AIX バージョン 7.2

ネットワーク管理

注記

本書および本書で紹介する製品をご使用になる前に、761 ページの『特記事項』 に記載されている情報をお読みください。

本書は AIX バージョン 7.2 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典：
AIX Version 7.2
Network management

発行：
日本アイ・ビー・エム株式会社

担当：
トランスレーション・サービス・センター

目次

本書について

本書は、アプリケーション・プログラマーを対象に、AIX® オペレーティング・システムのグローバリゼーション用アプリケーションを使用可能にする方法を詳しく説明します。また、本書は、システム管理者を対象に、AIX オペレーティング・システムのグローバリゼーション 用ネットワーク環境を使用可能にする方法を詳しく説明しています。プログラマーおよびシステム管理者は、本書を使用して、グローバリゼーションのガイドラインおよび原則についての知識を得ることができます。各トピックには、ロケール、コード・セット、入力メソッド、サブルーチン、コンバーター、文字マッピング、各国固有の情報、およびメッセージ機能が含まれています。

強調表示

本書では、以下の強調表示の規則を使用しています。

項目	説明
太字	名前がシステムによって事前定義されているコマンド、サブルーチン、キーワード、ファイル、構造体、ディレクトリー、およびその他の項目を示します。 また、ユーザーが選択するボタン、ラベル、アイコンなどのグラフィック ・オブジェクトも示します。
イタリック	ユーザーが実際の名前や値を指定するパラメーターを示します。
モノスペース	具体的なデータ値の例、画面に表示されるものと類似したテキストの例、プログラマーが作成するものと類似したプログラム・コードの例、システムからのメッセージ、または実際に入力する情報を示します。

AIX における大/小文字の区別

AIX オペレーティング・システムでは、すべてケース・センシティブとなっています。これは、英大文字と小文字を区別するということです。 例えば、**ls** コマンドを使用するとファイルをリストできます。 LS と入力すると、システムはそのコマンドが `not found` (見つからない) と応答します。 同様に、**FILEA**、**FiLea**、および **filea** は、同じディレクトリーにある場合でも、3 つの異なるファイル名です。 予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

ネットワーク管理

このトピック集は、システム管理者とユーザーが、さまざまなネットワーク通信の作業を行う際に役立ちます。 システム管理者はこのトピック集で、TCP/IP 設定の構成、ネットワーク・セキュリティーの改善、システムのモニターなどの作業を行う方法に関する情報を見つけることができます。 ユーザーは、オペレーティング・システム用の通信アプリケーションおよびサービスの使用などの作業を行う方法に関する詳細情報を参照できます。

ネットワーク管理における新機能

「ネットワーク管理」トピック集の新規情報または大幅に変更された情報をお読みください。

新規情報または変更情報の参照方法

技術的な変更が加えられた箇所を参照するのに役立つように、インフォメーション・センターは以下のイメージを使用しています。

- >| のイメージは、新規情報や変更情報の先頭を示すマークです。
- |< のイメージは、新規情報や変更情報の終了を示すマークです。

2020 年 11 月

以下の説明は、このトピック集に加えられた更新の要約です。

- **チーミング**集約テクノロジーに関する情報が、425 ページの『イーサチャネル、IEEE 802.3ad リンク集約、チーミング』トピックに追加されました。

2019 年 11 月

以下の説明は、このトピック集に加えられた更新の要約です。

- 以下のトピックにおいて、`submit.cf` ファイルに関する情報が追加されました。
  - 7 ページの『メールの管理』
  - 104 ページの『メールのファイルとディレクトリー』
  - 46 ページの『システム・ブート時に sendmail デーモンを始動』
- 717 ページの『Serial over Ethernet デバイス・ドライバー』のトピックに関する情報が追加されました。Ethernet Device Server (EDS) を使用して、仮想シリアル・デバイスおよびテレタイプ・デバイスを AIX 論理区画に作成できます。
- 以下のトピックにおいて、複数の iSCSI ソフトウェア・イニシエーターに対するサポートに関する情報が追加されました。
  - 453 ページの『複数の iSCSI ソフトウェア・イニシエーター・デバイスの構成』
  - 451 ページの『iSCSI ソフトウェア・イニシエーターの構成』

ネットワークおよび通信の概念

ここに記載されている情報は、一般的なネットワークの原則に詳しくないシステム管理者を対象としています。 UNIX ネットワークに精通している場合は、ここに記載されている情報をスキップしてかまいません。

ネットワークとは、複数のコンピューターとそれらの接続リンクを組み合わせたものです。 物理 ネットワークとは、ネットワークを形成するハードウェア (アダプター、ケーブル、電話回線などの装置) のことです。 ソフトウェアと概念モデルは、論理 ネットワークを形成します。 ネットワークやエミュレーターは、タイプによってその機能が異なります。

最近のコンピューター・ネットワークが複雑になるにつれて、ネットワークの仕組みを説明するための概念モデルがいくつか作られるようになりました。 その中で最も一般的なモデルの 1 つが国際標準化機構のオープン・システム間相互接続 (OSI) 参照モデル (OSI の 7 層モデルとも呼ばれる) です。

OSI モデルの 7 つの層は次のように説明されます。

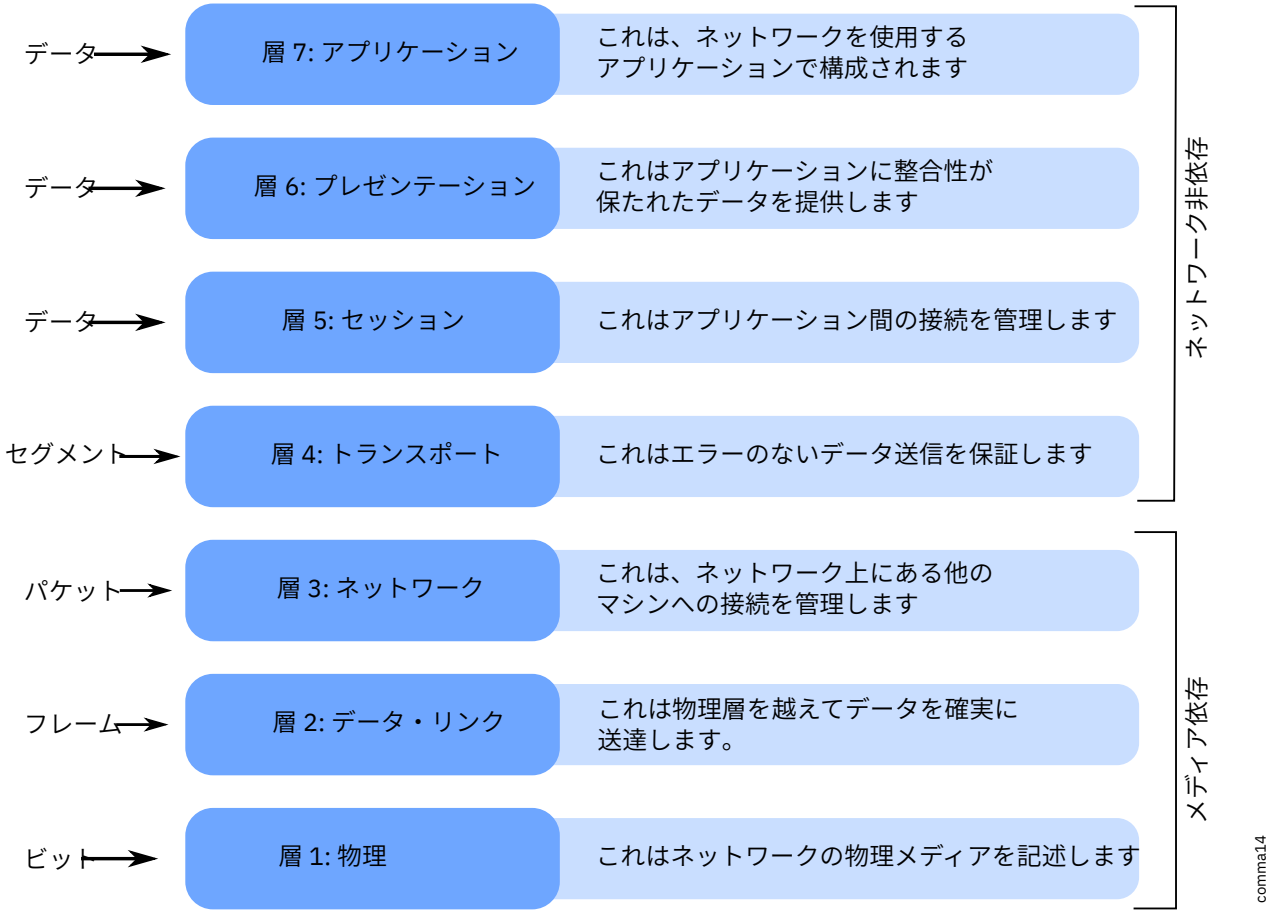


図 *1. OSI* 参照モデル

注 : OSI 参照モデルはネットワークの概念を説明するのには便利ですが、多くのネットワーク・プロトコルは OSI モデルに厳密には準拠していません。 例えば、TCP/IP を説明する場合は、アプリケーション層とプレゼンテーション層の機能をまとめることができ、セッション層とトランスポート層、データ・リンク層と物理層についても同様です。

ネットワークでは、複数のユーザーとアプリケーションの通信機能を使用できます。以下に例を示します。

電子メール (E メール) の送信
メッセージを別のユーザーに送信できます。 この 2 人のユーザーは、同一システム上にいても、別々の建物にある別々のシステム上にいても、別々の国にいてもかまいません。 ユーザーは、ソフトウェアとハードウェアの基礎となる層を物理ネットワークとともに使用して、メッセージ、手紙、メモ、招待状、およびデータ・ファイルを生成、送信、受信、および処理できます。 これらの通信は、物理ネットワーク上に存在するユーザーであれば、どのユーザーとの間でも行うことができます。

別の端末のエミュレーションまたは別のコンピューターへのログイン
コンピューターは、通信ネットワークを使用して、他のコンピューターをエミュレート (すなわち再現) したり、あたかも別のタイプのコンピューターや端末であるかのように情報にアクセスしたりできます。 ユーザーは、リモート・ログイン機能と対話式コマンド・ライン・インターフェースを 使用してリモート・システムにログインし、あたかもマシンをローカルで使用しているかのように同じプログラムとファイルにアクセスできます。

データの転送
システム間でデータを転送できます。 ネットワークを使用してマシン間でファイル、ディレクトリー、およびファイルシステム 全体を移行できます。 これにより、データをリモートでバックアップできる

ようになり、マシンで障害が発生した場合の冗長性が確保されます。 パスワード保護は、一般にプロトコルの一部として提供されています。 ファイル転送プロトコルには表示機能と制御機能が含まれていることが多く、 これによって読み取り/書き込みアクセス権をもつユーザーがファイルやディレクトリーを表示、定義、または削除できます。

リモート・ノードにあるプログラムを実行
システムのユーザーとアプリケーションが 他のシステムのプロシージャーとアプリケーションを 起動できるプロトコルがいくつかあります。 これは、科学技術計算アプリケーションで計算の多いルーチンを多量にオフロードするような、 さまざまな環境で役立ちます。

データ入力
データ入力とは、データをローカル・データ・ファイルまたはリモート・データ・ファイルに直接入力することです。 データ転送を 1 つのステップで行うことにより、精度と効率が当然のごとく向上します。

データ照会
データ照会とは、指定の情報をデータ・ファイルで検索することです。 データ更新とは、ローカル・ファイルまたはリモート・ファイルに保管されているデータを変更、追加、または削除することです。

リモート・バッチ入力
リモート・バッチ入力は、リモート・ロケーションからバッチ・データを入力することで、夜間やシステムがあまり使用されない時間帯によく行われます。 このようにさまざまな機能があるため、通信およびネットワークは、あれば便利という程度のものではなく、なくてはならないものになっています。

リソースの共用
リソースの共用は、ネットワークのもう 1 つの機能です。 ユーザーはプログラム、ファイル記憶スペース、周辺装置 (例えば、プリンター、モデム、端末、ハード・ディスクなど) のほか、データも共用できます。

データの共用
このようなシステム・リソースの共用はコスト効率に優れ、(プログラムとファイルを共有した場合は)、プログラムのコピーをいくつも保持したり、データの整合性を保ったりする手間を省きます。

他のオペレーティング・システムとの通信
ネットワークでは、さまざまなタイプのコンピューターを接続できます。 接続するコンピューターは異なるメーカーのコンピューターでも、同一のメーカーの異なるモデルでもかまいません。 複数のコンピューターが異なるオペレーティング・システムを 使用している場合でも、通信プログラムによるアクセスが可能となります。そのような通信プログラムは、ネットワーク上に別のプログラムがインストールされていることを要求することもあります。 また、TCP/IP やシステム・ネットワーク体系 (SNA) のような通信接続プロトコルがネットワーク上にあることを要求するプログラムが存在する可能性もあります。

物理ネットワーク

物理ネットワークは、ネットワーク上にある別のハードウェアを接続するケーブル (同軸ケーブル、対より線、光ファイバー、および電話回線)、ネットワーク (ホスト) に接続されたコンピューター上で使用されるアダプター、ネットワークで使用される任意の集線装置、中継器、ルーター、またはブリッジで 構成されます。

物理ネットワークは、その規模と、使用されるハードウェアのタイプにおいて 多種多様です。 一般的なネットワークは、ローカル・エリア・ネットワーク (LAN) と広域ネットワーク (WAN) の 2 種類です。 LAN の場合、通信は 1 つのオフィス・ビル、倉庫、大学構内のような、地理的にあまり広くない、1 km から 10 km まで (1 マイルから 6 マイルまで) の範囲内に限られます。 WAN では、LAN よりも地理的に広い範囲にわたって (例えば、国内や大陸間で) データ通信を行うことができます。 また、大都市圏ネットワーク (MAN) と呼ばれる中規模クラスのネットワークも存在します。 本書では、MAN を区別せず、WAN に含めて説明します。

LAN は通常、物理ネットワークとして標準 Ethernet (イーサネット)、IEEE 802.3 イーサネット、またはトークンリングのハードウェアを使用しますが、WAN および非同期ネットワークは、一般の電信電話会社が提供する通信ネットワークを使用します。 いずれの場合も、物理ネットワークの操作は通常、米国電子工業会 (EIA) または国際電気通信連合 (ITU) の規格に準じるネットワーク規格で制御されます。

ネットワーク・システム

すべてのネットワーク通信でハードウェアとソフトウェアが使用されます。 ネットワーク通信サポートは、ハードウェアと、そのハードウェアを作動させてネットワークに接続するために必要となるソフトウェアによって決定されます。

ハードウェア は、物理ネットワークに接続されている物理装置で構成されます。 ソフトウェア は、プログラムと、特定システムの操作に関連付けられているデバイス・ドライバーで構成されます。 システム・ハードウェアは、システム・ソフトウェアと物理ネットワークの間のパスまたはインターフェースを 提供するアダプターまたは 他のデバイスで構成されます。 アダプター には、システムにおける 入出力カード・スロットが必要です。 アダプターはすべてのインバウンド /アウトバウンド・データを用意し、アドレス検索を実行し、ドライバー、受信装置、およびサージ保護を提供し、さまざまなインターフェースをサポートし、一般的に、システム・プロセッサーが、多数の通信タスクを行わないですむようにします。

ネットワーク管理情報全体で以下の用語が使用されます。

プロトコル
プロトコルはセマンティクスと 構文規則のセットです。 このセットにより、情報の送達方法、情報を安全に宛先に送達するための情報の格納方法、および情報のパスが定義されます。 また、プロトコルにより、メッセージの流れとその受信の確認が調整されます。

アドレス
ネットワーク・ドメインは、同一インフラストラクチャー内における複数のコンピューター・ネットワーク/ホストの管理グループです。 ドメインは、共通制御に基づいてデータ処理リソースをネットワーク内に配置します。

例えば、インターネットの構造は、ドメインによるインターネット・プロトコル (IP) アドレスの定義方法を示しています。 インターネットは広範囲のネットワークであり、多種多様な小規模ネットワークで構成されています。 経路指定とアドレッシングを容易にするために、インターネット・アドレスはいくつかのドメインに分けて階層状に構成されており、例えば、最上層は企業関係のユーザーを表す com、教育機関のユーザーを表す edu、および政府機関のユーザーを表す gov というように、大きなカテゴリーに分類されています。 com ドメインには、例えば ibm のように、個々の企業に対応する多くのより小さいドメインが含まれています。ibm.com ドメインには、例えば austin.ibm.com や raleigh.ibm.com のような、さまざまなロケーションのインターネット・アドレスに対応する、さらに小さなドメインが含まれています。 このレベルでは、ホストの名前を識別できます。 ここでいうホストとは、ネットワークに接続されている何らかのコンピューターのことです。 austin.ibm.com 内に hamlet や lear という名前のホストがある場合、これらのホストのアドレスは、それぞれ hamlet.austin.ibm.com および lear.austin.ibm.com となります。

ゲートウェイとブリッジ
インターネット上には、多くの場合において、異なるハードウェアを使用し、異なるソフトウェアを実行する多種多様なネットワークが存在します。 ゲートウェイとブリッジは、このように異なるネットワーク間の通信を可能にします。ブリッジは、イーサネットのように、使用している論理リンク制御 (LLC) 手続きは同じでもメディア・アクセス制御 (MAC) 手続きは異なる可能性がある 2 つの LAN を接続する機能装置です。 ゲートウェイは、ブリッジより広い範囲をカバーします。 ゲートウェイは、リンク層より上位の層で機能し、必要に応じて、あるネットワークで使用されているインターフェースやプロトコルを、別の異なるネットワークで使用されているインターフェースやプロトコルに 変換します。 ゲートウェイでは、インターネットを 構成する各種ネットワーク間でデータを転送できます。

データの経路指定
アドレッシングにドメイン名を使用し、変換にゲートウェイを使用すると、転送しようとしているデータの経路指定が非常に容易になります。 経路指定とは、メッセージが宛先に到達するまでのパスを割り当てることです。 ドメイン名を使用すると、メッセージの宛先を効率よく定義できます。 インターネットのような大規模なネットワークでは、情報は宛先に到達するまでに通信ネットワークから通信ネットワークに次々と経路指定されます。 通信ネットワークごとにドメイン名が検査され、当該ネットワークで認識されているドメインに基づいて次の論理デバイス上に情報が経路指定されます。 このように、データを受信する各通信ネットワークが経路指定プロセスに関係します。

ローカル・ノードとリモート・ノード
物理ネットワークは、そのネットワーク上に存在するホストによって使用されます。 各ホストは、ネットワーク上のノードです。ノードは通信ネットワーク内のアドレッシング可能なロケーションであり、ホスト処理サービスを提供します。 このような各種ノードの相互通信はローカルまたはリモートとして定義されます。 ローカル とは、通信回線を使用せずに自分のシステムから直接アクセスできる

デバイス、ファイル、またはシステムに関係します。 リモート とは、自分のシステムから通信回線経由でアクセスできるデバイス、ファイル、またはシステムに関係します。 ローカル・ファイルが自分のシステムに存在するのに対し、リモート・ファイルは、物理ネットワーク (イーサネット、トークンリング、電話回線など) を使用して通信する相手となるファイル・サーバーや他のノードに存在します。

クライアントとサーバー

サーバーとは、ネットワーク上の他のコンピューターからアクセスされるデータまたは機能を備えたコンピューターのことです。クライアントとは、サーバーにサービスやデータを要求するコンピューターのことです。 一般的なサーバーのタイプとしては、ファイル・サーバー (ファイルを保管する)、ネーム・サーバー (名前とアドレスを保管する)、アプリケーション・サーバー (プログラムやアプリケーションを保管する)、印刷サーバー (印刷ジョブをスケジュールしたり宛先に送ったりする) があります。

クライアントは、コード・サーバーに対して、更新されたプログラム・コードやコード・サーバーに入っているアプリケーションの使用を要求できます。 名前またはアドレスを入手したい場合には、クライアントはネーム・サーバーに連絡します。 また、クライアントは、データ入力、照会、レコード更新を行うためにファイル・サーバーにファイルやデータを要求することもできます。

エミュレーター

エミュレーター は、ユーザーがそのシステムを、別の端末やプリンターを使用しているかのように機能させることができるソフトウェア・アプリケーションです。端末エミュレーター は、ホスト・システムに接続して、データまたはアプリケーションにアクセスします。 端末エミュレーターの中には、ホストとの間でファイルを転送する機能を提供するものもあります。 また別の端末エミュレーターには、プログラム間通信やホストのタスクの自動化のために、 アプリケーション・プログラミング・インターフェース (API) 機能をもつものもあります。プリンター・エミュレーター を使用すると、ホストはローカル・プリンター上でファイルを印刷するか、後で印刷または編集するために印刷可能なフォーマットで格納できます。

エミュレーション用の TCP/IP コマンド

伝送制御プロトコル/インターネット・プロトコル (TCP/IP) ソフトウェアには、**telnet** および **rlogin** コマンドが含まれており、これによってリモート TCP/IP システムに接続してアクセスできます。

項目	説明
telnet	**TELNET** プロトコルによりユーザーがリモート・ホストにログインできるようにします。 このコマンドは、トラステッド・コマンドである点で **rlogin** コマンドと異なります。 トラステッド ・コマンドとは、使用中のコンピューター上で構成されたすべてのセキュリティー・レベルの要求を満たすものです。 特別なセキュリティーを必要とするシステムは、トラステッド・コマンドのみを実行できるようにするべきです。 トラステッド・コマンド、プロセス、プログラムに関する規格は、米国国防総省によって設定され管理されています。
tn	**telnet** コマンドと同じ機能を実行します。
rlogin	ユーザーがリモート・ホストにログインできるようにします。 これは非トラステッド ・コマンド であり、特別なセキュリティーが必要なシステムでは使用できない場合があるという点で、**telnet** コマンドとは異なります。

TCP/IP については、106 ページの『伝送制御プロトコル/インターネット・プロトコル 』を参照してください。

注 : **bterm** コマンドは、両方向 (bidi) モードの端末をエミュレートします。

エミュレーション用の BNU コマンド

基本ネットワーク・ユーティリティー (BNU) ソフトウェアには、AIX オペレーティング・システムを使用するリモート・システムに接続するための **ct**、**cu**、および **tip** コマンドが含まれています。

項目	説明
ct	3161 などのリモート端末上で、電話回線を通じて別の端末装置と通信するためのものです。 これによりリモート端末上のユーザーは、もう一方の端末にログインして、そこで作業を実行できます。 **ct** コマンドは、**cu** コマンドに似ていますが、**cu** コマンドほど柔軟ではありません。 例えば、**ct** コマンドを通じてリモート・システムに接続中に、ローカル・システム上でコマンドを発行することはできません。 しかし、**ct** コマンドの場合は、接続が確立されるまでダイヤルを呼び出し続けたり、または一度に複数の電話番号を指定したりするように指示できます。
cu	端末を、UNIX または非 UNIX システムのいずれかに接続された別の端末に接続します。 接続が確立されると、BNU 通信リンクをドロップさせずに、どちらかのシステムでコマンドを実行して、同時に両方のシステムにログインできます。 リモート端末も UNIX で稼働している場合は、2 つのシステム間で ASCII ファイルを転送できます。 **cu** コマンドは複数のシステムを接続するために使用することもできます。 複数のシステムが接続された場合には、接続されたどのシステム上でもこのコマンドを実行できます。
tip	端末をリモート・システムに接続して、直接ログインしているかのようにそのリモート端末上で作業できるようにします。 **tip** コマンドは、 リモート・システムとの間でファイルを転送するために使用できます。 **tip** コマンドを使用すると、会話を記録するためにスクリプトを使用できます。 **注: tip** コマンドを使用するには、 リモート・システムでログインする 必要があります。

BNU については、489 ページの『基本ネットワーク・ユーティリティー 』を参照してください。

一般的なネットワーク・コマンド

以下のコマンドは、ユーザー、システム、およびロギングに関する基本情報を表示する場合に使用します。

表 *1.* 一般的に使用される通信コマンド

コマンド	説明
whoami	これはログイン名を表示します。
uname	これは、ご使用のシステムがネットワーク上にある場合に、ご使用のシステムの名前を表示します。

host *system_name*	これは、ご使用のローカル・システムが指定のシステムにアクセスできるかどうかを判別します。
	
	`# host system_name`
	
	システムに該当する情報があった場合は、次のような表示が戻されます。
	
	`<system_name> is 192.9.200.4 (300,11,310,4)`
	
	この場合は、メッセージをシステムに 送信できます。 メールを経路指定するためにシステムでアドレス 192.9.200.4 が使用されます。 システムに該当する情報がなかった場合は、次のような表示が戻されます。
	
