GNT ファイルと OBJ ファイルは、このリリースを使用して再コンパイルする必要があります。

3.1.3 IDY ファイル

コンパイラ辞書構造体が修正されています。つまり、旧リリースで作成された IDY ファイルとこのシステムの IDE には、互換性がありません。そのため、このリリースのコンパイラを使用して、アニメートするプログラムをすべて再コンパイルする必要があります。

3.1.4 LBR ファイル

このリリースにより作成されたアプリケーションで使用する Micro Focus ライブラリ形式の (.LBR) ファイルについては、すべてリビルドすることをお勧めします。以前の .LBR ファイルには、上位互換性を持つものもありますが、必ずしも保証されているわけではありません。NetExpress 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 でのデバッグだけに使用されるファイルです。ライブラリ ファイルには、データ ファイルだけが格納されます。アプリケーションの再パッケージ化については、「付録 A 開発ツール」に記載された「Build」の項目を参照してください。

3.1.5 実行コード

Micro Focus COBOL の旧製品を使用して作成された実行可能ファイルは、すべてこのリリースを使用してリビルドする必要があります。

3.1.6 データ ファイル

16 ビット COBOL V3.1 J 旧製品で作成されたデータ ファイルには、NetExpress との上位互換性があります。

3.2 バッチ ファイル

3.2.1 コンパイル

3.2.1.1 ANIM 指令

ANIM 指令を指定するだけでは、自動的に .int ファイルを作成することはできません。.int ファイルを作成するためには、NOGNT 指令も指定する必要があります。

3.2.1.2 PROCO

16 ビット V3.1 J 製品では、PROCO システムを通してプログラムをコンパイルすることができます。NetExpress には、PROCO システムがないため、次の表にしたがってバッチ ファイルを変更する必要があります。

PROCO のコマンド行	NetExpress のコマンド行
PROCO CHECK filename.cbl [指令]	COBOL filename.cbl ANIM NOGNT [指令] ;
PROCO C filename.cbl [指令]	COBOL filename.cbl ANIM NOGNT [指令] ;
PROCO ANIMATE filename.int	MFNETX /debug:filename.int
PROCO A filename.int	MFNETX /debug:filename.int
PROCO ZOOM filename.int	MFNETX /debug:filename.int
PROCO Z filename.int	MFNETX /debug:filename.int
PROCO COMPILE filename.int [指令]	COBOL filename.int OBJ [指令] ;
PROCO CO filename.int [指令]	COBOL filename.int OBJ [指令] ;
PROCO GENERATE filename.int [指令]	COBOL filename.int GNT [指令] ;
PROCO G filename.int [指令]	COBOL filename.int GNT [指令] ;
PROCO RUN filename.int	RUN filename.int
PROCO R filename.int	RUN filename.int
PROCO ADISCF	RUN ADISCF

この場合、

[指令] には、オプションで 1 つのコンパイラ指令を指定するか、複数のコンパイラ指令を連続して指定します。

* デバッガが起動し、最初の実行文が実行されますが、プログラム全体はズームされません。

3.2.1.3 実行

NetExpress では、グラフィックの RUN トリガにより新しいプロセスが生成されます。そのため、バッチファイルが次のようなファイルで構成されている場合、これらのファイルは連続して呼び出されません。

```
RUN A.GNT
RUN B.GNT
```

この場合、バッチファイルを次のように修正する必要があります。

```
RUNC A.GNT
RUNC B.GNT
```

3.2.2 リンク

NetExpress では、COBOL アプリケーションの静的リンクと動的リンクがサポートされています。

アプリケーションをリンクさせるためにバッチ ファイルを使用する場合、次の変更点を考慮する必要があります。このリリースでは、リンカ インターフェイスを使用して、簡単にオペレーティング システムのリンカにアクセスすることができます。このインターフェイスは、ユーティリティ CBLLINK.EXE で構成されています。

たとえば、.CMD ファイルのコマンド行で次のように指定すると、アプリケーションが動的にリンクされます。

```
LINK TICTAC+ADIS+ADISINIT+ADISKEY,,,COBLIB+DOSCALLS;
```

NetExpress では、これを次のように変更する必要があります。

```
CBLLINK TICTAC.OBJ
```

また、.CMD ファイルのコマンド行で次のように指定すると、アプリケーションが静的にリンクされます。

```
LINK TICTAC+ADIS+ADISINIT+ADISKEY,,,LCOBOL+DOSCALLS;
```

NetExpress では、これを次のように変更する必要があります。

```
CBLLINK TICTAC.OBJ -B
```

CBLLINK ユーティリティは、アプリケーションをリンクさせる必要のある .OBJ ファイルを判断し、これらを自動的にリンクさせます。

3.3 機能説明

3.3.1 CCI ファイル名

32 ビット製品では、ユーザーレベルの 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、次のいずれかになります。

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、常に ccitcp2.exe として提供されます。

3.3.2 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、NetExpress を含む 32 ビットの 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	FASTLINK	PROTMODE
64KSECT	FIXING	REGPARM
AUXOPT	MASM	SEGCROSS
BADSIGNS	MODEL	SEGSIZE
CHECKNUM	OPTSIZE	SIGNCOMPARE
DATALIT	OPTSPEED	SMALLDD
EANIM	PARAMCOUNTCHECK	TABLESEGCROSS
EXPANDDATA	PARAS	TRICKLECHECK

コンパイラ指令については、「付録C 指令とダイアレクト」を参照してください。

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 ビットで使用するためのものです。そのため、NetExpress 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、通常必要ありません。

3.3.3 LITLINK 呼び出し

先頭に "__" (二重アンダースコア) が記述された呼び出しは、呼び出し規則 8 を使用するために記録する必要があります。LITLINK 呼び出しを強制する場合、この方法を使用してください。二重アンダースコアの規則を使用して

LITLINK 呼び出しを行う場合、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。

呼び出し規則 8 を使用すると、個々の呼び出しを静的にリンクすることができます。CALL *literal* 文にこの呼び出し規則を使用すると、コンパイラによりこのリテラル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。この呼び出し規則は、他の呼び出し規則と組み合わせることができます。たとえば、呼び出し規則 10 は、スタックからパラメータを削除するサブプログラムに対して静的リンク呼び出しを行います。

3.3.4 Panels Version 2

Panels Version 2 を使用して描画された 3D オブジェクトを NetExpress で表示した場合、16 ビット製品で表示した場合よりも小さく見えます。NetExpress 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 の窪んだ境界線が描かれます。ネイティブの効果では、この境界線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。

3.3.5 実行時の構成

実行時スイッチに加えて、NetExpress では、実行時に構成可能な次の要素を使用して特定の実行時動作を構成することもできます。

- 環境変数
- 実行時調整可能変数

3.3.6 実行時スイッチ

16 ビットの COBOL システムで使用可能な次の実行時スイッチは、NetExpress 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	Z2
C*	K3	M*	/h
C1	L	M2	/s
C2	L1	P	/m
F1	L2	P1	/p
G	L3	S2	
I1	L4	S3	

* NetExpress 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、依存関係にある部分も削除する必要があります。

3.3.7 構成可能な属性バイト

16 ビット COBOL システムの属性バイトは、実装しているプラットフォームによって異なります。また、選択した属性モードによっても異なることがあります (DOS/V の COBATR03 や 73、OS/2 の COBFMT00.01.70 や 80 など)。ただし、NetExpress では、デフォルトの属性バイトが修正され、既存のアプリケーションとの互換性を確保するために構成ユーティリティが用意されています。旧製品の属性バイトについては、まず、リリース ノートを参照してください。その後で、旧システムとの互換性を確保するために NetExpress を構成する方法について、「NetExpress の構成可能な属性バイト」の項を参照してください。

第4章　日本語 COBOL/2 V1.1 からの移行

この章では、Micro Focus 日本語 COBOL/2 V.1.1 を使用して作成されたアプリケーションの維持方法について説明します。具体的には、問題の発生原因となりうる相違点やこれらの問題の解決方法について説明します。

4.1 上位互換性

4.1.1 ソース コードとファイル データ

COBOL/2 V1.1 で作成された有効なソース コードとファイル データは、NetExpress に対して上位互換性をもっています。そのため、COBOL/2 V1.1 で作成されたソース プログラムやファイルを NetExpress に移行することができます。

アプリケーションで COBOL システム ライブラリ ルーチンを使用している場合、NetExpress へ移行するために修正が必要な場合があります。詳細については、「付録 B COBOL システム ライブラリ ルーチン」を参照してください。

NetExpress は、旧 COBOL システムと異なり、小文字と大文字の DBCS ローマ字 COBOL データ項目を同様に処理します。そのため、同じ DBCS ローマ字変数名を大文字と小文字で別々に表記しても、これらを 2 つの別個の変数名として使用することはできません。

