Micro Focus Net Express[R]

UNIX オプションガイド

第 4 版
2003 年 5 月

このマニュアルでは、Net Express の UNIX オプションについて説明します。UNIX オプションにより、Net Express 上で作成されたアプリケーションを UNIX システムにディプロイできます。

このマニュアルは、UNIX システムに移植する COBOL プログラムを作成するために Net Express を使用するすべてのプログラマとシステム設計者を対象としています。ビジネスコンピューティング、Microsoft Windows の使用法、UNIX システムの使用と管理に関する知識があることを前提としています。

Micro Focus UNIX 製品、Object COBOL for UNIX または Server Express に付属するマニュアルセットの参照が必要な場合もあります。

第 1 章 : はじめに

UNIX オプションにより、アプリケーション開発を UNIX システムから Net Express にオフロードできます。つまり、Net Express のツールを使用して PC 上でアプリケーションを作成し、準備ができたら、Micro Focus COBOL for UNIX がインストールされている UNIX システムに、作成したアプリケーションをディプロイできることを意味します。

注:

- 用語「COBOL for UNIX」は、Micro Focus COBOL for UNIX システムのサポートされているすべてのバージョンを示します。現在サポートされているバージョンは次のとおりです。
    - Micro Focus Server Express
- UNIX オプションを使用するには、必要なソフトウェアを UNIX システムにインストールし、その構成ファイルを変更する必要があります。 これを行うには、UNIX システムへのスーパーユーザのアクセス権と、UNIX システムの構成に関連するスキルが必要です。特に、次の操作に精通している必要があります。
    - ftp の使用
    - ファイルとディレクトリのコピー
    - UNIX テキストファイルエディタの使用 (たとえば、vi、emacs)

ファイルとディレクトリへのアクセス権の設定

これらの作業に精通していない場合は、UNIX システム管理者に問い合わせてください。
また、UNIX システム上での DNS の使用による UNIX システムへの影響、および基本的なネットワークの問題についても、精通しておく必要があります。 わからない場合は、UNIX システム管理者に問い合わせてください。

UNIX オプションを構成するものを次に示します。

- パブリッシャ - アプリケーションを UNIX システムにコピーし、UNIX システム上のビルド処理を制御します。 パブリッシャのセットアップユーティリティにより、ディプロイ処理を制御できます。 たとえば、ディプロイ先の UNIX システム、アプリケーションのコピー先ディレクトリ、コピーファイルに使用するディレクトリなどを指定できます。

  パブリッシャの使用法の詳細については、『*アプリケーションのパブリッシュ*』の章を参照してください。パブリッシャの構成方法の詳細については、『*パブリッシャの設定*』の章を参照してください。

- インポートウィザード - Object COBOL for UNIX アプリケーションを Net Express にインポートできます。詳細については、『*Net Express への UNIX アプリケーションのインポート*』の章を参照してください。

- Samba - 標準的な Windows のネットワークを使用して、UNIX システムへのリモートドライブアクセスを提供します。

Net Express COBOL で使用できる構文には、COBOL for UNIX と互換性がないものもあります。Net Express 上で WARNINGS(2) コンパイラ指令を使用してコンパイルした場合には、コンパイラが非互換性構文を見つけたときに警告メッセージを表示します。 ただし、コンパイラが互換性のないすべての箇所にフラグを立てることはできません。互換性のない箇所の一部については、自分自身でチェックする必要があります。『*移植性の問題*』の章では、コンパイラが非互換性としてフラグを立てることができる構文を一覧表示し、自分自身でチェックする必要のある非互換の構文について説明します。

アプリケーションのパブリッシュについての概要

UNIX システムにアプリケーションをディプロイし、リビルドする処理は、「パブリッシュ」と呼ばれます。 UNIX オプションには、パブリッシャと呼ばれる、パブリッシュ処理を処理するツールがあります。

注: Net Express のデータアクセスウィザードを使用して生成した アプリケーションは UNIX システムにパブリッシュできません。

ディプロイ可能なアプリケーションを作成するための開発処理は次のとおりです。

1. Net Express ツールを使用してアプリケーションを作成し、開発します。

2. WARNING"2" コンパイラ指令を設定します。 アプリケーションをコンパイルします。 COBOL for UNIX と非互換の構文にはフラグが立てられます。 必要に応じて構文を変更します。COBOL for UNIX に問題を生じさせる可能性がある他の構文の詳細については、『*移植性の問題*』の章を参照してください。 必要に応じて再コンパイルします。

3. エラーなしでアプリケーションをビルドできた場合は、次の操作を行います。

    a. このプロジェクトに対してパブリッシャを設定します。詳細については、『*パブリッシャの設定*』の章を参照してください。 通常は、プロジェクトに対してこの設定を一度のみ行います。

    b. アプリケーションをパブリッシュします。詳細については、『*アプリケーションのパブリッシュ*』の章を参照してください。

4. アプリケーションを UNIX システムで実行します。ランタイムエラーが発生した場合は、Net Express を使用して修正します。そのアプリケーションをリビルドし、再度パブリッシュします。

Net Express を使用して UNIX システム上の既存の COBOL アプリケーションを編集し、リビルドする場合には、そのアプリケーションを Net Express にインポートする必要があります。このインポート方法については、『*Net Express への UNIX アプリケーションのインポート*』の章を参照してください。

サーバコントロールプログラム

サーバコントロールプログラム (SCP) には、Net Express と UNIX システムの間の通信を可能にする機能の標準セットが付属しています。 アプリケーションをパブリッシュするには、まず SCP を UNIX システムにインストールする必要があります。 SCP のインストール方法について Server Express のマニュアルを参照するか、SupportNet から SCP をダウンロードしてください。

PowerTerm

Ericom 社の PowerTerm は日本語環境では正しく動作しないため、日本語版の Net Express 製品には含めていません。

PowerTerm は、PC から UNIX システムへのターミナル接続 (Telnet) を開始できるように、UNIX オプションに付属しています。 PowerTerm を使用すると、次のことが可能になります。

- ディプロイしたアプリケーションを PC からテストする
- PC からリモートシステム上で COBOL for UNIX のツール (たとえば Editor、Animator) を使用する

PowerTerm が、リモートシステム上で COBOL ツールを使用可能にするので、「PowerTerm」ウィンドウの下部のボタンバーが標準の設定と変わります。 UNIX オプションがボタンバーの設定を変更し、F1 ～ F10 のボタンの他に、/A (Alt)、/C (Ctrl) ボタンが追加されます。 Toolbox (COBOL V3.2 および V4.0 for UNIX) または 開発環境メニューシステム (Object COBOL V4.1 および Server Express) に付属するツールとして、ボタンが設定されます。Toolbox や開発環境メニューシステムを操作するには、F1 ～ F10 のファンクションキーと、Alt キーおよび Ctrl キーを使用してください。 このようにボタンバーを設定することによって、マウスでボタンバーのボタンをクリックして COBOL ツールのメニューを操作できます。
PowerTerm も、デフォルトで SCO-ANSI のターミナルエミュレーションとなるように、UNIX オプションによって構成されます (ターミナル情報は COBOL for UNIX 製品に付属)。そのため、デフォルトの 24 行ではなく、COBOL ツールに合うように 25 行で表示するように設定されます。
他のキーやホットキーをエミュレートするようにボタンバーを設定する必要がある場合、または表示行数を再設定したい場合は、PowerTerm のヘルプを参照してください。

大文字と小文字の区別

UNIX システムにアプリケーションをパブリッシュする際に、定数 (特にファイル名) の大文字と小文字の区別によって問題が発生する場合があります。 一般的に、PC ではファイル名の大文字と小文字の区別は無視されます。たとえば、filename1 は FILENAME1 または FiLeName1 と同等です。UNIX システムでは、filename1 は FILENAME1 とは異なるファイル名と見なされます。 その結果、次のようになります。

- Net Express 開発環境では (ユーザが読みやすいように) ファイル名を大文字にマ

ップしたり、ファイルシステムの大文字小文字を表示したりしますが、アプリケーションがビルドされるときに大文字と小文字がそのままである保証はありません。

- PC で作成されたプログラムの CALL 文と COPY 文で使用される文字では、大文字と小文字が混在する場合があります。 この大文字と小文字の混在は PC では問題ありませんが、UNIX システムでは重要です。

- Microsoft や Novell の古いプロトコルを使用する PC ベースのファイルサーバは、ファイル名の大文字と小文字の区別を無視します。 新しいファイルサーバ (たとえば、Windows NT Server または Novell V4) では、大文字と小文字の混在をサポートする場合があります。

たとえば、**myapp.cbl** というプログラムを作成したとします。アプリケーション、およびプログラムを保存するために使用したオペレーティングシステム、または UNIX システムへの接続に使用しているネットワークプロトコルによって、実際のファイル名は、たとえば、**myapp.cbl** (Windows NT) または **MYAPP.CBL** (Novell の一部のバージョン) となります。 Windows ではこれらのファイル名はすべて同じであると見なされますが、UNIX ではすべて異なるファイルと見なされます。 そのため、コード call myapp.cbl を含むプログラムを作成した場合には、呼び出しは Windows では予期したとおりに機能しますが、UNIX システムでは問題が発生します。パブリッシュされたプログラムのファイル名が **myapp.cbl** または **Myapp.cbl** である場合があるからです。
これらの問題を解決するには、ファイル名を特定の方法で処理するように パブリッシャを構成できます。詳細については、『*パブリッシャの設定*』の章を参照してください。

第 2 章：移植性の問題

ここでは、Net Express コンパイラが、警告または備考メッセージを使用してフラグを立てることのできる Net xpress と COBOL for UNIX との間の非互換性構文について説明します。ただし、Net Express コンパイラはすべての非互換性の箇所にフラグを立てることができないため、ここでは UNIX システムにプログラムを移植する前に手動でチェックして変更する必要がある構文についても説明します。

Server Express をインストール済みの UNIX システムにパブリッシュする場合には、「***Server Express 以外***」と表記されている項に記述されている移植性の問題は無視できます。

構文のチェック

Net Express および COBOL for UNIX の両方の INTLEVEL 指令はデフォルトで 2 に設定されています。これは移植可能な中間コードを作成するために必要な値です。ただし、Net Express COBOL の構文の中には、INTLEVEL"2" を指定してコンパイルした場合でも、UNIX システムで実行できない中間コードを生成するものもあります。

このため、WARNINGS"2" コンパイラ指令を設定した場合、Net Express コンパイラは UNIX システムに移植されたときに問題が発生する可能性のある構文構造にフラグを立てます。

フラグが立つ構文

Server Express 以外

プログラムを INTLEVEL"2" および WARNINGS"2" 指令を指定して Net Express でコンパイルした場合には、構文構造にフラグが立てられます。フラグは備考メッセージまたは警告メッセージの形式です。CHANGE-MESSAGE コンパイラ指令を使用して、メッセージの重大度を備考または警告から重大な警告に上げることができます。 次の構文にフラグが立てられます。

- 160 バイトを超える定数。COBOL for UNIX の制限は現在 160 バイトです。
- 呼び出し規約。Net Express コンパイラは COB0L for UNIX コンパイラで受け付けない呼び出し規約を受け付けます。たとえば、Net Express では呼び出し規約 256 と 512 の両方を受け付けますが、COBOL for UNIX では受け付けません。
- 空白文字を含むファイル名。
- それぞれ異なる呼び出し規約が指定されたエントリポイント。 COBOL for UNIX では、最初のエントリポイントで残りのエントリポイントの呼び出し規約が設定されるものと仮定されます。
- 4 バイトを超える BY VALUE 項目。COBOL for UNIX では、CALL 文の BY VALUE で渡される最大バイト数は 4 バイトです。これより大きな値を渡そうとした場合には、オペレーティングシステムで障害が発生する場合があります。

- IS EXTERNAL DEFINITION および IS EXTERNAL REDEFINITION 指定。これらの指定は Net Express のみで使用できます。
- 255 (COBOL for UNIX の最大数) を超えた場合のエントリポイントのパラメータ数。
- マルチスレッド構文。この構文は、COBOL for UNIX では使用できません。
- ソースプログラムで検出される OO COBOL サポート。OO COBOL をサポートしているのは、Object COBOL V4.1 for UNIX 以降のバージョンのみです。

フラグが立たない構文

次の構文にはフラグを立てません。プログラム中に存在するかどうかをチェックする必要があります。

- UNIX では機能しないライブラリルーチン (つまり、COBOL for UNIX のマニュアルで DOS、OS/2、または Windows のみと示されているものと、PC に特有のもの)。たとえば、PC_FIND_DRIVES などの PC_ library ルーチンは、COBOL for UNIX では使用できません。一部の call-by-number ライブラリルーチンは、Net Express と COBOL for UNIX では異なります。たとえば、マウスルーチンを使用することも、x"AF" ルーチンを使用してマウス機能を制御することもできません。
- MF_CLIENT 呼び出し、および Windows API 呼び出しは UNIX では使用できません。
- Panels2 への呼び出し。これらの呼び出しはサポートされていません。
- .dll ファイルへの明示的な呼び出し。
- UNIX で無効なファイル名。たとえば、拡張子 .dat をもつファイル、または空白文字やドライブ識別子を含むファイル名などです。
- 定数中の埋め込み PC 文字 (つまり、印字不可能な文字)。
- ファイル記述中の COMP-5 データ項目。ファイルレコード記述で COMP-5 データ項目を指定した場合には、バイト順位が PC と UNIX システムでは異なる可能性があることに注意してください。
- プログラムがオブジェクト指向の構文を含む場合、COBOL for UNIX に対しては、ソースコードの先頭で $SET 文を使用するなどして、MFOO 指令を設定する必要があります。詳細については、次を参照してください。
- 添字に算術が含まれているプログラムの場合には、Server Express より前に、そのソースコードを UNIX ベースの COBOL システムに移植することはできません (中間コードの場合を除く)。同様に、Net Express と Mainframe Express に先立って、このようなソースコードを Windows ベースの COBOL システムに移植することはできません。

COBOL for UNIX と Net Express COBOL の相違については、次の点についても注意してください。

- バイト順位。 アプリケーションがプログラム間でパラメータの受け渡しをする場合には、システムがパラメータを渡す方法に注意する必要があります。 UNIX システムの中には、PC で使用される逆のバイト順位ではなく、COBOL のバイト順位を使用するものもあります。このため、Intel チップを搭載した PC で動作するテスト済みのプログラムが、SPARC システムなどに移植したときに予期しない動作をすることがあります。 この問題は、COBOL と COBOL 以外のプログラムの間でパラメータを受け渡す場合や、データを受け渡しするプログラムがそれぞれ異なるマシンで実行されている場合のみに発生します。

- デフォルトのファイルタイプは PC と UNIX で異なります。

- 行順ファイルに使用される区切り文字は、PC と UNIX で異なります。

LINKCOUNT 指令の使い方

Server Express 以外
COBOL for UNIX コンパイラがデータ項目に記憶スロットを割り当てる機構は、Net Express コンパイラのものと異なります。このため、Net Express を使用してコンパイルした正常に動作するプログラムを UNIX システムにパブリッシュすると、次のコンパイラエラーが発生する可能性があります。

```
0067 Please recompile using a larger value for the LINKCOUNT directive
```

プログラムを UNIX システムにパブリッシュしたときにこのコンパイルエラーが発生した場合には、COBOL 原始プログラムで $SET 文 を使用して、エラーが発生したプログラムに LINKCOUNT 指令を設定する必要があります。

デバッグと .idy ファイル

Server Express 以外
.idy ファイルの形式は、Net Express と、Object COBOL Development System 4.1 および以前のバージョンの COBOL for UNIX で異なります。このため、これらのシステムの Net Express で作成された **.idy** ファイルは使用できません。

オブジェクト指向プログラム

Server Express 以外
Object COBOL for UNIX では、クラスと OO プログラムをチェックするために MFOO コンパイラ指令を指定する必要があります。この指令によって OO COBOL の予約語がロードされます。Net Express は MFOO 指令を受け付けますが、それを無視します。これは、すべての OO COBOL の予約語がデフォルトですでにロードされているからです。

OO プログラムまたはクラスを Net Express で開発し、それを COBOL for UNIX を使用して再コンパイルするには、MFOO 指令を指定する必要があります。この指定は、最良の方法として、ソースコードで $SET 文 を使用して行います。

次のことに気を付けてください。

- ユーザクラスがクラスライブラリからデータとともにクラスを継承した場合には、これらのユーザクラスのコンパイルは UNIX で失敗する可能性があります。これは、クラスライブラリが Net Express のクラスライブラリと異なるからです。データなしでクラスを継承することをお奨めします。

- Net Express ではクラス制御節の意味が変更されました。この変更により、COBOL for UNIX で再コンパイルするときに問題が発生する場合があります。クラス制御項の形式は次のとおりです。

  ```
  class-control. logicalClassName is class "physicalFileName".
  ```

  Net Express では、logicalClassName の対象範囲はそれぞれのクラス、または、それが存在するプログラムです。これ以外の範囲には影響しません。OO COBOL の以前のバージョンでは、logicalClassName の対象範囲はグローバルな範囲です。複数のコンパイル単位で指定された logicalClassName に複数のエントリが存在する場合には、それらのエントリは一致する必要があります。一致しない場合には、ランタイムエラー 119 が発生します。