	`<system_name>: unknown host`

表 1. 一般的に使用される通信コマンド (続き)

コマンド	説明

finger *system_name*| *user_name*	これは、特定のシステム/ホスト上のログオン・ユーザーを示し、特定のユーザーに関する情報を表示します。
	`# finger system_name`
	次のような表示が戻されます。
	`brown  Console Mar 15 13:19`
	`smith  pts0    Mar 15 13:01`
	`jones  tty0    Mar 15 13:01`
	`finger brown@<system_name>`
	または
	`finger brown`
	次のような表示が戻されます。
	`Login name: brown`
	`In real life: Marta Brown`
	`Directory:/home/brown  Shell: /bin/ksh`
	`On since May 8 07:13:49 on console`
	`No Plan.`

ユーザーのシステムで他のタイプの端末のエミュレーションを可能にするいくつかのアプリケーションが用意されています。

メールの管理

メール機能を使用すると、同一システム上のユーザーまたはネットワークで接続した複数のシステム上のユーザーと電子メールを交換できます。 ここでは、メール・システム、標準メール・ユーザー・インターフェース、**IMAP (Internet Message Access Protocol)**、および **POP (Post Office Protocol)** について説明します。

メール・システムはネットワーク間のメール送達機能であり、ユーザー・インターフェース、メッセージ経路指定プログラムとメッセージ送達プログラム (メール・プログラム) で構成されています。 メール・システムは、同一ホスト上のあるユーザーから別のユーザーへ、またはホスト間で、あるいはネットワークの境界を超えて、メッセージの受け渡しを行います。 また、メッセージを受信側ホストに適したフォーマットに変換するために一定の範囲内でメッセージ・ヘッダーの編集も行います。

メールのユーザー・インターフェース を使用すると、メッセージを作成して他のユーザーとの間で送受信することができます。 メール・システムは、**mail** と **mhmail** という 2 つのユーザー・インターフェースを提供します。 **mail** コマンドは、 すべての UNIX システムで使用できる標準的なメールのユーザー・インターフェースです。 **mhmail** コマンドはメッセージ・ハンドラー (MH) ユーザー・インターフェース、つまり、熟練ユーザー用に設計された拡張メール・ユーザー・インターフェースです。

メッセージ経路指定プログラム は、メッセージを宛先に経路指定します。 メール・システムのメッセージ経路指定プログラムは **sendmail** プログラムで、これは基本オペレーティング・システム (BOS) の一部であり、BOS とともにインストールされます。 **sendmail** プログラムは、`/etc/mail/sendmail.cf` ファイル、または `/etc/mail/submit.cf` 構成ファイル、および`/etc/mail/aliases` ファイルに納められている情報を使用して必要な経路指定を実行するデーモンです。

sendmail コマンドは、宛先までの経路のタイプに応じて、異なるメール・プログラムを使用してメッセージを送達します。

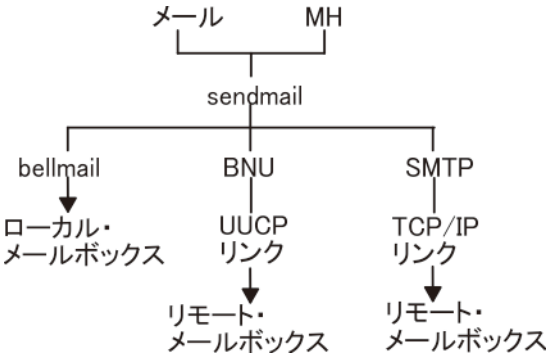


図 *2. sendmail* コマンドで使用するメール・プログラム

この図では、トップに Mail と MH があるトップダウン編成図のタイプを示します。 トップから bellmail、BNU、および SMTP が分岐しています。 そのすぐ下のレベルはそれぞれ、ローカル・メールボックス、UUCP リンク、および **TCP/IP** リンクです。 UUCP リンク と **TCP/IP** リンクの下は、リモート・メールボックスです。

図に示すとおり、

- ローカル・メールを送達する場合、**sendmail** プログラムは **bellmail** プログラムへメッセージを経路指定します。 **bellmail** プログラムは、ユーザーのシステム・メール・ボックスにメッセージを付けることによって、すべてのローカル・メールを配布します。このメールボックスは /var/spool/mail ディレクトリーにあります。
- メールを UNIX 間コピー・プログラム (UUCP) リンク上で配布する場合、**sendmail** プログラムは基本ネットワーク・ユーティリティー (BNU) を使ってメッセージを経路指定します。
- **TCP/IP** に経路指定されたメールを配布する場合、**sendmail** コマンドは、リモート・システムへの **TCP/IP** 接続を確立してから、**SMTP (Simple Mail Transfer Protocol)** を使用してリモート・システムにメッセージを転送します。

メールのユーザー・エージェント・プログラム

メール・システムを使用するには、事前にユーザー・エージェント・プログラムを選択しておく必要があります。 メール・プログラム (`mail`)、メッセージ・ハンドラー (`mh`)、または `bellmail` コマンドのいずれかを選択できます。

ユーザー・エージェント・プログラムは、メールの作成、送受信、 およびファイリングの機能を提供します。 また、他のシステムやパッケージからの着信メールの配布、送信メール項目の配布、 および 1 つ以上のリモート・システムの同様のプログラムへメール転送を行う転送エージェント・プログラム **sendmail** が必要です。

> **注:** `mail` と `mh` プログラムのメールの格納方法には互換性がありません。 メール・ハンドラーを 1 つだけ選ぶ必要があります。

メール・プログラム・インターフェース

mail プログラムは、 ローカル・ネットワーク・ユーザーおよびリモート・システム・ユーザーの両者の間で、 メールの送受信を可能にするユーザー・インターフェースを 提供します。

メールの本文には、エディターを使って入力したテキストや、ASCII ファイルを使用できます。 タイプ入力したメッセージやファイルに加えて、次に挙げるものを送信できます。

項目	説明
システム・メッセージ	システムが更新されたことをユーザーに知らせます。 システム・メッセージは、ブロードキャスト・メッセージと同じですが、ローカル・ネットワークでのみ送信できます。
機密メール	機密扱いの情報の送信に使用します。 機密メール・メッセージは暗号化されます。 このようなメッセージを読むには、受信者はパスワードを入力する必要があります。

項目	説明
不在メッセージ	受信者が不在であることをユーザーに知らせます。 受信者の留守中にメールを受信すると、受信者のシステムはメッセージを 送信元に送り返します。 このメッセージは、受信者が不在であることを伝えます。 不在の受信者あてのすべてのメールを、転送することもできます。

mail のサブコマンドを使用してメールを受信する場合に、 次のことを行うことができます。

- メールをシステム・メールボックスに残しておく。
- メールを読み、削除する。
- メールを転送する。
- メールに注釈を追加する。
- メールをユーザーの個人用メールボックス (mbox) に格納する。
- メールをユーザーが作成したフォルダーに格納する。
- メールやメール・メッセージの送信先を指定する別名ファイルまたは配布ファイルを作成し、 維持する。

sendmail は自動的にインストールされます。

mail プログラムの詳細は、10 ページの『メール機能』を参照してください。

メッセージ・ハンドラー (mh)

mh プログラムは、 コマンド・ラインから各メール処理機能を直接実行できるようにするコマンド群です。

これらのコマンドは、**mail** のサブコマンドよりも広範な機能を提供しています。 また、これらのコマンドは、コマンド・プロンプトが表示されている場合には、 いつでも出せるので、メールを作成し、受信メールを処理する上で大変便利で、柔軟性に富んでいます。 例えば、メール・メッセージを読み取り、 ファイルを検索したり、特定の処理をさせるためにプログラムを実行して、メッセージに応答できますが、 これらの処理をすべて同じシェル内で実行できます。

mh プログラムでは、 次に挙げるコマンドを使用すると、メッセージの作成、配布、受信、 表示、処理、および格納を行うことができます。

項目	説明
ali	メールの別名とそのアドレスをリストします。
anno	メッセージに注釈を付けます。
ap	アドレスを解析して再フォーマットします。
burst	ダイジェストをメッセージに拡張します。
comp	メッセージの作成や変更を行うためにエディターを始動します。
dist	メッセージを追加アドレスに再配布します。
dp	日付を解析してフォーマットを設定し直します。
folder	フォルダーやメッセージを選択し、リストします。
folders	フォルダーおよびメール・ディレクトリー内のメッセージをすべて、リストします。
forw	メッセージを転送します。
inc	新規のメールをフォルダーに取り込みます。
mark	メッセージ・シーケンスの作成、変更、表示を行います。
mhl	メッセージの定様式リストを作成します。
mhmail	メールの送受信を行います。
mhpath	メッセージおよびフォルダーの 絶対パス名を印刷します。
msgchk	メッセージが来ているかどうかを確認します。

項目	説明
msh	メール・ハンドラー (mh) シェルを作成します。
next	次のメッセージを表示します。
packf	フォルダーの内容をファイルに圧縮します。
pick	内容ごとにメッセージを選択し、シーケンスの作成および変更を行います。
prev	直前のメッセージを表示します。
refile	フォルダー間でファイルを移動します。
repl	メッセージに返信します。
rmf	フォルダーとフォルダー内に含まれるメッセージを除去します。
rmm	メッセージをアクティブな状況から除去します。
scan	1 メッセージにつき 1 行のスキャン可能なリストを作成します。
send	メッセージを送信します。
show	メッセージを表示します。
sortm	メッセージをソートします。
vmh	mh コマンドに使用するビジュアル・インターフェースを 始動します。
whatnow	ドラフトの後処理用のプロンプト・インターフェースを 始動します。
whom	mh アドレスを操作します。

bellmail コマンド

bellmail コマンドは AT&T UNIX の mail コマンドのオリジナルであり、同一システム上のユーザー、および UNIX 間コピー・プログラム (UUCP) の別名を持つ基本ネットワーク・ユーティリティー (BNU) を使用してアクセスできるリモート・システム上のユーザーのためにメールを処理します。

これらのプログラムは、ダイヤルアップ通信回線または専用 2 地点間通信回線によって接続されたシステムのネットワークのみをサポートします。 コマンドは、次に挙げることを実行するためのサブコマンドをもつシェルをオープンします。

- (入力されたか、または既存のファイルからリダイレクトされた) 標準入力からデータを取り込み、(コマンドの引数として与えられた) 1 つ以上のアドレスおよびタイム・スタンプを追加し、単一のコピーを各受信者のシステム・メールボックス・ファイル (/var/spool/mail/*UserID*) に付加する。
- ユーザーのシステム・メールボックス・ファイルからメール項目を読み取る。
- ユーザー専用のメールボックス・ファイル ($HOME/mbox) または指定されたファイルにメール項目を追加する。
- BNU を使って、メールを他のシステム上のユーザーに送信する。
- ユーザーのシステム・メールボックス・ファイルの先頭に *.forward* ステートメントを追加することによって、 ユーザーのシステム・メールボックスから他のシステム上のシステム・メールボックスにすべてのメールを自動的にリダイレクトする。

ただし、このメール・ハンドラーを活用できるようになるには、ユーザーは UNIX に精通していなければなりません。 詳しくは、**bellmail** コマンドを参照してください。

メール機能

ここでは、**mail** プログラムの機能について説明します。

mail プログラムを使用すると、ローカルまたはリモート・システムのユーザーと、メールの受信、作成、および送信を実行できます。

メールの格納
メールは特定の状況に応じてさまざまな方法で格納されます。

メールはユーザーのアドレスに送信されると、 メール用に特別に設けられたシステム・ディレクトリーに格納されます。 このシステム・ディレクトリーには、ローカル・システムの各ユーザーごとに 1 つのファイルが入っています。 このディレクトリーは、メールに対して何らかの処置を行うまで、そのメールを保持します。

システム・メールボックス
システム・メールボックスは私書箱に似ています。 郵便局は私書箱を所有する個人あてに手紙を配達します。

同様に、システム・メールボックスは、メッセージを特定のユーザーに配達するためのファイルです。 このファイルが存在しない場合には、このファイルはメールが到着したときに作成されます。 このファイルは、すべてのメッセージが除去されると、削除されます。

システム・メールボックスは、/var/spool/mail ディレクトリーにあります。 各システム・メールボックスの名前は、 それに関連するユーザー ID となります。 例えば、ユーザー ID が karen である場合、 そのシステム・メールボックスは

```
/var/spool/mail/karen
```

デフォルトの個人用メールボックス
個人用メールボックスはオフィスのインバスケットのようなものです 。 メールを受信した後、それをファイルする前にインバスケットに入れます。

各ユーザーには個人用のメールボックスがあります。 システム・メールボックスからメールを読むときに、保存されたか削除されたかを示すマークがファイルに付いていなければ、そのメールは個人用メールボックス $HOME/mbox ($HOME はログイン・ディレクトリー) に書き込まれます。 mbox ファイルは、中にメッセージがあるときだけ存在します。

未完メッセージのための *dead.letter* ファイル
他の作業を完了させるためにメッセージ作成作業を中断することが必要な場合、システムは未完メッセージを $HOME ディレクトリー内の dead.letter ファイルに保存します。

dead.letter ファイルが存在しない場合には、このファイルは新しく作成されます。 このファイルを後で編集して、メッセージを完了させることができます。

> **注意:** メッセージを格納する目的で dead.letter ファイルを使用しないでください。 割り込みが出されて dead.letter ファイルにメッセージの一部が保存されるたびに、このファイルの内容は上書きされます。

メール・フォルダー
フォルダーを使用すると、メッセージを系統的に保存できます。 メール・プログラムを使用すると、システム・メールボックス、個人用メールボックス、または他のフォルダーからメッセージをフォルダーに 入れることができます。

各フォルダーはテキスト・ファイルです。 各フォルダーは、**set folder** オプションを使って、.mailrc ファイル内で指定されたディレクトリーに入れられます。 メッセージを格納するためのフォルダーを使う前に、このディレクトリーを作成しておく必要があります。 ディレクトリーを作成すると、メール・プログラムは必要に応じてそのディレクトリー内にフォルダーを作成します。 .mailrc ファイル内にディレクトリーを指定しない場合、フォルダーは現行ディレクトリー内に作成されます。 17 ページの『メールの編成』を参照してください。

> **注:** メールを送受信するプログラムには、メッセージ・ハンドラー (MH)、**bellmail** プログラムをはじめとするいくつかのプログラムがあります。 どのプログラムを使うかは、システムに何がインストールされ、構成されているかによって異なります。 システム構成情報については、システム管理者に連絡してください。

メールの処理および受信

mail プログラムを使用すると、メールボックス内の各メッセージを検査し、個人用メール・ディレクトリー内のメッセージを削除またはファイルすることができます。

コマンド・シェルはメールが到着したことを知らせます。 MAIL 環境変数が設定されており、しかもシェルがメールを最後に検査した後に MAILCHECK によって指定された期間が経過していれば、そのことを知らせる通知が、次のプロンプトが出される前に表示されます。 この通知メッセージは MAILMSG 環境変数の値で設定されます。 この通知はどのシェル (bourne、korn または C シェル) を使用しているかによって異なりますが、次のようになります。

```
YOU HAVE NEW MAIL
```

メールボックスの始動

システム・メールボックスからのメッセージの読み取りや除去には、**mail** コマンドを使用します。

メッセージを格納するのにシステム・メールボックスを使用しないでください。 メッセージは自分の個人用メールボックスやメール・フォルダーに格納してください。

システム・メールボックスのメールのチェック

mail コマンドを使用して、システム・メールボックスのメールをチェックすることができます。

システム・コマンド・ラインのプロンプトから、次のように **mail** コマンドを入力します。

```
mail
```

システム・メールボックスにメールがない場合、システムは次のメッセージを出して応答します。

```
No mail for YourID
```

メールボックスにメールがある場合は、システムは次のようにシステム・メールボックス内のメッセージのリストを表示します。

```
Mail Type ? for help.
"/usr/mail/lance": 3 messages 3 new
  >N    1 karen Tue Apr 27 16:10 12/321 "Dept Meeting"
   N    2 lois  Tue Apr 27 16:50 10/350 "System News"
   N    3 tom   Tue Apr 27 17:00 11/356 "Tools Available"
```

現行メッセージには、先頭に「より大」記号 (>) が付けられています。 各行のエントリーはそれぞれ、次に示すフィールドを表示しています。

項目	説明
status	メッセージのクラスを示します。
number	メール・プログラムに対してメールを識別する番号です。
sender	メールを送信した人のアドレスを識別します。
date	メッセージを受信した日付を示します。
size	メッセージに含まれる行数と文字数 (ヘッダーも含む) を定義します。
subject	存在する場合には、メッセージの件名を識別します。

status は次のいずれです。

項目	説明
N	新規メッセージ
P	システム・メールボックスに保存されるメッセージ
U	未読メッセージ。 これは、前回のメール・プログラムを使用したときにメールボックス内にリストされたが、 その内容は読まれなかったメッセージです。

項目	説明
*	ファイルまたはフォルダーに保存されたか、または書き込まれたメッセージ。

状況表示の付いていないメッセージは、既に読まれたが、削除または保存されていないメッセージです。

個人用メールボックスまたはメール・フォルダーのメールのチェック
mail コマンドを使用して、個人用メールボックスまたはメール・フォルダーのメールをチェックすることができます。

システム・コマンド・ラインのプロンプトから、次の手順のように **mail** コマンドを使用できます。

1. 個人用メールボックス $HOME/mbox のメッセージのリストを表示するには、次のように入力します。

   ```
   mail -f
   ```

   個人用メールボックスにメールがない場合、システムは次のようなメッセージを出して応答します。

   ```
   "/u/george/mbox": 0 messages
   ```

   または

   ```
   A file or directory in the path name does not exist
   ```

2. dept フォルダーのメッセージのリストを 表示するには、次のように入力します。

   ```
   mail -f +dept
   ```

   メール・フォルダーにメールがない場合、システムは次のようなメッセージを出して応答します。

   ```
   A file or directory in the path name does not exist
   ```

メールボックスの内容の表示オプション

メールボックスのプロンプトからメールボックスのサブコマンドを入力することにより、メールボックスの内容を管理できます。

前提条件

1. システムにメール・プログラムがインストールされていること。
2. メール・プログラムが始動されていること。
3. メールボックス内にメールがあること。

特定の範囲のメッセージ
すべてのメッセージをブラウズしないで済むように、指定したメッセージのリスト内にあるメッセージを表示するには、**h** サブコマンドを使用します。

メールボックスのプロンプトから、次の例のように **h** サブコマンドを使用できます。

項目	説明
h	一度に約 20 個のメッセージを表示できます。 実際に表示される数は、使用している端末の種類と .mailrc ファイル内の **set screen** オプションによって決まります。 **h** サブコマンドを再度入力すると、同じ範囲内のメッセージが表示されます。
h 21	メッセージ 21 から 40 までのメッセージが (メールボックス内にその数のメッセージがあれば) 表示されます。 すべてのメッセージが表示されるまで、それ以降のメッセージ番号と共に **h** サブコマンドを入力します。
h 1	20 個のメッセージの最初のグループに戻るには、1 から 20 の範囲の番号のいずれかを入力します。

メールボックスのスクロール
メールボックスをスクロールするには、**z** サブコマンドを使用します。

メールボックスのプロンプトから、次の例のように **z** サブコマンドを使用できます。

項目	説明
z	一度に約 20 個のメッセージを表示できます。 実際に表示される数は、使用している端末の種類と .mailrc ファイル内の **set screen** オプションによって決まります。 **z** サブコマンドを再度入力して、次の 20 個のメッセージにスクロールします。
z +	プラス記号 (**+**) 引数を指定すると、次の 20 個のメッセージにスクロールします。 メッセージ 21 から 40 までのメッセージが (メールボックス内にその数のメッセージがあれば) 表示されます。 すべてのメッセージが表示されるまで **z+** サブコマンドを入力し続けます。 システムは次のメッセージで応答します。 `On last screenful of messages.`
z -	マイナス記号 (**-**) 引数を指定すると直前の 20 個のメッセージにスクロールします。 メッセージの最初のメッセージ・セットに達すると、 システムは次のメッセージを出して応答します。 `On first screenful of messages.`

メッセージの特定の情報のフィルター
メールボックスのプロンプトで、次の例のように **f** サブコマンドを使用して、必要な情報に応じてメッセージをフィルターできます。

項目	説明
f	現行メッセージのヘッダー情報を表示します。
f 1 4 7	特定のメッセージ 1、4、および 7 のヘッダー情報を表示します。
f 1-10	メッセージ 1 から 10 までの範囲内にあるヘッダー情報を表示します。
f *	すべてのメッセージを表示します。

f ron	ユーザー ron からのメッセージがある場合は、表示されます。 アドレスとして入力する文字は、検索するアドレスと完全に一致する必要はありません。 検索要求でアドレス ron が大文字または小文字のどちらで入力されても、以下のすべてのアドレスと一致します。
	`RoN`
	`ron@topdog`
	`hron`
	`rOn`

fmeet	**Subject:** フィールドに文字 meet が含まれているメッセージがある場合は、表示されます。 パターンとして入力された文字は、**Subject:** フィールドと完全に一致する必要はありません。 これらの文字は **Subject:** フィールド内に含まれていれば、大文字または小文字のどちらでもかまいません。 したがって、件名 meet に対する要求は次の件名のすべてと一致します。
	`Meeting on Thursday`
	`Come to meeting tomorrow`
	`MEET ME IN ST. LOUIS`

現行メッセージ番号
= サブコマンドは、メッセージ番号を表示します。

メールボックスのプロンプトから、次の例のように **=** サブコマンドを使うことができます。

項目	説明
=	現行のメッセージ番号が表示されます。

メールボックス内のメール総数
メールボックス内のメールの数をチェックするには、**folder** サブコマンドを使用します。

メールボックスのプロンプトから、次の例のように **folder** サブコマンドを使用できます。

項目	説明
folder	フォルダーまたはメールボックスに関する情報をリストします。 システムは次のようなメッセージを出して応答します。 `"/u/lance/mbox": 29 messages.`

メールの読みオプション

メールを読むにはいくつかの方法があります。 ここでは、それぞれの方法の例について説明します。

メールを読むための適切な方法を選択してください。 メールを読む前に、以下の条件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。
2. メール・プログラムが始動されていること。
3. システム・メールボックス内にメールがあること。

メールボックス内のメッセージの表示
メールボックス内のメッセージを読むには、**t** サブコマンドまたは **p** サブコマンドを使用します。

次の例に示すように、メールボックスのプロンプトから **t** サブコマンドまたは **p** サブコマンドを使用できます。

項目	説明
3	メッセージの番号を使用すると、デフォルトで、現行メッセージのテキストが表示されます。
t	**t** サブコマンドを使用すると、 デフォルトで、現行メッセージのテキストが表示されます。
t 3	メッセージ 3 のテキストが表示されます。
t 2 4 9	メッセージ 2、4、9 のテキストが表示されます。
t 2-4	メッセージ 2 から 4 の範囲にあるテキストが表示されます。
t	**p** サブコマンドを使用すると、 デフォルトで、現行メッセージのテキストが表示されます。
p 3	メッセージ 3 のテキストが表示されます。
p 2 4 9	メッセージ 2、4、9 のテキストが表示されます。
p 2-4	メッセージ 2 から 4 の範囲にあるテキストが表示されます。

メールボックス内の次のメッセージの表示
メールボックス内の次のメッセージを読むには、**n** サブコマンドを使用します。

メールボックスのプロンプトから、次の例のように (**n**)ext またはプラス記号 (+) サブコマンドを使用できます。

項目	説明
n または +	次のメッセージのテキストを表示し、このメッセージが現行メッセージになります。

Enter キーを押しても、 次のメッセージのテキストを表示できます。

メールボックス内の前のメッセージの表示
前のメッセージを読むには、**-** サブコマンドを使用します。

メールボックスのプロンプトから、次の例のように **-** サブコマンドを使うことができます。

項目	説明
-	直前のメッセージのテキストが表示されます。

メールの削除

メッセージを削除する際には、 現行のメッセージ、特定のメッセージまたはある範囲内のメッセージを削除できます。

また、複数のサブコマンドを組み合わせることによって現行メッセージを削除し、 次のメッセージを表示できます。 以下の条件を満たしていることを確認します。

1. システムにメール・プログラムがインストールされていること。
2. システム・メールボックス内にメールがあること。
3. メール・プログラムが始動されていること。