4.1.2 INT、GNT および OBJ コード

INI 形式にコンパイルしたプログラムを NetExpress で実行するためには、すべて再コンパイルすることをお勧めします。旧バージョンの COBOL から上位互換できるものもありますが、必ずしも保証されているわけではありません。このリリースで作成される GNT 形式ファイルや OBJ 形式ファイルと、旧リリースで作成される GNT 形式ファイルや OBJ 形式ファイルとの間では互換性がない場合があります。そのため、すべての GNT ファイルと OBJ ファイルは、このリリースを使用して再コンパイルする必要があります。

4.1.3 IDY ファイル

コンパイラ辞書構造体が修正されています。つまり、旧リリースで作成された IDY ファイルとこのシステムの IDE には、互換性がありません。そのため、このリリースのコンパイラを使用して、アニメートするプログラムをすべて再コンパイルする必要があります。

4.1.4 LBR ファイル

このリリースにより作成されたアプリケーションで使用する Micro Focus ライブラリ形式の (.LBR) ファイルについては、すべてリビルドすることをお勧めします。以前の .LBR ファイルには、上位互換性を持つものもありますが、必ずしも保証されているわけではありません。NetExpress 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 でのデバッグだけに使用されるファイルです。ライブラリ ファイルには、データ ファイルだけが格納されます。アプリケーションの再パッケージ化については、「付録 A 開発ツール」に記載された「Build」の項目を参照してください。

4.1.5 実行コード

Micro Focus COBOL の旧製品を使用して作成された実行可能ファイルは、すべてこのリリースを使用してリビルドする必要があります。

4.2 バッチ ファイル

4.2.1 コンパイル

次の説明では、Micro Focus 日本語 COBOL/2 V.1.1 製品とこのリリースの相違点について示します。また、問題の解決方法についても説明します。

COBOL/2 V1.1 のコンパイラ インターフェイスの形式は、変更されています。Micro Focus V3.1 J 以降の製品から移行する場合は、このインターフェイスを使用することになります。COBOL/2 V1.1 から移行する場合は、コンパイルするバッチ ファイルを COBOL への新しいインターフェイスを使用して変更する必要があります。

たとえば、バッチ ファイル

```
COBOL C TICTAC FLAG"MF"
COBOL CO TICTAC
```

と OS/2 コマンド ファイル

```
PCOBOL C TICTAC FLAG"MF"
PCOBOL CO TICTAC
```

は、次のように変更することができます。

```
COBOL TICTAC FLAG"MF" ANIM NOGNT;
COBOL TICTAC OBJ;
```

4.2.2 リンク

NetExpress では、COBOL アプリケーションの静的リンクと動的リンクがサポートされています。

アプリケーションをリンクさせるためにバッチ ファイルを使用する場合、次の変更点を考慮する必要があります。このリリースでは、リンカ インターフェイスを使用して、簡単にオペレーティング システムのリンカにアクセスすることができます。このインターフェイスは、ユーティリティ CBLLINK.EXE で構成されています。

たとえば、.BAT ファイルのコマンド行

```
LINK TICTAC+ADIS+ADISINIT+ADISKEY,,,RCOBOL;
```

は、次のように変更する必要があります。

```
CBLLINK TICTAC.OBJ
```

さらに、.CMD ファイルのコマンド行

```
LINK TICTAC+ADIS+ADISINIT+ADISKEY,,,PRCOBOL+DOSCALLS,TICTAC.DEF;
```

は、次のように変更する必要があります。

```
CBLLINK -OTICTAC.DLL -D TICTAC+ADIS+ADISINIT+ADISKEY
```

4.2.3 実行

NetExpress では、グラフィックの RUN トリガにより新しいプロセスが生成されます。そのため、バッチファイルが次のようなファイルで構成されている場合、これらのファイルは連続して呼び出されません。

```
RUN A.GNT
RUN B.GNT
```

この場合、バッチファイルを次のように修正する必要があります。

```
RUNC A.GNT
RUNC B.GNT
```

4.3 コンパイル

4.3.1 構文

ON EXCEPTION および ON OVERFLOW 句は、現在は、CHAIN 文では受け入れられていません。これらの句に与えられていた警告メッセージはエラーメッセージに変更されました。 (以前のリリースでは、この構文は受け入れられましたが、 ランタイム支援はありませんでした)。

4.3.1.1 入れ子プログラム

Micro Focus 日本語 COBOL/2 V1.1では、入れ子のプログラムが構成節を記述できました。Micro Focus COBOL V3.1 J では、記述することができません。

4.3.1.2 ACCEPT ... FROM DAY-OF-WEEK

文

```
ACCEPT データ項目 FROM DAY-OF-WEEK
```

上記の文は日曜日について、誤って値 0 を戻していました。現在はマニュアルに記載されているとおり、値 7 を戻

します。

4.3.1.3 COMP-5

USAGE COMP-5 データ項目の省略時動作が変更されました。以前は、COMP-5 は COMP と同様に処理されていました。このリリースでは、COMP-5 は COMP-X と同様に処理されます。よって、COMP-5 データ項目をマシン固有のバイト順を使用した真の 2 進数字項目とでき、データ型が有効なものとなります。

すでにプログラムで COMP-5 を使用しており、その符号処理に依存している場合、プログラムを COMP-5"1" 指令でコンパイルしてください。

4.3.1.4 NEXT SENTENCE

このバージョンの製品での NEXT SENTENCE の動作は、ANSI'85 規格に準拠します。つまり、制御は文の最後に移動します。

以前のバージョンでは、NEXT SENTENCE は制御を文の最後 (次のピリオドの後)、または、次の範囲終了区切り文字に続く文へ移動させています。

以前のバージョンのいくつかでは、NEXT SENTENCE の動作は、VSC2 指令を用いて 制御できました。この指令が設定されると NEXT SENTENCE が完結文の終わり (次のピリオドの後) へ制御を移動させ、設定されてないと次の範囲終了区切り文字に続く文へ移動しました。VSC2 指令は現在、NEXT SENTENCE の動作には影響を与えません。

プログラムが NEXT SENTENCE を使用し、特定の範囲終了区切り文字への制御の変更に依存していれば、MF"5" 指令 を使用します。あるいは、NEXT SENTENCE 指定 を CONTINUE に変更します。CONTINUE は、必要としている効果をもたらします。

この動作は OLDNEXTSENTENCE 指令を用いても得られます。

4.3.1.5 SYMBOLIC CHARACTERS 句

ANSI'85 規格に準拠するために、SYMBSTART の省略時の値は 1 に設定されています。以前のバージョンでは、これは 0 に設定されていました。SPECIAL-NAMES 段落に SYMBOLIC CHARACTERS 句を使用するプログラムの場合、必要な動作を保持するには、SYMBSTART"0" 指令を指定する必要があります。

4.3.1.6 RECORDING MODE

RECORDING MODE 指定が修正されました。このため、この指定が FD 句に明示的に含まれるとき、すべての場合において RECORD 句に優先し、RECORDING MODE は 常に固定長ファイルを作成させ、RECORDING MODE V は可変長ファイルを作成させます

RECORD 句が RECORDING MODE に優先する元の動作を必要とする場合は、MF"5" 指令を使用してください。

RECMODE 指令は、RECORD 句に優先しないことに注意してください。

4.3.1.7 COPY ファイル名

このバージョンでは現在、拡張子がなくて終了ピリオドの付いたファイル名と終了ピリオドの付かないファイル名とを区別しているので (前者を特定 の間隔文字の拡張子として扱う)、現存するプログラム内の一部のCOPY文は動作していないように見えることがあります。COPY ファイルに拡張子 .CPY を 付けて、プログラム内では拡張子なしでコード化したい場合は、その名前に終了ピリオドを付けないようにする必要があります。 例えば、

```
COPY "mycopy.".
```

to:

```
COPY "mycopy".
```

Note that:

```
 COPY mycopy.
```

このファイル名は終了ピリオドがないように扱われるので、以前のバージョンのように扱われることになります。

GOBACK 文と特殊レジスタ RETURN-CODE は、現在、標準の Micro Focus COBOL 言語の一部となっており、使用にあたっては VSC2 または OSVS を必要としません。MF レベル 5 またはそれ以上で使用可能 (省略時にはオン) で、したがって、MF(4) を指定すれば無効となります。

プログラム ID に続く注記は、以前のバージョンでは可能でしたが、省略時の値では不可となっています。これらの注記は、PROGID-COMMENT 指令を用いれば使用可能にできます。

4.3.1.8 REPORT-WRITER 予約語

REPORT-WRITER を支援する主要な方言に REPORT-WRITER 予約語を組み入れることによって、RW 指令は廃止されました。ある REPORT-WRITER 予約語をデータ名として使用し、NORW 指令でコンパイルしていたプログラムを持っている場合、NORW 指令を USE(NORW) 指令で置き換える必要があります。これにより REPORT-WRITER 予約語は予約語リストから除かれ、データ名として使用できるようになります。