- COBOL for UNIX では、クラスはそのクラスがロードされるときに一度実行されるメソッド以外に手続き部をもつことができます。この構造は Net Express では削除され、Initializeclass クラスメソッドで置き換えられました。このメソッドはクラスが最初にロードされたときに、OO COBOL ランタイムシステムにより呼び出されます (結果として、COBOL for UNIX の手続き部を使用するのと同じ効果があります)。このため、Initializeclass メソッドは Object COBOL for UNIX では自動的に実行されません。

CGI アプリケーション

COBOL for UNIX は、CGI アプリケーションが要求するさまざまな構文およびランタイムサポートを、複数のレベルでサポートしています。たとえば、Object COBOL for UNIX には ACCCGI ランタイムサポートモジュールが導入されています。

UNIX オプションは、COBOL for UNIX 製品すべてについて、標準レベルの CGI 構文およびランタイムサポートを提供しています。COBOL for UNIX 製品が独自の構文またはランタイムサポートを提供している場合には、UNIX オプションは自動的にこれを使用します。

COBOL への Java 構造体の渡し方

Net Express で Java アプリケーションが起動され、対応する Java パラメータが **mfcobol.lang.Datatype** インターフェイスを実装するオブジェクトの場合には、Java クラスのデ

ータを COBOL プログラムの連絡節と Object COBOL メソッドに渡ことができます。 Server Express V2.0.11 以降を使用している場合は、この機能を使用して Net Express から UNIX システムにパブリッシュできます。この機能は、COBOL for UNIX システムの以前のバージョンでは使用できません。

第 3 章 : パブリッシャの設定

パブリッシャを使用してアプリケーションをパブリッシュするには、次のことを行う必要があります。

1. アプリケーション用の Net Express プロジェクトを作成します。 まったく新規にプロジェクトを作成するか、または既存のソースファイルに基づいてプロジェクトを作成します。アプリケーションの作成とプロジェクトの使用法については、Net Express のヘルプを参照してください。

   既存の UNIX アプリケーションに基づいてアプリケーションを作成する場合には、その既存のアプリケーションを Net Express にインポートします。詳細については、『*Net Express への UNIX アプリケーションのインポート*』の章を参照してください。

2. UNIX システムを設定します。

   - ターゲットの UNIX システムでパブリッシャのログインを選択します。
   - パスワードなしで透過的にログインできるように **.rhosts** ファイルを設定します。詳細については、付録『*Samba と SCP のインストール*』を参照してください。

プロジェクトを作成し、UNIX システムを設定した場合には、パブリッシャのセットアップダイアログを使用してプロジェクトに対してパブリッシャを構成できます。

注:次に示すパブリッシャのセットアップダイアログには、UNIX システムのディレクトリやファイル情報を入力するためのいくつかのテキスト領域があります。たとえば、build ディレクトリや copyfile ディレクトリなどです。 これらのファイル名とディレクトリには絶対パス (スラッシュ (/) で始まる) と相対パスのどちらでも使用できます。 相対パスは指定したユーザのホームディレクトリを基にします。 たとえば、ユーザ foo のホームディレクトリが /usr/foo の場合には、そのユーザはビルドディレクトリを /usr/foo/mybuildir または mybuilddir として指定できます。どちらの指定も同じです。

環境変数も、サーバに渡されるパスでのワイルドカードの展開も指定できないことに注意してください。

次の説明では新規設定を仮定しています。 パブリッシャのセットアップダイアログを使用して、既存の構成を編集できます。その場合は、以前に設定された値がデフォルトとして表示されます。

1. IDE の [**UNIX**] メニューをクリックします。

2. [**セットアップ**] をクリックします。 次のダイアログボックスが表示されます。

図 3-1: 「UNIX オプションセットアップ」ダイアログボックスの [プロジェクト] タブ

タブをクリックして、次の設定を行います。

- プロジェクト
- サーバ
- その他の UNIX オプションの詳細

タブをクリックすると、各構成の詳細を入力するためのフォームが表示されます。 以前に構成の設定の詳細を指定した場合には、その構成の値が表示されます。

各プロジェクトに対して、プロジェクトのパブリッシュ先のさまざまなサーバを設定できます。 各サーバに対して、異なる構成情報を指定できます。

プロジェクトの詳細設定

プロジェクトの設定情報を入力するには、必要に応じて「**UNIX オプションセットアップ**」ダイアログの [**プロジェクト**] タブ (デフォルトのタブ) をクリックします。

最初のフィールドには、現在アクティブなサーバの構成が表示されます。 サーバが現在選択されていない場合には、このフィールドには「**何も選択されていません**」が表

示されます。UNIX システムに CGI アプリケーションをパブリッシュしたい場合は、「**CGI サポートを使う**」をチェックします。

ボタンを使用して、さらに詳細を設定できます。これらのボタンについては、次で説明します。

ファイル名マッピング

UNIX システムへパブリッシュされるときのファイル名、位置、各ファイルの種類を指定するには、[**ファイル名マッピング**] ボタンを使用します。CALL 文などの文で入力したファイル名と一致させるために、このマッピング方法の指定が必要になる場合があります。

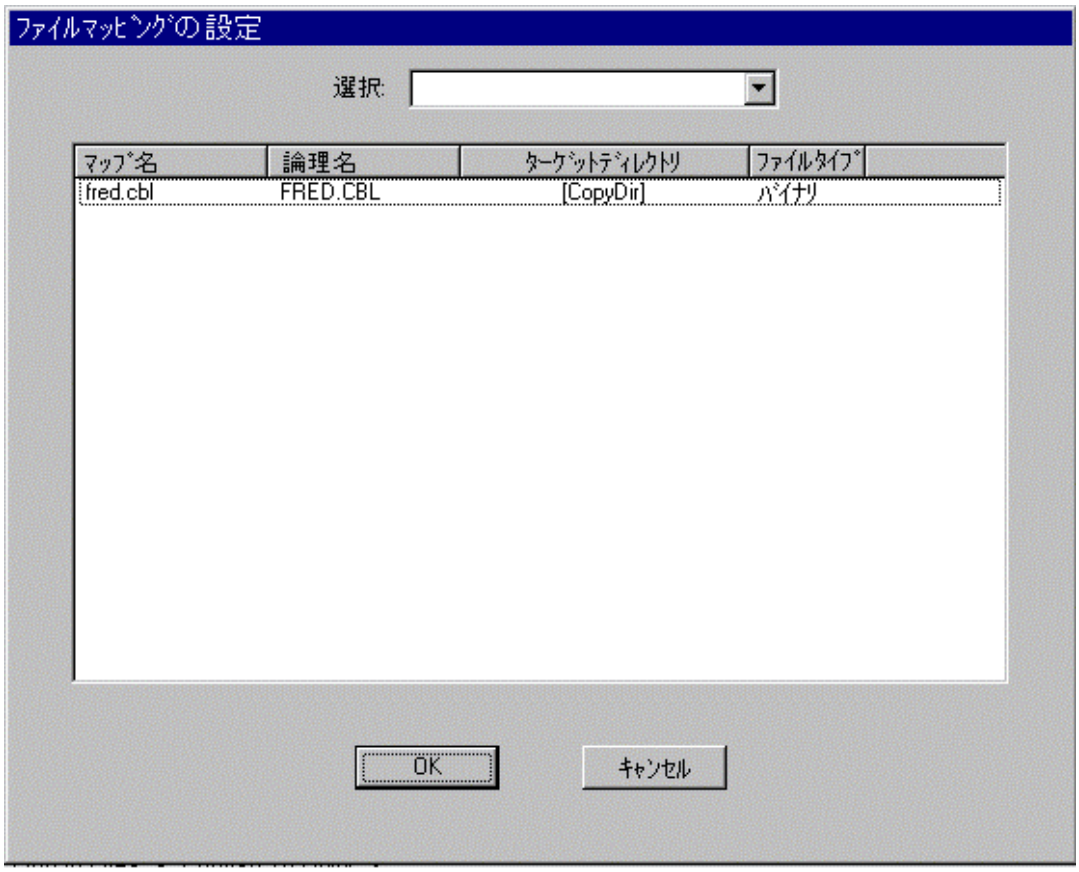

図 3-2:「ファイルマッピングの設定」ダイアログボックス

列「**マップ名**」は、UNIX へコピーされたときの実際のファイル名を大文字と小文字を区別して指定します。デフォルト値は、Windows ファイルシステムが決定するファイル名になります。

列「**論理名**」は、Net Express で表示されたファイル名を表示します。論理名は常に大文字であり、ユーザが編集できません。

たとえば、プログラムに CALL 文 `call "MYapp"` があるとします。 ファイル名マッピング手続きにより、Windows で **Myapp.cbl** として認識されているファイルが UNIX システム上では **MYapp.cbl** として作成されるように明示的に指定できます。

ビルド処理の一部としてマップされた名前で作成されたすべてのファイルでは、大文字と小文字が同じようにマップされます。 たとえば、**MYapp.cbl** は **MYapp.int** となり、**MYAPP.cbl** は **MYAPP.int** となります。

「**ファイルマッピングの設定**」ダイアログを使用してファイル名を変更できません。たとえば、**Myapp.cbl** を **Myapp2.cbl** に変更できません。

ファイル名マッピングと大文字と小文字の区別については、『*はじめに*』の章にある『*大文字と小文字の区別*』の項を参照してください。

列「**ターゲットディレクトリ**」は、ファイルのコピー先の論理ディレクトリを指定します。論理ディレクトリは、「**論理ディレクトリ**」ダイアログで定義されます。

列「**ファイルタイプ**」は、ファイルを UNIX サーバにバイナリファイルとして転送するか、またはテキストファイルとして転送するかを指定します。 新規ファイルのデフォルトは、テキストファイルです。

ファイルをテキストとして指定した場合は、次のようになります。

- 行の終了文字はすべて、PC 形式の文字 CR/LF から、UNIX 形式の文字 LF に変換されます。
- どのような検索 / 置換操作に対しても正しいファイルとなります。
- 「**サーバ設定**」ダイアログで「**EUC サーバ**」チェックボックスをチェックした場合は、EUC 文字コード体系に変換されます。

注：プロジェクト内のバイナリファイルを、テキストファイルとして定義しないでください。プロジェクト内のバイナリファイルをテキストファイルとして定義した場合、それらのファイルはサーバで壊れます。

[**選択**] プルダウンメニューは、事前に定義したパターンに一致する一連のファイルを簡単に選択できます。たとえば、COBOL ファイルすべて (***.cbl**)、またはユーザ自身が選択したパターンです。

ファイルマッピング情報項目を編集する方法は、2 通りあります。

- 編集したい行を選択します。選択した行内で編集したいフィールドをクリックします。ファイル名などの項目については、編集フィールドが表示されます。他のフィールド (ファイルタイプなど) については、有効な値のプルダウンリストが表示されます。

- [**選択**] プルダウンメニュー、または Windows の標準的な複数選択の方法を使用して、複数の項目を選択します。編集したい行で右クリックすると、その列の有効な操作を含むコンテキストメニューが表示されます。選択したファイルすべてに操作が適用されます。

その他のビルドオプションの設定

選択したプロジェクトに対して、[**その他のビルドオプション**] ボタンをクリックすると、プロジェクトをビルドするときに cob コマンド行に追加されるコンパイラ指令を指定できます。指定された指令は、現在のプロジェクトに対して定義されたすべてのサーバに適用されます。必要な指令を入力するためのダイアログが表示されます。

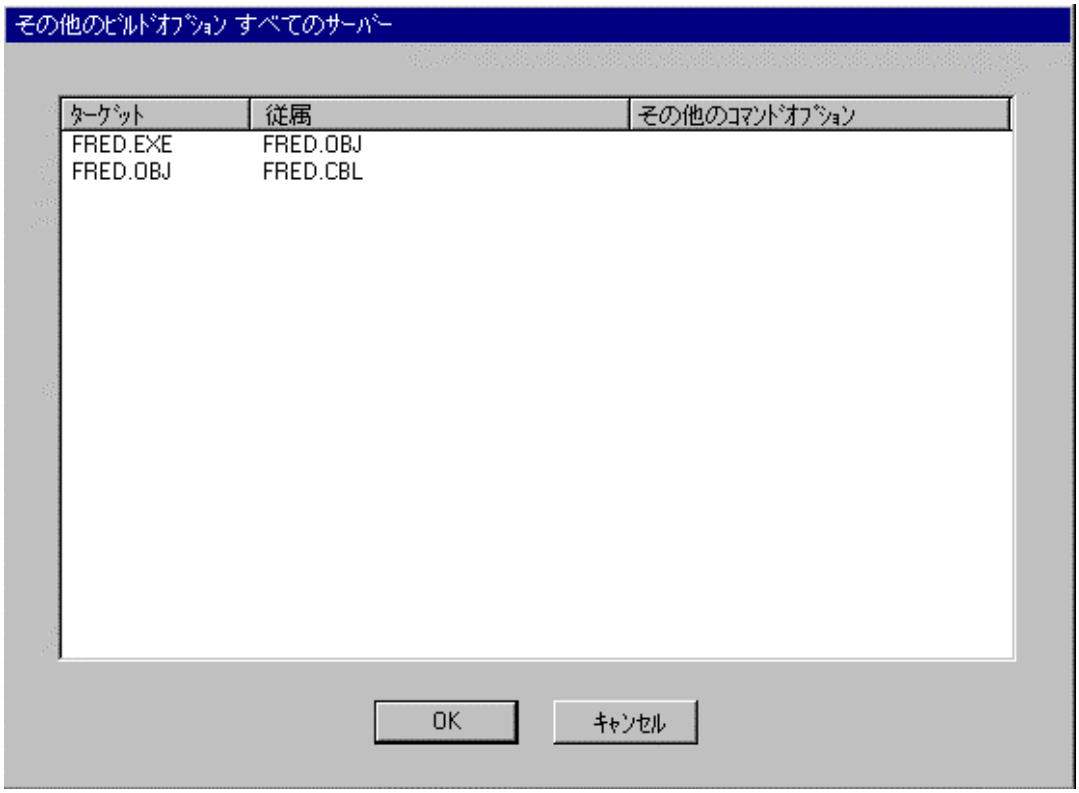

図 3-3: 「その他のビルドオプション すべてのサーバー」ダイアログボックス

列「**ターゲット**」は、作成されている論理名です。たとえば、**CGIPRG1.INT** です。

列「**従属**」は、ターゲットを作成するために使用される論理名をリストします。たとえば、**CGIPRG1.CBL** です。

列「**指令**」は、追加 cob コマンド行オプションを指定するフィールドです。設定を編集するには、編集したい行を選択し、選択した行で列「**指令**」をクリックします。または、編集したい行でマウスを右クリックし、メニュー項目 [**編集オプション**] を選択します。

注:ユーザが入力した指令は、パブリッシャによってチェックされません。そのため、変更されずに cob コマンド行に渡されます。

論理ディレクトリの設定

プロジェクトに対して論理ディレクトリ名を追加したり、削除したりできます。[**プロジェクト**] タブの [**論理ディレクトリ**] ボタンをクリックします。ダイアログボックスが表示され、論理ディレクトリを入力できます。

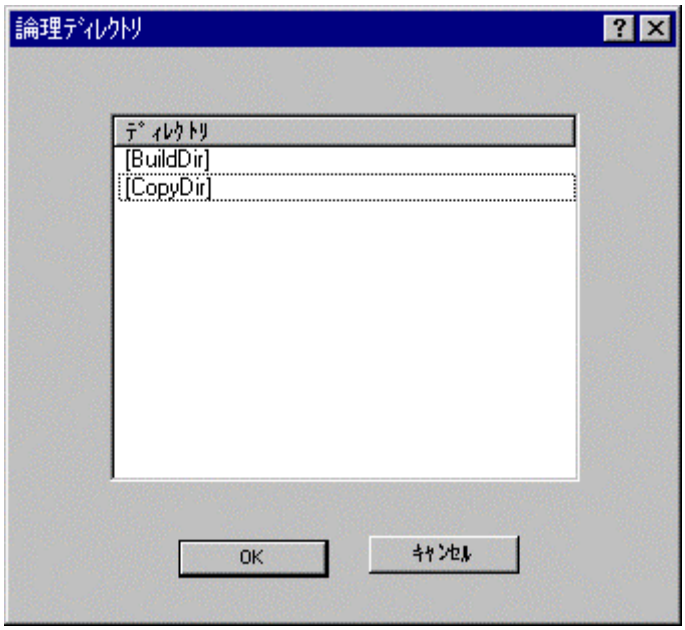

図 3-4:「**論理ディレクトリ**」ダイアログボックス

2 つの特別な論理ディレクトリが常時、表示されていますが、削除できません。

- [**BuildDir**] は、ビルド操作すべての基本となるディレクトリです。また、複数のユーザが同じ領域にアクセスしないようにロックされているディレクトリです。Net Express V2.0 UNIX オプションで使用可能な「**ビルドディレクトリ**」設定に対応しています。
- [**CopyDir**] は、Net Express V2.0 UNIX オプションの「**サーバコピーブックディレクトリ**」設定と互換性があります。

リストコントロール内で右クリックすることによって、論理ディレクトリを削除したり作成したりできます。論理ディレクトリ名はフリーフォーマットテキストであり、有効なテキスト文字を含むことができます。