メッセージの削除
メッセージを削除するには、さまざまな形式の **d** サブコマンドを使用します。

メールボックスのプロンプトから、 次の例のように (**d**)elete サブコマンドを使用できます。

項目	説明
d	現行メッセージが削除されます。
dp または **dt**	現行メッセージが削除され、次のメッセージが表示されます。 これは、`.mailrc` ファイルに **`set autoprint`** オプションを含めることによっても行えます。 これにより、**d** サブコマンドが **dp** や **dt** のサブコマンド組み合わせのように機能するように設定されます。
d 4	特定のメッセージ 4 を削除します。
d 4-6	4 から 6 までのメッセージを削除します。
d 2 6 8	メッセージ 2、6、8 を削除します。

メッセージの削除取り消し
メッセージの削除を取り消すには、**u** サブコマンドを使用します。

次の例に示すように、メールボックスのプロンプトで **u** サブコマンドを使用できます。

項目	説明
u	現行メッセージの削除を取り消します。
u 4	特定のメッセージ 4 の削除を取り消します。
u 4-6	4 から 6 までのメッセージの削除を取り消します。
u 2 6 8	メッセージ 2、6、8 の削除を取り消します。

メールの終了

メール・プログラムを終了する前に、以下の要件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。
2. システム・メールボックス内にメールがあること。
3. メール・プログラムが始動されていること。

メールの終了および変更の保存
メールを終了し、変更を保存するには、**q** サブコマンドを使用します。

システム・メールボックスを終了する場合:

項目	説明
q	**q** サブコマンドはシステム・メールボックスを終了して、 オペレーティング・システムに戻ります。 メールボックスを終了すると、削除マークの付いたすべてのメッセージがメールボックスから除去され、 リカバリーすることはできません。 メール・プログラムは、読み取られたメッセージを個人用メールボックス (mbox) に保存します。 メールを読まなかった場合には、 そのメッセージは次回に処理されるまでシステム・メールボックスに残ったままになります。

個人用メールボックスまたはメール・フォルダーを終了する場合

項目	説明
q	個人用メールボックスまたはメール・フォルダー内で **q** サブコマンドを使用すると、読み取られたメッセージおよび読み取られなかったメッセージは次回に処理されるまで、 個人用メールボックスまたはメール・フォルダー内にそのまま残されます。

変更の保存を行わないメールの終了
メールボックスを変更しないでメールを終了するには、**x** サブコマンドまたは **ex** サブコマンドを使用します。

項目	説明
x または **ex**	**x** サブコマンドや **ex** サブコマンドを使用すると、 メールボックスを終了し、 メールボックスの元の内容を変更せずにオペレーティング・システムに戻ることができます。 プログラムは **x** 要求の前に出されたすべての要求を無視しますが、 メッセージを別のフォルダーに保存した場合には、保存処理が実行されます。

メールの編成

フォルダーを使って、系統的にメッセージを保存してください。

必要な数だけフォルダーを作成できます。 オフィスのファイル・フォルダーと同じように、 各フォルダーにメッセージの件名に関係のある名前を付けてください。 各フォルダーは、**set folder** オプションを使って .mailrc ファイルに指定されたディレクトリー内に入っているテキスト・ファイルです。 メッセージを格納するためのフォルダーを使う前に、このディレクトリーを作成しておく必要があります。 ディレクトリーを作成すると、メール・プログラムは必要に応じてそのディレクトリー内にフォルダーを作成します。 **set folder** オプションを使って .mailrc ファイルにディレクトリーを指定しない場合、現行ディレクトリー内にフォルダーが作成されます。 メール・プログラムを使用すると、システム・メールボックス、個人用メールボックス、または他のフォルダーからメッセージをフォルダーに 入れることができます。

s サブコマンドまたは **w** サブコマンドを使用して、メッセージの内容をファイルまたはフォルダーに追加できます。 この 2 つのサブコマンドはどちらも既存のファイルに情報を付加し、既存のファイルがない場合は新しいファイルを作成します。 これらの情報は、現在ファイル内にある情報を破棄することはありません。 システム・メールボックスからファイルまたはフォルダーにメッセージを 保存した場合、 そのメッセージはシステム・メールボックスから削除され、指定されたファイルまたはフォルダーに転送されます。個人用メールボックスまたはフォルダーから別のファイルやフォルダーにメッセージを 保存した場合、そのメッセージは個人用メールボックスから削除されず、指定されたファイルまたはフォルダーにコピーされます。 **s** サブコマンドを使用すると、メッセージとそのヘッダー情報がフォルダーの最後に付加されるので、フォルダーをメールボックスのように読み取ることができます。 **w** サブコマンドを使用すると、メッセージがファイルの最後にヘッダー情報を付けずに追加されるので、フォルダーをファイルのように読み取ることができます。

メールを編成する前に、以下の要件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。

2. システム・メールボックス、個人用メールボックス、または自分で定義したフォルダー内に、メールがあること。
3. メール・プログラムが始動されていること。

フォルダーにメッセージを格納する *letters* メールボックス・ディレクトリーの作成

メールボックスのディレクトリー・フォルダーにメッセージを 保存するには、**set folder** サブコマンドを使用します。

フォルダーにメッセージを 格納するには、以下の手順を使用します。

1. **set folder** オプションが .mailrc ファイル内で使用可能になっているかどうかを検査するには、メールボックスのプロンプトに次のサブコマンドを入力します。

   ```
   set
   ```

   set サブコマンドにより、.mailrc ファイル内で使用可能なメール・オプションのリストが表示されます。

   set folder オプションが使用可能な場合、システムは次のようなメッセージを出して応答します。

   ```
   folder /home/george/letters
   ```

   この例では、letters がメール・フォルダーを格納するディレクトリーです。
2. **set folder** オプションが使用可能でない場合には、.mailrc ファイルに次のような行を追加してください。

   ```
   set folder=/home/george/letters
   ```

   この例では、/home/george は George のホーム・ディレクトリーであり、letters はメール・フォルダーが保存されるディレクトリーです。 **set folder** オプションにより、メールボックス・プロンプトに省略表現であるプラス記号 (+) を使用して、letters ディレクトリーにメッセージを 保存できます。
3. ホーム・ディレクトリーに letters ディレクトリーを作成します。 ホーム・ディレクトリーでシステム・コマンド・ラインのプロンプトに、次のように入力します。

   ```
   mkdir letters
   ```

ヘッダー付きでメッセージを保存する

s サブコマンドは、ヘッダー付きのメッセージを保存します。

s サブコマンドは、次のように使用します。

項目	説明
s 1-4 notes	メッセージ 1、2、3、4 にヘッダー情報を付けて、現行ディレクトリー内の notes というフォルダーに保存します。 メール・プログラムは次のメッセージを出して応答します。 `"notes" [Appended] 62/1610`
s +admin	現行メッセージを、フォルダー・ディレクトリー内の admin という既存のフォルダーに保存します。 フォルダー・ディレクトリーが .mailrc ファイルに /home/george/letters と定義されている場合には、システムは次のメッセージを出して応答します。 `"/home/george/letters/admin" [Appended] 14/321`

項目	説明
s 6 +admin	メッセージ 6 を、フォルダー・ディレクトリー内の admin という既存のフォルダーに保存します。 フォルダー・ディレクトリーが .mailrc ファイルに /home/george/letters と定義されている場合には、システムは次のメッセージを出して応答します。 `"/home/george/letters/admin" [Appended] 14/321`

ヘッダーなしでメッセージを保存する
フォルダーとしてではなくファイルとしてメッセージを 保存するには、**w** サブコマンドを使用します。

w サブコマンドを使って保存されたファイルを読み取りまたは編集するには、**vi** などの他のテキスト・エディターを使用する必要があります。 メールボックスのプロンプトから、次の例のように **w** サブコマンドを使用できます。

項目	説明

w 6 pass	メッセージ 6 のテキストのみを、現行ディレクトリー内の pass というファイルに保存します。
	pass ファイルがまだ存在しない場合には、システムは次のメッセージを出して応答します。
	`"pass" [New file] 12/30`
	pass ファイルが存在する場合には、システムは次のメッセージを出して応答します。
	`"pass" [Appended] 12/30`

w 1-3 safety	特定のメッセージ 1、2、3 のテキストのみを、現行ディレクトリー内の safety というファイルに保存します。 この例でメッセージのテキストは、1 つのファイルに 1 つずつ付加されます。safety ファイルがまだ存在しない場合には、システムは次のメッセージを出して応答します。 `"safety" [New file] 12/30`

現行のメールボックスまたはフォルダーの判別
現行のメールボックスまたはフォルダーを判別するには、**folder** サブコマンドを使用します。

mail コマンドは始動時に現行メールボックス名を表示しますが、そのうちにどのメールボックスを処理中かが分からなくなることがあります。 メールボックスのプロンプトから、次の例のように **folder** サブコマンドを使用できます。

項目	説明
folder	現行のメールボックスまたはフォルダーの 名前を検索します。 現行のメールボックスが /home/lance/mbox の場合、次のように表示されます。 `/home/lance/mbox: 2 messages 1 deleted` このメッセージは、現在処理中の現行メールボックスが /home/lance/mbox であり、メッセージが 2 つあり、そのうちの 1 つがメールボックスの処理の終了時に削除されることを示しています。

別のメールボックスへの移動
別のメールボックスへの移動は、メールボックスまたはフォルダーの終了に似ています。

削除マークの付いたメッセージはすべて、メールボックスを終了するときに削除されます。 削除されたメッセージをリカバリーすることはできません。 メールボックスのプロンプトに、次の例のように **file** または **folder** サブコマンドを使用できます。

項目	説明

folder +project	あるメールボックスでメール・プログラムを開始したら、**file** または **folder** のサブコマンドを使用して別のメールボックスに移動します。
	
	mbox ファイルから mbox フォルダーに移動し、mbox ファイル内のすべてのメッセージを削除した場合には、メール・プログラムは次のメッセージを表示します。
	
	`/home/dee/mbox removed`
	`+project: 2 messages 2 new`
	
	その後、project フォルダー内のメッセージ・リストを表示します。

メールの作成と送信
mail プログラムを使用すれば、メッセージの作成、送信、返信、および他のユーザーへの転送を行ったり、ASCII ファイルを他のユーザーへ送信できます。

ASCII ファイルとしては、例えば、好みのエディターを使って書いた文書や、プログラムのソース・ファイルを指定できます。

メッセージとファイルはローカル・ネットワーク上のローカル・システムのユーザーまたは接続されている別のネットワーク上のユーザーに送信することもできます。 情報の送信時には、受信者はそのシステムにログオンする必要はありません。 メールはユーザーのアドレスに送信されます。

メールのアドレッシング
メールはユーザーのアドレスに送信されます。 ログイン名とシステム名を含むアドレスは、メールのメッセージの送達を指示します。

一般に、メッセージを他のユーザーに送信するには、次のように **mail** コマンドとアドレスを入力します。

```
mail User@Address
```

Address パラメーターの形式は、受信者のロケーションによって異なります。 この概念はメモをオフィスの同僚あてに送る方法に似ています。 6 人から 8 人の小さな部に所属する Ryan にメモを送るには、封筒の表にその名前を書いて、オフィスのメール・システムに入れます。 しかし、Ryan が別の部に所属する場合には、封筒の表にさらに次のように書き加えなければなりません。

```
Ryan
Payroll
```

Ryan が別の地理的位置にいる場合には、メッセージが確実に届けられるように、さらに次のような別の情報を付け加えなければなりません。

```
Ryan
Payroll
Gaithersburg
```

電子メールを送信するには、同じようなアドレッシング方法を使用します。

項目	説明
mail ryan	ローカル・システム上のユーザーにメールを送信するには、アドレスの情報として必要なものは、ログイン名のみです。
mail ryan@tybalt	メールをローカル・ネットワークのユーザーに送信するには、完全なシステム (ノード) アドレスを入力します。

項目	説明
mail ryan@mars.aus.dbm.com	メールを接続されている別のネットワークのユーザーに送信するには、完全なシステム・アドレスとネットワーク・アドレスを入力します。
mail dept71	別名または配布リストを使用して、特定のグループにメールを送信できます。 そのためには、.mailrc ファイル内に別名または配布リストを作成する必要があります。 別名の作成に関する詳細は、37 ページの『別名および配布リスト』を参照してください。

複数ユーザーへのメール・アドレッシング
メールを一度に複数のユーザーに送信するように宛先を指定するには、各ユーザー名をスペースで区切ります。

例:

```
ryan@tybalt suemc@julius dmorgan@ophelia
```

ローカル・システム上のユーザーへのメール・アドレッシング
ローカル・システムのユーザー (ログイン名が /etc/passwd ファイル内にリストされているユーザー) にメッセージを送信するためには、アドレスにログイン名を使用します。

システム・コマンド・ラインのプロンプトに、次の例のように **mail** コマンドを使用できます。

```
mail LoginName
```

項目	説明
mail ryan	Ryan がシステム上に存在し、ログイン名 ryan が与えられている場合、このコマンドはメール・プログラムを起動し、メッセージの作成を可能にし、メッセージをローカル・ログイン名 ryan に送信しようとします。 メッセージが正常に配布された場合には、何の通知も受信しません。 Ryan がシステム上に存在しない場合には、メール・システムは直ちにエラー・メッセージを戻し、未送信メッセージをシステム・メールボックスに戻します。

現在のネットワーク内のユーザーへのメール・アドレッシング
現在のネットワーク内のユーザーにメッセージを送信するには、**mail** コマンドを使用します。 アドレス内にユーザーのログイン名とシステム名を組み込みます。

ローカル・ネットワークを通じてメッセージを他のシステム上のユーザーに送信するには、コマンド・ラインに次のように入力します。

項目	説明
mail *LoginName*@*SystemName*	例えば、Ryan が zeus というシステム上に存在する場合には、次のコマンドを使ってメッセージを作成し、送信します。 `mail ryan@zeus` このコマンドはメール・プログラムを起動し、メッセージの作成を可能にし、zeus というシステム上のログイン名 ryan にメッセージを送信しようとします。 メッセージが正常に配布された場合には、何の通知もなくシステム・プロンプトを受信します。 メール・アドレスが正しくない場合は、エラー・メッセージを受信します。

注: ローカル・ネットワークを通じてメッセージを別のシステム上のユーザーに送信するには、ログイン名とそのシステム名を知っている必要があります。 ユーザーを識別する情報の表示に関する詳細については、6 ページの『一般的なネットワーク・コマンド』を参照してください。

別のネットワーク内のユーザーへのメール・アドレッシング
現在のネットワークが他のネットワークに接続されている場合、メールを他のネットワーク上のユーザーに送信できます。

アドレス・パラメーターは、現在のネットワークと他のネットワークがお互いに相手をどのようにアドレッシングするか、およびどのように接続されているかによって異なります。 ネットワーク構成の方法に応じて、以下のいずれかのアクションを実行してください。

- 名前とアドレスの中央データベースを使用している場合は、次の例のように **mail** コマンドを使用します。

  ```
  mail LoginName@SystemName
  ```

  ネットワークに名前の中央データベースが使用されている場合には、接続されているネットワーク上のユーザーにメールを送信するための追加情報は必要ありません。 ローカル・ネットワーク上のユーザーへのアドレッシングと同様のアドレッシング・フォーマットを使用します。

  この種のアドレッシングは、ネットワークの性質により名前の中央データベースを維持できる場合にのみ機能します。
- ネットワークでドメイン名アドレッシングを使用する場合は、次の例のように **mail** コマンドを使用します。

  ```
  mail LoginName@SystemName.DomainName
  ```

  広範囲の地域にまたがる、大規模で相互に関連のないネットワークの場合には、名前の中央データベースは使用できません。 *DomainName* パラメーターは、相互接続された複数のネットワークからなる大規模なグループについて定義された構造の中で、現在のローカル・ネットワークからの相対位置によりリモート・ネットワークを定義します。

  例えば、次のコマンドを入力するとします。

  ```
  mail kelly@merlin.odin.valryanl
  ```

  このメールは、valryanl と呼ばれるドメインを持つ第 2 のネットワークに接続される odin というローカル・ネットワーク上にあるシステム merlin のユーザー kelly に送信されます。

BNU または *UUCP* リンクによるメールのアドレッシング
基本ネットワーク・ユーティリティー (BNU) または UNIX 間コピー・プログラム (UUCP) リンクでメッセージを別のシステムのユーザーに送信することができます。

BNU または別バージョンの UUCP により、現在のシステムに接続された別のシステム上のユーザーにメッセージを送信するには、次に挙げる項目を理解している必要があります。

- ログイン名
- 他のシステムの名前
- 他のシステムへの物理経路指定

他のシステムにアドレッシングするための 経路指定情報は、そのシステムと 他のシステムとの接続に関する責任者から提供されます。

コンピューターに BNU または UUCP リンクがある場合: システム・コマンド・ラインのプロンプトで、次の例のように **mail** コマンドを使用します。

項目	説明
mail *UUCPRoute*!*LoginName*	ローカル・システムに、リモート・システムに到達するために使用できる BNU または UUCP 接続がある場合には、この例のフォーマットを使用してメッセージをアドレッシングします。 変数 *LoginName* は、メッセージ受信者のリモート・システム上のログイン名です。 変数 *UUCPRoute* は、UUCP ネットワーク上でメッセージが通らなければならない物理的な経路指定を記述します。 システムが中間 UUCP システムを介さずにリモート・システムに接続されている場合には、この変数はリモート・システム名となります。
mailarthur!lancelot!merlin!ken	メッセージがメール送信先のリモート・システムに到着するまでに 1 つ以上の中間 UUCP システムを通る必要がある場合には、この変数は個々の中間システムのリストになります。 このリストは、最寄りのシステムから始まり、感嘆符 (!) で区切りながら、最も遠いシステムまで記述します。 メッセージが merlin に到達するまでにシステム arthur と lancelot を (この順番で) 通る場合には、この例のようにしてください。
mail merlin!ken	ローカル・システムと merlin というシステムとの間に UUCP リンクがあり、ローカル・システムと merlin との間にそれ以外の UUCP システムが存在しない場合には、メッセージをそのシステム上の ken に送信できます。

BNU または UUCP リンクが他のコンピューター上にある場合: ローカル・エリア・ネットワークまたは広域ネットワークの環境で、ネットワーク上に、リモート・システムに対して BNU または他の種類の UUCP で接続されているシステムがある場合があります。 UUCP 接続を使用してリモート UUCP システム上のユーザーにメッセージを送信できます。 システム・コマンド・ラインのプロンプトで、次の例のように **mail** コマンドを使用します。

mail @arthur:merlin!ken
インターネット・システム arthur から UUCP システム merlin 上の ken にメールを送信します。 区切り文字 @ はインターネットのアドレッシングに、区切り文字 ! は UUCP のアドレッシングに使用し、区切り文字 : は 2 つのアドレスを連結するのに使用します。 このフォーマットでは、中間システム上にあるユーザーにはメールを送信しないので、ログイン名をドメイン・アドレスの @ の前に付けないことに注意してください。

mail @arthur:odin!acct.dept!kelly
インターネット・システム arthur からシステム odin を経由して、UUCP システム acct.dept 上の kelly にメールを送信します。

mail@odin.uucp:@dept1.UUCP:@dept2:bill@dept3
odin と dept1 の UUCP リンクを経由して、次にシステム dept2 と dept3 の間のローカル・ネットワーク・リンクを経由して、bill@dept3 にメールを送信します。 /etc/sendmail.cf ファイルは、この種類の UUCP アドレス表記が使用できるように適切に構成されている必要があります。 詳細はシステム管理者に相談してください。

頻繁に他のネットワークのユーザーにメールを送信する場合には、ユーザーのアドレスを含む別名を作成すると時間の節約になります。 37 ページの『別名および配布リスト』を参照してください。

メール・エディターの始動

mail プログラムには、メッセージを作成するための行単位のエディターが含まれています。

1. システムにメール・プログラムがインストールされていること。
2. メール・プログラムが始動されていること。

このエディターを使用すると、メッセージを行ごとに入力でき、Enter キーを押すと新しい行ができて、テキストをさらに入力できます。 Enter キーを押した後は、その行を変更できません。 しかし、Enter キーを押す前であれば、バックスペース・キーや削除キーを使用して削除することによってその行の情報を変更

できます。 また、メール・エディター・サブコマンドを使って、フルスクリーン・エディターに入ったり、メッセージを変更できます。

メール・エディターを使ってメールを作成するときに、システムは **date:** フィールドと **from:** フィールドを自動的に完成します。 **subject:** フィールドと **cc:** フィールドを完成させるオプションもあります。 これらのフィールドは標準ビジネス・レターの本文と似ています。

メール・エディターには多数の制御サブコマンドが含まれており、メッセージをさまざまに処理できます。これらの各サブコマンドは新しい行に入力し、先頭に特別なエスケープ 文字を付けます。 デフォルトでは、エスケープ文字はティルドです。 `.mailrc` ファイルに **set escape** オプションを記入することにより、 これを他の文字に変更できます。

システム・コマンド・ラインのプロンプトまたはメールボックスのプロンプトで、次の例のように **mail** コマンドを使用できます。

項目	説明
mail *User@Address*	コマンド・ラインのプロンプトからこのコマンドを出します。 メッセージは *User@Address* にアドレッシングされます。 *Address* パラメーターは受信者のロケーションによって異なります。
m *User@Address*	メールボックスのプロンプトからこのサブコマンドを出します。 メッセージは *User@Address* にアドレッシングされます。 *Address* パラメーターは受信者のロケーションによって異なります。

R サブコマンドまたは **r** サブコマンドを使用してメッセージに応答する場合は、メール・エディターも活動化されます。 メッセージの応答については、29 ページの『メールの送信』、および 30 ページの『メールへの応答』を参照してください。

メッセージの編集

メールボックスの使用中に、メールボックスのプロンプトから (**e**)dit または (**v**)isual サブコマンドを入力して、既存のメッセージに情報を追加できます。

メール・エディターの使用中に、Enter キーを押して次の行に移ると、前の行の情報を変更することはできません。 メッセージを別のエディターで編集することにより、送信する前にメッセージの内容を変更できます。

別のエディターでメッセージを 編集する前に、以下の条件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。
2. 代替エディターが、次のように `.mailrc` ファイルに定義されていること。

   ```
   set EDITOR=PathName
   ```

   これは、**~e** サブコマンドで活動化されるエディターを定義します。 *PathName* の値は使用したいエディター・プログラムの絶対パス名でなければなりません。 例えば、`set EDITOR=/usr/bin/vi` は、**~e** サブコマンドで使用する **vi** エディターを定義します。
3. メールボックス内のメッセージに情報を追加するためには、**mail** コマンドを始動してシステム・メールボックス、他のメールボックス、またはフォルダー内のメールを読み取ること。
4. メッセージの作成中に代替エディターを始動するには、メール・エディターのプロンプトで行うこと。

メールボックス内の特定のメッセージへの情報の追加

メールボックス内のメッセージに情報を追加するには、メッセージ番号の前に **e** サブコマンドまたは **v** サブコマンドを入力します。

メールボックスのプロンプトで、次の例に示すように **e** サブコマンドまたは **v** サブコマンドのいずれかを使用できます。

項目	説明
e 13	**e** エディター (または .mailrc ファイルに定義されたエディター) を使用して、情報をメッセージ 13 に追加します。
v 15	**vi** エディター (または .mailrc ファイルに定義されたエディター) を使用して、 情報をメッセージ 15 に追加します。

メッセージ番号を指定しない場合には、**mail** コマンドは現行メッセージに使用したエディターを活動化します。 エディターを終了すると、メールボックスのプロンプトに戻り、 メールボックス内のメッセージ処理を継続できます。

メール・エディターでの 現行メッセージの変更
次の例に示すように、メール・エディターで行の先頭に **~e** サブコマンドまたは **~v** サブコマンドのいずれかを使用できます。

項目	説明
~e	**e** エディターまたは .mailrc ファイルに定義した他のエディターを活動化します。
~v	**vi** エディターまたは .mailrc ファイルに定義した他のエディターを活動化します。