4.3.1.9 指令

- RW 指令は削除されました。
- PANVALET 指令の省略時の値は、現在 NOPANVALET です。++INCLUDE は、PANVALET が指定されていると、現在ではフラグを立てられていません。また、-INC は LIBRARIAN が指定されていると、現在ではフラグを立てられていません。 LIBRARIAN と PANVALET を共に指定すると、コンパイルされたプログラムがメインフレーム互換でなくなるという警告が表示されます。
- 指令ファイルの操作を変更して、指令ファイルが正しく入れ子にできるようにしました。以前は、指令ファイルで DIRECTIVES 指令を指定すると、コンパイラは新しい指令ファイルに切り替わり、元のファイルに戻ることはありませんでした。現在は、戻るようになっています。この結果、元の指令ファイル内の続く指

令も処理されます。一方、以前のリリースでは、これらの指令は無視されていました。

- VSC2 は現在、VSC2(3) と同等です。以前は、VSC2 は VSC2(2) と同等でした。VSC2 OLDVSC2 指令は、VSC2(1) で置き換えられました。

- 指令 OLDFILEIO は、必要な索引ファイルの型を定義する指令 IDXFORMAT で置き換えられました。IDXFORMAT"2" は、OLDFILEIO と同等です。

- ANSI'85 規格に準拠するために、ALPHASTART の省略時の値は 1 に設定されています。以前のバージョンでは 0 に設定されました。Special-Names 段落に ALPHABET 句を使用するプログラムの場合、必要な動作を保持するために、指令 ALPHASTART"0" を指定する必要があります。

- 以前に ANS85 指令を用いないでコンパイルしたプログラムは、この省略時システムの下で再コンパイルされると、正しく動作しないことがあります。最も顕著にこの問題が発生するのは、新しいシステムで、省略時に ANS85 がオンになっている場合です。このとき、すべてのファイル状態コードは ANS74 状態ではなく ANS85 になります。この問題は、ANS74 ファイル状態で動作するように設計されているプログラムをコンパイルするとき、NOANS85 指令を用いて回避することができます。

- このリリースでは、OBJ 指令によりエントリ ポイント名を変更することはできません。

- RTNCODE-SIZE 指令の省略時のパラメータは、このリリースでは "2" から "4" に変更されました。アプリケーションにおいて無効なデータが RETURN-CODE に引き渡されることがないことを確認するか、RTNCODE-SIZE"2" 指令を使用して再度コンパイルするかしてください。

4.3.1.10 コンパイラの動作

- COPY...REPLACING の動作は、ANS85 または VSC2(3) 指令の設定によって変更されました。以前は、COBOL 以外の文字はすべて COPY...REPLACING に対するオペランドでは許可されておらず、また、小文字とコロンが文字集合に付加されていました。この動作は、現在、ANS85 または VSC2(3) 指令が使用される場合にだけ発生します。そうでない場合は、オペランドに COBOL 以外の文字が入ることがあります。

- アセンブラ レベルのインターフェイスでは、COBOL コンパイラで作成されたオブジェクト区分クラス名が、このリリースでは変更になっているので注意してください。

4.3.1.11 80 桁目以降の原始コード

- COBOL/2 は不正に 80 桁目以降の原始コードを次の行の 1 桁目として処理しま す。これは違法で ANSI 規格に準拠していません。したがって、80 桁目以降に原始コードがあるプログラムは、チェッカエラーを発生します。プログラムの原始コードは、80 桁を超えないようにしてください。

4.4 実行

- 以前のリリースでは､可変長ファイルについて指定された最大・最小レコード サイズは、そのファイルを参

照するどのプログラムでも同じである必要がありました。この制限が除かれ、プログラムは、ファイルの作成に使用されたプログラム中で指定されたものと、異なるレコード長を指定できるようになりました。しかし、定義された最小サイズより小さい、あるいは、定義された最大サイズより大きいレコードを書き出そうとすると、ランタイム エラー 9/044が戻されます。

- このソフトウェアの以前のバージョンでは、ファイルのレコード長が FD 句で 与えられたレコード長と全く同じにならないと、索引ファイルが開かれた時にエラーとなりました。これが変更され、最大サイズより大きい、あるいは 最小サイズより小さいレコードを書き出そうした場合にだけエラーとなりま す。

- 報告書作成モジュールが変更され、OSVS 指令が使用される場合、IBM OS/VS COBOL と互換が取れるようになっています。これは ANSI'68 COBOL 規格です。以前のバージョンでは、OSVS の場合の省略時解釈は ANSI'74 COBOL 規格の報告書作成機能でした。OSVS 機能を使用しているが、ANSI'74 報告書作成機能との互換性を要求するプログラムを作成することがありました。この組合わせはできなくなったため、OSVS 機能を除くか (VSC2 を使用)、あるいは、ANSI'68 COBOL 報告書作成規格を受け入れるかのどちらかとする必要があります。

- 以前の製品では、資料に不可と書かれているにも関わらず、部分参照演算で負の結果が許されていました。しかし、この結果は依然未定義です。

  部分参照を目的とした結果は、ゼロ以外の正の整数でなければならないという規則は現在も明記されています。これは『言語リファレンス』の「COBOLの概念 」の見出し「部分参照」の 4 項 a と 4 項 b に記述されています。4b.

  以前の製品では、ランタイム システムのバグにより、部分参照の目的でデータ項目の一部の長さを求めるために負の結果を使用すると、負の結果は正に変換されました。

  部分参照計算に負の値を使用することは許されませんが、計算結果が正のゼロでない整数の場合、ランタイム システムのバグによりそれらの値は正の数に変換されていました。したがって、全体的に正の結果を導くために負の値を使用すると部分参照は誤った動作をしました。次のプログラムに例を示します。

```
working-storage section.
01 a pic x(10) value "ABCDEFGHIJ".
01 b pic s9(1) value -3.
procedure division.
    display a(1:b + 4)
    stop run.
```

  上記のプログラムの正しい動作は 'A' を表示することです。しかし、以前の製品におけるランタイム システムのバグにより、データ項目 'b" の値は正の値 (+3) に変換され、プログラムは 'ABCDEFG' を表示しました。

この動作のもう一つの結果として、使用されている負の値が正に変換されるため、次のコードを実行することができます。ここでもこの製品では負の結果を使って部分参照を行うことは許されず、以下のプログラムは、不正な値を使用しているため違法であることに注意してください。

```
working-storage section.
01 a pic x(10) value "ABCDEFGHIJ".
01 b pic s9(1) value -3.
procedure division.
    display a(1:b).
    stop run.
```

RTS 内の負を正に変えるバグは MOVE 文のロジックにありました。明示の MOVE に 対しても、部分参照中に作られる暗黙の MOVE に対しても暗黙処理をしていました。当然、この問題が修正された結果、負の値は負のままになりました。したがって、上記のプログラムは別な動作をします。

このリリースにおける不正な部分参照の結果は未定となり、通常オペレーティング システム エラーを起こします。現在の製品レベルではこの不正処理はコンパイル時にも実行時にも検出できず、ランタイム エラーを生成することができないため、オペレーティング システム エラーとなります。この現象は、新規のプログラムで不正な値を使用するか、上記のように不正な値を含む既存のプログラムを実行した場合に発生します。

部分参照の目的で長さあるいはデータ項目の一部の計算に負の結果を使用するコードがある場合、処理結果が正になるように修正してください。

- COBOL/2 のいくつかのバージョンでは、場合によってはリダイレクトを行うのに S6 ランタイム スイッチが使用されました。現在では、S5 スイッチでこの機能を利用できます。
- COBOL/2 のいくつかのバージョンでは、M6 スイッチはヒープの連結を禁止するために使われました。このスイッチが回避したヒープ連結の問題は発生しなくなりました。したがってこのスイッチは意味がなく、使用は避けてください。

4.5 アニメート

このリリースでは省略値で +F スイッチを設定してあります。これにより数字項目内の違法データは、アニメート中に、また、中間コードを実行しているときに捕らえられます (ランタイム システム エラー 163)。これにより、以前は動作していた既存のプログラムが正しく動作しなくなることがあります。しかし、そのようなプログラムの不正を捕らえることによって、その後に生成されるコードが正しく動作しなくなるのを防ぐ手助けとなります。 生成されたすべてのプログラムは、常に数字項目には数字データが入っているものとして、動作します。

4.6 リンク

このリリースでは、索引ファイルのファイル ハンドラは MFFH と呼ばれます。IXSIO ファイル ハンドラはこの製品には含まれていません。MFFH ファイル ハンドラは、アプリケーションに静的にリンクさせるか、IXSIO とまったく同様にスタンドアロン システム プログラムとしてリンクさせることができます。

4.7 日本語文字についての考慮事項

- JAPANESE 指令は、現在の新しい指令 NCHAR と同義です。
- JAPANESE"1" または NCHAR"1" は、以前の日本語製品の JAPANESE 指令と同等です。