論理ディレクトリはプロジェクト全体の設定です。論理ディレクトリは、論理的にファイルをコピーする場所を指定するために、「**ファイル名マッピング**」ダイアログで使用されます。また、「**サーバ設定**」ダイアログの [**ディレクトリ指定**] タブ内で、サーバ間の基礎となる実際の値が割り当てられます。

注:論理ディレクトリを削除した場合、この値を使用していたマッピング項目は、かわりに [**BuildDir**] を使用するように更新されます。

特定のプロジェクトの検索 / 置換パターンの設定

特定のプロジェクトのパターンを編集し、置換するには、[**検索 / 置換**] ボタンをクリックします。現在の検索 / 置換パターンのリストと適用されるファイルが表示されます。次に例を示します。

図 3-5:「**検索 / 置換の設定**:**すべてのサーバー**」ダイアログボックス

列「**検索パターン**」では、正規表現を指定します。正規表現は、付録『*正規表現*』で定義しています。

列「**置換パターン**」では、検索パターンが成功したときに置換するテキストを指定します。置換メタ文字については、付録『*正規表現*』で定義しています。

注：ファイル名が列「**ファイル指定**」で定義したファイルに一致しても、「**ファイルマッピングの設定**」ダイアログ内でテキストファイルとして定義されていなければ、処理されません。

パブリッシュ操作中は、リストボックスの表示順にパターンが実行されます。パターンの順番を変更するには、行を選択し、変更する位置へパターンをドラッグアンドドロップします。

パターンを編集するには、カーソルでそのパターンを選択し、左クリックします。検索パターンを編集する方法の例を次に示します。

11. マウスを使用してパターンを選択します。
12. 左クリックします。
13. パターンを編集します。

右クリックすることで、パターンを編集したり、新しいパターンを作成したりできます。新しいパターンを作成し、リストの末尾にそのパターンを追加する方法を次に示します。

14. 画面の空白領域を右クリックします。次のダイアログが表示されます。

図 3-6：「検索/置換の編集」ダイアログボックス

15. ダイアログに新しいパターンを入力します。

16. [**OK**] をクリックします。

    「**検索/置換の設定**」リストのパターンのリストの末尾に新しいパターンが追加されます。

新しいパターンを作成し、リストの特定の場所にそのパターンを追加する方法を次に示します。

17. 新しいパターンを追加したい行の直前の行のフィールドの 1 つを右クリックします。新しいパターンを追加するか、または現在のパターンを編集するかどうかを尋ねられます。

18. ダイアログに新しいパターンを入力します。

19. [**OK**] をクリックします。

マウスの右ボタンを使用してパターンを編集する方法を次に示します。

20. 変更したいパターンを含む行を右クリックします。ダイアログボックスのフィールドには、その行で定義されたパターンの指定とファイルの指定がすでに設定されています。

21. 必要に応じてパターンの指定とファイルの指定を編集します。

22. [**OK**] をクリックします。

サーバの詳細設定

サーバの設定情報を入力するには、[**サーバー**] タブをクリックします。次のフォームが表示されます。

図 3-7:「UNIX オプションセットアップ」ダイアログボックスの [サーバー] タブ

リストボックスは、現在定義されているサーバを表示します。これらの 1 つを選択した場合は、このタブのボタンを使用して行った変更が選択したサーバに適用されます。「新しいサーバー」エントリで、新しいサーバを定義できます。このエントリを選択すると、新しいサーバ名を指定するためのダイアログが表示されます。

このタブのボタンを使用して、さらに詳細に設定できます。 これらのボタンについては、次で説明します。

サーバ設定

[**設定**] ボタンをクリックすると、次のタブ付きのダイアログが表示されます。

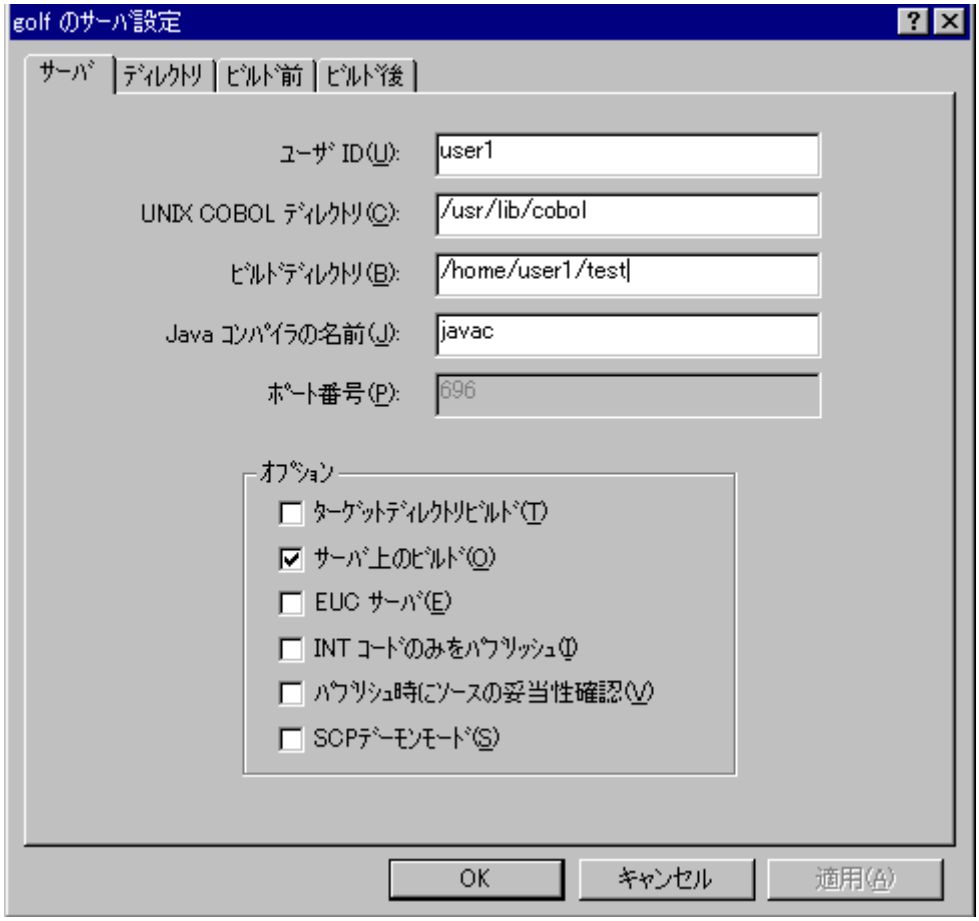

図 3-8:「*サーバ名*のサーバ設定」ダイアログボックス

タブをクリックして、次の設定を行います。

- **サーバ**
- **サーバのディレクトリ**
- **ビルド前のコマンド**
- **ビルド後のコマンド**

サーバの詳細設定

サーバの詳細を設定するには、必要に応じて「サーバ設定」ダイアログの [サーバー] タブ (デフォルトのタブ) をクリックします。 ここで指定する設定は、指定したサーバのみに適用されます。

次の詳細を入力してください。

ユーザ ID **サーバへの接続に使用するログイン識別子。 デフ**

	ォルトは、Windows にログインするときに使用するユーザ ID です。
UNIX COBOL ディレクトリ	Micro Focus COBOL for UNIX が格納されている UNIX システムのディレクトリ。このフィールドに詳細を入力すると、「**UNIX オプションセットアップ**」ダイアログ上の [**サーバの容量**] ボタンが有効化されます。
ビルドディレクトリ	プロジェクトのパブリッシュ先の UNIX システムのディレクトリ。このフィールドに詳細を入力すると、「**UNIX オプションセットアップ**」ダイアログ内の [**サーバの排他制御**] ボタンが有効化されます。
Java コンパイラの名前	プロジェクトのパブリッシュ先である UNIX システム上の Java コンパイラの名前。デフォルトは javac です。
ポート番号	SCP デーモンが監視するポート番号。このフィールドは、「**SCP デーモンモード**」チェックボックスをオンにした場合にのみ有効です。
ターゲットディレクトリビルド	Net Express ターゲットの種類 (たとえば、Debug) で指定したサブディレクトリにアプリケーションをビルドしたい場合は、これをチェックにします。このディレクトリは**ビルドディレクトリ**で指定されたディレクトリ下に作成されます。
サーバ上のビルド	ファイルをリビルドしないで UNIX システムにコピーしたい場合には、チェックを外します。
EUC サーバ	UNIX システム上で EUC サポートを有効化したい場合は、これをチェックします。
INT コードのみをパブリッシュ	ソースコードではなく、中間コードからプロジェクトをビルドしたい場合は、これをチェックします。
パブリッシュ時にソースの妥当性確認	各パブリッシュ操作でソースを検証したい場合に、これをチェックします。これは UNIX システム上の変更されたファイルを検索します。
SCP デーモンメソッドを有効化	SCP デーモンモードを有効化したい場合に、これをチェックします。SCP デーモンメソッドについては、付録『*Samba と SCP のインストール*』を参照してください。必要に応じて、「**ポート番号**」フィールドでポート番号を変更できます。

ディレクトリの指定

ディレクトリの詳細を設定するには、「**サーバー設定**」ダイアログの [**ディレクトリ**] タブをクリックします。次のダイアログが表示されます。

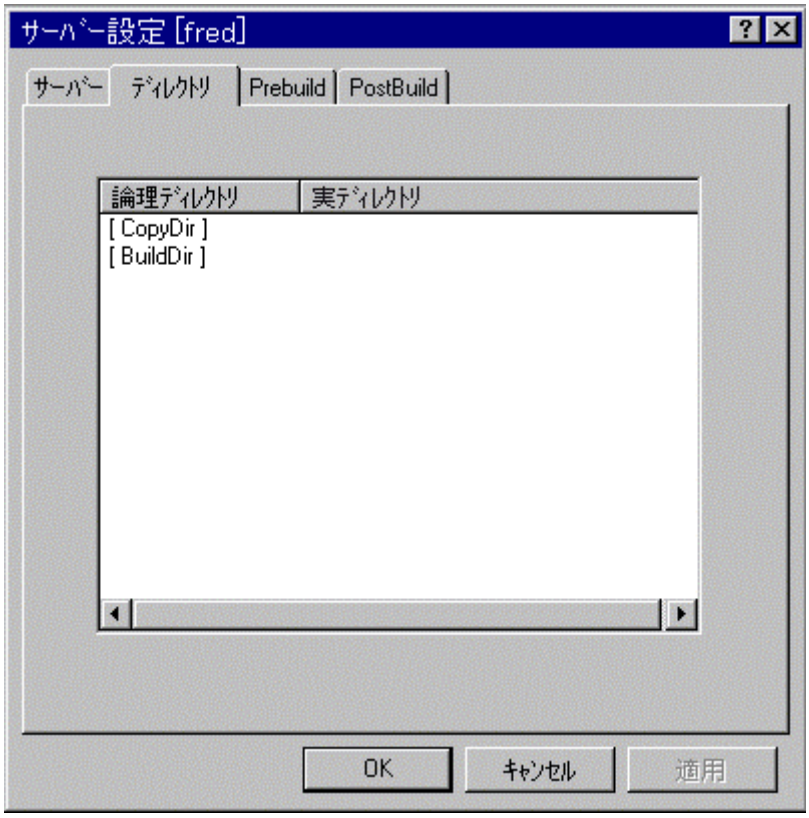

図 3-9:「*サーバ名のサーバ設定*」ダイアログボックス

リストボックスが表示されます。このリストボックスには、([**プロジェクト**] タブから)「**論理ディレクトリ**」ダイアログで定義した論理ディレクトリと現在のサーバ上の実際の物理ディレクトリが含まれます。

[**BuildDir**] の値は、[**サーバー**] タブで入力した値と常に一致します。どちらかの値を変更すると、自動的にもう一方へ反映されます。

どの論理ディレクトリにも値がない場合は、最初に使用された時点での現在の [**BuildDir**] の値が自動的に割り当てられます。いったん、この値が割り当てられると、[**BuildDir**] を変更しても他の論理ディレクトリ値には影響しません。

ビルド前およびビルド後のコマンドの設定

プロジェクトがビルドされる前後に実行する 1 つまたは一連のコマンドを定義できます。ビルド前のコマンドを定義するには、「**サーバ設定**」ダイアログの [**ビルド前**] タブをクリックします。ビルド後のコマンドを定義するには、[**ビルド後**] タブをクリックします。

注:UNIX の make プログラムの動作に従って、各コマンド行はそれぞれ別のシェル環境で実行されます。そのため、値をある 1 行に設定し、後でビルド前またはビルド後

のコマンドシーケンス中にこの値を読むことはできません。 for ループなどの処理を含め、このような処理をする必要がある場合は、サーバ上にスクリプトを作成するか、または、サブシェル内でコマンドを実行する必要があります。

サーバ名の削除

サーバリストから現在、選択しているサーバを削除するには、[**削除**] ボタンを使用します。

サーバ名を削除する手順は、次のとおりです。

27. 「**現在のサーバ**」ボックス内のプルダウンメニューリストから、削除するサーバ名を選択します。

28. [**削除**] ボタンをクリックします。 削除してよいかどうか確認されます。

サーバロックの変更

[**サーバの排他制御**] ボタンをクリックして、UNIX システムのビルドディレクトリのロック、およびロックの解除を指定できます。 現在のロック状態を示すダイアログが表示され、ビルドディレクトリをロックまたはロック解除できます。

サーバ設定の確認

[**サーバ設定の確認**] をクリックすると、次のことが実行できます。

- UNIX システム上 の Object COBOL for UNIX に関する情報を表示します。たとえば、Object COBOL for UNIX が共有オブジェクトファイルをサポートするかどうかなどです。
- 「セットアップ」ダイアログで入力されたエントリを確認します。 [パブリッシュ] オプションを正しく機能させるには、報告されたエラーを修正してください。

追加ビルドオプションの設定

[**その他のビルドオプション**] ボタンをクリックすることで、プロジェクトをビルドするときに cob コマンド行に追加される指令を、選択したサーバに対して指定できます。 ここで設定された指令は、現在のサーバに適用されます。 必要な指令を入力するためのダイアログが表示されます。

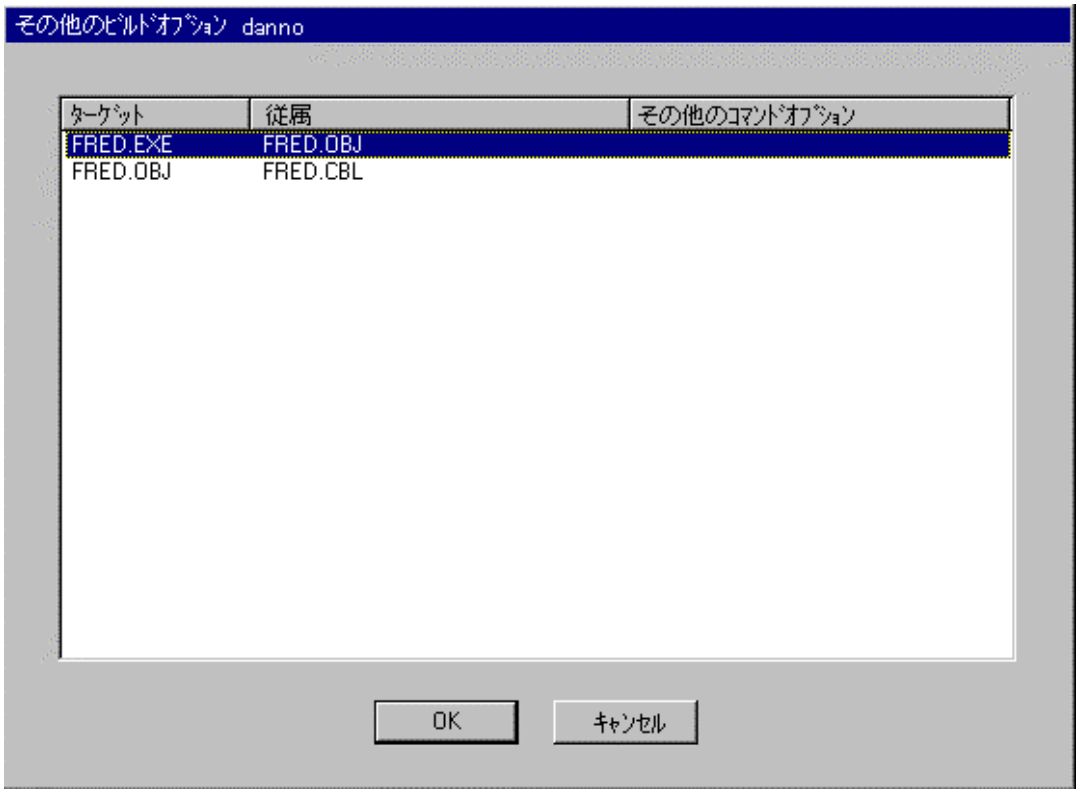

図 3-10:「*サーバ名*のその他のビルドオプション」ダイアログボックス

列「**ターゲット**」は、作成されている論理名です。たとえば、**CGIPRG1.INT** です。

列「**従属**」は、ターゲットを作成するために使用される論理名をリストします。たとえば、**CGIPRG1.CBL** です。

列「**指令**」は、追加 cob コマンド行オプションを指定するフィールドです。設定を編集するには、編集したい行を選択し、選択した行で列「**指令**」をクリックします。または、編集したい行でマウスを右クリックし、メニュー項目 [**編集オプション**] を選択します。