これにより、現行メッセージのテキストを編集できます。 別のエディターを終了するとメール・エディターに戻ります。

メール・エディターでのメッセージ行の表示
メール・エディターでメッセージ行を表示するには、**~p** サブコマンドを使用します。

1. システムにメール・プログラムがインストールされていること。
2. メール・エディターを使用中にメッセージを表示するためには、メール・エディターが始動されていること。 詳しくは、23 ページの『メール・エディターの始動』を参照してください。

次の例に示すように、メール・エディターで行の先頭に **~p** サブコマンドを使用します。

項目	説明
~p	エディターはメッセージのヘッダー情報を含むメッセージの内容を表示します。 テキストは画面の最下行からスクロールアップして表示されます。 メッセージの終わりの次には、メール・エディターの (Continue) プロンプトが続きます。

メッセージが 1 画面より大きく、 その端末のページ・サイズが **stty** コマンドによって設定されていない場合、テキストはスクロールされるので、 上の方は見えなくなります。 大きなメッセージの内容を見るには、 メール・エディター・サブコマンドを使用して、 他のエディターでメッセージを 表示します。 詳しくは、24 ページの『メッセージの編集』を参照してください。

メール・エディターの終了

メッセージを送信せずにメール・エディターを終了するには、**~q** サブコマンドまたは割り込みキー・シーケンス (通常は Alt-Pause キーまたは Ctrl-C キー・シーケンス) を使用します。

1. システムにメール・プログラムがインストールされていること。
2. メール・エディターを使用中にメッセージを表示するためには、メール・エディターが始動されていること。 詳しくは、23 ページの『メール・エディターの始動』を参照してください。

テキストを既に入力している場合には、**mail** コマンドによりメッセージが dead.letter ファイルに保存されます。

次の例に示すように、メール・エディターで、行の先頭に **~q** サブコマンドを使用できます。

項目	説明
~q	メール・エディターを終了しますが、メッセージは送信されません。 それ以上にテキストを入力していない場合、メッセージはホーム・ディレクトリーの dead.letter ファイルに保存されます。 システム・プロンプトが表示されます。

Ctrl-C	割り込みキー・シーケンスを使ってエディターを終了するには、break (Ctrl-C キー・シーケンス) または割り込み (Alt-Pause キー・シーケンス) を押します。 次のメッセージが表示されます。
	`(Interrupt -- one more to kill letter)`
	再度 break または interrupt を押します。
	`(Last Interrupt -- letter saved in dead.letter)`
	メッセージは送信されません。 それ以上にテキストを入力していない場合、メッセージはホーム・ディレクトリーの dead.letter ファイルに保存されます。 システム・プロンプトが表示されます。

注: メッセージを送信せずにメール・エディターを終了すると、 直前の dead.letter ファイルの内容は不完全なメッセージに置き換わります。 ファイルを検索するには、26 ページの『メッセージ内でのファイルおよび特定のメッセージの追加オプション』を参照してください。

メッセージ内でのファイルおよび特定のメッセージの追加オプション

メール・メッセージでファイルおよび特定のメッセージを追加するには、事前にいくつかの要件を満たしておく必要があります。

前提条件

1. システムにメール・プログラムがインストールされていること。
2. メールの受信者の名前とアドレスを知っていること。
3. メール・エディターが始動されていること。

メッセージへのファイルの組み込み

メッセージにファイルを追加するには、**~r** サブコマンドを使用します。

次の例に示すように、メール・エディターで、行の先頭に **~r** サブコマンドを使用できます。

項目	説明
~r schedule	schedule は組み込むファイル名です。 この例では、schedule ファイル内の情報が書き込み中のメッセージの現行の末尾に組み込まれます。

メッセージ内への特定のメッセージの組み込み

メッセージに特定のメッセージを組み込むには、**~f** サブコマンドまたは **~m** サブコマンドを使用します。

次の例に示すように、メール・エディターで、新しい行の先頭に **~f** サブコマンドまたは **~m** サブコマンドのいずれかを使用できます。

項目	説明
~f *MessageList*	指定された 1 つ以上のメッセージを現行メッセージの末尾に付加します。 ただし、付加したメッセージを字下げしません。 また、**~m** サブコマンドで対応できないほど広いマージンをもつ参照用のメッセージを付加するのに使用します。 **注:** パラメーター *MessageList* は、メールで処理されるメールボックスまたはフォルダー内の有効なメッセージ番号を参照する整数のリストです。 また、次の例のように単に番号の範囲を入力することもできます。 以下に例を示します。 **~f 1-4** メッセージ 1、2、3、4 を書き込み中のメッセージの最後に付加します。 これらのメッセージは左マージンに位置合わせされます (字下げなし)。
~m 2	指定されたメッセージを現行メッセージの末尾に追加します。 組み込まれたメッセージはメッセージの 通常の左マージンから 1 タブ文字分だけ字下げされます。 この例では、メッセージ 2 は現行メッセージに付加されます。
~m 1 3	メッセージ 1、次にメッセージ 3 を書き込み中のメッセージの末尾に追加し、 左マージンから 1 タブ文字分だけ字下げします。

現行メッセージに *dead.letter* ファイルの内容を追加する
メッセージに `dead.letter` の内容を追加するには、**~d** サブコマンドを使用します。

次の例に示すように、メール・エディターで、新しい行の先頭に **~d** サブコマンドを使用できます。

項目	説明
~d	`dead.letter` ファイルの内容を取り出して、現行メッセージの末尾に付加します。 (Continue) プロンプトで、メッセージに追加するかまたはメッセージを送信することにより処理を継続します。

ヘッダー情報の編集
メッセージのヘッダーには、経路指定情報と件名の短いステートメントが含まれています。 メッセージの受信者は少なくとも 1 人指定する必要があります。

1. システムにメール・プログラムがインストールされていること。
2. メール・エディターを始動して、メッセージの編集を開始すること。 詳しくは、メール・エディターの始動を参照してください。

その他のヘッダー情報は必須ではありません。 ヘッダー情報には、次の情報を含めることができます。

項目	説明
To:	メッセージを送信するための 1 つ以上のアドレスが含まれます。
Subject:	メッセージのトピックの要約が入ります。
Cc:	メッセージのコピーを送信するための 1 つ以上のアドレスが入ります。 このフィールドの内容はメッセージの一部であり、 メッセージを受信するすべての人に送られます。
Bcc:	メッセージのブラインド ・コピーを送信するための 1 つ以上のアドレスが入ります。 このフィールドは、メッセージを受信するすべての人へのメッセージの一部分には含められません。

メール・プログラムをカスタマイズし、`.mailrc` ファイル内にエントリーを入力して、これらのフィールド内の情報を自動的に要求できます。 詳しくは、34 ページの『メール・プログラムのカスタマイズ・オプション』を参照してください。

Subject: フィールドの設定または再設定
~s サブコマンドを使って **Subject:** フィールドに特定の句または文を設定します。

このサブコマンドを使用すると、**Subject:** フィールドの直前の内容が (ある場合は) 置き換えられます。 次の例に示すように、メール・エディターで、新しい行の先頭に **~s** サブコマンドを使用できます。

項目	説明

~s Fishing Trip	これは現行の **Subject:** フィールド
	`Subject: Vacation`
	を、次のように変更します。
	`Subject: Fishing Trip`

注: このサブコマンドを使って **Subject:** フィールドに付加することはできません。 27 ページの『ヘッダー情報の編集』に記述されているように、**~h** サブコマンドを使用します。

To:、*Cc:*、および *Bcc:* フィールドにユーザーを追加する
ヘッダー・フィールドにユーザーを追加するには、**~t**、**~c**、または **~b** の各サブコマンドを使用します。

次の例に示すように、メール・エディターで新しい行の先頭に **~t**、**~c**、または **~b** サブコマンドを使用できます。

項目	説明

~t geo@austin mel@gtwn	これは現行の **To:** フィールド
	`To: mark@austin`
	を、次のように変更します。
	`To: mark@austin geo@austin mel@gtwn`

~c geo@austin mel@gtwn	これは現行の **Cc:** フィールド
	`Cc: mark@austin amy`
	を、次のように変更します。
	`Cc: mark@austin amy geo@austin mel@gtwn`

~b geo@austin mel@gtwn	これは現行の **Bcc:** フィールド
	`Bcc: mark@austin`
	を、次のように変更します。
	`Bcc: mark@austin geo@austin mel@gtwn`

注: **~t**、**~c**、または **~b** サブコマンドを使用して、**To:**、**Cc:**、および **Bcc:** フィールドの内容を変更したり削除したりすることはできません。 27 ページの『ヘッダー情報の編集』に記述されているように、**~h** サブコマンドを使用します。

メール・エディターでのメッセージの再フォーマット
メッセージを入力したなら、それを送信する前に、**fmt** シェル・プログラムを使用してメッセージを再フォーマットし、体裁を整えることができます。

メッセージの再フォーマット設定を行う前に、以下の条件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。

2. システムに **fmt** コマンドがインストールされていること。

次の例に示すように、メール・エディターで、新しい行の先頭に **fmt** コマンドを使用できます。

項目	説明
~| fmt	定義されているマージン内で、各パラグラフごとに情報を組み込み直してメッセージの外観を変更します (各パラグラフはブランク行で区切ります)。 パイプ (**|**) サブコマンドはメッセージをコマンドの標準入力にパイプ接続し、そのメッセージをそのコマンドの標準出力で置き換えます。

> **注意:** メッセージに外部ファイルからの組み込みメッセージまたは事前にフォーマットされた情報が含まれている場合には、**fmt** コマンドは使用しないでください。 **fmt** コマンドはヘッダー情報を組み込みメッセージに再フォーマットし、事前にフォーマットされた情報のフォーマットを変更する場合があります。 **~e** または **~v** サブコマンドを代わりに使用して、フルスクリーン・エディターを始動し、メッセージを再フォーマットしてください。

メール・エディターでのミススペルの 検査

spell コマンドは、メッセージ内のスペルを検査します。

メッセージのミススペルを検査する前に、以下の条件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。
2. システムにテキスト・フォーマット・プログラムがインストールされていること。

spell コマンドは、メール・エディターでメッセージにミススペルがないかどうかを 検査する場合に使用します。

1. メッセージを一時ファイルに書き込みます。 例えば、checkit ファイルにメッセージを書き込むには、次のように入力します。

   ```
   ~w checkit
   ```

2. 一時ファイルを入力として使用して **spell** コマンドを実行します。 次のように入力します。

   ```
   ~! spell checkit
   ```

   この例では、感嘆符 (!) はシェルを始動してコマンドを実行し、メールボックスに戻るサブコマンドです。 **spell** コマンドの応答として、登録されているワード・リストにない語のリストが表示され、これに続けて感嘆符 (!) が表示されてメール・プログラムに戻ったことが示されます。
3. 単語のリストを検査します。 エディターを使って訂正する必要があるかどうか判別します。
4. 次のように入力して、一時ファイルを削除します。

   ```
   ~! rm checkit
   ```

メールの送信

作成したメッセージを送信するには、以下の手順を使用します。

- システムにメール・プログラムがインストールされていること。
- メールの受信者の名前とアドレスを知っていること。

1. 次のようにコマンド・ラインに **mail** コマンドを入力し、次に 1 人以上のメッセージの受信者の名前とアドレスを入力します。 以下に例を示します。

   ```
   >mail jan@brown
   ```

   システムは次のメッセージを表示して応答します。

   ```
   Subject:
   ```

2. メッセージの件名を入力します。 例えば、次のとおりです。

```
Subject: Dept Meeting
```

Enter キーを押します。 これでテキストの本文が入力できるようになります。

3. メッセージを入力します。 例えば、次のとおりです。

```
There will be a short department meeting this afternoon
in my office. Please plan on attending.
```

4. メール・エディターを使って入力したメッセージを送信するには、メッセージ内の新しい行の先頭でテキスト終結文字を押します。テキスト終結文字は、通常 Ctrl-D キー・シーケンスまたはピリオド (.) です。

   Cc: フィールドか表示されます。

```
Cc:
```

5. 次のようにメッセージのコピーを受信することになる各ユーザーの名前とアドレスを入力します。 以下に例を示します。

```
Cc: karen@hobo cliff@cross
```

   注: コピーを送る必要のない場合には、 入力せずに Enter キーを押します。

Enter キーを押すと、 メッセージは指定されたアドレスに送信されます。

注: システムに認識されていないアドレス、または別名または配布リスト内に定義されていないアドレスを入力した場合、システムはログイン 名の後に `[user ID]... User unknown` というエラー・メッセージを表示して応答します。

メールへの応答

メールボックスのプロンプトで、次の例に示すように **r** サブコマンドと **R** サブコマンドを使用して、メールに返信できます。

1. システムにメール・プログラムがインストールされていること。
2. システム・メールボックス内にメールがあること。

項目	説明
r	選択したメッセージの送信者を宛先とする新しいメッセージを作成します。元のメッセージの **Cc:** フィールドに指定されているユーザーにコピーが送られます。 新しいメッセージの **Subject:** フィールドは選択されたメッセージを参照するものになります。 **r** サブコマンドのデフォルト値は現行メッセージです。 このデフォルト値は **r** の後にメッセージ番号を入力して上書きできます。
R	メッセージの送信者だけへの応答を開始します。 **R** サブコマンドのデフォルト 値は現行メッセージです。

R 4	メッセージの送信者だけへの応答を開始します。 デフォルト現行メッセージは **R** の後にメッセージ番号を入力することにより上書きできます。 この例ではメッセージ 4 への応答が開始されます。
	`To: karen@thor`
	`Subject: Re: Department Meeting`
	応答は次のように入力できます。
	`I'll be there.`
	テキストの入力を終えたら、ピリオド (.) または Ctrl-D キー・シーケンスを押して、メッセージを送信します。 応答が送信されると、メールボックス・プロンプトに戻ります。

メールボックス内の新しいメッセージの作成

次の例に示すように、メールボックスのプロンプトで **m** サブコマンドを使用して、新しいメッセージを作成できます。

項目	説明
m *Address*	*Address* パラメーターには、正しいユーザー・アドレスであれば何でも指定できます。 このサブコマンドによりメール・エディターが始動し、メールボックスの使用中に新規メッセージを作成できるようになります。 メッセージを送信すると、メールボックスのプロンプトに戻ります。

メールの転送

メールの読み取り中に、 特定のメモを別のユーザーに転送したい場合があります。

1. システムにメール・プログラムがインストールされていること。
2. 選択したメッセージを転送する場合には、**mail** コマンドでメール機能を始動すること。 転送したいメール・メッセージの番号をメモすること。

この作業は、**~f** および **~m** サブコマンドを使用することにより実行できます。

通常のネットワーク・アドレスから離れる場合は、`.forward` ファイルを作成して、別のネットワーク・アドレスに自分のメールを送信させることができます。 32 ページの『.forward ファイル』を参照してください。 新しいアドレスは、ネットワーク上またはネットワークに接続された他のネットワーク上の有効なメール・アドレスであれば何でも指定できます。 留守中にメッセージを取り扱う同僚のアドレスを指定することもできます。 ネットワーク・メールを転送する場合には、 自分のメールボックス内に着信メールのコピーを受信することはありません。 すべてのメールは自分が指定した 1 つ以上のアドレスに直接転送されます。

メールボックス内からの選択したメッセージの転送

メールボックス内の特定のメール・メッセージを転送するには、以下の手順を使用します。

特定のメール・メッセージを転送するには、

1. **m** サブコマンドを使って新しいメッセージを作成し、受信者を指定するには、メールボックスのプロンプトから次のように入力します。

   ```
   m User@Host
   ```

   User は別のユーザーのログイン名、*Host* はそのユーザーのシステム名です。 指定したユーザーが自分のシステム上に存在する場合は、アドレスの `@Host` の部分を省略できます。
2. **Subject:** のプロンプトから、件名を入力します。
3. 転送したいメール・メッセージの番号を指定するには、次のように入力します。

   ```
   ~f MessageNumber
   ```

   または

   ```
   ~m MessageNumber
   ```

   MessageNumber は転送したいメールを識別します。

   mail コマンドにより次のようなメッセージが表示されます。

   ```
   Interpolating: 1
   (continue)
   ```
4. メールを終了するには、ブランク行にピリオド (.) を 1 つ入力します。 **Cc:** のプロンプトから、 他にメール・メッセージを転送したいユーザーの名前を入力します。

全メールの転送

すべてのメールを別のユーザーに転送するには、以下の手順を使用します。

全メールを別のユーザーに転送するには、

1. パラメーターを付けずに **cd** コマンドを入力し、ホーム・ディレクトリーにいることを確認します。
   例えば、ログイン名 mary: の場合、次のように入力します。

   ```
   cd
   pwd
   ```

   システムは次のメッセージを表示して応答します。

   ```
   /home/mary
   ```

2. ホーム・ディレクトリー内に .forward ファイルを作成します。
   32 ページの『.forward ファイル』を参照してください。

   注: .forward ファイルを削除するまで、転送元のユーザーはどのようなメールも受信しません。

.forward ファイル
.forward ファイルには、転送したネットワーク・メールの受信者となる 1 つ以上のネットワーク・アドレスが含まれています。

アドレス・フォーマットは *User*@*Host* になります。 *User* は他のユーザーのログイン名を指し、*Host* はそのユーザーのシステム名を指します。 指定したユーザーが自分のシステム上に存在する場合は、アドレスの @*Host* の部分を省略できます。 .forward ファイルを作成するには、 次のように **cat** コマンドを使用できます。

```
cat > .forward
mark
joe@saturn
[END OF FILE]
```

[END OF FILE] は EOF 文字を表し、 ほとんどの端末上では Ctrl-D キー・シーケンスです。 この文字はブランク行に入力する必要があります。

.forward ファイルにはメールを送信したいユーザーのアドレスが入っています。 メールはローカル・システム上の mark と saturn システム上の joe に転送されます。

このファイルには有効なアドレスが入っている必要があります。 このファイルが null ファイル (長さゼロ) の場合には、メールは転送されず、 転送元のユーザーのメールボックス内に格納されます。

注: .forward ファイルを削除するまで、転送元のユーザーはどのようなメールも受信しません。

メール転送の取り消し
メールの転送を停止するには、次のように .forward ファイルを削除します。

次のように **rm** コマンドを使用して、ホーム・ディレクトリーから .forward ファイルを除去します。

```
rm .forward
```

不在メッセージ通知の送信
不在メッセージ通知を用意して送信するには、以下の手順を使用します。

システムにメール・プログラムがインストールされていること。

1. $HOME (ログイン) ディレクトリーで次のように入力して、不在メッセージを初期化します。

   ```
   vacation -I
   ```

   これにより、 メッセージの送信者名を保持する .vacation.dir ファイルと .vacation.pag ファイルが作成されます。
2. .forward ファイルを変更します。
   例えば、carl により、 次のステートメントが .forward ファイルに入力されます。

```
carl, |"/usr/bin/vacation carl"
```

最初の carl のエントリーは、メールを転送するユーザー名です。 2 番目の carl のエントリーは、不在メッセージの送信者のユーザー名です。 carl に対してどれだけのメッセージを送ろうと、メール・メッセージの送信者が carl から受け取る不在メッセージは週に 1 回となります。 別のユーザーに自分のメールの転送を依頼する場合、送信者からのメール・メッセージは、.forward ファイルに定義されたユーザーに送信されます。

-f フラグを使って、メッセージを送信する頻度を変更します。 例えば、carl により、次のステートメントが .forward ファイルに入力されます。

```
carl, |"/usr/bin/vacation -f10d carl"
```

carl に対してどれだけのメッセージを送ろうと、メール・メッセージの送信者が carl から受け取る不在メッセージは 10 日ごとになります。

3. 受信したメールの各送信者に個別にメッセージを送信するには、$HOME/.vacation.msg ファイルを作成して、そのファイルに自分のメッセージを追加します。
次に不在メッセージの例を示します。

```
From: carl@odin.austin (Carl Jones)
Subject: I am on vacation.
I am on vacation until October 1. If you have something urgent,
please contact Jim Terry <terry@zeus.valhalla>.
  --carl
```

送信者は $HOME/.vacation.msg ファイル内にあるメッセージを受信するか、そのファイルがない場合は、/usr/share/lib/vacation.def ファイル内にあるデフォルト・メッセージを受信します。 どちらのファイルもない場合は、メール・メッセージの送信者に対する自動的な応答は実行されず、エラー・メッセージも生成されません。

不在のメッセージを取り消すには、次のように .forward ファイル、.vacation.dir ファイル、.vacation.pag ファイル、および .vacation.msg を $HOME (ログイン) ディレクトリーから除去します。

```
rm .forward .vacation.dir .vacation.pag .vacation.msg
```

機密メールの送受信

機密メールを送信するには、システム・コマンド・ラインのプロンプトから、次の例に示すように **xsend** コマンドを使用します。

1. システムにメール・プログラムがインストールされていること。
2. **enroll** コマンドを使ってパスワードを設定しておくこと。

項目	説明
xsend barbara	この例では、機密メールはログイン名 barbara にアドレッシングされています。 Enter キーを押すと、単一行のライン・エディターを使って、メッセージのテキストを入力できます。 メッセージの入力を終えたら、Ctrl-D のキー・シーケンスまたは .(ピリオド) を押してメール・エディターを終了し、メッセージを送信します。 **xsend** コマンドにより、メッセージが送信前に暗号化されます。

1. 機密メールを受信するには、システム・コマンド・ラインのプロンプトから、次のように入力します。

```
mail
```

システムはシステム・メールボックス内のメッセージのリストを表示します。 機密メール・プログラムにより、機密メールを受信したという通知が送信されます。 メッセージ行は次のようになります。

```
Mail [5.2 UCB] Type ? for help.
"/usr/spool/mail/linda": 4 messages 4 new
>N 1 robert Wed Apr 14 15:23 4/182 "secret mail from robert@Zeus"
```

メッセージのテキストは、**xget** コマンドを使ってホスト上の機密メールを読むように指示します。

2. システム・コマンド・ラインのプロンプトから、次のように入力します。

```
xget
```

enroll コマンドを使って以前に設定したパスワードを求めるプロンプトが表示されます。 パスワードを入力すると、**xget** コマンドのプロンプトが表示され、その後に機密メールがリストされます。 機密メールを表示するには、メール・プログラムを使用します。 既読メッセージと未読メッセージを機密メールボックス内に残しておいて、**xget** コマンドでメッセージが削除されないようにするには、**q** サブコマンドを入力する必要があります。

メール・ヘルプ情報

?、**man**、または **info** コマンドを使えば、メール・プログラムの使用法に関するヘルプ情報を表示できます。

項目	説明
メールボックス内でヘルプを表示するには、	メールボックスのプロンプトから ? または help を入力します。 **?** および **help** サブコマンドにより、一般的なメールボックス・サブコマンドの要約が表示されます。 **(l)**ist サブコマンドを入力すると、全メールボックス・サブコマンドの (要約情報なしの) リストが表示されます。
メール・エディター内でヘルプを表示するには、	メール・エディターのプロンプトから ~? と入力します。 **~?** サブコマンドにより、一般的なメール・エディターのサブコマンドの要約が表示されます。
機密メールでヘルプを表示するには、	メール・エディターのプロンプトから ? と入力します。 **?** サブコマンドにより、一般的な機密メールのサブコマンドの要約が表示されます。
マニュアル・ページを使用してヘルプを表示するには、	システム・コマンド・ラインのプロンプトで man mail と入力します。 この例では、mail は検索対象のコマンド名です。 システムは **mail** コマンドに関して ASCII フォーマットの資料を提供します。 継続マーカー (:) が表示されているときに Enter キーを押すと、資料の残りの部分が表示されます。 **man** コマンドを使用すると、コマンド、サブルーチン、およびファイルに関する参照トピックの情報が ASCII 形式で提供されます。

メール・プログラムのカスタマイズ・オプション

.mailrc ファイルと /usr/share/lib/Mail.rc ファイル内のコマンドとオプションは、個人的なメールの要件に合わせてカスタマイズすることができます。