4.7.1 2バイトの空白詰め

日本語 COBOL/2 では、2 バイトの空白詰めはランタイム システムをパッチすることによってだけ構成できます。詳細については、ランタイム システムの「リリースノート」を参照してください。しかし、「リリースノート」にこの機能についての説明がない場合、ご使用の日本語 COBOL/2 システムは、2 バイト空白文字については自動的に X"2020" に設定されています。

この方法はコンパイラ指令によって次のように置き換えられました。

DBSPACE - X"8140" を 2 バイト間隔文字として指定します。NODBSPACE - X"2020" を 2 バイト間隔文字として指定します。

送り出し側が SBCS で受取側が DBCS である MOVE 文を実行するとき、送出し側項目に含まれている任意の X"20" スペースは現在次のように変換されます。

送り出し側	受取側: NCHAR"1" を使用	NCHAR"2" を使用
X"20"	X"2020"	X"2020"
X"2020"	X"2020"	X"20202020"
X"202020"	X"20202020"	X"202020202020"
X"20202020"	X"20202020"	X"2020202020202020"

したがって、NCHAR"1" は Micro Focus 日本語 COBOL/2 V1.1 製品上で JAPANESE 指令の動作と互換があります。

4.7.2 STRING 文と UNSTRING 文

日本語 COBOL/2 の最近のバージョンの一部では、STRING 文と UNSTRING 文内で PIC X と PIC N の混在を可能にしていました。これは標準の COBOL 動作ではなく、そのような動作の結果を定義することはしばしば困難です。したがって、このシステム では STRING および UNSTRING のほとんどの項目について、PIC X または PIC N を混在させることを許していません。

現存するプログラムの保守のために、OLDSTRMIX 指令を指定すればこの動作を行えます。

4.7.3 互換用の指令

Micro Focus 日本語 COBOL/2 V1.1 との完全互換のために、次の指令を設定する必要があります。

```
          OLDSTRMIX
          DBSPACE
または    NODBSPACE       - 先に述べた 2 バイト空白づめの注を参照
          COMP-5"1"
          DBCS"1"         - 標準 DBCS サポート用
または    NCHAR"1"        - fMicro Focus 日本語拡張用
          RTNCODE-SIZE"2"
```

第5章　LEVEL II COBOL V2.5.54 からの移行

この章では、Micro Focus LEVEL II COBOL V2.5.54 を使用して作成されたアプリケーションの維持方法について説明します。具体的には、問題の発生原因となりうる相違点やこれらの問題の解決方法について説明します。

5.1 上位互換性

5.1.1 ソース コード

LEVEL II COBOL V2.5.54 で作成された有効なソース コードは、NetExpress に対して上位互換性をもっています。そのため、LEVEL II COBOL V2.5.54 で作成されたソース プログラムやファイルを NetExpress に移行することができます。このバージョンの予約語と古いソースでユーザーが定義した単語は、競合することがあります。この場合、プログラムのコンパイル時に、MF(1) 指令または MF(2) 指令を使用してください。

5.1.2 INT、GNT および OBJ コード

INI 形式にコンパイルしたプログラムを NetExpress で実行するためには、すべて再コンパイルすることをお勧めします。旧バージョンの COBOL から上位互換できるものもありますが、必ずしも保証されているわけではありません。このリリースで作成される GNT 形式ファイルや OBJ 形式ファイルと、旧リリースで作成される GNT 形式ファイルや OBJ 形式ファイルとの間では互換性がない場合があります。そのため、すべての GNT ファイルと OBJ ファイルは、このリリースを使用して再コンパイルする必要があります。

5.1.3 IDY ファイル

コンパイラ辞書構造体が修正されています。つまり、旧リリースで作成された IDY ファイルとこのシステムの IDE には、互換性がありません。そのため、このリリースのコンパイラを使用して、アニメートするプログラムをすべて再コンパイルする必要があります。

5.1.4 LBR ファイル

このリリースにより作成されたアプリケーションで使用する Micro Focus ライブラリ形式の (.LBR) ファイルについては、すべてリビルドすることをお勧めします。以前の .LBR ファイルには、上位互換性を持つものもありますが、必ずしも保証されているわけではありません。NetExpress 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 でのデバッグだけに使用されるファイルです。ライブラリ ファイルには、データ ファイルだけが格納されます。アプリケーションの再パッケージ化については、「付録 A 開発ツール」に記載された「Build」の項目を参照してください。

5.1.5 実行コード

Micro Focus COBOL の旧製品を使用して作成された実行可能ファイルは、すべてこのリリースを使用してリビルド

する必要があります。

5.1.6 データ ファイル

行順、レコード順、相対ファイルはすべて NetExpress に対する上位互換性をもっています。

Level II COBOL V2.1 で作成された索引順編成ファイルは、Level II COBOL V2.5.4 およびそれ以降の製品のファイル形式とは互換がありません。

Level II COBOL V2.5.4、Microsoft COBOL Version 1、Microsoft COBOL Version 2 から作成される索引順ファイルは、この COBOL システムの索引ファイル ハンドラで作成される索引順ファイルとは形式が異なります。索引ファイル ハンドラは現存するファイルの形式を認識、読み込み、更新しますが、新規ファイルはすべて新しい形式で作成します。古いファイルを新しい形式のファイルに変換するには、Rebuild に /s:lii、 /s:ms1、/s:ms2 オプションを組み合わせて使用します。

```
rebuild 旧ファイル .dat, 新規ファイル .dat /s:lii [/i] [/v]
    - Level II COBOL V2.5.4 形式にファイルを更新
rebuild 旧ファイル .dat, 新規ファイル .dat /s:ms1 [/i] [/v]
    - Microsoft COBOL Version 1 形式にファイルを更新
rebuild 旧ファイル .dat, 新規ファイル .dat /s:ms2 [/i] [/v]
    - Microsoft COBOL Version 2 形式にファイルを更新
```

索引ファイルを古い形式で作成したい場合、プログラムのコンパイル時に IDXFORMAT"2" 指令を使用することによって可能です。IDXFORMAT"2" 指令を使用すると、その結果できる実行可能ファイルは古い形式の索引ファイルを作成します。古い形式のままだと、この索引ファイルハンドラの提供する新しい機能 (動作性能の向上、可変長レコード、READ PREVIOUS 等)を使用できません。

索引ファイル (古い形式あるいは新しい形式いずれの場合も)を使用するすべてのプログラムは、索引ファイル ハンドラ MFFH にプログラムをリンクするか、あるいは、スタンドアロン モジュールとして使用可能とするかによって、アクセスする必要があることに注意してください。

5.2 バッチ ファイル

5.2.1 コンパイル

次の説明では、Micro Focus LEVEL II COBOL V2.5.54 製品とこのリリースの相違点について説明します。また、問題の解決方法についても説明します。

LEVEL II COBOL のコンパイラ インターフェイスの形式は、変更されています。Micro Focus V3.1 J 以降の製品から移行する場合は、このインターフェイスを使用することになります。COBOL/2 V1.1 から移行する場合は、コンパ

イルするバッチ ファイルを COBOL への新しいインターフェイスを使用して変更する必要があります。

たとえば、バッチ ファイル

```
COBOL C TICTAC FLAG"MF"
COBOL CO TICTAC
```

と OS/2 コマンド ファイル

```
PCOBOL C TICTAC FLAG"MF"
PCOBOL CO TICTAC
```

は、次のように変更することができます。

```
COBOL TICTAC FLAG"MF" ANIM NOGNT;
COBOL TICTAC OBJ;
```

出力は、ANIM、GNT、OBJ などの指令により制御されます。たとえば、.GNT ファイルは次のコマンドを使用して作成することができます。

```
COBOL TICTAC GNT;
```

5.2.2 リンク

NetExpress では、COBOL アプリケーションの静的リンクと動的リンクがサポートされています。

アプリケーションをリンクさせるためにバッチ ファイルを使用する場合、次の変更点を考慮する必要があります。このリリースでは、リンカ インターフェイスを使用して、簡単にオペレーティング システムのリンカにアクセスすることができます。このインターフェイスは、ユーティリティ CBLLINK.EXE で構成されています。

例えば、以下のような .BAT コマンド行は、

```
LINK TICTAC+ADIS+ADISINIT+ADISKEY,,,RCOBOL;
```

次のように変更する必要があります。

```
CBLLINK TICTAC.OBJ
```

また、以下のような .CMD コマンド行は、

```
LINK TICTAC+ADIS+ADISINIT+ADISKEY,,,PRCOBOL+DOSCALLS,TICTAC.DEF;
```