注:ユーザが入力した指令は、パブリッシャによってチェックされません。そのため、変更されずに cob コマンド行に渡されます。

特定のサーバの検索 / 置換パターンの設定

選択したサーバに対して、テキストファイルがサーバにパブリッシュされるときに処理されるファイル名のパターンと正規表現を指定できます。 このパターンはリストとして作成され、そのリストに追加された順番に従って実行されます。

パターンを編集し、置換するには、[**検索/置換**] ボタンをクリックします。現在の検索 / 置換パターンのリストと適用されるファイルが表示されます。 次に例を示します。

図 3-11:「検索/置換の設定: *サーバ名*」ダイアログボックス

列「**検索パターン**」では、正規表現を指定します。正規表現は、付録『*正規表現*』で定義しています。

列「**置換パターン**」では、検索パターンが成功したときに置換するテキストを指定します。置換メタ文字については、付録『*正規表現*』で定義しています。

注: ファイル名が列「**ファイル指定**」で定義したファイルに一致しても、「**ファイルマッピングの設定**」ダイアログ内でテキストファイルとして定義されていなければ、処理されません。

パブリッシュ操作中は、リストボックスの表示順にパターンが実行されます。パターンの順番を変更するには、行を選択し、変更する位置へパターンをドラッグアンドドロップします。

パターンを編集するには、カーソルでそのパターンを選択し、左クリックします。検索パターンを編集する方法の例を次に示します。

31. マウスを使用してパターンを選択します。

32. 左クリックします。

33. パターンを編集します。

右クリックすることで、パターンを編集したり、新しいパターンを作成したりできます。新しいパターンを作成し、リストの末尾にそのパターンを追加する方法を次に示します。

34. 画面の空白領域を右クリックします。次のダイアログが表示されます。

図 3-12:「検索/置換の編集」ダイアログボックス

35. ダイアログに新しいパターンを入力します。

36. [**OK**] をクリックします。

    「**検索/置換の設定**」リストのパターンのリストの末尾に新しいパターンが追加されます。

新しいパターンを作成し、リストの特定の場所にそのパターンを追加する方法を次に示します。

37. 新しいパターンを追加したい行の直前の行のフィールドの 1 つを右クリックします。新しいパターンを追加するか、または現在のパターンを編集するかどうかを尋ねられます。

38. ダイアログに新しいパターンを入力します。

39. [**OK**] をクリックします。

マウスの右ボタンを使用してパターンを編集する方法を次に示します。

40. 変更したいパターンを含む行を右クリックします。ダイアログボックスのフィールドには、その行で定義されたパターンの指定とファイルの指定がすでに設定されています。

41. 必要に応じてパターンの指定とファイルの指定を編集します。

42. [**OK**] をクリックします。

ソースコードの検証

PC のバージョンと UNIX システムのバージョンの間でソースコードのステータスを確認できます。この検証を行うには、[**ソース検証**] ボタンをクリックします。リストボックスが表示されます。

図 3-13:「ソースの検証」ダイアログボックス

列「**ステータス**」は、ファイルのステータスを表示します。ステータスには次のものがあります。

OK	PC 上のファイルと UNIX 上のファイルのパブリッシュ履歴は、一致します。
次回のパブリッシュ時にコピーされる	ファイルはこれまでパブリッシュされたことがない。または、次のパブリッシュでコピーしたいファイルを指定しました。

サーバファイルが最新でない	PC 上のファイルが最後にパブリッシュされてから更新されています。
サーバファイルが更新されました	UNIX 上のファイルが最後にパブリッシュされてから更新されています。
サーバファイルが見つかりません	サーバ上でソースファイルが見つかりませんでした。

行を右クリックすると、コンテキストメニューが可能になり、次にパブリッシュするときにファイルをコピーできます。

その他の詳細設定

「**セットアップ**」ダイアログの [**一般設定**] タブを使用して、プロジェクト固有のものでもサーバ固有のものでもない、その他の詳細を設定できます。このタブは、PC から UNIX システムへのリモートターミナルアクセス用に使用されるターミナルエミュレータを指定するために使用されます。

[**一般設定**] タブをクリックすると、次のダイアログが表示されます。

図 3-14: 「UNIX オプションセットアップ」ダイアログボックスの [一般設定] タブ

デフォルトでは、UNIX オプションは PowerTerm ターミナルエミュレータを使用します。別のターミナルエミュレータを使用したい場合には、次に示す手順に従ってください。

43. 「**PowerTerm を使う**」チェックボックスのチェックを外します。

44. エントリフィールドで、選択したいターミナルエミュレータのパス名と実行可能ファイル名を指定します。必要な実行可能ファイルの場所またはファイル名が確かでない場合には、[**参照**] ボタンを使用します。システムの telnet プログラムが存在する場合は、デフォルトで telnet が選択されます。

第 4 章 : アプリケーションのパブリッシュ

UNIX オプションを使用すると、Net Express でアプリケーションを作成し、UNIX システムにそのアプリケーションをパブリッシュできます。そのため、アプリケーション開発が UNIX 環境からオフロードされ、Net Express のツールと機能を使用できます。アプリケーションのパブリッシュは、Net Express パブリッシャにより処理されます。 すべてのソースコード管理は PC 上で行われ、パブリッシャに透過的です。たとえば、Net Express に付属の PVCS ソースコード管理パッケージを使用できます。

パブリッシャ

パブリッシャへのアクセスは Net Express 統合開発環境 (IDE) の [**UNIX**] メニューで行います。UNIX オプションがインストールされると、[**UNIX**] メニューは、アプリケーションのパブリッシュを可能にする次のオプションを提供します。

- **パブリッシュ** - アプリケーション内の修正したプログラムを UNIX システムにパブリッシュします。
- **すべてをパブリッシュ** - アプリケーション内のすべてのプログラムをパブリッシュします。
- **パブリッシュ中止** - 可能な場合には現在のパブリッシュ操作を中止します。
- **セットアップ** - 作業に適したパブリッシャの設定を使用可能にします。『*パブリッシャの設定*』の章を参照してください。

注:Net Express データアクセスウィザードを使用して生成されたアプリケーションは UNIX システムにパブリッシュできません。

パブリッシャの使い方

UNIX システムにアプリケーションをパブリッシュするには、[**UNIX**] をクリックします。[**パブリッシュ**] をクリックして、最後にパブリッシュしてから修正したソースファイルのみを、指定したサーバにパブリッシュするか、または、[**すべてをパブリッシュ**] をクリックしてプロジェクト内のすべてのファイルをパブリッシュします。

「**サーバ設定**」ダイアログで指定したディレクトリがサーバ上にみつからない場合は、パブリッシャにディレクトリを作成させるかどうか尋ねられます。 ディレクトリを作成したくない場合、またはディレクトリの作成に失敗した場合は、パブリッシュ操作は失敗します。 サーバ設定をチェックし、ディレクトリを手動で作成する必要があります。

「**サーバ設定**」ダイアログで「**パブリッシュ時にソースの妥当性確認**」をチェックした場合は、サーバ上のファイルがいっさい変更されていないことをチェックするために、プロジェクト内のファイルすべてが検証されます。変更されたファイルがあった場合は通知があり、パブリッシュを継続するか、または中止するかのオプションが表示されます。

注:ソース検証オプションは、ソースファイル上で CRC32 チェックを実行します。CRC32 チェックは信頼性の高い変更を検出しますが、ファイルをすべて読み込む必要があります。これは時間のかかる処理であり、UNIX システム上でファイル変更を行った可能性がある環境で操作している場合のみに、このオプションを使用可能にしてください。

プロジェクトに関連した設定を変更し、[**パブリッシュ**] をクリックした場合は、新しい設定を使用可能にするためにすべてのファイルがパブリッシュされます。

パブリッシュ操作の進捗は、Net Express IDE の「出力」ウィンドウでモニタされます。

CGI プログラムのパブリッシュ

CGI プログラムのパブリッシュには注意が必要な手順が他にもいくつかあります。

注:パブリッシャは UNIX システムへの CGI プログラムのパブリッシュを処理できます。ただし、UNIX システムで CGI アプリケーションをディプロイするときに考慮する必要がある他の要素について注意が必要です。『*分散コンピューティング*』の次の項を参照してください。

- UNIX システムへの CGI アプリケーションのディプロイを準備する方法については、『*UNIX サーバでのディプロイ手順*』の項を参照してください。
- UNIX への CGI アプリケーションのパブリッシュの詳細については、『*UNIX へのアプリケーションのパブリッシュ*』の項を参照してください。

次に示す手順は CGI アプリケーションのパブリッシュの概要を示します。

1. サーバの設定を指定するには、「**UNIX オプションのセットアップ**」ダイアログ ([**UNIX** > **セットアップ**] を順にクリック) を使用します。「**UNIX オプションのセットアップ**」ダイアログ内で「**CGI サポートを使う**」をクリックし、アプリケーションが CGI アプリケーションであることを指定する必要があります。

CGI プログラムの中で、Form Designer が大文字でフォームのファイル名を指定するため、ファイル名マッピングが正しく構成されていることを確認してください。ACCCGI モジュールは小文字の **.htm** 拡張子をもつファイルのみを検索できます。大文字の **.HTM** または小文字の **.html** 拡張子をもつファイルは検索できません。

2. CGI アプリケーションの Net Express プロジェクト定義では、アプリケーションがシステム実行可能 (**.exe**) ファイルとしてコンパイルされることを確認してください。

3. [**UNIX**] メニューで [**パブリッシュ**] をクリックします。アプリケーションファイル、必要なプリプロセッサ、ランタイムサポートモジュールがリモートシステムにコピーされ、ビルド処理が修正されます。

   HTML ファイルがソースプールにある場合には、これらのファイルも同様に UNIX システムに転送されます。HTML ファイルを (手動で、または パブリッシャの検索 / 置換機能を使用して) 修正し、CGI トリガプログラム (次の手順を参照) の URL を示すようにする必要があります。トリガへのパスは web サーバの構成により異なります。

4. CGI を実行するには、UNIX システム上でトリガプログラム (通常はシェルスクリプト) を作成します。パブリッシャが作成した **mfenv.sh** シェルスクリプトを使用できます。この場合には、完全なトリガは次のようになります。

5. ```
   #!/bin/sh
   ./mfenv.sh cgiapp
   ```

   トリガを保存します。

注：Server Express にパブリッシュする場合には、この手順は必要ありません。

---

6. トリガファイル、すべてのアプリケーションおよび HTML ファイルを UNIX システムの指定されたディレクトリにコピーします。通常の HTML ファイルは、web サーバの適切な場所に置く必要があります。CGI 実行可能ファイルとサポートファイルは、web サーバで実行可能である必要があるため、実行可能な位置に置かなければなりません (NCSA または Apache などのほとんどの UNIX web サーバの場合には、実行可能な位置は ScriptAlias 構成パラメータにより制御されます)。

   次の場合には、ファイルは正しい場所に自動的にコピーできます。

   - 追加論理ディレクトリを定義する場合
   - ファイルをコピーするためのビルド後のコマンドを指定する場合

7. ファイルをすべてのユーザが読み取ることができることを確認します。たとえば、ファイルのアクセス権を次のように変更します。`chmod +r *.htm`
   ```

CGI ディレクトリにビルドディレクトリの別名を指定できます。これにより、ディレクトリ間でファイルをコピーする必要がないため、すばやい変更が可能です。

「セットアップ」ダイアログの「**ビルド後のコマンド**」フィールドにコマンドを入力して、ファイルを正しい CGI ディレクトリにコピーできます。

CGI アプリケーションをパブリッシュするときに、ランタイムサポートモジュール **ACCCGI.int** がビルドディレクトリにコピーされます。そこで再コンパイルされ、CGI アプリケーションにリンクされます。アプリケーションを **.int** ファイルまたは **.gnt** ファイルとしてパブリッシュしたい場合には、UNIX システム上で Acccgi ランタイムサポートモジュールを手動でリビルドする必要があります。これを行う場合は、アプリケーションとともに Acccgi モジュールを配布する必要があります。

Java アプリケーションと Enterprise JavaBeans アプリケーションのパブリッシュ

ここでは、Java と Enterprise JavaBeans を含むアプリケーションをパブリッシュする場合に必要となる追加手順について説明します。この追加手順は、Java と Enterprise JavaBeans をパブリッシュする場合の手順と同じです。Net Express は、アプリケーションのタイプを自動的に判別し、それに応じてパブリッシュの詳細を仕立て上げます。COBOL bean をパブリッシュすると、**.class** ファイルと JAR ファイルを含む、必要なすべてのファイルが UNIX システム上に自動的に作成されます。

Java アプリケーションや Enterprise JavaBeans アプリケーションを含め、すべてのアプリケーションをパブリッシュする前に、UNIX システムで Java 環境の詳細を指定する必要があります。この指定は、『*環境変数*』で説明する **.mfenv** ファイルを使用して行います。

.mfenv ファイル内の環境変数に設定する実際の値は、使用するプラットフォームによって異なります。設定が必要な一般設定を次に説明します。それぞれのプラットフォームの設定例については、Server Express のマニュアルを参照してください。

- Java の CLASSPATH には **$COBDIR/lib/mfcobol.jar** を必ず指定します。
- PATH には Java 環境を含むディレクトリを必ず指定します。
- Sun Java ランタイムシステムを使用する場合は、PATH に Sun サブディレクトリを追加する必要があります。PATH に追加するサブディレクトリは、使用するプラットフォームによって異なります。特定のプラットフォームの設定例については、Server Express のマニュアルを参照してください。
- オペレーティングシステムの共有ライブラリパスには、**libjava** を含むディレクトリを必ず指定します。このディレクトリ名はプラットフォームに固有ですが、ほとんどの場合は Java の **jre/lib** ディレクトリのサブディレクトリとなります。プラットフォームによっては、複数のディレクトリを指定する必要がある場合があります。たとえば、Java ネイティブスレッドサポートを含むディレクトリも同時に指定します。
- Java ソースをコンパイルするには、COBCPY に **$COBDIR/cpylib** を指定する必要があります。

注：**.mfenv** ファイル内に埋め込み環境変数を指定する場合は、その環境変数の名前をかっこで囲む必要があります。たとえば、**.mfenv** ファイルに次の項目を設定する場合を示します。

```
export PATH=$JH/bin:$PATH
```

この場合には、次のように指定します。

```
export PATH=$(JH)/bin:$(PATH)
```

Net Express IDE 設定とパブリッシャ

Net Express IDE の設定にはパブリッシャの操作に影響を与えるものがあります。

コンパイラ指令

「**プロジェクト** > **プロパティ** > **プロジェクト指令**」ダイアログで設定されたコンパイラ指令は、UNIX システムでコンパイラ指令ファイルを作成するためにパブリッシャで使用されます。指令ファイルは *projectname*.dir と呼ばれます。

Net Express によりデフォルトで設定された次のコンパイラ指令は、パブリッシュ中に無視されます。

- ENSUITE
- WB3
- FLAGEUC
- % 文字を含む指令。% は Net Express の変数エントリを示すために使用されます。たとえば、%TARGETDIR のように使用します。

「**ビルド設定** > **指令**」ダイアログで指定されたコンパイラ指令はチェックフェーズ指令としてのみ、UNIX システムのコンパイラコマンド行にインクルードされます。これらの指令をこのダイアログから UNIX のコンパイルプロセスの生成フェーズに渡すことはできません。

エントリポイント

UNIX システムにパブリッシュされたシステム実行可能ファイルの場合には、エントリポイントは「**プロジェクト** > **ビルド設定**」ダイアログの [**リンク**] タブの「**エントリポイント名**」で定義されています。

環境変数

UNIX 上のリモートシェルサーバは、標準的なシステム初期化ファイル (たとえば、**.profile**、**.cshrc**、**.kshrc** など) を実行せず、最小限の環境変数の設定のみを提供します。パブリッシュ中の環境に対して、変数の変更または追加指定を行うには、ファイル **.mfenv** に環境変数を記述します。**$HOME** ディレクトリとビルドディレクトリの両方で **.mfenv** ファイルを定義できます。実際に、**$HOME** ディレクトリの **.mfenv** ファイルはグローバルファイルとして機能します。プロジェクトがパブリッシュされると、UNIX オプションは **$HOME** ディレクトリの **.mfenv** ファイルを検索します。ファイルが見つかった場合には、その環境変数はビルド環境に追加されます。その後、UNIX オプションはビルドディレクトリの **.mfenv** ファイルを検索します。**.mfenv** ファイルが見つかった場合には、その環境変数がビルド環境に追加されます。

.mfenv ファイル は、UNIX ボーンシェルのサブセットであり、次のものが含まれます。

- 第 1 列がハッシュ (#) 文字であるコメント行
- 次の形式の環境変数

  variable=*value*

value が別の環境変数を示す場合には、次のようになります。

- 環境変数はかっこまたは波かっこで囲まれている場合のみ評価されます。
- 代替パラメータはサポートされていません。

たとえば、次のように記述します。

MYPATH=$PATH	$PATH は展開されません。環境変数 MYPATH は定数値 $PATH に設定されます。
MYPATH=$(PATH)	変数 $PATH を現在の環境で調べます。それが見つかった場合には、展開され、${PATH} の値に置き換わります。見つからない場合には、空白の文字列と見なされます。
MYPATH=$(PATH:="/usr/mybin")	サポートされていません。環境変数 PATH:="/usr/mybin" を見つけようとします。これは失敗し、空白の文字列で置き換えられます。