メール・オプションについては、35 ページの『メールの使用可能化および使用不可化オプション』を参照してください。

個別設定できるメール・セッションの特性は次のとおりです。

- **メッセージの件名の入力を求めるプロンプト。** **mail** コマンドを入力すると、プログラムは **Subject:** フィールドを完成するように要求します。 このプロンプトが表示されると、メッセージの件名の要約を入力できます。 受信者が読むとき、この要約はメッセージの冒頭に表示されます。 37 ページの『Subject: および Carbon Copy (Cc:) のフィールド・プロンプト』を参照してください。
- **メッセージのコピーを受信するユーザー名の入力を求めるプロンプト。** メッセージの送信時に、メール・プログラムがメッセージのコピーを受信するユーザー名の入力を求めるプロンプトが表示されるように、.mailrc ファイルをカスタマイズすることができます。 37 ページの『Subject: および Carbon Copy (Cc:) のフィールド・プロンプト』を参照してください。
- **別名または配布リスト。** 大規模なネットワーク上でメールを送信する場合、または多数のユーザーに同一のメッセージを頻繁に送信する場合、受信者ごとに長いアドレスを入力するのは面倒になります。 この処理を簡略化するには、.mailrc ファイル内に別名または配布リストを作成します。 別名 は、各ユーザーのアドレスの代わりに使用できるように定義された名前です。 配布リスト は、ユーザーのアドレスのグループの代わりに使用できるように定義された名前です。 37 ページの『別名および配布リスト』を参照してください。
- **メッセージの読み取り時に表示される行数。** 画面上でスクロールされるメッセージ・ヘッダーまたはメッセージ・テキストの行数を変更できます。 38 ページの『メール・プログラムに表示されるメッセージ・ヘッダーまたはメッセージ・テキストの行数の変更』を参照してください。
- **メッセージ内にリストされる情報。** コンピューターで設定された message-id フィールドなどのメッセージ・ヘッダーをオフにすることができます。 39 ページの『メッセージ内の情報の表示』を参照してください。
- **メッセージを格納するフォルダー・ディレクトリー。** メッセージを格納するための特別なディレクトリーを作成できます。 簡略コマンドであるプラス記号 (+) サブコマンドを使用して、メッセージの格納時またはフォルダーの参照時にそのディレクトリーを指定できます。 41 ページの『メッセージを格納するデフォルト・フォルダーの作成』を参照してください。
- **発信メッセージを記録するログ・ファイル。** ホーム・ディレクトリー内のファイルまたはサブディレクトリーに発信メッセージをすべて記録するように、**mail** プログラムに指示できます。 41 ページの『メッセージを格納するデフォルト・フォルダーの作成』を参照してください。
- **メッセージを入力するためのエディター。** メール・エディターの他に、2 つの異なるエディターを選択して、メッセージを編集できます。 42 ページの『メッセージ入力用テキスト・エディター』を参照してください。

メール・プログラムのカスタマイズに関する詳細は、次のトピックを参照してください。

メールの使用可能化および使用不可化オプション

オプションは 2 値オプションまたは値指定オプションのいずれかです。

2 値オプションは **set** (設定) または **unset** (未設定) のいずれかであり、値指定オプションは特定の値に**設定**できます。

> **注: unset** オプション は、**set no** オプション に相当します。

/usr/share/lib/Mail.rc ファイルを表示するには、**pg** コマンドを使用します。 /usr/share/lib/Mail.rc ファイルには、mail プログラムの構成が定義されています。 メール・プログラムのシステム構成を変更するには、$HOME/.mailrc ファイルを変更します。 **mail** コマンドを実行すると、**.mailrc** ファイル内のサブコマンドによって、/usr/share/lib/Mail.rc ファイル内の同種のサブコマンドが上書きされます。 .mailrc オプションはカスタマイズすることができ、mail プログラムを使用するたびに有効になります。

ファイルに格納されたメール・コマンドを実行するには、**source** サブコマンドを使用します。

前提条件

システムにメール・プログラムがインストールされていること。

メール・オプションの使用可能化
以下のメールボックス・サブコマンドは、メール・セッションの特性を変更する上で最も一般的に使用されます。

項目	説明
set	メール・オプションを使用可能にします。
source	ファイルに格納されたメール・オプションを使用可能にします。 メールの読み取り時、メールボックスのプロンプトから、このサブコマンドを出すことができます。 `source PathName` *PathName* はパスおよびメール・コマンドが入っているファイルです。 このファイル内のコマンドによって、 以前に設定された同種のコマンドが現行セッション中にすべて上書きされます。 また、メールボックス・プロンプトからコマンドを入力して、 現在のメール・セッションの特性を変更することもできます。

メールボックスの使用中、または .mailrc ファイルにエントリーを作成することによって、これらのオプションを設定できます。

使用可能メール・オプションの表示
メールを読む際に、使用可能な .mailrc オプションをすべてリストするには、引数を付けずに **set** サブコマンドを入力します。

このリストでは、フォルダー・ディレクトリーが選択されているかどうか、また発信メッセージを記録するログ・ファイルが設定されているかどうかも調べることができます。

メールボックスのプロンプトから、次のように入力します。

```
set
```

次のようなメッセージが表示されます。

```
ask
metoo
toplines 10
```

この例では、2 つの 2 値オプション **ask** と **metoo** が使用可能になっています。 このリストには、**askcc** エントリーはありません。 これは **askcc** オプションが使用可能になっていないことを示します。 **toplines** オプションには値 10 が代入されています。 **ask**、**metoo**、**askcc**、および **toplines** オプションについては、ファイル参照の『.mailrc File Format』セクションを参照してください。

メール・オプションの使用不可化
以下のメールボックス・サブコマンドは、メール・セッションの特性を変更する上で最も一般的に使用されます。

項目	説明
unset	メール・オプションを使用不可にします。
unalias	指定された別名を削除します。
ignore	メッセージのヘッダー・フィールドを抑制します。

メールボックスの使用中、または .mailrc ファイルにエントリーを作成することによって、これらのオプションを設定できます。

注: unset オプション は、**set no** オプション に相当します。

Subject: および *Carbon Copy (Cc:)* のフィールド・プロンプト

Subject: と **Cc:** のフィールド・プロンプトを編集する場合は、以下の前提条件を満たしている必要があります。

前提条件

システムにメール・プログラムがインストールされていること。

Subject フィールドのプロンプトを使用可能または使用不可にする

Subject: フィールドを使用可能または使用不可にするには、**set** コマンドと **unset** コマンドを使用します。

Subject: フィールドを使用可能または使用不可にするには、次のようにします。

項目	説明
set ask	.mailrc ファイルの ask オプションを編集することによって、**Subject:** フィールドのプロンプトが使用可能になります。
unset ask	.mailrc ファイルの ask オプションを編集することによって、**Subject:** フィールドのプロンプトが使用不可になります。

Cc: フィールドのプロンプトの使用可能化または使用不可化

Cc: フィールドを使用可能または使用不可にするには、**set** コマンドと **unset** コマンドを使用します。

Cc: フィールドを使用可能または使用不可にするには、次のようにします。

項目	説明
set askcc	.mailrc ファイルの **askcc** オプションを編集することによって、Carbon copy (**Cc:**) フィールドのプロンプトが使用可能になります。
unset askcc	.mailrc ファイルの **askcc** オプションを編集することによって、Carbon copy (**Cc:**) フィールドのプロンプトが使用不可になります。

別名および配布リスト

別名と配布リストを作成すると、通常使用する受信人とアドレスを管理しやすくなります。

別名または配布リストを作成する前に、以下の条件を満たしていることを確認してください。

1. システムにメール・プログラムがインストールされていること。
2. 別名または配布リストに組み込みたいユーザーの名前とアドレスを知っていること。

別名または配布リストを作成するには、次のようにします。

項目	説明
alias	kath kathleen@gtwn この例では、アドレス gtwn に存在するユーザー kathleen の別名 kath がリストされています。 この行を $HOME/.mailrc ファイルに追加した後、Kathleen にメッセージを送信するには、コマンド・ラインのプロンプトから次のように入力します。 `mail kath` これで、kath 別名を使って、Kathleen にメールを送信することができるようになります。

項目	説明
alias	dept dee@merlin anne@anchor jerry@zeus bill carl この行を $HOME/.mailrc ファイルに追加した後、自分の部門にメッセージを送信するには、コマンド・ラインのプロンプトから次のように入力します。 `mail dept` 作成し送信したメッセージは、システム merlin 上の dee、システム anchor 上の anne、システム zeus 上の jerry およびローカル・システム上の bill と carl に送信されます。

別名と配布リストを表示するには、メールボックスのプロンプトから次のように入力します。

alias

または

```
a
```

別名および配布リストの一覧が表示されます。

メール・プログラムに表示されるメッセージ・ヘッダーまたはメッセージ・テキストの行数の変更

.mailrc ファイルを変更することにより、メールボックス・リスト内または実際のメッセージ内でスクロールする機能をカスタマイズすることができます。

これらの変更を行うためには、システムにメール・プログラムがインストールされている必要があります。

メッセージ・リストの表示行数の変更

メールボックス内の各メッセージには、メッセージ・リスト内に 1 行の見出しがあります。 メッセージが 25 以上ある場合は、メッセージ・リストの最初の見出しはスクロールして画面の最上端から外に出てしまいます。 一回に表示されるリストの行数は、**set screen** オプションによって設定します。

1 回の表示メッセージ行数を変更するには、**$HOME/.mailrc** ファイルに次のように入力します。

```
set screen=20
```

この例では、システムは一度に 20 個のメッセージ・ヘッダーを表示します。 他のグループのヘッダーを表示するには、**h** または **z** サブコマンドを使用できます。 また、メールボックスのプロンプトからもこのサブコマンドを入力できます。

長いメッセージの表示行数の変更

25 行以上あるメッセージを表示する場合、メッセージの最初の数行はスクロールして画面の最上端から外に出てしまいます。 .mailrc ファイルに **set crt** オプションが含まれている場合には、メール内から **pg** コマンドを使用することによって、長いメッセージを表示できます。

pg コマンドを始動させる前に、**set crt** オプションにより、1 つのメッセージの行数を制御できます。

例えば、**t** サブコマンドを使って長いメッセージを読み取る場合は、1 画面 (つまり 1 ページ) しか表示されません。 ページの後にはコロンのプロンプトが続いて表示され、ページの続きがあることを知らせます。 Enter キーを押してメッセージの次のページを表示してください。 メッセージの最後のページが表示されると、次のようなプロンプトが表示されます。

```
EOF:
```

このプロンプトから、有効な **pg** サブコマンドをすべて入力できます。 前のページを表示したり、メッセージから特定の文字列を検索したり、またはメッセージの読み取りを終了してメールボックスのプロンプトに戻ることができます。

.mailrc ファイルに **set crt** オプションを次のように入力します。

set crt=*Lines*

例:

```
set crt=20
```

は、**pg** コマンドを始動する前に、メッセージが 20 行になることを指定します。 21 行以上あるメッセージを読み取るときは、**pg** コマンドが始動します。

メッセージの先頭の表示行数の変更
top サブコマンドにより、メッセージ全体を読み取らなくてもメッセージをスキャンできます。

次のように **toplines** オプションを設定して、メッセージの表示行数を制御します。

```
set toplines=Lines
```

このサブコマンドでは、変数 *Lines* は **top** サブコマンドで表示される、すべてのヘッダー・フィールドを含む最上行からの行数を表します。

例えば、ユーザー Amy の .mailrc ファイル内に次の行が含まれる場合、

```
set toplines=10
```

Amy が **mail** コマンドを実行して新しいメッセージを読むと、次のメッセージが表示されます。

```
Mail Type ? for help.
"/usr/mail/amy": 2 messages 2 new>
N 1 george  Wed Jan 6 9:47  11/257 "Dept Meeting"
N 2 mark    Wed Jan 6 12:59 17/445 "Project Planner"
```

Amy が **top** サブコマンドを使用してメッセージ内をブラウズすると、メッセージの一部が次のように表示されます。

```
top 1
Message 1:
From george Wed Jan 6 9:47 CST 1988
Received: by zeus
    id AA00549; Wed, 6 Jan 88 9:47:46 CST
Date: Wed, 6 Jan 88 9:47:46 CST
From: george@zeus
Message-Id: <8709111757.AA00178>
To: amy@zeus
Subject: Dept Meeting
Please plan to attend the department meeting on Friday
at 1:30 in the planning conference room. We will be
```

このメッセージは **toplines** が 10 に設定されているので部分的に表示されます。 第 1 行 (**Received:** フィールド) から第 10 行 (メッセージ本文の第 2 行) までのみ表示されます。 第 1 行の From george Wed Jan 6 9:47 CST 1988 は必ず存在しますが、**toplines** オプションではカウントされません。

メッセージ内の情報の表示

.mailrc ファイルを変更すると、メッセージ内に表示されるヘッダー情報を制御できます。

ヘッダー情報の中には既にオフになっているものがあるかもしれません。 無視されたヘッダー・フィールドについては、/usr/share/lib/Mail.rc ファイルを調べてください。

前提条件

システムにメール・プログラムがインストールされていること。

Date、*From*、および *To* ヘッダーが表示されないようにする
各メッセージには最上行にいくつかのヘッダー・フィールドがあります。 これらのヘッダー・フィールドはメッセージを読み取るときに表示されます。 メッセージを読み取るとき、**ignore** サブコマンドを使用してヘッダー・フィールドの表示を制御できます。

ignore サブコマンドのフォーマットは次のとおりです。

```
ignore [FieldList]
```

FieldList は、メッセージを表示するときに無視したい 1 つ以上のフィールド名で構成できます。 例えば、ユーザー Amy が自分の .mailrc ファイルに次の行を含めると、

```
ignore date from to
```

/usr/share/lib/Mail.rc ファイルには次の行が入ります。

```
ignore received message-id
```

t サブコマンドを使用した結果は次のようになります。

```
t 1
Message 1:
From george Wed Jan 6 9:47 CST 1988
Subject: Dept Meeting
Please plan to attend the department meeting on Friday
at 1:30 in the planning conference room. We will be
discussing the new procedures for using the project
planning program developed by our department.
```

Received:、**Date:**、**From:**、**Message-Id:**、および **To:** の各フィールドは表示されません。 これらのフィールドを表示するためには、**T** または **P** サブコマンド、または **top** サブコマンドを使用します。

> **注:** この例では **From** 行が表示されています。 これは **ignore** サブコマンドの *FieldList* 内にリストされた **From:** フィールドと同じではありません。

無視されているヘッダー・フィールドのリスト表示
無視されているヘッダー・フィールドをリストするには、**ignore** サブコマンドを使用します。

現在無視されているヘッダー・フィールドのリストを表示するには、メールボックスのプロンプトから、次のように入力します。

```
ignore
```

現在無視されているすべてのヘッダーのリストが表示されます。 以下に例を示します。

```
mail-from
message-id
return-path
```

ヘッダー・フィールドのリセット
ヘッダー・フィールドを再設定するには、**retain** サブコマンドを使用します。

例:

```
retain date
```

保持されているヘッダー・フィールドのリスト表示
保持されているヘッダー・フィールドをリストするには、**retain** サブコマンドを使用します。

現在保持されているヘッダー・フィールドを確認するには、ヘッダー・フィールド・パラメーターを付けないで **retain** サブコマンドを入力します。

バナーが表示されないようにする

メールのバナーは、**mail** コマンドを出したときにメール・プログラム名を示すメッセージのリストの最上行です。

メール・バナーとは次のようなものです。

```
Mail [5.2 UCB] [Workstation 3.1] Type ? for help.
```

メール・プログラムの始動時にバナーを表示させないようにするには、$HOME/.mailrc ファイルに次の行を追加します。

```
set quiet
```

mail バナーの表示を制御するオプションには、次のものもあります。

```
set noheader
```

このオプションを .mailrc ファイル内で使用すると、メールボックス内のメッセージのリストは表示されません。 **mail** プログラムを始動するときに応答として表示されるのは、メールボックスのプロンプトだけです。 (**h**)eader サブコマンドを入力すると、メッセージのリストが表示されます。

delete コマンドと *print* コマンドの結合

delete サブコマンドと print サブコマンドを組み合わせるには、autoprint オプションを使用します。

メッセージを読んだ後、**d** サブコマンドを使ってそれを削除できます。 **p** サブコマンドを使うと、次のメッセージを表示できます。 .mailrc ファイルに次の行を入力することにより、これらのサブコマンドを結合します。

```
set autoprint
```

.mailrc ファイル内で **set autoprint** オプションを使うと、**d** サブコマンドにより現行メッセージが削除され、次のメッセージが表示されます。

メッセージを格納するデフォルト・フォルダーの作成

デフォルト・フォルダーを使用して、メッセージを保管することができます。

システムにメール・プログラムがインストールされていること。

letters メールボックス・ディレクトリーを作成して、フォルダーにメッセージを 保管するには、以下の手順を使用します。

1. .mailrc ファイル内で **set folder** オプションが使用可能になっているかどうかを検査するには、メールボックスのプロンプトから次のように入力します。

   ```
   set
   ```

   set folder オプションが使用可能である場合、システムは次のメッセージを表示して応答します。

   ```
   folder /home/george/letters
   ```

   この例では、letters がメール・フォルダーを格納するディレクトリーです。
2. **set folder** オプションが使用可能でない場合は、.mailrc ファイルの中に、次のような **set folder** エントリーを作成します。

   ```
   set folder=/home/george/letters
   ```

   この例では、/home/george は George のホーム・ディレクトリーであり、letters はメール・フォルダーが保存されるディレクトリーです。 **set folder** オプションにより、簡略表記であるプラス記号 (+) を使用して、letters ディレクトリーにメッセージを 保存できます。

3. letters ディレクトリーが存在しない場合は、ホーム・ディレクトリー内に letters ディレクトリーを作成してください。 ホーム・ディレクトリー内で、システム・コマンド・ラインから次のように入力します。

```
mkdir letters
```

他のユーザーに送信したメッセージの記録を保持するには、以下の手順を使用します。

1. .mailrc ファイルに次のように入力します。

```
set record=letters/mailout
```

2. letters ディレクトリーが存在しない場合は、ホーム・ディレクトリー内に letters ディレクトリーを作成してください。 ホーム・ディレクトリー内で、システム・コマンド・ラインから次のように入力します。

```
mkdir letters
```

3. 他のユーザーに送信したメッセージのコピーを読み取るには、次のように入力します。

```
mail -f +mailout
```

この例では、mailout ファイルに他のユーザーに送信したメッセージのコピーが入れられます。

メッセージ入力用テキスト・エディター

メッセージを入力するテキスト・エディターを定義するには、**set EDITOR=***PathName* オプションを使用します。

システムにメール・プログラムがインストールされていること。

項目	説明
set EDITOR= *PathName*	.mailrc ファイル内のこのオプションにより、**~e** キー・シーケンスで活動化されるエディターが定義されます。 *PathName* の値は使用したいエディター・プログラムの絶対パス名でなければなりません。 メール・プログラムの使用中に e エディターに変更するには、次のように入力します。 `~e` このシーケンスにより、e エディターまたは .mailrc ファイルの中で定義されている他のエディターが活動化されます。 このエディターを使って、メール・メッセージを編集します。
set VISUAL=*PathName*	.mailrc ファイル内のこのオプションにより、**~v** キー・シーケンスで活動化されるエディターが定義されます。 *PathName* の値は使用したいエディター・プログラムの絶対パス名でなければなりません。 デフォルト値は /usr/bin/vi です。 メール・プログラムの使用中に vi エディターに変更するには、次のように入力します。 `~v` このシーケンスにより、vi エディターまたは .mailrc ファイルの中で定義されている他のエディターが活動化されます。 このエディターを使って、メール・メッセージを編集します。

mail コマンドのサブコマンド

mail コマンドは、さまざまな機能を実行する複数のサブコマンドを使用します。

このトピックは、**mail** コマンドとそのサブコマンドの参照用に使用します。

メールを実行するコマンド
メールを実行するには、以下のシステム・コマンドを使用します。

項目	説明
mail	システム・メールボックスを表示します。
mail -f	個人用メールボックス (**mbox**) を表示します。
mail -f +*folder*	メール・フォルダーを表示します。
mail*user@address*	指定されたユーザーにメッセージをアドレッシングします。

メール・プログラムのメールボックス・サブコマンド
メール・プログラムは、メールボックスを処理するとき、 それ自身が入力を待っていることを示すメールボックスのプロンプトを表示します。

メールボックスのプロンプトは新しい行の先頭に表示されるアンパーサンド (&) です。 このプロンプトから、メールボックスのサブコマンドを入力できます。

メール・プログラムの制御サブコマンド
メール・プログラムを制御するには、以下のサブコマンドを使用します。

項目	説明
q	終了し、このセッションに入力されたメールボックス・サブコマンドを受け付けます。
x	終了し、メールボックスを元の状態に復元します。
!	シェルを始動してコマンドを実行し、メールボックスに戻ります。
cd *dir*	ディレクトリーを `dir` または `$HOME` に変更します。

メール・プログラムの表示サブコマンド
メール・プログラムの表示を制御するには、以下のサブコマンドを使用します。

項目	説明
t	*msg_list* 内のメッセージまたは現行メッセージを表示します。
n	次のメッセージを表示します。
f *msg_list*	*msg_list* 内のメッセージの見出しを表示するか、*msg_list* が与えられていない場合には、現行メッセージの見出しを表示します。
h *num*	メッセージ *num* を含むグループの見出しを表示します。
top *num*	メッセージの一部を表示します。
set	使用可能になっている **.mailrc** オプションのリストをすべて表示します。
ignore	無視されたヘッダー・フィールドのリストをすべて表示します。
folder	現在のフォルダー内のメッセージ数をフォルダーのパス名と共に表示します。

メッセージの処理
メッセージを編集、削除、再呼び出し、追加、または保存するには、以下のサブコマンドを使用します。

項目	説明
e *num*	メッセージ *num* を編集します (デフォルトのエディターは e)。
d *msg_list*	*msg_list* 内のメッセージまたは現行メッセージを削除します。
u *msg_list*	*msg_list* 内の削除されたメッセージを再び呼び出します。
s *msg_list* **+***file*	メッセージ (見出し付き) を *file* に付加します。
w *msg_list* **+***file*	メッセージ (テキスト部分のみ) を *file* に付加します。

項目	説明
pre *msg_list*	メッセージをシステム・メールボックス内に保持します。

新しいメールの作成サブコマンド
新しいメール・メッセージを作成するには、以下のサブコマンドを使用します。

項目	説明
m *addrlist*	新しいメッセージを作成し、*addrlist* 内のアドレスに送信します。
r *msg_list*	応答をメッセージの送信者と受信者に送ります。
R *msg_list*	応答をメッセージの送信者のみに送ります。
a	別名とそのアドレスのリストを表示します。

メール・エディターのサブコマンド

メール・エディターは、それが処理されるときに、それ自身が入力を待っていることを示すメール・エディターのプロンプトを表示します。

このプロンプトから、メール・エディターのサブコマンドを入力できます。

メール・エディターの制御サブコマンド
メール・エディターを制御するには、以下のサブコマンドを使用します。

項目	説明
~q	現行メッセージを保存または送信せずに、エディターを終了します。
~p	メッセージ・バッファーの内容を表示します。
~: *mcmd*	メールボックス・サブコマンド (*mcmd*) を実行します。
EOT	メッセージを送信します (多くの端末では Ctrl-D)。
.	現行メッセージを送信します。

見出しへの追加サブコマンド
メッセージに見出し項目を追加するには、以下のサブコマンドを使用します。

項目	説明
~h	**To:**、**Subject**:、**Cc:**、および **Bcc:** のフィールドに追加します。
~t *addrlist*	*addrlist* 内のユーザー・アドレスを **To:** フィールドに追加します。
~s *subject*	**Subject:** フィールドに、*subject* によって指定される文字列に設定します。
~c *addrlist*	*addrlist* 内のユーザー・アドレスを、**Cc:** フィールドに追加します。
~b *addrlist*	*addrlist* 内のユーザー・アドレスを、`Bcc:` フィールドに追加します。

メッセージへの追加サブコマンド
メッセージに内容を追加するには、以下のサブコマンドを使用します。

項目	説明
~d	`dead.letter` の内容をメッセージに付加します。
~r *filename*	*filename* の内容をメッセージに付加します。
~f *numlist*	メッセージ番号 *numlist* の内容を付加します。
~m*numlist*	メッセージ番号 *numlist* の内容を付加して字下げします。