次のように変更する必要があります。

```
CBLLINK -OTICTAC.DLL -D TICTAC+ADIS+ADISINIT+ADISKEY
```

5.2.3 実行

NetExpress では、グラフィックの RUN トリガにより新しいプロセスが生成されます。そのため、バッチファイル

が次のようなファイルで構成されている場合、これらのファイルは連続して呼び出されません。

```
RUN A.GNT
RUN B.GNT
```

この場合、バッチファイルを次のように修正する必要があります。

```
RUNC A.GNT
RUNC B.GNT
```

5.3 コンパイル

5.3.1 指令

VSC2 指令は IBM VS COBOL II リリース 3 の動作に従います。以前の製品では、リリース 1 だけとしか互換がありませんでした。新しいシステムを用いてこの動作を得るには、VSC2"1" 指令を使用する必要があります。この指令の詳しい説明については、『COBOL システム リファレンス』を参照してください。

以前の製品では、OSVS と VSC2 指令は表アクセスに対する添字の範囲外チェックを行わない中間コードファイルを作成しました。このシステムでは、NOBOUND コンパイラ指令が使用されない限り、省略時解釈として添字チェックコードが作成されます。

CHECK 指令は BOUND 指令と完全に同義で、互換を図るためにのみ保持されています。将来の製品では削除されることが考えられます。この指令は添字および指標についての境界チェックコードを生成させます。

詳細については、「付録 C 指令とダイアレクト」を参照してください。

5.3.2 コマンド行からのパラメータ受取り

LEVEL II COBOL ではコマンド行からパラメータを受け取る方法は次のとおりでした。

```
ACCEPT ... FROM CONSOLE
```

これは現在、次のように変更になりました。

```
ACCEPT ... FROM COMMAND-LINE
```

5.3.3 NUMBER 句

ADIS の ACCEPT または DISPLAY 文で NUMBER 句が指定されると (例えば、LINE NUMBER 指定)、次の構文エラーが報告されます。

利用者語が必要である

以前の製品ではこれは許されていました。NUMBER 句はコードの読みやすさを向上させるためにだけ使用されるの

で、ACCEPT あるいは DISPLAY 文では使用しないことをお勧めします。

5.3.4 2 進ライブラリ値

『言語リファレンス マニュアル』に正しい値が記載されていなかったため、LEVEL II COBOL では、誤った 2 進ライブラリ値を使用している可能性があります。この製品では、+F ランタイムスイッチ (デフォルトで、2 進値の妥当性をチェックするために設定されている)が誤った 2 進ライブラリ値が発生した場所を示します。原始コードが正しい値を使用するように修正してください。例えば、

```
MOVE X"3004" TO A-VAL.
DISPLAY A-VAL.
```

は、次のように変更する必要があります。

```
MOVE X"3034" TO A-VAL.
DISPLAY A-VAL.
```

(ここで、A-VAL は数字データ項目です)

5.3.5 HIGH-VALUES

このシステムは、表意定数 HIGH-VALUES に対しては、16 進 FF を用います。以前の製品では、x"7F" が使用されました。したがって、アプリケーションが HIGH-VALUES を英数字項目に転記していると、その後の x"7F" との比較は失敗することになります。

5.3.6 指標

以前のバージョンでは、表に対する指標を変更するのに算術演算 (加算や減算など) を使用することができました。この製品では、これは省略時には使用できなくなっています。算術演算を SET 文で置き換えることによってこの問題を避けるか、あるいは OLDINDEX コンパイラ指令を用いてこの動作を可能とする方法があります。前者の方法をお勧めします。

5.3.7 リンク パラメータ

割り付けられてないリンク項目を後続の CALL 文に対するパラメータとして参照する機能は、このソフトウェアではサポートされていません。 例えば次のような場合、

プログラム A

```
CALL "B" USING a1 a2
```

プログラム B

```
PROCEDURE DIVISION USING b1 b2 b3
CALL "C" USING b1 b2 b3
```

プログラム C

```
PROCEDURE DIVISION USING c1 c2 c3
```

これは、"C" に対する呼出しをするとエラー 203 となります。

5.3.8 ゼロによる除算エラー

動作性能上の理由により、現在、ジェネレータは「ゼロによる除算」エラーをチェックするコードを生成しません。したがって、このようなエラーが発生しないようにするか、あるいは、ON SIZE ERROR 指定のような適当な機能でこのようななエラーを把握するようにするのは、アプリケーション側で確実に行ってください。

5.3.9 システム ドライブ

以前の製品では、ランタイム システムにシステムのドライブの中でファイルを見つけさせるために、プログラムがドライブ識別子 "S:" をファイル名に使用できるようになっていました。

ネットワーク化されたマシンを使用するとき、ドライブ識別子間の矛盾が発生するので、これは現在では使用されていません。プログラムが COBOL システム ディレクトリにあるファイルにアクセスする場合、COBDIR 環境変数をプログラムが使用することをお勧めします。"S:" に対する参照を "$COBDIR¥" に対する参照に変更することによりこれが可能になります。例えば、

```
SELECT FILE1 ASSIGN TO "S:FRED.DAT"
.
.
   CALL "S:HELP" USING HELP-ID.
```

これは次のようになります。

```
SELECT FILE1 ASSIGN TO "$COBDIR¥FRED.DAT"
.
.
   CALL "$COBDIR¥HELP" USING HELP-ID.
```

COBDIR 環境変数が設定されている限り、そのディレクトリを探索して FRED と HELP を探します (COBDIR 環境変数の設定方法に関する詳細は、『始動マニュアル』を参照してください)。

5.3.10 装置ファイル

:CO: コンソール・キーボードまたは画面用 :LP: パラレル プリンタ用

CON または CON: ファイル名を使って COBOL 構文をチェックすることはできなくなりました。例えば、PROCO CHECK CON: はサポートされていません。

5.3.11 開いてないファイルを閉じる

以前の製品では、開いてないファイルを閉じることが可能でした。このシステムでは、そのような動作は RTS エラー 142、または、そのファイルが状態領域を定義されている場合は、ファイル状態リターン 9/142を引き起こします。RTSエラー 142 が発生した場合は、プログラムが開かれてないファイルを閉じようとしないようにプログラムを修正してください。これを行いたくない場合、問題のファイルについてファイル状態を定義できますが、閉じる時に戻される状態は無視してください。

5.3.12 COBOL システム ライブラリ ルーチン

LEVEL II COBOL で使用可能だった次のルーチンは、このシステムではサポートされていません。

X"91" ファンクション1 - 新しいファイルへ連鎖

このルーチンの現存する呼出しは、COBOL の CHAIN 文を使用するように変更する必要があります。 COBOL システム ライブラリ ルーチンの詳細については、「付録 B COBOL システム ライブラリ ルーチン」を参照してください。

5.3.13 複数リール ファイル

複数リール ファイルのヘッダー レコードが、以前の製品のものから変更されました。最初の 49 バイトには次の文字列が入っています。

COBOL MULTIPLE FILE HEADER V2.0

以前の製品においてバージョン チェックを無効にするのに使用されていたランタイム スイッチ -v は、このリリースでは使用できません。したがって、以前の製品を用いて作成された複数リール ファイルは使用できません。

5.3.14 アセンブラ サブプログラム

Micro Focus LEVEL II COBOL では、次のレジスタを任意のアセンブラ サブプログラムによって保存しておく必要があります。

CS DS ES SS BX ディレクション フラグ

この製品では、次のレジスタを保存しておく必要があります。

CX SS:SP DS BP ディレクション フラグ

5.3.15 2 進ファイル (.BIN)

以前の製品でサポートされていた 2 進ファイル (.BIN) は、このシステムではサポートされていません。

5.3.16 RTS.BIN および RTSBIN.ASM ファイル

以前の製品は、RTS.BIN、または、RTSBIN.ASM と呼ばれるどちらかのファイルに保持されたユーザー記述の数字による呼出しルーチンをサポートしていました。この機能は、このリリースではサポートされていません。

5.3.17 APPEND の使用

APPEND コマンドが導入されました。これを用いて、現行ディレクトリ以外のディレクトリを、現行ディレクトリ下にあるもののように動作させることが可能です。

COBOL ランタイム システムは、まず現行ディレクトリを、次に COBDIR ディレクトリを検索してファイルを探すので、APPEND パスにある、以前の Micro Focus ベースのソフトウェアを見つけることができます。

例えば、APPEND ディレクトリの 1 つに以前の製品を用いて書かれたアプリケーションが格納されており、このアプリケーションが、開こうとしているライブラリと同じ名前のライブラリファイルを持っている場合、その APPEND パス内のライブラリが、今まで述べた互換性に関する制限に従っているかを確認する必要があります。

第6章　Embedded SQL Toolkit V2.1 からの移行

Embedded SQL Toolkit for Microsoft SQL Server は NetExpressでは使用できません。かわりに、NetExpress 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。

この章では、OpenESQL と Embedded SQL Toolkit for Microsoft SQL Server の相違点について説明します。

6.1 変更点のまとめ

SQL のコンパイラ指令には、新規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。たとえば、

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

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。)