システムコピーファイル

COBOL の構成要素には、アプリケーションをビルドするときにシステムコピーファイルを呼び出すものがあります。Net Express では、これらのすべてのシステムコピーファイルが 1 つの場所に格納されます (**Net Expess\Base\Source**)。COBOL for UNIX では、システムコピーファイルは多数のディレクトリに格納され、拡張子 **.cpy** または **.CPY** が付けられています。

システムコピーファイルは、アプリケーションが UNIX システムにパブリッシュされるときに UNIX サーバにコピーされません。COBOL for UNIX は固有のシステムコピーファイルを使用する必要があるからです。 パブリッシュ操作中に、システムコピーファイルがコピーされなかったことを伝えるメッセージが IDE のパブリッシュペインに表示されます。

パブリッシャは、UNIX システム上でこれらのシステムコピーファイルを動的に検索することができないため、それらがディレクトリ **$COBDIR/cpylib** にあると仮定します。 UNIX システム上のパブリッシャが作成した Makefile は、システムコピーファイルがこのディレクトリにあると予期します。

ほとんどのシステムコピーファイルは **$COBDIR/cpylib** にあり、問題を起こすことはありません。 ただし、必要なシステムコピーファイルを検索するときに問題が発生したためにビルドが失敗した場合には、システムコピーファイルをディレクトリ **$COBDIR/cpylib** にコピーし、make コマンドを使用してアプリケーションをリビルドする必要があります。また、Net Express で使用されているシステムコピーファイルの名前の大文字と小文字が COBOL for UNIX のそれと異なる場合には、ビルドが失敗する可能性があります。 この場合には、ファイル名を変更してください。かわりに、「パブリッシャセットアップ」ダイアログのファイルマッピング機能を使用できます。

AIX へのアプリケーションのパブリッシュ

ここを読む必要があるのは、次の場合のみです。

- Server Express 以前の COBOL for UNIX バージョンの UNIX システムへパブリッシュする場合
- UNIX システムのオペレーティングシステムが AIX である場合
- 共有ライブラリ (DLL) を使用して作業している場合

この場合は、作成しようとしている共有ライブラリに対するエクスポートファイルが必要です。このファイルは自動的には作成されません。 エクスポートファイルは AIX リンカの標準的な部分であり、1 行に 1 つずつのエクスポートされたシンボルを含む必要があります。 エクスポートファイルの名前は、ファイル拡張子 **.exp** をもつターゲットのベース名です。

たとえば、共有ライブラリ **mylib.dll** を作成するプロジェクトがある場合には、そのプロジェクトは **prog1.obj** と **prog2.obj** を含みます。prog と呼ばれる有効なエントリポイントが 1 つのみあります。

AIX にパブリッシュするには、ファイル **mylib.exp** が必要です。このファイルは、テキスト prog を含む 1 行のテキストファイルです。このファイルをプロジェクトに追加すると、パブリッシュするたびに AIX に自動的に転送されます。 かわりに、AIX システム上でこのファイルをメンテナンスするだけでもかまいません。

注:エクスポートファイルは AIX の場合にのみ必要です。他の UNIX システムにパブリッシュする場合には必要ありません。

状態メンテナンスルーチンの使い方

状態メンテナンスライブラリルーチン (MF_CLIENT_STATE_*n*) を使用すると、web サーバ上でユーザとアプリケーションの状態情報を格納できるインターネットアプリケーションを作成できます。アプリケーションでこれらのルーチンを使用した場合には、これらのルーチンのランタイムサポートモジュールがアプリケーションとともに UNIX システムにパブリッシュされていることを確認することが必要です。

これを行うには、ファイル **sstate.int** を **Net Express\Unix\CGI-SUP** からプロジェクトにコピーします。アプリケーションをパブリッシュするときに、**sstate.int** もまた UNIX システムにパブリッシュされます。

第 5 章 : Net Express への UNIX アプリケーションのインポート

UNIX システムに常駐している開発済みのアプリケーションを編集するために Net Express を使用する場合には、そのアプリケーションを Net Express にインポートする必要があります。インポートは一度のみ行います。アプリケーションが PC 上にインポートされると、Net Express を使用してそのアプリケーションを維持できます。

ほとんどの複雑な UNIX アプリケーションは、複数のアプリケーションモジュールから構成されています。アプリケーションモジュールは、アプリケーションの明確な一部分であるプログラムのグループと見なすことができます。 たとえば、会計のアプリケーションは、購買伝票、元帳、販売伝票などのいくつかのモジュールから構成されています。 各アプリケーションモジュールは多くのプログラムから構成されている場合があります。 各モジュール (購買伝票など) には固有のディレクトリ構造があり、そこでモジュールが維持管理されています。 一般的に、アプリケーションソースコード、コピーライブラリ、データファイルおよび実行可能ファイル用に別々のディレクトリがあります (中間コード、生成コード、実行可能コードについてもディレクトリが別になっていることがよくあります)。 アプリケーションをインポートするときには、複数のソースディレクトリを指定して、複数のターゲットディレクトリへのコピーを指定できます。

アプリケーションをインポートする前に、Net Express ではソースコードが複数のディレクトリに存在することは可能であっても (この方法はお奨めできません)、各プロジェクトがそれぞれ 1 つのディレクトリに存在することを前提にしている点に注意してください。Net Express はプロジェクトの全モジュールを同じディレクトリにビルドするので、アプリケーション内でのファイルの配置方法に制限があります。たとえば、別のディレクトリに同じソースファイル名をもつことはできません。UNIX からアプリケーションをインポートするときは、ソースディレクトリの数を最小限に抑え、アプリケーション内のモジュールごとにそれぞれ別の Ne Express プロジェクトを作成することお奨めします。

UNIX オプションのインポートウィザードを使用すると、UNIX アプリケーションを Net Express にインポートできます。ただし、アプリケーションモジュールを開発したり維持したりするには、Net Express を使用する前に、プロジェクトに手動で変更を行う必要があります。

インポートウィザードを起動するには、[**UNIX** > **インポート**] を順にクリックします。

注: インポートウィザードを開始する前に、プロジェクトのソースコードすべてが、Samba 経由でネットワーク上で共有可能であることを確認してください。PC からアクセス可能である必要があるからです。Samba のインストールと構成の詳細については、付録『*SCP と Samba のインストール*』を参照してください。

インポートウィザードの手順は次のとおりです。

手順 1 - プロジェクトの省略時設定

Net Express プロジェクトとプロジェクトディレクトリの名前を指定します。 新規に作成するプロジェクトまたは追加する既存のプロジェクトの名前を指定します。Net Express では、プロジェクトディレクトリは特に重要です。Net Express は、プロジェクトディレクトリにソースコードを置き、アプリケーションをビルドすることを前提としています。

手順 2 - 送り側ディレクトリ

プロジェクト用のソースコードがある UNIX システム上のディレクトリすべてを入力します。 すべての COPY ファイルディレクトリを含めるする必要があります。

これらのディレクトリにファイルが存在していない場合には、手順 4 でファイルを追加できません。

手順 3 - 受け側ディレクトリ

入力した各送り側ディレクトリに対して、ディレクトリ内のファイルをコピーする場所を指定できます。 デフォルトのコピー先は、プロジェクトディレクトリです。 プロジェクトディレクトリを受け側ディレクトリの値にしてください。必要な場合のみに、この値を変更してください。

手順 4 - ソースファイルの指定

プロジェクトを作成するソースファイルすべてを入力します。 入力できるのは、手順 2 で指定した送り側ディレクトリのどれかに存在するファイルのみです。

[**COPY ファイル検索**] ボタンをクリックすると、インポートウィザードが入力したファイルすべてについて、COPY 文を含んでいるかどうか調べます。検索が終わると、参照する COPY ファイルを位置指定しようとします。 検索するディレクトリは、手順 2 で指定した送り側ディレクトリのみです。 見つかった COPY ファイルはすべて、ソースファイルのリストに追加されます。 COPY 文で参照されていて、見つからない COPY ファイルがある場合は、それらが一覧表示されます。

注:COPY ファイルの検索は、参照する COPY ファイルの定数どおり、位置指定するのみです。たとえば、COPY "filename.cpy" です。

列「**ファイルタイプ**」は、ファイルがテキストまたはバイナリかを指定します。デフォルトはテキストです。 テキストファイルは、手順 5 で指定されるインポート時テキスト変換操作の有効なターゲットです。バイナリファイルは、送り側から受け側に、変更されずにそのままコピーされます。

手順 5 - インポート時のテキスト変換

手順 4 でタイプをテキストと定義されたソースファイルに対して、インポートウィザードで処理できます。

ここで指定する検索 / 置換パターンは、『*検索 / 置換パターンの設定*』で説明したものと同じです。正規表現についての詳細は、付録『*正規表現*』を参照してください。

テキストファイルの UNIX 形式の行の終了文字を、PC 形式の行の終了文字に変換するには、「**UNIX テキストファイルを PC 形式に変換**」チェックボックスをチェックします。

テキストファイルを、UNIX システム上で使用される日本語 EUC 文字コード体系から、PC 上で使用される日本語 SJIS 文字コード体系へ変換するには、「**テキストファイルを EUC から SJIS に変換**」をチェックします。

注：SJIS 文字コード体系も使用する UNIX システムもあります。日本語のプロジェクトをインポートする場合は、どの文字集合でエンコードされているかを知る必要があります。

手順 6 - その他のプロジェクト設定

UNIX 上に指令ファイルがない場合は、自動的に UNIX 上から Net Express プロジェクトにチェッカー設定が追加されます。cobol.dir と cobopt ファイルの両方が、サポートされています。

注：**cobopt** ファイルからはチェッカーオプションのみがインポートされます。他のオプションは無視されます。

インポートウィザードがプロジェクトにソースファイルを追加して、各 COBOL ファイルに対してデフォルトのターゲットを作成するようにするには、「**各ソースファイルに対して省略時のビルドターゲットを作成する**」をチェックします。このコントロールのチェックを外した場合は、ソースファイルがプロジェクトにコピーされますが、ターゲットは作成されません。

手順 7 - 設定の確認

[**完了**] をクリックすると、指定したソースファイルがコピーされて変換され、必要に応じて新しいプロジェクトが作成されるか、または、既存のプロジェクトが更新されます。

インポートウィザードは、自動的に INTLEVEL"2" および WARNINGS"2" 指令をユーザのプロジェクト設定に追加します。さらに、手順 5 で「**テキストファイルを EUC から SJIS に変換**」をチェックした場合は、FLAGEUC 指令がプロジェクト設定に追加されます。

COBOL COPY ファイル

指定したソースファイルが PC 上で複数のソースディレクトリへインポートされる場合は、Net Express が COPY ファイルを位置指定できるように COBCPY 環境変数を設定する必要があります。COBCPY 環境変数は、Net Express IDE の「**プロジェクトのプロパティ**」で設定できます。

COBOL データファイル

Net Express プロジェクトディレクトリ以外のディレクトリに COBOL データファイルがある場合には、実行中に外部ファイル名マッピングを使用してデータファイルを検索できます。外部ファイル名マッピングは、テキストファイル (マッパーファイル) 内でのファイル名マッピングの解決を使用可能にすることにより、COBOL プログラムで使用された割り当てファイル名を物理的なファイル名に変換するための柔軟な方法を提供します。後でファイルを編集することで、ファイル名マッピングを変更できます。たとえば、マッパーファイルには次のエントリを含むことができます。

```
apfile c:\tmp\unixapp\ap\data\apfile
arfile c:\tmp\unixapp\ar\data\arfile
```

外部ファイルマッパーを使用するには、指令 ASSIGN"EXTERNAL" を使用してプロジェクトをコンパイルし、ランタイムチューナ environment_mapper を TRUE に設定する必要があります。使用するすべてのプロジェクトで、マッピングファイルはメインプロジェクトディレクトリになければなりません。

外部ファイル名マッピングの詳細については、『*ファイル処理*』(FHpubb.htm) の『*ファイル名*』の章を参照してください。

他のプロジェクトでのプログラムの実行

プロジェクトが現在のプロジェクトの外部のプログラムを呼び出す場合には、呼び出されるプログラムの位置を COBDIR 環境変数に追加してください。パスが絶対パス名の場合には、プログラムへのパスを完全に削除し、COBDIR 環境変数にそのパスを追加できます。より柔軟なアプローチとしては、環境変数を使用してパスを置き換え、その環境変数を設定してから、Net Express を起動する方法があります。

第 6 章 : データベースアプリケーションでの COBSQL の使い方

COBSQL は、リレーショナルデータベースのベンダが提供する COBOL プリコンパイラで機能するように設計された統合プリプロセッサです。使用するプリコンパイラのリストについては、『*COBSQL*』の章を参照してください。

旧バージョンの Micro Focus COBOL 製品でこれらのプリコンパイラをすでに使用し、アプリケーションを Net Express に移行する場合は、COBSQL を使用してください。それ以外の埋め込み SQL の開発には、OpenESQL を使用することをお奨めします。

UNIX プラットフォームにディプロイするアプリケーションを作成していて、Oracle または 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場合には、COBSQL を使用する必要があります。

ディプロイ可能なデータベースアプリケーションの開発

UNIX システムにディプロイ可能なデータベースアプリケーションを、COBSQL を使用して作成する場合には、COBSQL を設定し、必要に応じてランタイムを再リンクする必要があります。これらの手順は機能によって異なるため、Server Express の『***データベースアクセス***』マニュアルの『*COBSQL*』の章、または OCDS の『***ファイル処理のためのプログラマーズガイド***』の『*COBSQL*』の章を参照してください。

Net Express で COBSQL とデータベースプリコンパイラを使用してデータベースアプリケーションを開発します。Net Express で COBSQL を呼び出す方法については、『*データベースアクセス*』マニュアルを参照してください。通常、データベースプリコンパイラには埋め込み SQL 文を使用して COBOL プログラムをコーディングする方法を示すサンプルプログラムが付属しています。Net Express でアプリケーションをアニメートまたは実行し、そのアプリケーションの動作を確認した場合は、データベースアプリケーションを UNIX システムにコピーします。

1. ファイル **cobsql.dir** をプロジェクトのソースプールに追加します。このファイルは UNIX に必要なすべての COBSQL 指令を含みます。このファイルの追加は一度のみ行う必要があります。

2. [**UNIX** > **セットアップ**] を順にクリックします。このプロジェクトに合わせてパブリッシャを構成します (パブリッシャの設定の詳細については、『*パブリッシャの設定*』の章を参照してください)。特に、「**サーバ上のビルド**」のチェックを外します。これは、パブリッシャが UNIX システムにプロジェクトをコピーするが、プロジェクトをリビルドしないことを指定します。

   プロジェクトをパブリッシュします。ファイルが UNIX システムにコピーされ、Makefile が作成されますが、アプリケーションはリビルドされません。

3. UNIX マシンの publish ディレクトリにある、Makefile ファイルを新しいファイルにコピーします。たとえば、CSQLMakefile という名前でコピーします。

4. CSQLMakefile を変更し、プログラムをコンパイルするために使用するコマンド行に -k オプションと -C"p(cobsql)" 指令を含むようにします。 たとえば、次に例を示します。

```
5.   Animtst1.int: Animtst1.pco
6.
7.    ./mfenv.sh cob Animtst.pco
```

この場合は、次のようになります。

```
Animtst1.int: Animtst1.pco

./mfenv.sh cob -k Animtst.pco -C"p(cobsql)"
```

8. この修正した CSQLMakefile で make を使用して、UNIX システム上でこのプログラムのデバッグ可能なバージョンを作成します。

   注:必要に応じて、CSQLMakefile を正しく設定した場合には、パブリッシャを使用してこの手順を自動化できます。 ビルドが行われる前に実行する UNIX コマンドをパブリッシャに指定できます。

   1. [**UNIX** > **セットアップ** > **サーバ** > **設定** > **ビルド前**] の順にクリックします。

   2. 「**ビルド前のコマンド**」フィールドにコマンド make -f *makefilename* を入力します。*makefilename* は編集した Makefile の名前です (この例では、CSQLMakefile)。

      [**OK**] をクリックします。

   3. [**サーバ**] タブをクリックし、「**サーバ上のビルド**」のチェックを外します。

   この場合には、プロジェクトをパブリッシュするときに自動的に生成されたビルドは失敗しますが、makefile を使用して作成したビルド (この例では、CSQLMakefile) は、アプリケーションを正常にリビルドします。

9. UNIX システムのアプリケーションをアニメートまたは実行します。

第 7 章 : ヒントとトラブルシューティング

ここでは、次の内容を説明します。

- UNIX オプションの使用時に役に立つヒント
- UNIX オプションの使用時に発生する問題と解決方法

ヒント

CGI アプリケーションのパブリッシュ

CGI アプリケーションは、しばしばファイルシステム上の複数の位置に存在します。 たとえば、CGI プログラムは **cgi-bin** ディレクトリに、静的 HTML ページは別のディレクトリに、フォームはさらに別のディレクトリにある、という場合です。