メッセージの変更サブコマンド
メッセージを編集するには、以下のサブコマンドを使用します。

項目	説明
~e	e エディターを使ってメッセージを変更します (デフォルトは e)。
~v	vi エディターを使ってメッセージを変更します (デフォルトは vi)。
~w *filename*	メッセージを *filename* に書き込みます。
~! *command*	シェルを始動して *command* を実行し、エディターに戻ります。
~| *command*	メッセージを *command* の標準入力にパイプ接続し、そのメッセージをコマンドからの標準出力で置き換えます。

機密メールのサブコマンド
機密メール・プログラムは、 機密メールボックスを処理するときに、それ自身が入力を待っていることを示す機密メールボックスのプロンプトを表示します。

機密メールボックスのプロンプトは新しい行の先頭に表示される疑問符 (?) です。 このプロンプトから、機密メールボックス・サブコマンドを入力できます。

機密メールのサブコマンド
機密メールを送信するには、以下のサブコマンドを使用します。

項目	説明
xsend barbara	指定されたユーザーにメッセージをアドレッシングします。
xget	機密メールボックスを表示します。

メールボックスのタスク
以下のサブコマンドは、さまざまなメールボックスのタスクを実行します。

項目	説明
q	未読メッセージを残して終了します。
n	現行メッセージを削除し、次のメッセージを表示します。
d	現行メッセージを削除し、次のメッセージを表示します。
Return キー	現行メッセージを削除し、次のメッセージを表示します。
!	シェル・コマンドを実行します。
s	指定されたファイルまたは mbox にメッセージを保存します。
w	指定されたファイルまたは mbox にメッセージを保存します。

メール管理タスク
メール・マネージャーは、以下のタスクを管理します。

1. /etc/rc.tcpip ファイルを構成し、**sendmail** デーモンがシステムのブート時に始動するようにします。 46 ページの『システム・ブート時に sendmail デーモンを始動』を参照してください。
2. 構成ファイル /etc/mail/sendmail.cf をカスタマイズします。 デフォルトの /etc/mail/sendmail.cf ファイルは、ローカル・メールと TCP/IP メールの両方を送達できるように構成されています。 BNU を経由してメールを送達するには、/etc/mail/sendmail.cf ファイルをカスタマイズする必要があります。 詳しくは、ファイル参照の sendmail.cf ファイルを参照してください。
3. システム・ワイドとドメイン・ワイドのメール別名を /etc/mail/aliases ファイルの中に定義します。 詳しくは、46 ページの『メールの別名』を参照してください。
4. メール・キューを管理します。 詳しくは、48 ページの『メール・キュー』を参照してください。
5. メール・ログを管理します。 詳しくは、53 ページの『メールのロギング』を参照してください。

システム・ブート時に sendmail デーモンを始動

システム・ブート時にも **sendmail** デーモンが始動するように /etc/rc.tcpip ファイルを構成するには、以下の手順を使用します。

1. 任意のテキスト・エディターを使用して /etc/rc.tcpip ファイルを編集します。
2. システム・ブート時に sendmail MTA デーモンを自動的に始動するには、start /usr/lib/sendmail で始まる行を見つけます。

   デフォルトでこの行はコメント行になっていません。 つまり、行頭に # (ハッシュ記号) が付いていません。 ただし、この行がコメント行になっている場合は、# (ハッシュ記号) を削除してください。
3. システム・ブート時に sendmail クライアント・キュー・ランナーを開始する場合は、次の行を /etc/rc.tcpip ファイルに追加して **sendmail** コマンドをメール配信プログラム (MSP) として開始します。

   ```
   /usr/lib/sendmail -L sm-msp-queue -Ac -q 30m
   ```
4. /etc/rc.tcpip ファイルを保存します。

メールの別名

別名は、パーソナル、システム・ワイド、ドメイン・ワイドのそれぞれの別名ファイルを使用して、名前をアドレス・リストへマップします。

次の 3 つのタイプの別名が定義できます。

項目	説明
personal	個々のユーザーが自身の $HOME/.mailrc ファイル内に定義します。
local system	メール・システム管理者が /etc/mail/aliases ファイル内に定義します。 これらの別名は、ローカル・システム上で **sendmail** プログラムが処理するメールに適用されます。 ローカル・システムの別名は、ほとんど変更する必要はありません。
domainwide	デフォルトでは、**sendmail** が /etc/alias を読み取り、別名を解決します。 デフォルトを変更して NIS を使用するには、/etc/netsvc.conf を編集または作成して、次の行を追加します。 `aliases=nis`

/etc/mail/aliases ファイル

ここでは、/etc/mail/aliases ファイルの属性、内容、およびロケーションについて説明します。

/etc/mail/aliases ファイルは、次のフォーマットの一連のエントリーで構成されます。

```
Alias: Name1, Name2, ... NameX
```

alias には、ユーザーが選択する任意の英数字文字列を使用できます (ただし、@ または ! などの特殊文字は使用できません)。 *Name1* から *NameX* は、1 つ以上の受信側の名前です。 この名前のリストは、複数の行にわたってもかまいません。 各継続行は、スペースまたはタブで始まります。 ブランク行および # (シャープ) で始まる行はコメント行です。

/etc/mail/aliases ファイルには、次の 3 つの別名を収めておく必要があります。

項目	説明
MAILER-DAEMON	メール・プログラム・デーモン宛てのメッセージを受信するユーザーの ID です。 この名前は、次のように、当初は root ユーザーに割り当てられます。 `MAILER-DAEMON: root`

項目	説明
postmaster	ローカル・メール・システムの操作に責任を負うユーザーの ID です。 **postmaster** という別名は、ネットワーク内にある 1 つのシステムごとに 1 つの有効なメールボックス・アドレスを定義します。 このアドレスを使用すると、あるシステムにいるユーザーの正しいアドレスを知らなくても、そのシステムの **postmaster** 別名に照会を送信できます。 この名前は、次のように、当初は root ユーザーに割り当てられます。 `postmaster: root`
nobody	**news** および **msgs** などのプログラム宛てのメッセージを受信する ID です。 この名前は、次のように、当初は /dev/null: に割り当てられます。 `nobody: /dev/null` これらのメッセージを受信するには、この別名を有効なユーザーとして定義します。

このファイルを変更した場合は、必ず、再コンパイルして、**sendmail** コマンドが使用できるデータベース・フォーマットにする必要があります。 48 ページの『別名データベースの構築』を参照してください。

メール用ローカル別名の作成

ローカル・メール別名を作成すると、メールの送信先のグループまたは配布リストを作成できます。

このシナリオでは、geo@medussa、mark@zeus、ctw@athena、および dsf@plato が testers メール別名に追加されます。 testers 別名の作成後に、glenda@hera には別名の所有権が付与されます。

testers 別名が /etc/mail/aliases ファイルに追加された後で、別名データベースが **sendmail** コマンドを使用して再コンパイルされます。 データベースの再コンパイル後に、E メールを testers 別名に送信できるようになります。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

以下の手順を使用して、ローカル・メール別名を作成します。

1. 任意のテキスト・エディターを使用して /etc/mail/aliases ファイルを開きます。
2. ブランク行で、別名とそれに続くコロンおよびコンマで区切られた受信側のリストを追加します。 例えば、次のエントリーは testers 別名を定義します。

   ```
   testers: geo@medussa, mark@zeus, ctw@athena, dsf@plato
   ```

3. 別名の所有者を作成します。 メールを別名に送信する際の **sendmail** コマンドが失敗した場合は、所有者にエラー・メッセージが送信されます。

   所有者を指定するために /etc/mail/aliases ファイルに行を追加します。 この行の形式は owner-*groupname*: *owner* です。ここで、*groupname* は別名の名前、*owner* は所有者の電子メール・アドレスです。 この例では、glenda@hera に testers 別名の所有者が作成されています。

   ```
   testers: geo@medussa, mark@zeus, ctw@athena, dsf@plato owner-testers: glenda@hera
   ```

4. 別名の作成後に、別名データベースを再コンパイルするために **sendmail -bi** コマンドを実行します。 /etc/mail/aliases ファイルを更新するたびに、このコマンドを実行する必要があります。

これで、E メールを testers 別名に送信することができます。

別名データベースの構築

sendmail コマンドは、ローカル・システムの /etc/mail/aliases ファイルに収められている別名定義を直接使用することはありません。 その代わりに、**sendmail** コマンドは、処理されたデータベース管理プログラム (dbm) バージョンの /etc/mail/aliases ファイルを読み取ります。

この別名データベースをコンパイルするには、次のいずれかのコマンドを実行します。

- **-bi** フラグを使用して、/usr/sbin/sendmail コマンドを実行します。
- **newaliases** コマンドを実行します。 このコマンドにより、**sendmail** コマンドはローカル・システムの /etc/mail/aliases ファイルを読み取って、別名データベース情報が収められた新規ファイルを作成します。 このファイルは、次のようなより効率的な Berkeley フォーマットを使用します。

  /etc/mail/aliases.db
- **別名の再構築**フラグを使用して、**sendmail** コマンドを実行します。 このフラグは、別名データベースが古くなった場合、自動的に別名データベースを再構築します。 自動再構築は、大きな別名ファイルがある負荷の大きいマシンでは危険な場合があります。 データベースの再構築に再構築タイムアウト (通常は 5 分) を超える時間がかかるような場合は、複数のプロセスが同時に再構築プロセスを開始する恐れがあります。

注 :

1. これらのファイルが存在しない場合、**sendmail** コマンドはメールを処理できず、エラー・メッセージを生成します。
2. 複数の別名データベースが指定されている場合、**-bi** フラグは、認識できるすべてのタイプのデータベースを再構築します (例えば、NDBM データベースは再構築できますが、NIS データベースは再構築できません)。

/etc/netsvc.conf ファイルには、システム・サービスの順序付けが含まれます。 別名のサービス順序を指定するには、次の行を追加します。

```
aliases=service, service
```

この場合、service には files と nis のどちらかを指定できます。 例えば、次のとおりです。

```
aliases=files, nis
```

この場合、**sendmail** コマンドは最初にローカル別名ファイルを使用し、それが失敗したら nis を使用します。 nis はサービスとして定義されている場合には、実行中でなければなりません。

/etc/netsvc.conf ファイルについて詳しくは、ファイル参照 を参照してください。

メール・キュー

メール・キューとは、**sendmail** コマンドが送達するメール・メッセージについて、データの保管とファイルの制御を行うディレクトリーのことです。 デフォルトでは、メール・キューは /var/spool/mqueue です。

メール・メッセージは、いくつかの理由からキューに入れられます。

例:

1. **sendmail** コマンドは、キューをすぐに処理するのではなく、一定のインターバルで処理するように構成されている場合があります。 その場合、メール・メッセージを一時的に保管しておく必要があります。
2. リモート・ホストがメール接続の要求に応答しない場合、メール・システムは、あとで再試行するために、メッセージをキューに入れます。

メール・キューの印刷
キューの内容は、**mailq** コマンド (または **-bp** フラグを指定した **sendmail** コマンド) を使用して印刷できます。

これらのコマンドにより、キュー ID、メッセージのサイズ、そのメッセージがキューに入った日付、送信側、受信側を示したリストが生成されます。

メール・キュー・ファイル
キュー内にある個々のメッセージには、いくつかのファイルが関連付けられています。

それらのファイルには、次の規則に従った名前が付いています。

```
TypefID
```

ID は固有なメッセージ・キュー *ID* で、*Type* はファイルのタイプを示す次のいずれかの文字です。

項目	説明
d	見出し情報を除く、メッセージ本体が入っているデータ・ファイル。
q	キュー制御ファイル。 このファイルには、ジョブの処理に必要な情報が入っています。
t	一時ファイル。 このファイルは q ファイルの再構築中のイメージです。 このファイルは、直ちに q ファイルに名前変更されます。
x	セッションが継続している間だけ存在するトランスクリプト・ファイルで、セッション中に発生したすべてのことを示します。

例えば、メッセージのキュー ID が AA00269 である場合、**sendmail** コマンドがメッセージを送達しようとしている間に、次のファイルがメール・キュー・ディレクトリー内に作成され、削除されます。

項目	説明
dfAA00269	データ・ファイル
qfAA00269	制御ファイル
tfAA00269	一時ファイル
xfAA00269	トランスクリプト・ファイル

q 制御ファイル
q 制御ファイルには、コード文字で始まる一連の行が含まれています。

項目	説明
B	body type を指定します。 この行の残りの部分は、body type を定義するテキスト・ストリングです。 このフィールド全体が存在していない場合、body type はデフォルトで 7 ビットで、特別な処理は行われません。 有効な値は、**7BIT** と **8BITMIME** です。
C	制御ユーザーを含みます。 ファイルまたはプログラムである宛先アドレスの場合、**sendmail** はそのファイルまたはプログラムのオーナーとして送達を行います。 制御ユーザーは、ファイルまたはプログラムのオーナーに設定されます。 **.forward** または **:include:** ファイルから読み取られる受信側アドレスも、そのファイルのオーナーに設定された制御ユーザーを持ちます。 **sendmail** がこれらの宛先にメールを送達する場合、制御ユーザーとして送達してからルートへと戻ります。
F	エンベロープ・フラグが含まれています。 フラグは、**w** (**EF_WARNING** フラグを設定)、**r** (**EF_RESPONSE** フラグを設定)、**8** (**EF_HAS8BIT** フラグを設定)、および **b** (**EF_DELETE_BCC** フラグを設定) の任意の組み合わせです。 その他の文字は無視されます。
H	見出し定義が入っています。 この行は、ファイル内に何行あっても構いません。 **H** 行が出現する順序は、最終的なメッセージでのそれらの行の順序を決定します。 これらの行には、 /etc/mail/sendmail.cf 構成ファイル内の見出し定義と同じ構文が使用されます。

項目	説明
I	df ファイルの i ノードおよびデバイス情報を指定します。これは、ディスク・クラッシュのとき、メール・キューをリカバリーするために使用できます。
K	最後に送達を試みた時刻 (秒数) を指定します。
M	送達を試みた際にエラーが発生したためにメッセージがキューに入れられた場合、エラーの特性が **M** 行に保管されます。
N	送達を試みた合計回数を指定します。
O	ESMTP からのオリジナルの MTS 値を指定します。 Delivery Status Notification (DSN) にだけ使用されます。
P	現行メッセージの優先順位が入っています。 この優先順位は、キューの順序を決めるために使用されます。 数値が大きくなるほど、優先順位は低くなります。 メッセージがキュー内に待機している間に、優先順位は高くなります。 最初の優先順位は、メッセージ・クラスとメッセージのサイズによって決まります。
Q	ESMTP トランザクションで ORCPT= フィールドによって指定されたオリジナルの受信側を含みます。 Delivery Status Notification にのみ使用されます。 直後の **R** 行のみに適用されます。
R	受信側アドレスが入っています。 この行は、受信側ごとに 1 行ずつ存在します。
S	送信側アドレスが入っています。 この行は、ファイル内に 1 行だけ存在します。
T	メッセージのタイムアウト時刻を計算するために使用するメッセージ作成時刻が入っています。
V	旧バージョンで作成したキュー・ファイルを新しい **sendmail** バイナリーで読み取れるようにするために使用するキュー・ファイルの形式のバージョン番号を指定します。 デフォルトのバージョンは**ゼロ**です。 これを使用する場合は、ファイルの先頭の行にしなければなりません。
Z	オリジナルのエンベロープ ID (ESMTP トランザクションからの) を指定します。 Delivery Status Notification にのみ 使用されます。
$	マクロ定義が入っています。 特定のマクロの値 (**$r** および **$s**) は、そのままキュー実行フェーズへ渡されます。

amy@zeus へ送信されたメッセージの q ファイルは、次のとおりです。

```
P217031
T566755281
MDeferred: Connection timed out during user open with zeus
Sgeo
Ramy@zeus
H?P?return-path: <geo>
Hreceived: by george (0.13 (NL support)/0.01)
        id AA00269; Thu, 17 Dec 87 10:01:21 CST
H?D?date: Thu, 17 Dec 87 10:01:21 CST
H?F?From: geo
Hmessage-id: <8712171601.AA00269@george>
HTo: amy@zeus
Hsubject: test
```

この場合、

項目	説明
P217031	メッセージの優先順位
T566755281	秒単位で示した発信時刻
MDeferred: Connection timed out during user open with zeus	状況メッセージ
Sgeo	送信側の ID
Ramy@zeus	受信側の ID

項目	説明
H *lines*	メッセージのヘッダー情報

sendmail での時間値
メッセージのタイムアウトおよびキュー処理の時間インターバルを設定するには、時間値に特定のフォーマットを使用しなければなりません。

時間値のフォーマットは次のとおりです。

```
-qNumberUnit
```

ここで、*Number* は整数値であり、*Unit* は単位を表す英字です。 *Unit* には次のいずれかの値を使用できます。

項目	説明
s	秒
m	分
h	時
d	日
w	週

Unit を指定しなかった場合、**sendmail** デーモンはデフォルトとして分 (**m**) を使います。 時間値を指定する 3 つの例を次に示します。

```
/usr/sbin/sendmail -q15d
```

このコマンドは、**sendmail** デーモンに 15 日ごとにキューを処理するよう指示します。

```
/usr/sbin/sendmail -q15h
```

このコマンドは、**sendmail** デーモンに 15 時間ごとにキューを処理するよう指示します。

```
/usr/sbin/sendmail -q15
```

このコマンドは、**sendmail** デーモンに 15 分ごとにキューを処理するよう指示します。

停滞したメール・キュー
場合によっては、何らかの理由でキューの処理が遅くなることがあります。 **-q** フラグを (値なしで) 使用すると、キュー処理を強制的に実行させることができます。

また、**-v** フラグ (詳細) を使用して、何が起きているかを知ることもできます。

```
/usr/sbin/sendmail -q -v
```

いずれかのキュー修飾コードを使用して、特定のキュー ID、送信側、または受信側のみにジョブを制限することもできます。 例えば、**-qRsally** と指定すると、受信側アドレスの 1 つに `sally` という文字列があるジョブのみにキュー処理を行うことができます。 同様に、**-qS***string* と指定すると、特定の送信側のみキュー処理を実行でき、**-qI***string* と指定すると、特定のキュー ID のみにキュー処理を実行できます。

キュー処理のインターバルの設定
sendmail デーモンがメール・キューを処理する時間インターバルは、デーモンが始動したときの **-q** フラグの値によって決まります。

通常、**sendmail** デーモンは、システム始動時に `/etc/rc.tcpip` ファイルによって始動されます。 `/etc/rc.tcpip` ファイルには、キュー処理インターバル (QPI) という変数が入っています。この変

数は、**sendmail** デーモンの始動時に **-q** フラグの値を指定するのに使用します。 デフォルトでは、**qpi** の値は 30 分です。 別のキュー処理インターバルを指定するには、次のようにします。

1. エディターを使用して /etc/rc.tcpip ファイルを編集します。
2. *qpi* 変数に値が割り当てられている次のような行を検索します。

```
qpi=30m
```

3. *qpi* 変数に割り当てられた値を、指定したい時間値に変更します。

これらの変更は、次のシステム再始動時に有効となります。 変更を直ちに有効にしたい場合は、**sendmail** デーモンを停止し、**-q** フラグに新しい値を指定して再始動します。 詳しくは、53 ページの『sendmail デーモンの停止』 および 52 ページの『sendmail デーモンの始動』 を参照してください。

メール・キューの移動
あるホストが長時間ダウンした場合、そのホスト宛てに (またはそのホストを介して) 経路指定された数多くのメッセージがシステム管理者のメール・キューに保管されることがあります。 その結果、sendmail コマンドを出すとキューのソート処理に長時間を要し、システムのパフォーマンスが大幅に低下します。 キューを一時的な場所へ移し、新しいキューを作成すると、あとでホストがサービスに戻ったときに古いキューを実行できます。

キューを一時的な場所へ移して新しいキューを作成するには、次のようにします。

1. 53 ページの『sendmail デーモンの停止』 の指示に従って、**sendmail** デーモンを停止します。
2. 次のように入力してキュー・ディレクトリー全体を移動します。

```
cd /var/spool
mv mqueue omqueue
```

3. 52 ページの『sendmail デーモンの始動』 の指示に従って、**sendmail** デーモンを再始動します。
4. 次のように入力して古いメール・キューを処理します。

```
/usr/sbin/sendmail -oQ/var/spool/omqueue -q
```

   -oQ フラグは、代替のキュー・ディレクトリーを指定します。 **-q** フラグは、キュー内のすべてのジョブを実行するように指定します。 操作の進行状況についてのレポートを入手するには、**-v** フラグを使用します。

   注 : この操作は、長時間を要する場合があります。
5. キューが空の場合は、次のように入力してログ・ファイルと一時ディレクトリーを削除します。

```
rm /var/spool/omqueue/*
rmdir /var/spool/omqueue
```

sendmail デーモンの始動
sendmail デーモンを始動するコマンドには、以下の 2 つがあります。

sendmail デーモンを始動するには、次のいずれかのコマンドを入力します。

```
startsrc -s sendmail -a "-bd -q15"
```

```
/usr/lib/sendmail -bd -q15
```

上記のコマンドの 1 つを入力したときに、**sendmail** デーモンが既にアクティブである場合は、次のメッセージが画面に表示されます。

```
The sendmail subsystem is already active. Multiple instances are not supported.
```

sendmail デーモンがアクティブでなかった場合は、**sendmail** デーモンが始動されたことを示すメッセージが表示されます。

sendmail デーモンの停止

sendmail デーモンを停止するには、**stopsrc -s sendmail** コマンドを実行します。

sendmail デーモンが **startsrc** コマンドによって始動されていない場合は、次のようにします。

- **sendmail** プロセス ID を検索します。
- **kill** *sendmail_pid* コマンドを入力します (この *sendmail_pid* は、**sendmail** プロセスのプロセス ID です)。

メールのロギング

sendmail コマンドは、**syslogd** デーモンを通してメール・システムのアクティビティーを 記録します。

ロギングを行うには、**syslogd** デーモンを構成して始動しておかなければなりません。 つまり、/etc/syslog.conf ファイルに、次のようなコメント行でない行を入れておく必要があります。

```
mail.debug          /var/spool/mqueue/log
```

この行がコメント行になっている場合は、任意のエディターを使用して変更します。パス名が正しいことを確認してください。 **syslogd** デーモンの実行中に /etc/syslog.conf ファイルを変更した場合は、コマンド・ラインに次のコマンドを入力して **syslogd** デーモンをリフレッシュします。

```
refresh -s syslogd
```

/var/spool/mqueue/log ファイルが存在しない場合は、次のコマンドを入力してこのファイルを作成しておく必要があります。

```
touch /var/spool/mqueue/log
```

ログ・ファイル内のメッセージは、次のフォーマットになっています。

システム・ログ内の個々の行は、タイム・スタンプ、それを生成したマシンの名前 (ローカル・エリア・ネットワークを介した複数のマシンからのログの場合)、「sendmail:」というワード、およびメッセージで構成されます。 ほとんどの メッセージは、一連の定義名 = 値 のペアで構成されます。

メッセージの処理時に記録される最も一般的な行は、**受信**行と**送達試行**行の 2 つです。 **受信**行には、メッセージの受信が記録され、1 つのメッセージにつき 1 行が記録されます。 一部のフィールドは省略される場合があります。 それらのメッセージ・フィールドは次のとおりです。

項目	説明
from	エンベロープ送信側のアドレスを指定します。
size	メッセージのサイズをバイト 単位で指定します。
class	メッセージのクラス (数値優先順位) を指定します。
pri	初期のメッセージ優先順位 (キューのソートに使用される) を指定します。
nrcpts	そのメッセージのエンベロープ受信側の数 (別名付けと転送を行ったあとの) を示します。
proto	メッセージの受信に使用するプロトコル (例えば、ESMTP または UUCP) を指定します。
relay	ここに指定したマシンからそのメッセージを受信しました。

送達試行行は、送達が 1 回試みられるたびに 1 行ずつ記録されます (したがって、送達が据え置かれたり受信側が複数の場合などには、1 つのメッセージにつき 複数の行が記録されます)。 フィールドは次のとおりです。

項目	説明
to	このメール・プログラムへ渡された受信側をコンマで区切ったリストが入っています。
ctladdr	制御側ユーザー、つまり、送達に使用する証明書のオーナーであるユーザーの名前を指定します。

項目	説明
delay	このメッセージの受信時刻から送信時刻までの合計遅延時間を指定します。
xdelay	この送達試行に要した時間量を指定します。
mailer	この受信側への送達に使用されたメール・プログラムの名前を指定します。
relay	この受信側を実際に受け入れた (またはリジェクトした) ホストの名前を指定します。
stat	送達状況を指定します。