また、

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

は、次のようになります。

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

さらに、

```
SQL(MSSQL)SQLPROT
```

は、次のようになります。

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

その他の相違点は、次のとおりです。

- 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、上位の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 の 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、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7章　Dialog System V2.5 J からの移行

この章では、NetExpress の Dialog System に移行する場合の注意点について、Dialog System V2.5 J と比較しながら説明します。

Dialog System V2.5 J を使用している場合、大きな問題はありません。NetExpress の Dialog System には、ソース コード テンプレートが追加されおり、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 サポートを使用することができます。

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 ビットから 32 ビットに移行する場合と OS/2 などのそ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 ユーザー ガイド』の「*同一画面と異種プラットフォーム*」を参照してください。

7.1 注意点

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

  ただし、NetExpress 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することができます。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勧めします。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 dslink32.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、NetExpress プロジェクト テンプレートが使用されるため、将来に備えてこれ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 ユーザー ガイド』を参照してください。

- NetExpress コマンド プロンプトから Dialog System を実行することができます。

  32 ビットの Dialog System V2.5J では、Dialog System トリガ名は DSNT.EXE です。一方、NetExpress では、Dialog System トリガ名は DSWIN.EXE に変更されています。

- ダイアログ エディタとデータ定義エディタによる切り取り、コピー、貼り付け、その他スクリーンセット アニメータのステップ実行のような機能などに対するデフォルトのアクセラレータ キーは、Microsoft 規格と NetExpress IDE に対して互換性をもつように変更されています。新しいアクセラレータ キーを調べるには、メニュー項目の隣を参照してください。

- DBCS ローマ字

  NetExpress では、DBCS ローマ字オブジェクト名を大文字として処理します。そのため、小文字の DBCS ロ

ーマ字オブジェクト名を含むスクリーンセットを処理する上で問題が発生します。この場合、旧バージョンの Dialog System を使用して DBCS ローマ字名を大文字に変更することをお勧めします。

第8章　Panels V2 アプリケーションの移行

NetExpress 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勧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 マニュアルで説明されています。

Panels V2 を使用して作成した既存のアプリケーションをさらに開発する場合、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勧めします。この章では、この方法について説明します。

8.1 方法

この章では、プログラム例 ¥Netexpress¥Base¥Demo¥Pan2oo¥Pan2oo.cbl を参照してください。このプログラム例では、Panels V2 とクラス ライブラリを一緒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

次の項では、Pan2oo.cbl のコードについて説明します。

8.1.1 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 ライブラリを一緒に使用するには、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。

8.1.2 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ためのコードです。

```
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"
```