この問題への簡単な対処方法として、論理ディレクトリ機能を使用する方法があります。「論理ディレクトリ」ダイアログで対になる名前を定義します。たとえば、HTML Pages と HTML Forms です。「**ファイルマッピングの設定**」ダイアログを表示して、HTML ファイルに論理名を割り当てます。 ワイルドカードを指定したり、シフトクリックしたりすると、複数のファイルを選択できます。

[**サーバ設定** > **ディレクトリ**] タブを順に選択して、HTML ページと HTML フォームを含む物理ディレクトリを正確に入力します。ファイルはその後、自動的に指定されたディレクトリにコピーされます。

CGI プログラム自身は、常にビルドディレクトリにビルドされます。 これを自動的に正しい位置に置くには、[**サーバ設定** > **ビルド後**] タブを順に選択して、postbuild コマンドを指定する必要があります。

ファイルの自動修正

UNIX オプションの検索 / 置換パターンは正規表現を使用しており、とても強力です。ただし、正規表現を使いこなすには、ある程度の経験が必要です。 経験を積むために最適な方法は試してみること、そして正規表現を使用したことがある人のアドバイスを受けることです。

検索 / 置換機能を使い始めたばかりの場合は、まず、定数検索を使用してみてください。 定数検索は、直接、あるテキストを別のテキストに置換します。

おそらく最も難しいのは、繰り返し文字を処理する正規表現です。 繰り返しメタ文字 (*+?) はすべて、直前の正規表現に対して機能します。メタ文字自身には、文字そのものとしての意味はありません。 直前の文字とは、単に 1 文字のみです。ただし、かっこを使用してグループ化された正規表現は例外です。注意してください。 ファイル名形式のパターンの使用に慣

れている場合は、混乱するかもしれません。付録『*正規表現* 』で正規表現の使用例を参照し、理解してください。

トラブルシューティング

パブリッシュできない

サーバコントロールプログラム (SCP) は、アプリケーションのパブリッシュ先の UNIX システムにインストールする必要があります。SCP は、Net Express と UNIX の間のインターフェイスを提供するからです。 SCP をインストールしない場合には、アプリケーションをパブリッシュできません。 詳細については、Server Express のマニュアルを参照するか、SupportNet から SCP をダウンロードしてください。

サーバが scp の実行を拒否したメッセージが表示された場合は、ユーザの **.rhosts** 設定をチェックします。構成方法の詳細については、付録『*Samba と SCP のインストール*』を参照してください。

.dll ファイルを含むアプリケーションをパブリッシュできない

この問題は、Net Express と COBOL for UNIX の間の COBOL の機能の相違によって発生します。Net Express 上では、プログラムは **.dll** ファイルを直接呼び出すことができます。COBOL for UNIX 上では、共有オブジェクトをアクセス可能にするには、共有オブジェクトを実行形式 (またはランタイムシステム) にリンクする必要があります。

注:Server Express は、Net Express と同じ方法で共有オブジェクトを直接呼び出すことができます。

このようなアプリケーションを UNIX にパブリッシュするには、IDE をうまく操作して、実行形式モジュールと **.dll** ファイルの間に従属関係を作成する必要があります。アプリケーションをパブリッシュしたときに、この従属関係を見分けて、共有オブジェクトが実行形式モジュールに正確にリンクするようになります。

次の手順では、同じ Net Express プロジェクト内の **.exe** ファイルによって呼び出される **.dll** ファイルがあると仮定しています。 使用される技術が複数のプロジェクト内の複数の **.dll** ファイルに同様に適用されます。

1. ビルドペインで **.dll** ファイルを選択し、右クリックして [**ビルド設定**] を選択します。
    1. [**リンク**] タブを選択します。
    2. [**カテゴリ**] プルダウンメニューから「**高度な設定**」を選択します。

3. 「**一時リンカーファイルの保持**」オプションをチェックします。

4. ダイアログボックスを閉じます。

2. ビルドペインで **.dll** ファイルを選択し、右クリックして [**オブジェクトをリビルド**] を選択します。 または、[**すべてをリビルド**] をクリックします。 これで、**.dll** ファイルに対応するターゲットディレクトリに **.lib** ファイルが作成されます。

3. ソースペインで、右クリックして [**ソースプールへファイルを追加**] を選択します。 ターゲットディレクトリに作成された **.lib** ファイルを選択します。プロジェクトディレクトリにファイルをコピーするかどうか、Net Express が尋ねてくるので、[いいえ] を選択します。

4. **.lib** ファイルをビルドペインにドラッグして、実行形式モジュールの一部に含めます。

5. これで、Net Express 内にプログラムをビルドし、実行できます。

6. 必要な場合は、パブリッシャをセットアップします。 これで、このプロジェクトを UNIX へパブリッシュできます。そして、共有オブジェクトにリンクされた実行形式が作成されます。

ユーザ ID をビルド領域で変更できない、または共同作業者と共有できない

ビルド領域は、1 人のユーザのみが使用するようになっています。このため、ビルド領域はロックされるようになっています。

共有ビルド領域の主な問題は、ある人が行ったソースコードの変更が、別の人が同じビルド領域で行ったソースコードの変更と混在することです。 この結果、微妙なエラーが原因でコンパイルに失敗してしまいます。

実際に共同作業者とビルド領域を共有する必要がある場合は、共通ユーザ ID を設定して、自分のユーザ ID も共通ユーザ ID も使用できるようにすることを推奨します。 同期的にユーザの変更を保持するために、ソースコードコントロールシステムを使用することを、強く推奨します。 [**サーバ設定**] タブで「**パブリッシュ時にソースの妥当性確認**」オプションをオンにすると、パブリッシャによって、UNIX オプション の予期するバージョンとサーバ上のファイルが比較され、相違のあるファイルが通知されます。

ビルド領域をあるユーザ ID から別のユーザ ID に変更するには、現在、ロックを取得しているユーザが、「**サーバの排他制御**」ダイアログを使用してエリアのロックを解放する必要があります。 新しいユーザがパブリッシュに成功するには、ビルド領域内のファイルの所有権を、古いユーザから新しいユーザに手動で変更することが必要な場合があります。 UNIX システムによっては、root アクセス権が要求されます。

システムコピーファイルの問題

パブリッシュ操作中に、パブリッシャが UNIX システム上でシステムコピーファイルを見つけることができないためアプリケーションが正常にビルドされなかったことを示すエラーを受け取る可能性があります。 このエラーが発生した場合には、システムコピーファイルをディレクトリ

$COBDIR/cpylib にコピーする必要があります。場合によっては、UNIX のシステムコピーファイルの拡張子の大文字小文字を変更する必要があります (たとえば、コピーファイルの拡張子が **.cpy** の場合は、**.CPY** に変更することが必要になる場合があります)。

詳細については、『*アプリケーションのパブリッシュ*』の章にある『*システムコピーファイル*』の項を参照してください。

CGI アプリケーションの問題

CGI アプリケーションが正常に機能しない場合には、次のチェックを行ってください。

- HTML 出力フォームは CGI プログラムと同じディレクトリに存在する必要があります。
- フォームのファイル名は Form Designer で大文字でハードコーディングされるため、ファイル名マッピングが正しく構成されていることを確認してください。 Acccgi モジュールは小文字の **.htm** 拡張子をもつファイルのみを検索できます。 大文字の **.HTM** または小文字の **.html** 拡張子をもつファイルは検索できません。
- ファイルをすべてのユーザが読み取ることができることを確認します。たとえば、ファイルのアクセス権を次のように変更します。`chmod +r *.htm`

また、『***分散コンピューティング***』の次の項を参照してください。

- UNIX システムへの CGI アプリケーションのディプロイを準備する方法については、『UNIX サーバでのディプロイ手順』の『*アプリケーションのディプロイ*』の項を参照してください。
- UNIX への CGI アプリケーションのパブリッシュの詳細については、『UNIX へのアプリケーションのパブリッシュ』の項を参照してください。

パブリッシュされたファイルがコンパイルできない

GMT オフセットを調べて、システムの日付と時刻が正しく設定されていることを確認してください。 SCP は、UNIX にコピーされるファイルの日付を設定するために、コンピュータの世界協定時刻 (GMT +/- タイムゾーンオフセット) を使用します。 実行可能ファイルが、コピーされたファイルより新しいタイムスタンプでビルドされると、1 回目より後のコンパイルは実行されません。

付録 A 章 : Samba と SCP のインストール

Samba は、標準的な PC 形式のネットワークを使用して、UNIX ファイルやプリンタを PC と共有できるようにします。Samba を使用するには、UNIX マシンから PC へのアプリケーションのインポートが必要です。

SCP は、 NET Express UNIX オプションと UNIX オペレーティングシステムおよび COBOL 製品間の標準化インターフェイスで、Server Express のインストールに組み込まれています。

Samba は Server Express CD の */abcc/os_name* ディレクトリに格納されています。この *abcc* はバージョン、*os_name* はオペレーティングシステムのニーモニックです。たとえば、HP-UX 11.x と SUN SPARC の場合には、プログラムは次のディレクトリに格納されています。

HP-UX 11.x	SUN SPARC
/2200/parisc/samba.tar	/2200/sunspc8/samba.tar

SCP の構成

SCP プログラムは、デフォルトで、Net Express UNIX オプションによって UNIX リモートシェル (RSH) プロトコルを使用して実行されます。 ほとんどすべての UNIX システムは、デフォルトで RSH サーバプログラムを使用可能です。他の特別なサーバソフトウェアをインストールしたり、構成したりする必要はありません。 ただし、UNIX オプションを使用するには、RSH セキュリティ構成がユーザに PC から SCP プログラムを実行可能としていることを確認する必要があります。 SCP はデーモンとして使用することもできます。この章の『*代替セキュリティ機構 (SCP デーモンメソッド)*』の項も参照してください。

クイックスタート

アプリケーションのパブリッシュ先である UNIX システムのユーザ ID のホームディレクトリに、**.rhosts** というファイルがあることを確認してください。 このファイルには、パブリッシュ元の PC の正式名が含まれている必要があります。

RSH セキュリティ機構

RSH セキュリティ機構は、UNIX システムの構成ファイルに基づいて「ユーザ等価性」を確立することによって機能します。 ユーザが等価である場合は、UNIX システムは呼び出されたプログラムへのアクセス権を、パスワードを要求しないで付与します。**.rhosts** と **hosts.equiv** ファイルは、この等価性を制御するために使用されます。これらの 2 つのファイルを *rhosts ファイル*と呼びます。

rhosts ファイルは、オリジナルのバークレイ UNIX 版に由来しますが、すべての UNIX バージョンで使用されています。 その過程で複数の異なるバージョンが生じています。SUN 拡張は

NIS (旧名 Yellow Pages) をサポートしており 、よく知られています。ただし、これらもまた、UNIX のさまざまな異なるバージョンで使用されています。

UNIX オプションが UNIX システムに接続するときに、「**サーバ設定**」ダイアログで入力したユーザ ID を提供します。 UNIX システムは、ユーザの PC の正式名を IP アドレスに基づいて決定します。これには、逆調査という技術を使用します。 ユーザの正式マシン名とユーザ ID を使用して、アクセスを許可するかどうかを次の方法で決定します。

1. ファイル **/etc/hosts.equiv** が存在するかどうかを確認します。 ファイルが存在する場合は、次の形式のテキストファイルである必要があります。

   Machine-Name [*User-ID*] [*#Comments*]

2. マシン名の次にユーザ ID が指定されていない場合は、全ユーザが有効であるとみなされ、アクセスが許可されます。

3. PC マシン名とユーザ ID が **hosts.equiv** ファイルの行のどれかに一致する場合は、アクセスが許可されます。

4. **/etc/hosts.equiv** 内のマシン名とユーザ ID が、 ユーザの PC マシン名またはユーザ ID に一致しない場合には、サーバは要求されたユーザ ID の **HOME** ディレクトリ内に **.rhosts** というファイルがあるかどうか確認します。 **.rhosts** ファイルは、**hosts.equiv** ファイルと同じ形式です。

警告:.rhosts ファイルの所有者と、このディレクトリの所有者は同じでなければなりません。また、このユーザのグループと他のユーザにファイルへの書込み許可を付与しません (つまり、アクセス権は `rw-r--r--` である必要があります)。 また、シンボリックリンクであってもいけません。

- PC マシン名が **.rhosts** ファイルの行のどれかに一致した場合は、アクセスが許可されます。
- 上記のどの手順でもアクセスが許可されなかった場合は、この時点でアクセスが拒否されます。

.rhosts ファイルのユーザ名フィールドは、UNIX ユーザのために設計されています。あるシステムにログインするユーザがリモートシェルを使用したり、別のユーザとして他のシステムにリモートコピーを実行したりするかもしれないからです。 UNIX オプションでは、.rhosts ファイルユーザ名フィールドは必要ありません。アクセスしようとしている **HOME** ディレクトリのユーザ ID をいつも提供するからです。

RSH に対する主な SUN 拡張は、rhosts ファイルの有効マシンリストにシンボル (+) を追加しています。これは、NIS 設定で使用するために設計されており、「すべての有効なマシン」を意味します。ただし、非 NIS 設定では「すべてのマシン」を意味します。一般に、システムのセキュリティを混乱させるので、このシンボルの使用は避けるべきです。

ほとんどの場合には、UNIX オプションに対する rhosts ファイルは次のように設定します。

- **hosts.equiv** ファイルは、一般にシステム管理者がリモートシステム上の全ユーザが等価であることを宣言する場合にのみ変更されます。これは PC クライアントシステムでは推奨できません。PC 上ではどのユーザも指定することができ、UNIX システムはそれを受け入れるからです。悪意の PC ユーザがシステム上の全ユーザアカウント (root を除く) にアクセスすることを、無制限に許してしまいます。

- ユーザ ID 用の **$HOME/.rhosts** ファイルは、パブリッシュに使用する PC のマシン名を含む必要があります。ユーザ ID は必要ありません。

  サーバが SUN 拡張をサポートしている場合は、**.rhosts** ファイルに + を追加して、ユーザ ID に対するマシン名のチェックを不要にしたいことがあります (**rhosts** の man ページをチェックして + をサポートしているか調べてください)。

.rhosts ファイル用の正式マシン名の決定

.rhosts ファイルに入力する PC のマシン名は、UNIX によって決定された正式なマシン名である必要があります。PC の通称は問題ではありません。名前は、IP アドレス接続に基づいて UNIX サーバによって決定されます。ほとんどのインストールでは、サーバが名前を決定し、クライアント名は同じである必要があります。

システムの正式名を決定するには、まず、PC の IP アドレスを決定します。

注：Windows 95 と Windows NT の構成ダイアログは異なります。また、構成ダイアログはさまざまなサービスパックで更新されています。そのため、使用する必要のあるダイアログは微妙に異なります。次に示す手順は、サービスパック 3 とインターネットエクスプローラ V4.01 がインストールされている Windows NT V4.0 の場合の手順です。

---

1. [**スタート**] をクリックし、[**設定**] を選択します。
2. [**コントロール パネル**] をクリックし、[**ネットワーク**] をクリックします。

注：[**識別**] タブの「**コンピュータ名**」フィールドは、TCP/IP 名とは関係のない NetBIOS 名です。

- Windows NT では、[**プロトコル**] タブをクリックし、[**TCP/IP プロトコル**] をクリックします。次に、[**プロパティ**] ボタンをクリックします。

  Windows 95 では、インストールされたネットワークコンポーネントがリストボックスに表示されるので、TCP/IP プロトコルを選択して、[**プロパティ**] ボタンをクリックします。

- [**IP アドレスを自動的に取得**] ボタンがチェックされている場合は、動的に IP アドレスが割り当てられます。次の『*IP アドレスの動的割り当て*』の項へ進んでください。

- [**IP アドレスを指定**] ボタンがチェックされている場合は表示された **IP アドレス** の値を記録して、次へ進みます。

まず、UNIX マシンにログインします。UNIX システムがマシン名と IP アドレスをクロスリファレンスするために使用している方法は、主に 3 つあります。

1. hosts ファイル。IP アドレスとマシン名を各行に含む単純なテキストファイルです。通常、**/etc/hosts** にあります。

2. DNS (Domain Name System)。インターネットによって使用されるシステムです。世界中にある特別なサーバを構成し、名称とアドレスを解読してお互いに接続します。最も普及した方法となりつつあります。

3. NIS (Network Information System)。NIS (旧名 Yellow Pages) は、ホストファイル、パスワード、グループ、エイリアス、サービス、その他の分散データベースです。

正式なホスト名を決定するには、UNIX システムで使用する名前の解決方法を決定する必要があります。次に、その方法を使用して、PC の IP アドレスに基づいた名前を調査します。

システム管理者に質問することもできます。システム管理者は、**.rhosts** ファイルに何を入力すべきか教えてくれます。

NIS が構成されているかどうかを確認する方法

/etc/nsswitch.conf というファイルがあるかどうか確認します。存在する場合は、hosts: で開始する行を見つけてください。次のような行が見つかります。

```
hosts: xfn nisplus dns [NOTFOUND=return] files
hosts: xfn nis [NOTFOUND=return] files
 hosts: files
```