このような大量の情報が記録されることがあるため、ログ・ファイルは連続したレベルに並べられます。 最下位のレベル 1 では、非常に異常な状況だけが記録されます。 最上位のレベルでは、重要でないイベントも記録されます。 慣例的に、10 以下のログ・レベルの情報が有用と見なされます。 64 を超えるログ・レベルは、デバッグ用に予約されています。 11 から 64 のレベルは、詳細情報用に予約されています。

sendmail コマンドがログ・ファイルに記録するアクティビティーのタイプは 、/etc/mail/sendmail.cf ファイル内の **L** オプションで指定します。

ログの管理

情報がログの末尾に随時追加されて行くため、そのファイルは非常に大きくなる可能性があります。 また、エラー条件により、メール・キューに予期しないエントリーが生成される場合もあります。 メール・キューとログ・ファイルが大きくなりすぎないようにするには、/usr/lib/smdemon.cleanu シェル・スクリプトを実行します。

このスクリプトを実行すると、**sendmail** コマンドに強制的にキューを処理させることができ、log.0、log.1、log.2 および log.3 という名前の 4 つのコピー (番号が進むほど古い) を維持できます。スクリプトが実行されるたびに、次のように移動されます。

- log.2 を log.3 へ
- log.1 を log.2 へ
- log.0 を log.1 へ
- log を log.0 へ

このスクリプトを実行させることで、ログを新規ファイルから新たに開始できます。 このスクリプトは手操作で実行するか、指定したインターバルで **cron** デーモンを使用して実行します。

トラフィック・ログ

トラフィックのロギングを設定する場合は、**sendmail** コマンドの **-X** フラグを使用します。

シンプル・メール転送プロトコル (SMTP) の多くは、そのプロトコルを完全には実装していません。 例えば、一部のパーソナル・コンピューター・ベースの **SMTP** は、応答コード内の継続行を認識しません。 それらの応答コードをトレースするのは非常に困難な場合があります。 そのような問題が疑われる場合は、**-X** フラグを使用してトラフィック・ログを設定できます。 例えば、次のとおりです。

```
/usr/sbin/sendmail -X /tmp/traffic -bd
```

このコマンドは、すべてのトラフィックを /tmp/traffic ファイルに記録します。

このコマンドは、即時に大量のデータを記録するので、通常の操作時には、このコマンドを決して使用しないでください。 このコマンドを実行後、**errant** のインプリメンテーションからご使用のホストにメッセージを強制的に送信させてください。 **sendmail** へ出入りしたすべてのメッセージ・トラフィックは、着信 **SMTP** トラフィックも含め、このファイルに記録されます。

sendmail を使用すると、**SIGUSR1** シグナルを送信することで、オープン・ファイルと接続キャッシュのダンプを記録できます。 結果は、**LOG_DEBUG** 優先順位で記録されます。

メール・プログラムの統計ログ

sendmail コマンドは、このコマンドとのインターフェースである 各メール・プログラムによって処理されるメールの総量を追跡します。

これらのメール・プログラムは、/etc/mail/sendmail.cf ファイル内に定義されています。

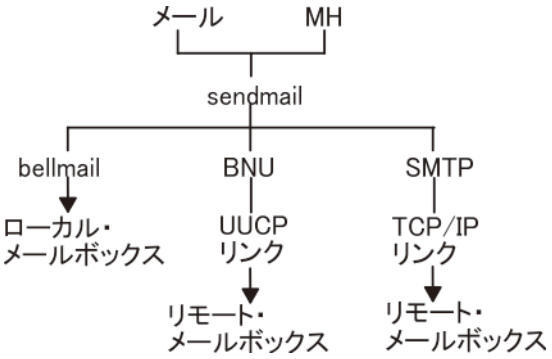


図 *3. sendmail* コマンドで使用するメール・プログラム

この図では、トップに Mail と MH があるトップダウン編成図のタイプを示します。 トップから bellmail、BNU、および SMTP が分岐しています。 そのすぐ下のレベルはそれぞれ、ローカル・メールボックス、UUCP リンク、および TCP/IP リンクです。 UUCP リンク と TCP/IP リンクの下は、リモート・メールボックスです。

メール・プログラムの統計の集計を開始するには、次のように入力して /etc/mail/statistics ファイルを作成します。

```
touch /etc/mail/statistics
```

sendmail コマンドは、統計情報の記録時にエラーを検出すると、syslog サブルーチンを使用してメッセージを書き込みます。 それらのエラーは、**sendmail** コマンドのその他の動作には影響を及ぼしません。

sendmail コマンドは、メールを処理するたびに、このファイル内の情報を更新します。 このファイルのサイズは大きくはなりませんが、ファイル内の数値は大きくなります。 それらの数値は、/etc/mail/statistics ファイルを作成またはリセットした時点以降のメールのボリュームを表します。

メール・プログラムの情報の表示

/etc/mail/statistics ファイル内に保持される統計はデータベース・フォーマットになっており、テキスト・ファイルとして読み取ることはできません。

メール・プログラムの統計を表示するには、コマンド・プロンプトに応じて次のように入力します。

```
/usr/sbin/mailstats
```

このコマンドは、/etc/mail/statistics ファイル内の情報を読み取り、そのフォーマットを設定し、それを標準出力に書き出します。 詳しくは、**/usr/sbin/mailstats** コマンドを参照してください。

sendmail メール・フィルター API

sendmail メール・フィルター API (*Milter* と呼ばれます) を使用すると、サード・パーティー・プログラムでメタ情報と内容をフィルターするために 処理中のメール・メッセージにアクセスできます。

sendmail フィルターの要件

フィルターはスレッドを 使用するため、フィルターはスレッド・セーフでなければなりません。 スレッドとの互換性を確保できるように、フィルターを 構成することができます。

多くのオペレーティング・システムで、標準 C ライブラリーの POSIX スレッドがサポートされています。スレッド・サポートとリンクするためのコンパイラー・フラグは、使用するコンパイラーとリンカーによって異なります。 使用されるローカル・フラグがわからない場合は、該当する obj.*/libmilter ビルド・サブディレクトリー内の Makefile を検査してください。

注 : フィルターはスレッドを使用するため、フィルターのプロセス制限の変更が必要な場合があります。例えば、フィルターが使用中になったとき、setrlimit を使用してオープン・ファイル・ディスクリプターの数を増やさないと、メールが拒否される場合があります。

sendmail フィルターの構成

sendmail を構成する際にフィルターを指定するには、以下のガイドラインに従ってください。

キー文字 X (eXternal 用) を使用してフィルターを指定します。 次の例では、3 つのフィルターが指定されています。

```
    Xfilter1, S=local:/var/run/f1.sock, F=R
    Xfilter2, S=inet6:999@localhost, F=T, T=C:10m;S:1s;R:1s;E:5m
    Xfilter3, S=inet:3333@localhost
```

次の構文を使用して .mc ファイルにフィルターを指定できます。

```
    INPUT_MAIL_FILTER(`filter1', `S=local:/var/run/f1.sock, F=R')
    INPUT_MAIL_FILTER(`filter2', `S=inet6:999@localhost, F=T, T=C:10m;S:1s;R:1s;E:5m')
    INPUT_MAIL_FILTER(`filter3', `S=inet:3333@localhost')
```

ここで、filter(番号) は、フィルターの名前です。 構文の先頭行は、フィルターが /var/run ディレクトリーの UNIX ドメインにあるソケットに関連付けられることを指定します。 2 行目は、フィルターがローカル・ホストのポート 999 上の IPv6 ソケットを使用することを指定します。 3 行目は、フィルターがローカル・ホストのポート 3333 上の IPv4 ソケットを使用することを指定します。

F= は、以下のフラグのいずれを適用するかを示します。

項目	説明
R	フィルターが使用不可の場合に接続が拒否されます。
T	フィルターが使用不可の場合に接続が一時的に失敗します。

いずれのフラグも指定されていない場合は、フィルターが存在しなかったかのように、メッセージは **sendmail** をパススルーします。

T= の値を指定すると、フィルターを使用して **sendmail** で使用されるデフォルトのタイムアウトを変更できます。 T= 等価は、以下のフィールドを使用します。

項目	説明
C	フィルターに接続するタイムアウト (0 の場合は、システム・タイムアウトを使用します)。
S	情報を MTA からフィルターに送信するタイムアウト。
R	応答をフィルターから読み取るタイムアウト。
E	フィルターへのメッセージ 終結通知の送信と最終確認通知の待機との間の全体のタイムアウト。

前記の例で示したとおり、各タイムアウトの間の区切りはセミコロン (;) です。各等価の間の区切りはコンマ (,) です。

タイムアウトのデフォルト値は、次のとおりです。

```
T=C:0m;S:10s;R:10s;E:5m
```

ここで、s は秒数、m は分数です。

InputMailFilters オプションは、使用されるフィルターとその順序を決定します。

注 : InputMailFilters オプションが指定されないと、フィルターは使用されません。

InputMailFilters オプションは、.mc ファイルの INPUT_MAIL_FILTER コマンドの順序に応じて自動的に設定されます。 この値をリセットするには、.mc ファイルの confINPUT_MAIL_FILTERS の値を設定します。 例えば、**InputMailFilters** オプションが次のように設定されているとします。

```
InputMailFilters=filter1, filter2, filter3
```

これらの 3 つのフィルターは、指定された順序と同じ順序で呼び出されます。

.mc ファイルで INPUT_MAIL_FILTER() の代わりに MAIL_FILTER() を使用することで、フィルターを入力フィルター・リストに追加しないでフィルターを定義することができます。

ライブラリー制御関数

sendmail フィルターは、制御を **libmilter** に引き渡す前に、**libmilter** パラメーターを設定するためにライブラリー制御関数を呼び出します。 **libmilter** パラメーターは、**smfi_main** 関数を呼び出して設定します。 また、フィルターは、そのコールバックを明確に登録するために、**smfi_register** 関数も呼び出します。 各関数は、操作の状況を示すために、MI_SUCCESS 値または MI_FAILURE 値のどちらかを返します。 これらの関数は、メール転送エージェント (MTA) とは通信しませんが、ライブラリーの状態を変更し、これが **smfi_main** 関数の内部の MTA に伝えられます。

表 2. ライブラリー制御関数

項目	説明
smfi_opensocket	**smfi_opensocket 関数は、インターフェース・ソケットを作成します。**
smfi_register	**smfi_register** 関数は、フィルターを登録します。
smfi_setconn	**smfi_setconn** 関数は、使用するソケットを指定します。
smfi_settimeout	**smfi_settimeout** 関数は、タイムアウトを設定します。
smfi_setbacklog	**smfi_setbacklog** 関数は、着信 **listen (2)** キュー・サイズを定義します。
smfi_setdbg	**smfi_setdbg** 関数は、**milter** ライブラリー・デバッグ (トレース) レベルを設定します。
smfi_stop	**smfi_stop** 関数は、正常シャットダウンを引き起こします。
smfi_main	**smfi_main** 関数は、制御を **libmilter** に引き渡します。

smfi_opensocket 関数

目的

smfi_opensocket 関数は、フィルターに接続するために使用される、インターフェース・ソケットのメール転送エージェント (MTA) の作成を試みます。

構文

```
#include <libmilter/mfapi.h>
int smfi_opensocket(
bool rmsocket
);
```

説明

smfi_opensocket 関数は、プログラムのメインラインからのみ呼び出され、**smfi_setconn** 関数と **smfi_register** 関数を呼び出した後、ただし **smfi_main** 関数を呼び出す前に呼び出されます。 **smfi_opensocket** 関数は、MTA とフィルターの間のインターフェースである **smfi_setconn** 関数を呼び出して前に指定されたソケットを作成します。 **smfi_opensocket** 関数は、呼び出し側アプリケーションがソケットを作成できるようにします。 **smfi_opensocket** 関数が呼び出されていない場合、**smfi_main** 関数は暗黙的にこの関数を呼び出します。

引数

表 *3.* 引数

項目	説明
rmsocket	ライブラリーは、新規ソケットを作成する前に、UNIX ドメインの既存のソケットの削除を試みる必要があるかどうかを示すフラグ。

戻り値

smfi_opensocket 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- インターフェース ・ソケットを作成できなかった。
- *rmsocket* 値が TRUE であり、ソケットを調査できなかったか、既存のソケットを削除できなかった。
- **smfi_setconn** 関数または **smfi_register** 関数が呼び出されていない。

関連情報

libmilter_smfi_register.dita

libmilter_smfi_setconn.dita

smfi_register 関数

目的

smfi_register 関数は、sendmail フィルターのコールバック 関数のセットを登録します。

構文

```
#include <libmilter/mfapi.h>
int smfi_register(
smfiDesc descr)
);
```

説明

smfi_register 関数は、**smfiDesc** 引数で提供された情報を使用して、sendmail フィルターを 作成します。 **smfi_register** 関数は、**smfi_main** 関数の前に呼び出す必要があります。

注 : 単一のプロセス内で複数の **smfi_register** 関数を正常に呼び出すことはできません。 単一の sendmail フィルターのみ、正常に登録できます。 ただし、ライブラリーはこの制限が守られているかどうかを検査できないことに注意してください。

xxfi_flags フィールドには、sendmail フィルターが実行できるアクションを記述した、ビット単位の値、ゼロ値、または以下のいずれかの値が入っていなければなりません。

表 *4.* 値

項目	説明
SMFIF_ADDHDRS	**smfi_addheader** 関数は、ヘッダーを追加します。
SMFIF_CHGHDRS	**smfi_chgheader** 関数は、ヘッダーを変更するか、ヘッダーを削除します。
SMFIF_CHGBODY	**smfi_replacebody** 関数は、フィルター処理中に本文を置き換えます。 このフィルターの後で他のフィルターが本文のフィルター処理を行う場合、フィルターはパフォーマンスに大きな影響を与えます。
SMFIF_ADDRCPT	**smfi_addrcpt** 関数は、受信側をメッセージに追加します。
SMFIF_ADDRCPT_PAR	**smfi_addrcpt_par** 関数は、Simple Mail Transfer Protocol (ESMTP) 引数を含めて、受信側を追加します。
SMFIF_DELRCPT	**smfi_delrcpt** 関数は、メッセージから受信側を削除します。
SMFIF_QUARANTINE	**smfi_quarantine** 関数は、メッセージを検疫します。
SMFIF_CHGFROM	**smfi_chgfrom** 関数は、エンベロープ送信側 (メールの発信元) を変更します。
SMFIF_SETSYMLIST	**smfi_setsymlist** 関数は、必要なシンボル (マクロ) のセットを送信します。

引数

表 *5.* 引数

項目	説明

descr	フィルター関数を記述する、タイプ smfiDesc のフィルター・ディスクリプター。 この構造体には、以下のメンバーがあります。
	`struct smfiDesc`
	`{`
	`  char  *xxfi_name;  /* filter name */`
	`  int  xxfi_version;  /* version code -- do not change */`
	`  unsigned long  xxfi_flags;  /* flags */`
	`  /* connection info filter */`
	`  sfsistat  (*xxfi_connect)(SMFICTX *, char *, _SOCK_ADDR *);`
	`  /* SMTP HELO command filter */`
	`  sfsistat  (*xxfi_helo)(SMFICTX *, char *);`
	`  /* envelope sender filter */`
	`  sfsistat  (*xxfi_envfrom)(SMFICTX *, char **);`
	`  /* envelope recipient filter */`
	`  sfsistat  (*xxfi_envrcpt)(SMFICTX *, char **);`
	`  /* header filter */`
	`  sfsistat  (*xxfi_header)(SMFICTX *, char *, char *);`
	`  /* end of header */`
	`  sfsistat  (*xxfi_eoh)(SMFICTX *);`
	`  /* body block */`
	`  sfsistat  (*xxfi_body)(SMFICTX *, unsigned char *, size_t);`
	`  /* end of message */`
	`  sfsistat  (*xxfi_eom)(SMFICTX *);`
	`  /* message aborted */`
	`  sfsistat  (*xxfi_abort)(SMFICTX *);`
	`  /* connection cleanup */`
	`  sfsistat  (*xxfi_close)(SMFICTX *);`
	`  /* any unrecognized or unimplemented command filter */`
	`  sfsistat  (*xxfi_unknown)(SMFICTX *, const char *);`
	`  /* SMTP DATA command filter */`
	`  sfsistat  (*xxfi_data)(SMFICTX *);`
	`  /* negotiation callback */`
	`  sfsistat (*xxfi_negotiate)(SMFICTX *,`
	`  unsigned long, unsigned long, unsigned long,`
	`  unsigned long,`
	`  unsigned long *, unsigned long *, unsigned long *,`
	`  unsigned long *`
	`  );`
	`  };`
	コールバック関数の NULL 値は、フィルターが指定のタイプの情報を処理しないことを示し、**SMFIS_CONTINUE** を返します。

headerf	ヘッダー名は、NULL 以外のヌル終了文字列です。

表 5. 引数 (続き)	
項目	**説明**
headerv	追加されるヘッダー値。NULL 以外のヌル終了文字列または空の文字列にすることができます。

戻り値

smfi_register 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- メモリーの割り当てに失敗した。
- 非互換バージョンまたは正しくないフラグ値。

関連情報

libmilter_smfi_addheader.dita

libmilter_smfi_chgheader.dita

libmilter_smfi_replacebody.dita

libmilter_smfi_addrcpt.dita

libmilter_smfi_addrcpt_par.dita

libmilter_smfi_delrcpt.dita

libmilter_smfi_quarantine.dita

libmilter_smfi_chgfrom.dita

libmilter_smfi_setsymlist.dita

smfi_setconn 関数

目的

smfi_setconn 関数は、このフィルターが **sendmail** コマンドとの通信に使用できるソケットを設定します。

構文

```
#include <libmilter/mfapi.h>
int smfi_setconn(
char *oconn;
);
```

説明

smfi_setconn 関数は、**smfi_main** 関数を呼び出す前に呼び出す必要があります。

UNIX またはローカル・ドメインのソケットを介して通信する場合、フィルターは root として実行してはなりません。

UNIX またはローカルのソケットに対する許可は、0600 (ソケットの所有者またはグループに対する読み取りまたは書き込み許可のみ) または 0660 (ソケットの所有者およびグループに対する読み取り/書き込み許可) に設定する必要があります。 これらの許可は、**sendmail RunAsUser** オプションが使用されている場合に便利です。

UNIX またはローカル・ドメインのソケットに対する許可は、**umask** によって決定され、これは 007 または 077 に設定する必要があります。 ソケットの許可を使用しないオペレーティング・システム (Solaris オペレーティング・システムなど) の場合は、ソケットを保護されたディレクトリーに入れます。

引数

表 6. 引数	
項目	**説明**

oconn	必要な通信ソケットのアドレス。 アドレスは、以下のような **proto:address** 形式のヌル終了文字列でなければなりません。
	`* {unix|local}:/path /to/file -- A named pipe.`
	`* inet:port @{hostname|ip-address} -- An IPV4 socket.`
	`* inet6:port @{hostname|ip-address} -- An IPV6 socket.`

戻り値

smfi_setconn 関数は、無効なアドレスであっても失敗しません。 ただし、メモリーがない場合、**smfi_setconn** 関数はソケットの設定に失敗します。 この障害は、**smfi_main** 関数でのみ検出されます。

関連情報

libmilter_smfi_main.dita

smfi_settimeout 関数

目的

smfi_settimeout 関数は、フィルターの 入出力タイムアウト値を設定します。

構文

```
#include <libmilter/mfapi.h>
int smfi_settimeout((
int otimeout
);
```

説明

smfi_settimeout 関数は、**smfi_main** 関数からのみ呼び出されます。 **smfi_settimeout** 関数は、**libmilter** パラメーターが、タイムアウトになる前にメール転送エージェント (MTA) 通信 (読み取りまたは書き込み) を待つ期間 (秒数) を設定します。

注 : smfi_settimeout 関数が呼び出されない場合、デフォルトのタイムアウト期間は 7210 秒です。

引数

表 7. 引数	
項目	**説明**
otimeout	**libmilter** パラメーターが、タイムアウトになる前に MTA を待つ期間 (秒数)。 *otimeout* 値は、ゼロより大きくなければなりません。 *otimeout* 値がゼロの場合、**libmilter** パラメーターは MTA を待ちません。

戻り値

smfi_settimeout 関数は、常に MI_SUCCESS 値を返します。

関連情報

smfi_main

smfi_setbacklog 関数

目的

smfi_setbacklog 関数は、フィルターの **listen(2)** バックログ値を設定します。

構文

```
#include <libmilter/mfapi.h>
int smfi_setbacklog(
int obacklog
);
```

説明

smfi_setbacklog 関数は、**smfi_main** 関数の呼び出しの前にのみ呼び出されます。 **smfi_setbacklog** 関数は、着信ソケットのバックログを設定し、これは **listen(2)** バックログ値によって使用されます。
smfi_setbacklog 関数が呼び出されない場合、デフォルトのオペレーティング・システムが使用されます。

引数

表 *8.* 引数

項目	説明
obacklog	listen キューで許容される着信接続の数。

戻り値

smfi_setbacklog 関数は、*obacklog* 引数がゼロ以下に設定されている場合、MI_FAILURE 値を返します。

関連情報

libmilter_smfi_main.dita

smfi_setdbg 関数

目的

smfi_setdbg 関数は、**milter** ライブラリーのデバッグ (トレース) レベルを設定します。

構文

```
#include <libmilter/mfapi.h>
int smfi_setdbg(
    int level;
);
```

説明

smfi_setdbg 関数は、**milter** ライブラリーの内部デバッグ・レベルを新規レベルに設定して、コードの詳細をトレースできるようにします。 ゼロのレベルは、デバッグをオフにします。 レベルが高くなるほど (正の値が大きくなるほど) デバッグがより詳細になります。 6 が、現行値であり、最高値であり、しかも実用的な値です。

引数

表 9. 引数	
項目	**説明**
level	新規のデバッグ・レベル。

戻り値

smfi_setdbg 関数は、デフォルトでは MI_SUCCESS 値を返します。

smfi_stop 関数

目的

smfi_stop 関数は、**milter** をオフにします。 この呼び出しの後は、接続は受け付けられません。

構文

```
#include <libmilter/mfapi.h>
int smfi_stop(void);
);
```

説明

smfi_stop 関数は、コールバック関数またはエラー処理ルーチンから随時に呼び出されます。 **smfi_stop** ルーチンは、新規の接続を許可しません。 ただし、この関数は、既存の接続 (スレッド) が終了するのを待ちません。 この関数により、**smfi_main** 関数は呼び出し側プログラムに戻り、呼び出し側プログラムはその後、終了するか、ウォーム・リスタートすることができます。

引数

表 10. 引数	
項目	**説明**
void	この引数は、どのような値も取りません。

戻り値

smfi_stop 関数は、以下の場合、SMFI_CONTINUE 値を返します。

- 内部ルーチンが **milter** ライブラリーを停止させる。
- ルーチンが **milter** ライブラリーを停止させる。
- 開始済みのプロセスを停止できない。

例

```
int ret;
SMFICTX *ctx;
...
ret = smfi_addheader(ctx, "Content-Type",
"multipart/mixed;¥n¥tboundary=¥"foobar¥"");
```

関連情報

コールバック関数

smfi_main 関数

目的

smfi_main 関数は、**libmilter** イベント・ループに制御を引き渡します。

構文

```
#include <libmilter/mfapi.h>
int smfi_main(
);
```

説明

smfi_main 関数は、フィルターの 初期化が完了した後で呼び出されます。

戻り値

smfi_main 関数は、接続を確立できなかった場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

障害はさまざまな理由で発生し、障害の理由がログに記録されます。 例えば、**smfi_setconn** 関数内で無効なアドレスを渡すと、関数は失敗します。

関連情報

libmilter_smfi_setconn.dita

データ・アクセス関数

データ・アクセス関数は、現行の接続またはメッセージに関する情報にアクセスするために、フィルター内に定義されたコールバック関数内から呼び出されます。

表 *11.* データ・アクセス関数

項目	説明
smfi_getsymal	**smfi_getsymal** 関数は、シンボルの値を返します。
smfi_getpriv	**smfi_getpriv** 関数は、専用データ・ポインターを取り出します。
smfi_setpriv	**smfi_setpriv** 関数は、専用データ・ポインターを設定します。
smfi_setreply	**smfi_setreply** 関数は、使用する特定の応答コードを設定します。
smfi_setmlreply	**smfi_setmlreply** 関数は、使用する特定の複数行応答を設定します。

smfi_getsymval 関数

目的

smfi_getsymval 関数は、**sendmail** マクロの値を取り出します。

構文

```
#include <libmilter/mfapi.h>
char* smfi_getsymval(
SMFICTX *ctx,
char *headerf,
```

```
char *symname
);
```