8.1.3 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.
```

付録A: 開発ツール

この章では、既存の Micro Focus COBOL システムで使用できる機能を列挙し、NetExpress 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、この章では、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。実行時機能を使用する既存のアプリケーションを NetExpress 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。

この章では、各章で説明されている分野は取り上げません。

A.1 除外されたコンポーネント

NetExpress では、次のような COBOL、Toolkit、および Workbench で使用できるコンポーネント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

  NetExpress は、グラフィカル アプリケーション用ですが、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 を構成するためのユーティリティも含まれています。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ためのツール

  NetExpress は、既存のアプリケーションのダウンサイズ化と PC で実行するためのアプリケーション開発を目的としています。

- 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、すべて大きく拡張されて組み込まれています。

- 広く使用されないツール

除外されたコンポーネントは、次のとおりです。

- バナー、バッチ ファイル ファシリティ、カラー、キー操作マクロ、Linein、メニュー ハンドラ、スクリーン
- ファイル転送エイド、パラメータ パサー (文字)、パラメータ パサー (グラフィック)、ソース コンバータ
- アナライザ、アニメータ、アニメータ V2、基本アニメータ、COBOL ソース情報 (CSI)、構造体アニメータ

上記のコンポーネントについては、特別言及する必要がないかぎり、この章では説明しません。

A.2 新規の画面処理アプリケーション

この章の各項目では、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。この章全体で繰り返し説明せずに、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 形式の新規 GUI アプリケーションを作成するには、Dialog System を使用します。

A.3 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 - 説明する機能を持つ NetExpress 以前の Micro Focus 製品を列挙します。
- NetExpress - 説明する機能が NetExpress 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 NetExpress へ移行するために必要な変更点を解説します。
- NetExpress で使用する機能 - 説明する機能が NetExpress にない場合、または下位互換性を確保するためだけに装備されている場合に、新しく同じ効果を実現するための機能を紹介します。

A.3.1 機能リスト

A.3.1.1 アドバンスト オーガナイザ

アドバンスト オーガナイザ デスクトップを作成するソフトウェア。トップレベル ウィンドウまたは開発環境です。

この機能を含む製品

- COBOL V4.0 J
- COBOL V5.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support

NetExpress

使用できません。

移行

NetExpress では アドバンスト オーガナイザ プロジェクトを使用することはできません。最初から NetExpress プロジェクトを作成する必要があります。

NetExpress で使用する機能

NetExpress 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、NetExpress 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、ここから実行することができます。

A.3.1.2 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

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1
- LEVEL II COBOL V2.5.54

NetExpress

IDE に組み込まれています。旧製品では、GUI バージョンはアニメータ V2 と呼ばれ、文字バージョンはアニメータと呼ばれます。 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。

NetExpress 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 つのわずかな違いが影響します。NetExpress では、ANIM 指令を指定するだけでは、自動的に .int ファイルを作成することはできません。.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)、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。

A.3.1.3 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ためのユーティリティ。文字バージョンとグラフィカル バージョンの両方があります。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

使用できません。

詳細

旧 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することができます。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。

NetExpress 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 でデバッグする場合だけに実行されます。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、.obj ファイルのうちどのファイルを作成して .dll ファイルと .exe ファイルにリンクさせるかを指定します。.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だけに使用されます。

NetExpress では、「ビルド」という用語が旧システムとは異なる意味で使用されています。旧システムでは、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、NetExpress 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することもできます。この場合、起動プログラムを .exe ファイルとしてリンクするだけです。

NetExpress で使用する機能

アプリケーションを作成するには、プロジェクトを使用します。

A.3.1.4 Cbllink

アプリケーションをリンクさせるためのコマンド行ユーティリティ

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット

NetExpress

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。

NetExpress で使用する機能

リンク処理は、通常、 NetExpress でプロジェクトを通して行われます。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、NetExpress コマンド行から Cbllink を呼び出すこともできます。

A.3.1.5 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、編集するためのツール。

この機能を含む製品

- COBOL V5.0 J

- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support

NetExpress

機能は、旧製品と同様ですが、拡張され、改良されたインターフェイスを装備しています。

NetExpress **で使用する機能**

IDE で [検索] メニューの [参照] を使用します。

A.3.1.6 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のライブラリ。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support

NetExpress

機能が拡張されています。NetExpress 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、OLE クラス ライブラリです。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。

A.3.1.7 COBATR ユーティリティと COBFMT ユーティリティ

DOS と OS/2 で属性モードを変更するためのユーティリティ。

この機能を含む製品

- COBOL および Workbench V3.1 J 16 ビット

NetExpress

使用できません。NetExpress の属性バイトは、旧 32 ビット Micro Focus COBOL システムと互換性があります。16 ビット COBOL V3.1 J システムで特定の COBATR または COBFMT ユーティリティを使用した場合、NetExpress でアプリケーションを実行すると問題が生じることがあります。この場合、32 ビット Windows の属性バイトを使用

してアプリケーションのコードを再作成するか、属性バイトを再構成する必要があります。実際に使用されたユーティリティによっては (COBATR ユーティリティまたは COBFMT ユーティリティ)、属性バイトを正確に再構成できないことがあります。詳細については、「構築可能な属性バイト」の項を参照してください。

A.3.1.8 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すことができるルーチンのライブラリ。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1
- LEVEL II COBOL V2.5.54

NetExpress

この機能は NetExpress にも含まれています。ただし、旧製品のルーチンによっては、除外されたものもあります。

移行

「付録 B COBOL ライブラリ システム ルーチン」で互換性に関する問題のリストを参照してください。

A.3.1.9 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

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support

- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

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 サーバー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

NetExpress は NetBEUI、TCP/IP、DDE および Novell IPX プロトコルを装備しています。

NetExpress では、タイムアウト期間を動的に構成することができます。旧製品でアプリケーションのコードに 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。NetExpress 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することができます。-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ー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、制御することができます。

CCITCP2 は、NT サービスとしてインストールされている場合、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 サーバーを追加している場合、削除し、必要なときだけに実行します。それ以外の場合、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、変更は必要ありません (ただし、「V3.1 J 16 ビット COBOL からの移行」の章を参照してください)。

NetExpress で使用する機能

クライアント サーバー結合を使用することをお勧めします。

A.3.1.10 コンパイラ

構文を確認し、実行可能形式に変換するための開発ツール。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1
- LEVEL II COBOL V2.5.54

NetExpress

この機能は、NetExpress にも含まれています。変更点はありません。ただし、コンパイルは、通常、プロジェクトを通して NetExpress で行われます。その際、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することが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。

NetExpress で使用する機能

アプリケーションを作成する場合、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 [プログラムのコンパイル] 機能を使用して実行することもできます。

また、NetExpress 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。

A.3.1.11 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ためのグラフィカル 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

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support

NetExpress

使用できません。

NetExpress で使用する機能

NetExpress 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ル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。

A.3.1.12 ディレクトリ ファシリティ (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ための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すことができます。文字バージョンは、ディレクトリ ファシリティと呼ばれ、グラフィック バージョンは、ディレクトリ ファシリティ バージョン 2 と呼ばれます。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

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、NetExpress の Win32API デモを参照してください)。

A.3.1.13 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

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

下位互換性を確保するために、変更されないまま NetExpress 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。

A.3.1.14 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ための文字モード 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 V3.1 J

NetExpress

使用できません。

NetExpress で使用する機能

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

A.3.1.15 フォーム

文字モードの画面表示を作成するための文字モードの画面ペインタ。

この機能を含む製品

- Workbench V3.1 J

NetExpress

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、手動で編集することができます。これは標準 COBOL で、データ部に記述されています。

A.3.1.16 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ためのユーティリティ。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

このユーティリティは、NetExpress にも含まれています。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、NetExpress コマンド行で h2cpy コマンドを使用して呼び出します。

A.3.1.17 Hexedit

16 進でファイルを列挙し、編集するためのユーティリティ。

この機能を含む製品

- Workbench V3.1 J

NetExpress

使用できません。

NetExpress で使用する機能

同等の機能はありません。

A.3.1.18 ライブラリ

ファイルを .lbr ファイルにパッケージ化するためのユーティリティ。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

パッケージ化できるファイルは、データ ファイルだけです。また、この操作は IDE を使用する場合にだけ可能になります。

移行

NetExpress でアプリケーションを配布するためにライブラリ ファイルを作成することはお勧めできません。詳細は、「*Build*」を参照してください。

A.3.1.19 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ための構文。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support

NetExpress

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

- CLASS-OBJECT ヘッダーと END CLASS-OBJECT ヘッダーは、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 ヘッダーと END CLASS-OBJECT ヘッダーで囲まれた範囲外の手続き部にクラスがある場合、クラスの読み込み時に実行されなくなります。この場合、"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

旧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、NetExpress で実行する前に再コンパイルする必要があります。

A.3.1.20 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

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V 3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

下位互換性を確保するために、変更されないまま NetExpress 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。

A.3.1.21 スクリーン

文字モードの画面表示を作成するための文字モード画面ペインタ。

この機能を含む製品

- Workbench V3.1 J

NetExpress

使用できません。

移行

変更は必要ありません。スクリーン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、手動で編集することができます。これは、スクリーン節に記述される標準 COBOL です。

A.3.1.22 整列ユーティリティ

ファイルを整列させるユーティリティ。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V 3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

このユーティリティは、NetExpress にも含まれています。変更点はありません。Windows 95、Windows 98 および Windows NT 4.0 に類似した機能があるため、説明しません。

このユーティリティを使用するには、NetExpress コマンド行で mfsort コマンドを使用して呼び出します。旧製品のマニュアルを参照してください。

NetExpress で使用する機能

Windows 95、Windows NT V4.0 の sort コマンドと Mfsort の機能は類似していますが、異なるキーの COBOL データ型や異なるファイル タイプを記録しません。

A.3.1.23 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ための COBOL 構文。

この機能を含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V 3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

下位互換性を確保するために、変更されないまま NetExpress に含まれています。新規アプリケーションには、使用しないでください。

移行

変更は必要ありません。

A.3.1.24 XM

640KB 超のメモリで 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

- COBOL および Workbench V3.1 J 16 ビット
- XM V1.4 J

NetExpress

使用できません。

NetExpress で使用する機能

同等の機能はありません。DOS だけに適用されます。

付録B: COBOL システム ライブラリ ルーチン

この付録では、既存の Micro Focus COBOL システムで使用できる COBOL システム ライブラリ ルーチンを列挙し、これらを NetExpress のアプリケーションで機能させるために必要な変更について説明します。32 ビットのシステムに移行したときは、特定のルーチンが発行されているので、これらについても適宜説明します。

B.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 NetExpress 以前の Micro Focus 製品を列挙します。
- NetExpress - 説明するルーチンが NetExpress 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したアプリケーションを NetExpress に移行するための変更について解説します。たとえば、システム ライブラリ ルーチンを Call-by-Number ルーチンから Call-by-Name ルーチンに切り替える場合などについて説明します。