これらの文は、NIS、DNS および **/etc/hosts** 内のファイルを使用して、ホスト名が解決される順序を決定しています。

ユーザの正式名を決定するには、**nsswitch.conf** で定義された名前解決の順序に従います。

DNS が構成されているかどうかを確認する方法

/etc/resolv.conf というファイルがあるかどうか確認します。存在する場合は、DNS が構成されています。このファイルの内容はここでは重要ではありません。

NIS を使用した正式名の決定

ypcat コマンドとともに NIS を使用して、正式名を決定できます。たとえば、PC の IP アドレスが 204.160.128.10 である場合は、次のように入力します。

```
ypcat hosts | grep 204.160.128.10
```

IP アドレスに関連する名前が複数ある場合は、最初の名前が正式名です。

DNS を使用した正式名の決定

nslookup コマンドとともに DNS を使用して、正式名を決定できます。このコマンドは、DNS サーバに問い合わせを行う一般的な方法です。たとえば、PC の IP アドレスが 204.160.128.10 である場合は、次のように入力します。

```
nslookup 204.160.128.10
```

入力した IP アドレスの名前とアドレスの次に、情報を取得した DNS サーバの IP アドレスと名前が表示されます。表示される名前は、いつも正式名です。

/etc/hosts を使用した正式名の決定

hosts ファイルは単純なテキストファイルなので、直接、調べることができます。たとえば、次のように入力します。

```
grep 204.160.128.10 /etc/hosts
```

IP アドレスに関連する名前が複数ある場合は、最初の名前が正式名です。

正式名を解決できない場合は、PC に正式名がない可能性があります。この場合は、(ドット区切りの 10 進) IP アドレスを直接 **.rhosts** ファイルに追加できます。ただし、この方法はマシンを特定します。システム管理者に依頼して、会社のマスターマシン名テーブルにユーザの PC 用の正式名を追加してもらう方がよいでしょう。

IP アドレスの動的割り当て

.rhosts アクセス機構は、IP アドレスの動的割り当てをサポートしていません。完全なセキュリティ機構があり、マシン名 (および IP アドレス) を定数で定義していると仮定します。

ユーザのネットワークシステムが DHCP (または BOOTP のような他の動的 IP 方式) を使用している場合は、静的 IP アドレス割り当てが可能かどうかをネットワーク管理者に確認してください。 可能な場合は、静的 IP アドレスを取得し、正常な方法で **.rhosts** を構成してください。

注:シリアルライン経由でダイアルアップしている場合は、おそらく PPP または SLIP を使用しています。その場合は、まず確実に IP アドレスの動的割り当てを行っています。

動的 IP アドレスを用いて UNIX オプションで作業するには、rhosts ファイル内でリストされている現在のマシン名を知る必要があります。 これには方法がいくつかありますが、利便性とセキュリティのどちらをとるかによります。

- **オプション 1:マシン名チェックを使用不能にする**

  セキュリティを考慮せず、サーバが rhosts ファイルに対して SUN の + 拡張をサポートしている場合は、+ を **.rhosts** ファイルに追加し、そのユーザ ID を使用してパブリッシュしようとしているホストすべてが成功します。 安全性は低いですが、簡単な方法です。

- **オプション 2:動的 IP アドレスすべてにユーザディレクトリへのアクセスを許可する**

  .rhosts ファイルへのサブネット上に存在する動的に割り当てられた IP アドレスすべてに対して、マシン名を追加できます。 たとえば、ネットワークが動的 IP マッピング用に *x.x.x.*100 から *x.x.x.*120 が割り当てられ、dhcp100 から dhcp120 の名前が割り当てられたとします。

  dhcp100 から dhcp120 の名前をすべて **.rhosts file** ファイル に追加した場合は、これらのアドレスのどれを割り当てられてもパブリッシュできます。 同じサブネット上で動的に IP アドレスを割り当てられたユーザは、まったく同じアクセス権をもちます。

  おそらく、ローカルな IT 部門に質問して、動的に割り当てられるアドレスを決定する必要があります。

- **オプション 3:動的に .rhosts ファイルを変更する**

  最も安全性の高い方法です。ただし、最も不便な方法でもあります。 PPP を使用してダイアルアップしたり、PC をリブート (DHCP) したりするときに必ず、新規に IP アドレスを割り当てることができます。 そのため、リブートしたりダイアルアップしたりするたびに、ユーザは **.rhosts** ファイルを編集して、古いマシン名を削除し、新しく書き換える必要があります。

注:実際は、DHCP はリブートしなくても IP アドレスを変更できます。ただし、ほとんどの環境ではこの方法をとることは、まずありません。

このオプションを使用するには、次の手順で実行します。

1. PowerTerm を使用して、UNIX システムに Telnet で接続します。

2. 現在の Windows IP アドレスを決定します。
   - Windows 95 では、**windows** ディレクトリにある **winipcfg.exe** プログラム (Microsoft 提供) を使用します。

   - Windows NT では、**winnnt\system32** にある **ipconfig.exe** プログラムを使用する必要があります。

3. UNIX システムでは、現在の実際の Windows IP アドレスであると UNIX システムが見なすマシン名を決定する必要があります (詳細については、『*.rhosts ファイル用の正式マシン名の決定*』の項を参照)。
   - ユーザのシステムが、DNS で構成されている場合は (ファイル **/etc/resolv.conf** が存在)、**nslookup** コマンドを使用して名前を決定できます。nslookup の後に、現在の Windows IP アドレス (ドット区切りの 10 進) を入力します。すると、IP アドレスに対応するマシン名が表示されます。

   - hosts ファイルを使用している場合は、hosts ファイルを編集したり、grep を使用して Windows IP アドレスを見つける必要があります。

4. **.rhosts** ファイルを編集して、古い動的 IP アドレス名を削除します。 新しい名前を追加してファイルを保存します。

5. これでパブリッシュできます。

代替セキュリティ機構 (SCP デーモンメソッド)

動的 IP アドレスのセキュリティ上の問題のため、Net Express UNIX オプションには、RSH セキュリティ機構にかわるセキュリティ機構が用意されています。 SCP をデーモンとして使用するこの代替方式を使用するためには、UNIX パブリッシャを構成する必要があります。 詳細については、『パブリッシャの設定』の章を参照してください。

サーバの構成

サーバを構成する手順は、次のとおりです。 これらのコマンドはスーパーユーザ権限で実行する必要があります。

1. SCP プログラムを scpd にリンクします。 たとえば、次のように入力します。
2. `ln /cobol_install_dir/bin/scp /usr/local/etc/scpd`

cobol_install_dir は、Server Express のインストール先のディレクトリです。

コマンドを実行する際には実際のディレクトリ名を入力してください。

3. 監視対象のネットワークポートを選択します。

   SCP デーモンが起動されると、特別なネットワークポートを監視します。 デフォルトのネットワークポートは 696 です。システムでこのネットワークポートがすでに使用されている場合、または別のポートを使用する場合には、SCP デーモンの起動時に -p 引数でネットワークポートを変更できます。 たとえば、scpd -p 900 のように指定します。

   ポートビジーなどのエラーは、標準のシステム syslog 機能に報告されます。 **/etc/syslog.conf** をチェックして、syslog の出力を受信するファイルを調べることができます。

4. デーモンモードを選択します。

   SCP デーモンは、コマンド行、シェルスクリプト、または **inetd** サーバから起動できます。

   コマンド行から起動する場合は、システムの起動プロセスの一部で、サーバを起動してください。 システムの起動ファイルは、システムによって異なります。ほとんどの SVR4 システムでは、**/etc/rc2.d** です。

   inetd サーバから起動する場合は、構成がさらに複雑になります。 ただし、システムは必要になったときにデーモンを自動的に起動し、必要なくなるとデーモンを終了します。 次のように構成してください。

   1. SCP デーモンが監視するポート番号をサービスファイル **/etc/services** に追加します。 追加する行は次のようになります。
   2. `mf-scpd 696/tcp # Micro Focus SCP Daemon`
   3. **inetd** の構成ファイル **/etc/inetd.conf** に次のような行を追加します。 この定義はシステムによって少し異なります。

      ```
      mf-scpd stream tcp nowait root /usr/local/etc/scpd scpd
      ```

   4. SIGHUP シグナルを **inetd** プロセスに送り、構成ファイルを再度読み込みます。 これを行うには、`ps -eaf | grep inet` などのコマンドを実行してプロセス ID を見つけ、`kill -1 processid` コマンドを実行します。
5. r-command のサーバへのアクセスを許可するかどうかを選択します。

r-command の UNIX パブリッシャへのアクセスを制限する場合は、SCP プログラムを削除するか、または、プログラムの名前を変更します。

セキュリティ拡張の使い方

セキュリティ設定を使用可能にした後、UNIX オプションがサーバとの接続を開始すると、サーバ上のユーザパスワードの入力を求めるダイアログボックスが表示されます。 正しくないパスワードを入力すると、UNIX パブリッシャは失敗します。

SCP デーモンは、UNIX サーバで使用される UNIX 認証方式を理解する必要があります。 次のものがサポートされています。

- 従来の UNIX パスワードファイル (**/etc/passwd**)
- シャドーパスワードファイル (**/etc/shadow**)
- SecureWare

次のものはサポートされていません。

- AFS
- DCE
- Kerberos

Samba のインストール

Samba は、標準的な PC 形式のネットワークを使用して、UNIX ファイルやプリンタを PC と共有できるようにします。 Samba は、UNIX マシンから PC へのアプリケーションのインポートを要求します。

注: Samba は、いわゆる「TCP/IP 上の NetBIOS」を実装します。これは、UNIX 用の NetBIOS サポートの実装方法の 1 つです。 システムに他の実装方法をパッケージしているシステムベンダもあります。 複数の実装方法を可能にしようとする場合は、2 番目の方法は初期化に失敗します。このサービス用のネットワークポートが使用中だからです。 このサービスを実装するには、1 つのパッケージを選択する必要があります。

既存の TCP/IP 上の NetBIOS パッケージがある場合は (ユーザ自身の Samba バージョンも含む)、UNIX オプションのインポート機能は既存のものを使用します。UNIX オプションは、API への標準的なファイルアクセス経由で UNIX ファイルをコピーするのみです。

Micro Focus は、Net Express CD に付属する Samba 製品以外のパッケージを使用して発生した問題については、サポートしません。

Windows のネットワークに精通していない場合は、Windows のエクスプローラを使用してネットワークドライブをマップする方法、または **net use** コマンドの詳細については、Windows のマニュアルを参照してください。

UNIX システムに Samba をイントールする方法を次に示します。

1. 使用しているオペレーティングシステムに適した Samba のバージョンを Net Express CD から UNIX システムの一時ディレクトリにコピーします。

2. Samba の構成ファイルがない場合は、構成ファイルのサンプルを含む **tar** ファイルを **unix\samba\smbconf.tar** から、UNIX システム上の一時領域へコピーします。

3. ルート権限があることを確認します。

4. すでに Samba が実行されていないことを確認します。ps コマンドを使用して、実行中のプロセスを調べます。たとえば、コマンド `ps -eaf | grep mbd` を入力して、nmbd または smbd というプロセスが実行されていないことを確認します。ps コマンドでこれらのどちらかのプロセスが表示された場合には、システムにすでにインストールされている Samba が実行されているため、インストールを中止する必要があります。

   現在インストールされている Samba を更新したい場合には、更新するときに現在のユーザの作業を中断しないようにする必要があります。 再び必要なる場合のために、既存の Samba ファイルを安全な位置にコピーします。

   `kill` コマンドを使用して既存の nmbd と smbd プロセスを終了してから、この後の指示に従ってインストールを実行します。 マスターデーモンプロセスのみを kill する必要があります。つまり、親プロセス ID (PPID) が 1 (init プロセス) であるプロセスです。これらの init プロセスは他のすべての Samba プロセスをシャットダウンします。

5. コマンド `tar xvf samba.tar` を使用して、tar アーカイブから Samba ファイルを展開します。

6. `installsmb.sh` を実行してバイナリを **/usr/local/samba** に、マニュアルページを **/usr/local/man** にそれぞれインストールします。

7. 有効な Samba 構成ファイルがない場合には、サンプルファイルを tar ファイルから展開します。

   ```
   tar xvf smbconf.tar
   ```

   一連の **smbconf.*description*** ファイルが作成されます。 各構成ファイルの先頭には、その目的に応じてコメントがあります。 早く開始したい場合は、**smbconf.basic** を選択し、**/usr/local/samba/lib/smb.conf** へコピーします。

   smb.conf ファイルの編集方法についての詳細は、次の『*Samba の構成*』の項を参照してください。

8. ネームデーモンを開始します。/usr/local/samba/bin/nmbd -D

9. セッションデーモンを開始します。/usr/local/samba/bin/smbd -D

10. エラーメッセージについては、**/usr/local/samba/var** のログファイル **log.smb** と **log.nmb** をチェックしてください。

11. **smbclient** コマンドを使用して、定義した共有ファイルが動作していることを確認してください。マシン名が **unixserv** である場合は、次のように入力します。

    /usr/local/samba/bin/smbclient -L unixserv

    パスワードの入力を求められた場合は、**Enter** キーを押します。エラーが生じた場合は、Samba のログファイルを確認してください。

12. Samba が開始されて実行中であることを確認したら、ユーザのシステムのスタートアップファイルに Samba を開始するコマンドを追加する必要があります。手順については、システム管理者または UNIX システムのマニュアルを参照してください。一般にシステムのスタートアップファイルの場所は、**/etc/rc2.d** です。

Samba の構成

Samba は複合製品であり、Samba 管理者が変更可能な 100 以上の異なる構成のオプションをもっています。その目的は、異なる構成すべてをカバーすることではなく、Samba サーバのセットアップ時に生じる共通の問題を解決することです。

Samba の man ページは Samba バイナリと同時にインストールされ、詳細な参照情報を含んでいます。man ページにアクセスするには、**man** コマンドがページを見つけられるように MANPATH 環境変数に **/usr/local/man** を追加します。最も重要な man ページは次のとおりです。

- samba - Samba の概要
- smbd - セッションマネージャデーモン
- nmbd - ネームマネージャデーモン
- smb.conf - Samba の構成ファイル

さらに、Samba にはソースコードの一部として多くのマニュアルと FAQ があります。Samba のソースコードは Net Express CD の **unix\samba\sambasrc.tar** に付属しています。ソースコードを展開すると、マニュアルが **docs** サブディレクトリに格納されます。このマニュアルは、初歩的な問題から高度に技術的な問題まで説明しています。Samba について問題がある場合、または構成方法について詳しく調べたい場合に、非常に役立ちます。このマニュアルを参照しても Samba の構成に問題がある場合は、**docs** ディレクトリに **DIAGNOSIS.txt** という非常に有用なマニュアルがあります。

ゲストアカウント

Samba のセットアップ時に生じる共通の問題の 1 つに、ゲストアカウントの値があります。ゲストアカウントは **smb.conf** ファイルで指定されます。このユーザ ID はクライアントが利用できる共有リストのようなものとして、Samba が使用します。これは、一般公開された共有ファイルを実際のゲストユーザとして明確に使用する場合と同じようなものです。

smb.conf ファイルで指定された値が存在し、有効な UID をもっていることは非常に重要です。ほとんどの UNIX システムは、特権のないネットワークユーザをデフォルトでもっています。一般例として、**nobody**、**nouser** および **network** がいろいろな UNIX システムで提供されています。システムの **/etc/passwd** ファイルを確認して、どれがサポートされている。または、**smb.conf** どれがファイルが正しい値をもっているかを確認してください。適切なユーザ名をもっていない場合は、作成する必要があります。

注：HP-UX では、**nobody** アカウントの UID (ときには GID も) は、負の値です (これは違反です)。Samba へのゲストアカウントにこのユーザ名を使用したい場合は、**nobody** アカウントの UID と GID を未使用の正の数値に変更する必要があります。

ユーザ認証

PC が OSR2 以前のバージョンの Windows 95、または Windows NT V4.0 SP3 を使用している場合には、Samba でのユーザ認証に問題は生じません。

ただし、Windows 95 OSR2 および Windows NT V4.0 SP3 のリリースで、Microsoft は、クライアントシステムが NetBIOS サーバにユーザ認証される方法を変更しました。変更点は、サーバに送信されるパスワード情報が、非暗号化形式から暗号化形式に変わったことです。Microsoft が使用している暗号化パスワード形式は、UNIX **passwd** ファイルが使用している形式と互換性がありません。このため、2 つの暗号化パスワード文字列が一致するかどうかは判別不能です。

注：非暗号化パスワードの使用は、UNIX の **telnet** または **ftp** セッションでパスワードを入力するのと同様に安全です。

Windows 95 と Windows NT の上記リリースに対して、Samba セキュリティを構成する方法は 3 つあります。

- Windows クライアントでの非暗号化パスワードの再使用可能

- UNIX での **smbpasswd** ファイルの構成
- Windows NT ドメインサーバを使用した認証

詳細については、次を参照してください。

Windows クライアントでの非暗号化パスワードの再使用可能

Windows クライアントの古い非暗号化パスワードサポートをレジストリ設定経由で、再使用可能にできます。 Windows 95 と Windows NT のレジストリ設定は、docs ディレクトリ内の Samba ソースコードに付属しています。

利点

- UNIX passwd データベースが、ユーザを認証するために使用されます。ユーザの UNIX パスワードと Samba パスワードはいつも同じです。

欠点

- NT サーバの新バージョンは、このレジストリ設定から非暗号化パスワードを受け付けず、ログインを拒否します。
- レジストリの変更は、各クライアントシステム上で適用する必要があります。

Samba の設定

```
[global]
security=user
  encrypt passwords = no
```

UNIX での smbpasswd ファイルの構成

Samba は、NetBIOS 形式の暗号化パスワードをサポートしてビルドされています。 NetBIOS 形式のパスワードは、特別な **smbpasswd** ファイルに格納されます。**smbpasswd** ファイルは、**smbpasswd** コマンドを使用して手作業で作成されます。

利点

- Windows クライアントを変更する必要がありません。

欠点

- UNIX パスワードと NetBIOS パスワードは異なるファイルに維持されるので、管理上の負荷が増大し、パスワードを誤る可能性があります。

Samba の設定

```
[global]
security=user
  encrypt passwords = yes
```

方法

- **root** アクセス権をもっていることを確認してください。
- **smbpasswd** 用の専用ディレクトリを作成します。デフォルトでは、`/usr/local/samba/private` です。このディレクトリは **root** によって所有され、**root** のみがアクセス可能である必要があります。アクセス権は、`rwx------` です。
- **smbpasswd** コマンドを使用して、要求されるユーザ ID を追加します。たとえば、`smbpasswd -a myuserid` のように追加します。

注：Samba のソースコード **docs** ディレクトリには、役に立つヒントやプログラムがあります。ユーザ ID を既存の **passwd** データベースから **smbpasswd** へ移行するときに参考になります。さらに、クライアントからユーザがパスワードを変更する方法についての詳細な説明もあります。

Windows NT ドメインサーバを使用した認証

Samba は Windows NT ドメインサーバに対してユーザを認証できます。Samba は実際にはドメインに参加できませんが、確認のために暗号化パスワードを Windows NT サーバに転送できます。

利点

- 共通ネットワークパスワードが、Windows NT と UNIX ネットワークとで共有されます。
- Windows クライアントを変更する必要はありません。

欠点

- Windows NT サーバが必要です。

Samba の設定

```
[global]
  encrypt passwords = yes
  security = server
  password server = nt-server-name
```

特定のユーザ ID とパスワードについて Windows NT の認証に失敗した場合は、セキュリティモードが `security = user` に戻り、Samba は **smbpasswd** ファイルが存在する場合はこのファイルでユーザを認証しようとします。

注: セキュリティ認証方法の使用に関係なく、実際の UNIX ユーザ ID は認証されたユーザ名用のファイル **/etc/passwd** に存在する必要があります。 Samba がこのファイルを要求するのは、ユーザに適切なアクセス権を設定したり、ファイル所有者のアクセス権を確立したりするためです。

複数の IP サブネット

NetBIOS は、本来、NetBEUI トランスポートを使用する単一の孤立した LAN 上で、小規模なワークグループ用に設計されました。 この設計の結果として、NetBIOS は、マシン名と共有可能情報のリストを作成して、LAN 内の全マシンにブロードキャストします。

TCP/IP 上の NetBIOS の導入によって、ブロードキャスト方式は機能しなくなりました。 TCP/IP は、サブネット内でのみブロードキャストを行うプロトコルです。 他の方法は、異なる IP サブネット間では、マシン名と共有情報を配布する必要があります。 解決方法は、Windows Internet Naming Service (WINS) です。

複数の IP サブネットがあるときは、サブネット間で情報を関連させるには WINS サーバをもつ必要があります。 Windows NT と Samba は両方とも WINS サーバを含みます。 一般に、Windows NT サーバがある場合は、付属の Samba よりはむしろ WINS サーバを使用してください。 WINS サーバはネットワークごとに 1 つのみです (ただし、Windows NT は、バックアップ用 WINS サーバをもてます)。

クライアントの構成

どのクライアントシステムも TCP/IP プロパティで構成された WINS サーバをもつ必要があります。 この構成は手動で行う。または、DHCP サーバを経由して一括で行います。 クライアントシステムは、このサーバに自身を登録し、マシン名と共有情報を検索するときはこのサーバに問い合わせます。

Samba の構成

Samba の構成は、Samba と Windows NT (またはリモートマシン上の Samba) のどちらを WINS サーバ上で実行しているかによります。

- **Samba が WINS サーバの場合**
- ```
  [global]
  wins support = yes
  ```
  ```

- **他のマシンまたはリモートの Samba が WINS サーバの場合**
- [global]
- wins support = no
  wins server = *wins-server-name*

ここで *wins-server-name* は、WINS サーバマシンの名前です。

Samba のインストール先の変更

基本的なインストール先 **/usr/local/samba** は、実際には Samba バイナリにコンパイルされています。Samba のインストール先を変更したい場合は、付属の Samba ソースコードを再コンパイルするか、または、コマンド行と構成ファイルオプションでファイルの位置を変更できます。

ほとんどの場合は、新しい基本ディレクトリで Samba ソースコードを再コンパイルする方法が簡単で確実です。Samba Makefile を 1 行変更するのみです。

注:Samba の機能によっては (たとえば、複数コードページのサポート)、機能させるために、新しい基本ディレクトリで Samba を再コンパイルする必要があるものもあります。

Samba の再コンパイルが実用的でない場合は、次の例のように変更を加える必要があります。この例では、新しいインストール先を **/home/samba** と仮定しています。

nmbd および **smbd** コマンドに、オプションを追加して、新しい構成ファイルの位置とログ出力を格納する位置を指定します。

```
nmbd -s /home/samba/lib/smb.conf -l
/home/samba/var/nmb.log -D
smbd -s /home/samba/lib/smb.conf -l
/home/samba/var/smb.log -D
```

Samba 構成ファイル内で、[global] セクションに次の記述を追加します。

```
lock directory = /home/samba/var/locks
smb passwd file = /home/samba/private/smbpasswd
smbrun = /home/samba/bin/smbrun
```

付録 B 章 : 正規表現

UNIX オプションでは、インポート中またはパブリッシュ中に正規表現を使用して検索操作や置換操作を実行します。

検索操作や置換操作では、テキストファイルは、連続した行として読み取られます。 各行は、適用可能なすべての検索パターンまたは置換パターンを使用して処理されます。 これらのパターンが適用される順序は、UNIX オプションのセットアップ時に指定した順序によって制御されます。

注:各行は個別に処理されるので、複数行にわたって検索する可能性のあるパターンは指定できません。

正規表現の構文は、UNIX の grep コマンドの構文とよく似ています。 正規表現には、通常の文字とメタ文字の両方を使用します。 メタ文字を使用すると、他の通常の文字に関する特別な意味を伝えたり、その意味を変更したりできます。 たとえば、PC で DOS プロンプトを使用したことのあるユーザにはなじみ深い **dir *.*** コマンドのメタ文字は、アスタリスク (ワイルドカード) で、0 個以上の通常の文字を表します。

検索パターン

検索パターンの定義に使用できるメタ文字は、次のとおりです。

メタ文字	説明
^	行の先頭を検索します。 文字クラスではクラスを無効化します。
$	行末
.	すべての文字を検索します。
[	文字クラスの先頭を示します。
]	文字クラスの末尾を示します。
*	直前にある正規表現を 0 個以上含む文字列を検索します。
+	直前にある正規表現を 1 つ以上含む文字列を検索します。
?	直前にある正規表現を含まない文字列、または、1 つ含む文字列を検索します。
|	左右にある正規表現を検索します。
(	サブ文字列の先頭を示します。

)	サブ文字列の末尾を示します。
"	定数文字列の文字を区切ります。
\	エスケープ文字です。

置換パターン

置換パターンの定義に使用できるメタ文字は、次のとおりです。

メタ文字	説明
&	検索パターンと一致する文字列を示します。この後に 1 ～ 9 までの数字 (*n*) が続く場合には、この文字列は、サブ文字列の数字 *n* と一致します。
\	エスケープ文字です。

エスケープ文字

エスケープ文字は、メタ文字がもつ特殊な意味を無効化するために使用します。

たとえば、検索パターン $HOME には、特殊な意味をもつドル記号 ($) が使用されているため、これを検索できません。この文字列を正しく検索するには、\$HOME のように指定します。円記号 (\) は、これに続く文字の特殊な意味を無効化する文字です。

さらに、エスケープ文字を使用して、別の方法で表示しにくい、または、表示できない特殊文字を定義できます。これらをエスケープシーケンスと呼びます。使用できるエスケープシーケンスは、次のとおりです。

エスケープシーケンス	説明
\b	バックスペース
\e	ASCII エスケープ文字
\f	用紙送り
\n	改行
\r	キャリッジリターン
\s	空白文字
\t	タブ
\\	円記号文字

ddd	1 ～ 3 桁の 8 進数 (*d*)
\x*dd*	1 ～ 2 桁の 16 進数 (*d*)
\x^*c*	文字 (*c*) で指定された制御文字

ファイル名パターン

検索や置換のダイアログでは、ファイル名パターンに完全な正規表現ではなく、構文と一致する標準 UNIX 形式のワイルドカードを使用します。 認識できるメタ文字は、次のとおりです。

メタ文字	説明
*	0 文字以上の文字列を検索します。
?	1 文字を検索します。
[]	1 文字の文字クラスを定義します。
\	直前にある文字の特殊な意味を無効化します。 円記号を検索するには、\\ と指定します。

検索例

次の例では、さまざまなメタ文字を紹介します。

検索パターン	説明
^Start	テキスト行の先頭にある Start という単語を検索します。
End$	テキスト行の末尾にある End という単語を検索します。
	注:UNIX オプションでは、CRLF や LF などの行の終了文字は検索対象になりません。
file\.dat	テキスト行で file.dat と完全に一致する文字列を検索します。
	注:ピリオドはメタ文字なので、. の前にエスケープ文字を使用します。
file.\.dat	メタ文字の例です。. は、有効な 1 文字を示します。 このパターンを指定すると、filea.dat、fileX.dat、file9.dat のような文字列を検索できます。
file..\.dat	メタ文字は、複数回使用できます。 この例では、file、任意の 2 文字、.dat の順に並ぶ文字を含む文字列を検索できます。

file..?\.dat	繰り返し用メタ文字の例です。文字 ? を指定すると、直前にある正規表現 (この場合は、2 番目のメタ文字 .) を含まない文字列、または、1 つ含む文字列を検索できます。この例では、file、任意の 1 文字または 2 文字、.dat の順に並ぶ文字を含む文字列を検索できます。
file.*\.dat	この例では、別の繰り返し用メタ文字を使用しています。* を指定すると、前にある正規表現 (この場合は、メタ文字 .) を 0 個以上含む文字列を検索できます。この例では、file、有効な文字 (0 個も含む)、.dat の順に続く文字列を検索できます。
file[ABC]\.dat	文字クラスの例です。文字クラスには、有効な文字のリストが含まれます。この場合は、文字 A、B、C が有効な文字として定義されています。このパターンでは、fileA.dat、fileB.dat、fileC.dat などが検索されます。
file[0-9]\.dat	文字クラスでは、ハイフン (-) を使用して文字範囲を指定できます。この例の文字クラスは、0 ～ 9 までの数字です。このパターンでは、file、数字、.dat の順に続く文字列を検索できます。
file[0-9A-F]+\.dat	ここまでで最も複雑な例です。文字クラスには、0 ～ 9 までと A ～ F までの 2 つの範囲が指定されています。これは、16 進数を示します。メタ文字 + を使用すると、直前にある正規表現 (この場合は、文字クラス) を 1 つ以上含む文字列を検索できます。このパターンでは、file、1 つ以上の 16 進数、.dat の順に続く文字列を検索できます。
file(\.dat)?	サブ文字列を使用すると、複数の文字を 1 つの論理正規表現にまとめることができます。この例では、パターン \.dat は、サブ文字列に含まれており、後にメタ文字 ? が続いています。? を使用すると、直前にある正規表現 (この場合は、サブ文字列全体) を含まない文字列、または、1 つ含む文字列を検索できます。この例では、file または file.dat を検索できます。
	注:サブ文字列がない場合には、パターン file\.dat? では、file.da または file.dat を検索することになります。
file\.(dat)|(idx)	この例では、サブ文字列とオプションのメタ文字 | が使用されています。オプションのメタ文字を使用すると、左側または右側の正規表現を検索できます。このパターンでは、file. の次に dat または idx が続く文字列を検索できます。
"file.dat"	検索文字列を二重引用符で囲むと、二重引用符内の他のメタ文字をすべて無効にできます (エスケープ文字は除きます)。この例では、file.dat を検索できます。

置換例

正規表現の本当の威力は、検索操作で検出された文字列を置換するときに顕著に発揮されます。サブ文字列の演算子は、置換対象にフォーカスを設定する場合に不可欠です。

検索パターン	置換パターン	備考
"file.dat"	newfile	定数文字列を検索し、別の定数文字列で直接、置換します。
(.*)\.htm	&1.html	末尾が .htm である文字列を検索し、次に、検索パターンと一致し、かつ、後に .html が続く文字列で置換します。
\"file([0-9A-F]+)\.dat\"	"newname&1.data"	この検索文は、上記の例を応用したものです。この文では、二重引用符内にある 16 進ベースのファイル名を検索します。二重引用符はエスケープ文字によりエスケープされています。16 進数を囲むサブ文字列の区切り文字によりフォーカスが設定されます。置換文字列は、newname、検索文字列の 16 進数、新しい拡張子の順に並ぶ文字列です。file9F.dat は、newname9F.data に置換されることになります。

付録 C 章 : 旧バージョンの UNIX オプションとの互換性

ここでは、Net Express Version 3 と Net Express Version 2 で提供される UNIX オプションの互換性について説明します。次の各項では、「旧 UNIX オプション」は、Net Express V2.0 で提供される UNIX オプションを指します。また、「新 UNIX オプション」は、Net Express V3.0 で提供される UNIX オプションを指します。

設定の保存

UNIX オプションの設定は、Net Express のプロジェクトファイルに保存されます。Net Express V3.0 では、これらの設定の保存形式が更新されています。UNIX オプションの設定が変更されるたびに、ファイルマッピング以外の旧 UNIX オプションの設定は、自動的に新 UNIX オプションの形式に変換され、デフォルト値が割り当てられます。

旧 UNIX オプションには、ファイルの種類という概念がなく、ファイルはすべてバイナリとして転送されます。 そのため、古いファイルマップがロードされると、すべてのファイルに、新 UNIX オプションのデフォルトで通常割り当てられるテキストではなく、バイナリが割り当てられます。

制限

旧 UNIX オプションでは、保存データをサーバ固有の部分とプロジェクト固有の部分に分けることはありません。 そのため、次のような制限があります。

- 前回パブリッシュ

  旧 UNIX オプションで定義された前回パブリッシュは、新 UNIX オプションでは無効になります。

- CGI サポート

  旧 UNIX オプションでは、CGI サポートはサーバごとに定義されます。 一方、新 UNIX オプションでは、プロジェクトごとに定義されます。 旧 UNIX オプションサーバから最初にロードされたサーバにより、CGI サポートフラグが有効化または無効化されます。

- ファイルマッピングの有効化

  旧 UNIX オプションでは、次の操作が可能です。

  - ファイル名マッピングを有効化するか、または無効化するかを選択できます。 ファイル名マッピングを無効化すると、ネイティブの PC ファイルシステム名が使用されます。

- ファイル名マッピング情報は、プロジェクトごとに適用されますが、ファイル名マッピングの有効化は、サーバごとに決定します。

新 UNIX オプションでは、ファイル名マッピングを無効化できません。すべてのファイル名マッピング情報は、プロジェクトごとに適用されます。プロジェクトファイルを最初にロードしたときに、旧 UNIX オプションで使用されていたデフォルトのサーバを基に、ファイル名マッピングが有効化されているか無効化されているかを判断します。ファイル名マッピングが有効化されている場合は、既存のファイル名マッピング情報が使用されます。無効化されている場合は、ファイルシステム情報を基にファイル名マッピングが行われます。

クライアント / サーバの互換性

旧 UNIX オプションで使用されるクライアント / サーバプロトコルは、新 UNIX オプションでは更新されています。新 UNIX オプションでは、機能が拡張され、性能が大幅に向上しています。プロトコルストリームには、バージョンを確認して、あるプラットフォームの新しい構成要素と他のプラットフォームの古い構成要素とが相互に操作できるようにする情報が含まれています。

Net Express V2.0 クライアントから新しい Net Express V3.0 SCP プログラムをインストールしたサーバへパブリッシュする場合には、互換性の問題はまったくありません。

Net Express V3.0 クライアントから古い Net Express V2.0 SCP プログラムをインストールしたサーバへパブリッシュする場合には、次のような制限があることにご注意ください。

- プロジェクトの論理ディレクトリを定義することができますが、論理ディレクトリに関連付けた実際のディレクトリは、[**BuildDir**] または [**CopyDir**] の実際のディレクトリと同一である必要があります。つまり、実際のターゲットディレクトリは 2 つまで定義できます。
- ソース妥当性確認操作は、サポートされません。
- サーバのディレクトリ自動作成機能は、サポートされません。
- Net Express IDE の出力ペインには、サーバでのビルド進捗は表示されません。出力ペインは、サーバでのビルドが完了したときのみ更新されます。
- サーバでビルドが開始されると、パブリッシュを中止できません。
- パブリッシャの性能は向上しません。