B.1.1 ルーチン リスト

B.1.1.1 PC_TEST_PRINTER

プリンタの状態をテストするためのライブラリ ルーチン。

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット

NetExpress

NetExpress 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、NetExpress では実装されていません。ただし、この名前を持つダミー ルーチンが含まれており、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

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

B.1.1.2 PC_WIN... **プリンタ ルーチン**

プリンタを処理するためのライブラリ ルーチン。

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット

NetExpress

NetExpress には、このルーチンはありません。

移行

先頭に 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。

B.1.1.3 x"82"**、画面への文字入力**

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress でもこのルーチンを使用できます。

移行

新規開発する場合は、Call-by-Name ルーチン CBL_WRITE_SCR_TTY を使用する必要があります

B.1.1.4 x"83"、キーボードからの文字の読み取り

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress でもこのルーチンを使用できます。

移行

新規開発する場合は、Call-by-Name ルーチン CBL_READ_KBD_CHAR を使用する必要があります

注記

- このサブプログラムでの呼び出しにより、ファンクション キー テーブルが確認され、結果バイトが更新されます。
- このサブプログラムから返された文字がキャリッジリターン文字 (16 進の 0D) であり、キーボードから読み取られた文字がファンクション キー テーブルで識別されている場合、そのテーブルの結果バイトを確認し、実際に押された文字を判断する必要があります。
- 読み取られた文字が 2 バイト文字 (ファンクション キーなど) であり、ファンクション キー テーブルに該当する文字がない場合、両方のバイトがこのサブプログラムに返されます。最初の呼び出しからはヌル バイト (16進の 00) が返され、次の呼び出しからは走査コードが返されます (走査コードの詳細については、『パーソナル コンピュータ テクニカル リファレンス マニュアル』を参照してください) 。
- 実行中のプログラムに割り込むには、ブレークを使用します。ただし、プログラムがこのサブプログラムを呼び出している間にブレークを使用した場合、呼び出されたこのサブプログラムがキーボードから渡される別の文字を読み取るまで何も起こりません。このルーチンからブレークは返されません。

B.1.1.5 x"84"、DOS 割り込みの実行

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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。

B.1.1.6 x"85"、メモリからのバイトの読み取り

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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。

B.1.1.7 x"86"、メモリのバイトへの書き込み

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット

- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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。

B.1.1.8 x"87"、ハードウェア ポートからのバイトの読み取り

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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、Win32API ルーチンを使用します。

B.1.1.9 x"88"、ハードウェア ポートへのバイトの書き込み

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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、Win32API ルーチンを使用します。

B.1.1.10 x"8C"、ファイル名の分割

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

拡張ルーチン CBL_SPLIT_FILENAME を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.11 x"8C"、ファイル名の結合

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット

- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

拡張ルーチン CBL_JOIN_FILENAME を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.12 x"91" 関数 5、デフォルト ドライブの読み取り

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン PC_READ_DRIVE を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.13 x"91" 関数 6、デフォルト ドライブの設定

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J

- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン PC_SET_DRIVE を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.14 x"91" 関数 7、デフォルト ディレクトリの読み取り

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_READ_DIR を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.15 x"91" 関数 8、デフォルト ディレクトリの設定

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_CHANGE_DIR を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.16 x"91" 関数 17、ファイル名変更

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_RENAME_FILE を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.17 x"91" 関数 18、ファイルの削除

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_DELETE_FILE を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.18 x"91" 関数 69、ディレクトリをスキャンするためのルーチン

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

NetExpress でもこのルーチンを使用できます。

移行

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選択されません。一方、16 ビットの Windows では選択されます。

B.1.1.19 x"94"、メモリからの単語の読み取り

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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。

B.1.1.20 x"95"、メモリの単語への書き込み

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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。

B.1.1.21 x"96"、ハードウェア ポートからの単語の読み取り

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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、Win32API ルーチンを使用します。

B.1.1.22 x"97"、ハードウェア ポートへの単語の書き込み

このルーチンを含む製品

- COBOL V3.1J および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。このルーチンは DOS でのみ機能するため、NetExpress 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、Win32API ルーチンを使用します。

B.1.1.23 x"B7" 関数 0、画面からの文字の読み取り

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_READ_SCR_CHARS を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.24 x"B7" 関数 1、画面からの文字の書き込み

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_WRITE_SCR_CHARS を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.25 x"B7" 関数 2、画面からの属性の読み取り

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_READ_SCR_ATTRS 利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.26 x"B7" 関数 3、画面からの属性の書き込み

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_WRITE_SCR_ATTRS を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.27 x"B7" 関数 4、画面からの文字の消去

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_WRITE_SCR_N_CHAR を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.28 x"B7" 関数 5、画面からの属性の消去

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット

- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_WRITE_SCR_N_ATTR を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.29 x"D9"、キーボード状態のテスト

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress でもこのルーチンを使用できます。

移行

このルーチンは、互換性を確保するためだけに組み込まれたものです。そのため、新規アプリケーションについては、ルーチン CBL_GET_KDB_STATUS を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.30 x"E3"、画面寸法の取得

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J

- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_GET_SCR_SIZE を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.31 x"E4"、画面全体の消去

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress でもこのルーチンを使用できます。

移行

このルーチンは、互換性を確保するためだけに組み込まれたものです。そのため、新規アプリケーションについては、ルーチン CBL_CLEAR_SCR を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.32 x"E6"、カーソル位置の設定

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット

NetExpress

NetExpress では、使用できません。

移行

ルーチン CBL_SET_SCR_POS を利用できるようにアプリケーションのコードを再作成する必要があります。

B.1.1.33 "_extname"、環境変数の内容を返すルーチン

このルーチンを含む製品

- COBOL V5.0 J
- COBOL V4.0 J
- COBOL V4.0 J および 5.0 J with Host Emulation Support
- COBOL V3.1 J 32 ビット
- COBOL および Workbench V3.1 J 16 ビット
- 日本語 COBOL/2 V1.1

NetExpress

NetExpress では、使用できません。

移行

_extname ルーチンは、COBOL 構文 DISPLAY ... UPON environment-name (X/Open ダイアレクトの一部) に換わっています。

付録C: 指令とダイアレクト

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することができます。この章では、旧 COBOLシステム以降に変更されたデフォルト設定について説明します。

また、この章では、DIALECT 指令を始めとして 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 V3.1 J、COBOL V4.0 J with Host Emulation Support、および COBOL V5.0 J with Host Emulation Support の USE 指令と wb*.dir ファイルに換わるものです。

SQL に関連した指令については、「Embedded SQL Toolkit」を参照してください。

C.1 デフォルト

NetExpress のデフォルト指令設定は、旧 Micro Focus COBOL システムのデフォルト設定と異なります。特定のデータ項目が予約語となっているため、NetExpress でアプリケーションをコンパイルするときに問題が生じる可能性があります。MF 指令は、Micro Focus COBOLシステムで予約語のデフォルト設定を制御するために使用します。次の表はこの指令の設定を示します。

製品	MF レベル設定
LEVEL II COBOL V2.5.54	1
日本語 COBOL/2 V1.1	4
COBOL V3.1 J 16 ビット	10
COBOL V3.1/V4.0/V5.0 J 32 ビット	10

NetExpress では、MF コンパイラ指令のデフォルト設定は 12 です。この MF レベルと競合するデータ項目をもつプログラムをコンパイルするためには、ソース コードを変更して、現在、予約語となっているデータ項目の名前を変更をするか、旧製品と互換性のある MF レベルでコンパイルします。旧製品の MF レベルを下げると、NetExpress で提供される新しい COBOL 構文を利用できません。

C.2 除外された指令

次の指令は NetExpress にはありません。

BROWSE EXTINDEX OLDSTRSUB STRUCT XNIM

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

INCLUDE-FILLER MFOO REF

C.3 メインフレーム指令

NetExpress 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。エミュレートされた特定のメインフレーム コンパイラは、これらの指令の設定に基づいています。これらの指令は、Workbench V3.1 J、COBOL V4.0、および COBOL V5.0 J with Host Emulation Support で使用できます。詳細は、個々の製品に添付されているマニュアルで説明されています。

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

C.4 DIALECT 指令

旧製品では、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ダイアレクトで 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*) 指令により指定することができます。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。

NetExpress 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 指令にダイアレクトを指定すると、必要な指令が自動的に設定されます。

C.4.1 定義

DIALECT 指令は、次のように定義されます。

ダイアレクト

指定したダイアレクトがコンパイラで受け付けられるようにします。そのダイアレクト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 に格納できる値: ANS85 COBOL370 DOSVS MF OSVS SAA1 SAA2 VSC21 VSC22 VSC23 VSC24

プロパティ

デフォルト: NODIALECT

$SET: initial

コメント

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(ANS85)

  NOCOBOLDIR COMS85 COPYLBR NODBCHECK NODBCS NODBSPACE FLAGCD"W" FLAGSTD"H C2 D2 S2 R O" FOLDCALLNAME"UPPER" FOLDCOPYNAME"UPPER" 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

C.5 予約語

コンパイル時にMF指令を使用すると、予約語リストが拡張されているので、一部のデータ名が違法となります。こ

の問題が発生した場合は、データ名を変更して REMOVE (予約語) 指令を使用するか、あるいは、適当な MF 指令で再度コンパイルしてください。

次の新しい予約語が、MF(11) を使用するとき含まれます。これらの語をデータ名として使用するプログラムは、MF(10) を用いて再コンパイルできます。

ABSTRACT	AS	AT
B-AND	B-EXOR	B-LEFT
B-NOT	B-OR	B-RIGHT
B-XOR	BROWSING	CALLED
CLASS-CONTROL	CLASS-ID	CLASS-OBJECT
COERCION	DEFAULT	DEFINITION
END-INVOKE	END-WAIT	EVENT-POINTER
EXTERNAL-FORM	FACTORY	INHERITING
INHERITS	INVOKE	INVOKED
METHOD	METHOD-ID	MONITOR-POINTER
MUTEX-POINTER	OBJECT	OBJECT-ID
OBJECT-STORAGE	OOSTACKPTR	PRIVATE
PUBLIC	SELF	SELFCLASS
SEMAPHORE-POINTER	SUPER	THREAD-LOCAL
THREAD-POINTER		

次の新しい予約語が、MF(10) を使用するとき含まれます。これらの語をデータ名として使用するプログラムは、MF(9) を用いて再コンパイルできます。

COMPUTATIONAL-6	COMP-6	REPEATED
TYPEDEF		

次の新しい予約語が、MF(8) を使用するとき含まれます。これらの語をデータ名として使用するプログラムは、MF(7) を用いて再コンパイルできます。

IGNORE	KANJI	LOWER
NATIONAL	NATIONAL-EDITED	UPPER
WAIT		

次の新しい予約語が、MF(7) を使用するとき含まれます。これらの語をデータ名として使用するプログラムは、MF(6) を用いて再コンパイルできます。

これらの語のほとんどは既に ANS85 または VSC2 指令の下で予約されているので、ANS85 または VSC2 指令な

しで MF でコンパイルする場合にだけ新しいといえることに注意してください。

ALPHABET	ALPHABETIC-LOWER	ALPHABETIC-UPPER
ALPHANUMERIC	ALPHANUMERIC-EDITED	ANY
BINARY	CLASS	COMMON
COMP-1	COMP-2	COMP-4
COMPUTATIONAL-1	COMPUTATIONAL-2	COMPUTATIONAL-4
CONTENT	CONTINUE	CONVERTING
CYCLE	DAY-OF-WEEK	DBCS
DISPLAY-1	EJECT	END-ADD
END-CALL	END-COMPUTE	END-DELETE
END-DISPLAY	END-DIVIDE	END-EVALUATE
END-IF	END-MULTIPLY	END-PERFORM
END-READ	END-RECEIVE	END-RETURN
END-REWRITE	END-SEARCH	END-START
END-STRING	END-SUBTRACT	END-UNSTRING
END-WRITE	EOL	EOS
EQUALS	EVALUATE	EXCEEDS
FALSE	FUNCTION	GLOBAL
ID	INITIALIZE	LOWLIGHT
NUMERIC-EDITED	ORDER	OTHER
PACKED-DECIMAL	PADDING	PARAGRAPH
PURGE	REFERENCE	REPLACE
SKIP1	SKIP2	SKIP3
SORT-RETURN	STANDARD-2	TEST
TIME-OUT	TIMEOUT	TITLE
TRUE	UNEQUAL	WHEN-COMPILED

次の新しい予約語が、MF(6) を使用するとき含まれます。これらの語をデータ名として使用するプログラムは、MF(5) を用いて再コンパイルできます。

LENGTH	NULL	NULLS

次の新しい予約語が、MF(5) を使用するとき含まれます。これらの語をデータ名として使用するプログラムは、MF(4) を用いて再コンパイルできます。

CONTROL	END-CHAIN	ENTRY
FH-FCD	FH-KEYDEF	GOBACK
LOCAL-STORAGE	NCHAR	PROCEDURE-POINTER
RETURN-CODE	RETURNING	

索引