説明

smfi_getsymval 関数は、メッセージにヘッダーを追加するために、どの **xxfi_* callback** 関数からも呼び出されます。 マクロ定義は、呼び出される関数によって異なります。

デフォルトでは、以下のマクロが有効です。

表 *12.* 説明

項目	説明
xxfi_connect	**daemon_name, if_name, if_addr, j, _**
xxfi_hello	**tls_version, cipher, cipher_bits, cert_subject, cert_issuer**
xxfi_envfrom	**i, auth_type, auth_authen, auth_ssf, auth_author, mail_mailer, mail_host, mail_addr**
xxfi_envrcpt	**rcpt_mailer, rcpt_host, rcpt_addr**
xxfi_data	なし
xxfi_eoh	なし
xxfi_eom	**msg_id**

マクロが受け取られた時点から、**xxfi_connect** 関数および **xxfi_hello** 関数の接続の終了まで、すべてのマクロが有効のままです。

xxfi_envfrom 関数および **xxfi_eom** 関数のメッセージの終了まで、すべてのマクロが有効のままです。

xxfi_envrcpt 関数の各受信側に対して、すべてのマクロが有効のままです。

sendmail.mc の **confMILTER_MACROS_*** オプションを使用して、マクロのリストを変更できます。 このようなマクロの有効範囲は、**sendmail** コマンドによってマクロが設定されるときに決定されます。 マクロの値の説明については、「*Sendmail Installation and Operation Guide*」を参照してください。

引数

表 *13.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
symname	**sendmail** マクロの名前。 **sendmail.cf** ファイル内と同様に、1 文字のマクロは任意選択により中括弧 (「{」と「}」) で囲むことができ、2 文字以上のマクロ名は中括弧で囲む必要があります。

戻り値

smfi_getsymval 関数は、指定されたマクロの値をヌル終了文字列として返します。 あるいは、マクロが定義されていない場合は、**smfi_getsymval** 関数は NULL を返します。

関連情報

libmilter_xxfi_connect.dita

libmilter_xxfi_helo.dita

libmilter_xxfi_envfrom.dita

libmilter_xxfi_envrcpt.dita

libmilter_xxfi_data.dita

libmilter_xxfi_eoh.dita

libmilter_xxfi_eom.dita

smfi_getpriv 関数

目的

smfi_getpriv 関数は、この接続用の接続固有のデータ・ポインターを取り出します。

構文

```
#include <libmilter/mfapi.h>
void* smfi_getpriv(
SMFICTX *ctx
);
```

説明

smfi_getpriv 関数は、任意の **xxfi_* callback** 関数から呼び出すことができます。

引数

表 *14.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。

戻り値

smfi_getpriv 関数は、**smfi_setpriv** 関数の前の呼び出しによって保管された専用データ・ポインターを返します。 あるいは、値が設定されていない場合は、**smfi_setpriv** 関数は NULL を返します。

関連情報

libmilter_smfi_setpriv.dita

smfi_setpriv 関数

目的

smfi_setpriv 関数は、この接続用の専用データ・ポインターを設定します。

構文

```
#include <libmilter/mfapi.h>
int smfi_setpriv
SMFICTX *ctx,
void *privatedata
();
```

説明

smfi_setpriv 関数は、*ctx* 用の専用データ・ポインターを設定するために、どの **xxfi_* callback** 関数からも呼び出されます。

注 : 接続ごとに 1 つの専用データ・ポインターがあります。**smfi_setpriv** 関数に対して異なる値を指定して複数の呼び出しを実行すると、前の値が失われます。 フィルターは、終了する前に、専用データを解放し、ポインターを NULL に設定する必要があります。

引数

表 *15.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
privatedata	この引数は、専用データを指します。 後続の **smfi_getpriv** 関数の呼び出しによって、*ctx* を使用して、この値が返されます。

戻り値

smfi_setpriv 関数は、*ctx* が無効なコンテキストである場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

関連情報

libmilter_smfi_setpriv.dita#smfi_setpriv

smfi_setreply 関数

目的

smfi_setreply 関数は、デフォルトの Simple Mail Transfer Protocol (SMTP) エラー応答コードを設定し、4XX および 5XX の応答コードのみを受け入れます。

構文

```
#include <libmilter/mfapi.h>
int smfi_setreply
SFICTX *ctx,
char *rcode,
char *xcode,
char *message
);
```

説明

smfi_setreply 関数は、**xxfi_connect** 関数以外のどの **xxfi_callback** 関数からも呼び出されます。 **smfi_setreply** 関数は、接続用の SMTP エラー応答コードを設定します。 このコードは、このフィルターによって実行されるアクションの結果生じる後続のエラー応答に使用されます。

smfi_setreply 関数に渡される値は、規格への準拠について検査されません。

message 引数は、印刷可能文字のみを含んでいる必要があります。 その他の文字は、未定義の動作を引き起こす可能性があります。 例えば、CR や LF のような文字があると呼び出しが失敗する原因となり、単一の「%」文字があるとテキストが無視される原因となります。

注 : パラメーター内で「%」文字列を使用する必要がある場合は、printf(3) の場合と同様に「%%」を使用します。

応答コードとその意味についての詳細は、RFC 821 または 2821、および RFC 1893 または 2034 を参照してください。

rcode 引数が 4XX として設定されているが、メッセージに SMFI_REJECT 値が使用されている場合、カスタム応答は使用されません。

rcode 引数が 5XX として設定されているが、メッセージに SMFI_TEMPFAIL 値が使用されている場合、カスタム応答は使用されません。

注 : 上記の 2 つのケースでは、**milter** パラメーターにエラーが返されます。 **Libmilter** パラメーターは、上記の応答コードを無視します。

milter パラメーターが SMFI_TEMPFAIL 値を返し、応答コードを 421 に設定した場合、SMTP サーバーは 421 エラー・コードで SMTP セッションを終了します。

引数

表 *16.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
rcode	3 桁 (RFC 821 または 2821) の SMTP 応答コードは、ヌル終了文字列です。 *rcode* は NULL にすることはできず、有効な 4XX または 5XX の応答コードでなければなりません。
xcode	拡張 (RFC 1893 または 2034) 応答コード。 *xcode* が NULL の場合、拡張コードは使用されません。 それ以外の場合、*xcode* は RFC 1893 または 2034 に準拠していなければなりません。
message	SMTP 応答のテキスト部分。 メッセージが NULL の場合、空のメッセージが使用されます。

戻り値

smfi_setreply 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *rcode* 引数または *xcode* 引数が無効である。
- メモリー割り当て障害が発生する。

関連情報

libmilter_xxfi_connect.dita

smfi_setmlreply 関数

目的

smfi_setmlreply 関数は、複数行の応答へのデフォルトの Simple Mail Transfer Protocol (SMTP) エラー応答コードを設定します。 **smfi_setmlreply** 関数は、4XX と 5XX の応答のみを受け入れます。

構文

```
#include <libmilter/mfapi.h>
int smfi_setmlreply(
SMFICTX *ctx,
char *rcode,
char *xcode,
...
);
```

説明

smfi_setmlreply 関数は、**xxfi_connect** 関数を除き、どの **xxfi_callback** 関数からも呼び出されます。 **smfi_setmlreply** 関数は、*xcode* の下に記述されている接続用の SMTP エラー応答コードを提供します。 引数のリストは、ヌル終了でなければなりません。 このコードは、このフィルターによって実行されるアクションの結果生じる後続のエラー応答に使用されます。

smfi_setmlreply 関数に渡される値は、規格への準拠について検査されません。

message パラメーターには、印刷可能文字のみを含める必要があります。その他の文字は、未定義の動作を引き起こす可能性があります。 例えば、CR や LF のような文字があると呼び出しが失敗する原因となり、単一の「%」文字があるとテキストが無視される原因となります。

注 : message パラメーター内で「%」文字列が必要な場合は、**printf(3)** 文字列を使用するのと同様に「%%」文字列を使用します。

応答コードとその意味については、RFC 821 または 2821、および RFC 1893 または 2034 を参照してください。

rcode が 4XX として設定されているが、メッセージに SMFI_REJECT 値が使用されている場合、カスタム応答は使用されません。

rcode が 5XX として設定されているが、メッセージに SMFI_TEMPFAIL 値が使用されている場合、カスタム応答は使用されません。

注 : 上記の 2 つのケースでは、**milter** パラメーターにエラーが返され、**Libmilter** パラメーターはそのエラーを無視します。

milter パラメーターが SMFI_TEMPFAIL 値を返し、応答コードを 421 に設定した場合、SMTP サーバーは 421 エラー・コードで SMTP セッションを終了します。

引数

表 *17.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
rcode	ヌル終了文字列としての 3 桁 (RFC 821 または 2821) の SMTP 応答コード。 *rcode* 引数は NULL にすることはできず、有効な 4XX または 5XX の応答コードでなければなりません。
xcode	拡張 (RFC 1893 または 2034) 応答コード。 *xcode* が NULL の場合、拡張コードは使用されません。 それ以外の場合、*xcode* は RFC 1893 または 2034 に準拠していなければなりません。
...	残りの引数は、最大 32 個の引数からなる単一のテキスト行で、これは SMTP 応答のテキスト部分として使用されます。 リストはヌル終了でなければなりません。

戻り値

smfi_setmlreply 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *rcode* 引数または *xcode* 引数が無効である。
- メモリー割り当て障害が発生する。
- テキスト行に復帰または改行が含まれている。

- テキスト行の長さが MAXREPLYLEN(980) を超えている。
- テキスト応答が 32 行を超えている。

例

```
ret = smfi_setmlreply(ctx, "550", "5.7.0",
"Spammer access rejected",
"Please see our policy at:",
 "http://www.example.com/spampolicy.html",
NULL);
```

前の例の結果は、次のようになります。

```
550-5.7.0 Spammer access rejected
550-5.7.0 Please see our policy at:
550 5.7.0 http://www.example.com/spampolicy.html
```

関連情報

libmilter_xxfi_connect.dita

メッセージ変更関数

メッセージ変更関数は、メッセージの内容および属性を変更します。 この関数は、**xxfi_eom** 関数内からのみ呼び出されます。 メッセージ変更関数は、メール転送エージェント (MTA) との追加の通信を呼び出すことができます。 これらの関数は、操作の状況を示すために、MI_SUCCESS 値または MI_FAILURE 値のどちらかを返します。

注 : パラメーター内のメッセージ変更関数に渡されるメッセージ・データ (送信側、受信側、ヘッダー、および本文のチャンク) はコピーされるので、保存する必要はありません (割り当てられたメモリーを解放できます)。

メッセージ変更関数を呼び出すには、フィルターは **smfi_register** 関数に渡される記述内に適切なフラグを設定する必要があります。 フラグが設定されていない場合、MTA はその関数の呼び出しをフィルターの障害として扱い、接続を終了します。

注 : 関数によって返される状況は、メッセージ・フィルターが MTA に正常に送信されたかどうかを示します。 この状況は、MTA が要求された操作を実行したかどうかを示すものではありません。 例えば、**smfi_header** 関数は、正しくないヘッダー名を指定して呼び出された場合、たとえ後で MTA がその正しくないヘッダーの追加を拒否する可能性があっても、MI_SUCCESS フラグを返します。

表 *18. Mod* 関数

項目	説明	関数
smfi_addheader	**smfi_addheader** 関数は、メッセージにヘッダーを追加します。	**SMFIF_ADDHDRS**
smfi_chgheader	**smfi_chgheader** 関数は、ヘッダーを変更または削除します。	**SMFIF_CHGHDRS**
smfi_insheader	**smfi_insheader** 関数は、メッセージにヘッダーを挿入します。	**SMFIF_ADDHDRS**
smfi_chgfrom	**smfi_chgfrom** 関数は、エンベロープ送信側のアドレスを変更します。	**SMFIF_CHGFROM**
smfi_addrcpt	**smfi_addrcpt** 関数は、エンベロープに受信側を追加します。	**SMFIF_ADDRCPT**

表 18. Mod 関数 (続き)		
項目	**説明**	**関数**
smfi_addrcpt_par	**smfi_addrcpt_par** 関数は、拡張 Simple Mail Transfer Protocol (ESMTP) パラメーターを含めて、受信側をエンベロープに追加します。	**SMFIF_ADDRCPT_PAR**
smfi_delrcpt	**smfi_delrcpt** 関数は、エンベロープから受信側を削除します。	**SMFIF_DELRCPT**
smfi_replacebody	**smfi_replacebody** 関数は、メッセージの本文を置き換えます。	**SMFIF_CHGBODY**

smfi_addheader 関数

目的

smfi_addheader 関数は、ヘッダーを現行メッセージに追加します。

構文

```
#include <libmilter/mfapi.h>
int smfi_addheader(
SMFICTX *ctx,
char *headerf,
char *headerv
);
```

説明

smfi_addheader 関数は、ヘッダーをメッセージに追加するために、**xxfi_eom** 関数から呼び出されます。

smfi_addheader 関数は、メッセージの既存のヘッダーは変更しません。

ヘッダーの現行値を変更する場合は、**smfi_chgheader** 関数を使用します。

smfi_addheader 関数を呼び出すフィルターは、`smfiDesc_str` 引数に **SMFIF_ADDHDRS** フラグを設定する必要があります。 これにより、フィルターはその 値を **smfi_register** 関数に渡します。

smfi_addheader 関数では、フィルターの 順序を指定する必要があります。 以前に作成されたフィルターを使用して、ヘッダー内の変更を表示できます。

ヘッダーの名前や値は、規格への準拠について検査されません。 ただし、ヘッダーの各行は 998 文字を下回っている必要があります。 これより長いヘッダー名が必要な場合は、複数行のヘッダーを使用します。 複数行のヘッダーを作成する必要がある場合は、改行 (`ASCII 0x0a`、または C プログラミング言語の `¥n`) と、その後にスペース (`ASCII 0x20`) やタブ (`ASCII 0x09`、または C プログラミング言語の `¥t`) などの空白文字を挿入します。 改行の前に復帰 (`ASCII 0x0d`) があってはなりません。 復帰は、メール転送エージェント(MTA) が自動的に追加します。 フィルターの 作成者の責任で、どの規格にも違反していないことを確認する必要があります。

SMFIP_HDR_LEADSPC フラグが設定されていない場合、MTA は追加されたヘッダー値に先行スペースを追加します。フラグが設定されている場合は、必要なすべての先行スペースが **milter** パラメーターに含まれている必要があります。

引数

表 19. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
headerf	ヘッダー名は、NULL 以外のヌル終了文字列です。
headerv	追加されるヘッダー値。NULL 以外のヌル終了文字列または空の文字列にすることができます。

戻り値

smfi_addheader 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *headerf* 引数または *headerv* 引数が NULL である。
- 現行接続状態でのヘッダーの追加は無効である。
- メモリーの割り当てに失敗する。
- ネットワーク・エラーが発生する。
- **smfi_register** 関数が呼び出されたときに **SMFIF_ADDHDRS** フラグが設定されていなかった。

例

```
int ret;
SMFICTX *ctx;
...
ret = smfi_addheader(ctx, "Content-Type",
"multipart/mixed;¥n¥tboundary=¥"foobar¥"");
```

関連情報

libmilter_xxfi_eom.dita

libmilter_smfi_chgheader.dita

libmilter_smfi_register.dita

smfi_chgheader 関数

目的

smfi_chgheader 関数は、メッセージ・ヘッダーを変更または削除します。

構文

```
#include <libmilter/mfapi.h>
int smfi_chgheader(
SMFICTX *ctx,
char *headerf,
mi_int32 hdridx,
char *headerv
);
```

説明

smfi_chgheader 関数は、現行メッセージのヘッダー値を変更するために、**xxfi_eom** 関数から呼び出されます。

smfi_chgheader 関数を使用して、新規ヘッダーを追加することができます。 ただし、**smfi_addheader** 関数を使用した方が効率的で安全です。

smfi_chgheader 関数を呼び出すフィルターは、smfiDesc_str 引数に **SMFIF_CHGHDRS** フラグを設定する必要があります。 これにより、フィルターはその値を **smfi_register** 関数に渡します。

smfi_chgheader 関数では、フィルターの順序を指定する必要があります。 以前に作成されたフィルターを使用して、ヘッダー内の変更を表示できます。

ヘッダーの名前や値は、規格への準拠について検査されません。 ただし、ヘッダーの各行は 998 文字を下回っている必要があります。 これより長いヘッダー名が必要な場合は、複数行のヘッダーを使用します。 複数行のヘッダーを作成する必要がある場合は、改行 (ASCII 0x0a、または C プログラミング言語の ¥n) と、その後にスペース (ASCII 0x20) やタブ (ASCII 0x09、または C プログラミング言語の ¥t) などの空白文字を挿入します。 改行の前に復帰 (ASCII 0x0d) があってはなりません。復帰は、メール転送エージェント (MTA) が自動的に追加します。 フィルターの 作成者の責任で、どの規格にも違反していないことを確認する必要があります。

SMFIP_HDR_LEADSPC フラグが設定されていない場合、MTA は追加されたヘッダー値に先行スペースを追加します。フラグが設定されている場合は、必要な先行スペースが **milter** パラメーター自体に含まれている必要があります。

引数

表 *20.* 引数

項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
headerf	ヘッダー名は、NULL 以外のヌル終了文字列です。
hdridx	ヘッダー索引値 (1 を基本とする)。 *hdridx* 値の 1 は、最初に出現した *headerf* という名前のヘッダーを変更します。 *hdridx* が *headerf* の出現回数より大きい場合、*headerf* の新規コピーが追加されます。
headerv	追加されるヘッダー値。NULL 以外のヌル終了文字列または空の文字列にすることができます。

戻り値

smfi_chgheader 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *headerf* 引数が NULL である。
- 現行接続状態でのヘッダーの変更は無効である。
- メモリーの割り当てに失敗する。
- ネットワーク・エラーが発生する。
- **smfi_register** 関数が呼び出されたときに **SMFIF_CHGHDRS** フラグが設定されていなかった。

例

```
int ret;
SMFICTX *ctx;
...

ret = smfi_chgheader(ctx, "Content-Type", 1,
"multipart/mixed;¥n¥tboundary=¥"foobar¥"");
```

関連情報

libmilter_xxfi_eom.dita

libmilter_smfi_addheader.dita

smfi_insheader 関数

目的

smfi_insheader 関数は、ヘッダーを現行メッセージの前に付加します。

構文

```
#include <libmilter/mfapi.h>
int smfi_insheader(
SMFICTX ,
int hdridx,
char *headerf,
char *headerv
);
```

説明

smfi_insheader 関数は、ヘッダーを現行メッセージの前に付加するために、**xxfi_eom** 関数から呼び出されます。

smfi_insheader 関数は、メッセージの既存のヘッダーは変更しません。

ヘッダーの現行値を変更する場合は、**smfi_chgheader** 関数を使用します。

smfi_insheader 関数を呼び出すフィルターは、smfiDesc_str 引数に **SMFIF_ADDHDRS** フラグを設定する必要があり、これが **smfi_register** 関数に渡されます。

smfi_insheader 関数では、フィルターの順序を指定する必要があります。 以前に作成されたフィルターを使用して、ヘッダー内の変更を表示できます。

フィルターは、Simple Mail Transfer Protocol (SMTP) クライアントによって送信されたヘッダーを受け取り、以前のフィルターによって変更されたヘッダーも受け取ります。 **sendmail** コマンドによって挿入されたヘッダーと、単独で挿入されたヘッダーは受け取りません。 ヘッダーを挿入する位置は、着信メッセージ内に存在するヘッダーによって異なり、さらに **sendmail** コマンドで追加するように構成されているヘッダーによっても異なります。

例えば、**sendmail** コマンドは常にヘッダーの先頭に **Received: header** を追加します。 *hdridx* 値を 0 に設定すると、ヘッダーは **Received: header** パラメーターの前に挿入されます。 ただしフィルターは追加されたヘッダーは受け取るが、**Received: header** は受け取らないため、後で判断を誤ることになります。 このようなことが、ヘッダーを固定された位置に挿入することを難しくしています。

hdridx 値がメッセージ内のヘッダーの数より大きい場合、ヘッダーが追加されます。

ヘッダーの名前や値は、規格への準拠について検査されません。 ただし、ヘッダーの各行は 998 文字を下回っている必要があります。 これより長いヘッダー名が必要な場合は、複数行のヘッダーを使用します。 複数行のヘッダーを作成する必要がある場合は、改行 (ASCII 0x0a、または C プログラミング言語の ¥n) と、その後にスペース (ASCII 0x20) やタブ (ASCII 0x09、または C プログラミング言語の ¥t) などの空白文字を挿入します。 改行の前に復帰 (ASCII 0x0d) があってはなりません。 メール転送エージェント (MTA) が自動的に復帰を追加します。 フィルターの作成者の責任で、どの規格にも違反していないことを確認する必要があります。

SMFIP_HDR_LEADSPC フラグが設定されていない場合、MTA は挿入されたヘッダー値に先行スペースを追加します。フラグが設定されている場合は、必要なすべての先行スペースが **milter** パラメーターに含まれている必要があります。

引数

表 21. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
headerf	ヘッダー名は、NULL 以外のヌル終了文字列です。
headerv	追加されるヘッダー値。NULL 以外のヌル終了文字列または空の文字列にすることができます。

戻り値

smfi_insheader 関数は、以下の場合、MI_FAILURE 値を返します。それ以外の場合、関数は MI_SUCCESS を返します。

- *headerf* 引数または *headerv* 引数が NULL である。
- 現行接続状態でのヘッダーの追加は無効である。
- メモリーの割り当てに失敗する。
- ネットワーク・エラーが発生する。
- **smfi_register** 関数が呼び出されたときに **SMFIF_ADDHDRS** フラグが設定されていなかった。

例

```
int ret;
SMFICTX *ctx;
...
ret = smfi_insheader( ctx, 0, "First", "See me?");;
```

関連情報

libmilter_xxfi_eom.dita

libmilter_smfi_register.dita

libmilter_smfi_chgheader.dita

smfi_chgfrom 関数

目的

smfi_chgfrom 関数は、現行メッセージのエンベロープ送信側 (メールの発信元) を変更します。

構文

```
#include <libmilter/mfapi.h>
int smfi_chgfrom(
SMFICTX *ctx,
const char *mail,
char *args
);
```

説明

smfi_chgfrom 関数は、現行メッセージのエンベロープの送信側およびメールの発信元 (MAIL From) を変更するために、**xxfi_eom** 関数から呼び出されます。

smfi_chgfrom 関数を呼び出すフィルターは、smfiDesc_str 引数に **SMFIF_CHGFROM** フラグを設定する必要があります。 これにより、フィルターはその値を **smfi_register** 関数に渡します。

この呼び出しを通じて、すべての拡張 Simple Mail Transfer Protocol (ESMTP) 引数を設定できます。 ただし、SIZE や BODY のような一部の引数の設定値は、問題を引き起こします。 そのため、引数を設定する際には十分に注意する必要があります。 呼び出しが成功した場合、メール転送エージェント (MTA) から **milter** パラメーターへのフィードバックはありません。

引数

表 22. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
mail	新しい送信側アドレス。
args	拡張 Simple Mail Transfer Protocol (ESMTP) 引数。

戻り値

smfi_chgfrom 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *mail* 引数が NULL である。
- 現行接続状態での送信側の変更は無効である。
- ネットワーク・エラーが発生する。
- **smfi_register** 関数が呼び出されたときに **SMFIF_CHGFROM** フラグが設定されていなかった。

関連情報

libmilter_xxfi_eom.dita

libmilter_smfi_register.dita

smfi_addrcpt 関数

目的

smfi_addrcpt 関数は、現行メッセージの受信側を追加します。

構文

```
#include <libmilter/mfapi.h>
int smfi_addrcpt(
SMFICTX *ctx
char *rcpt
);
```

説明

smfi_addrcpt 関数は、メッセージ・エンベロープに受信側を追加するために、**xxfi_eom** 関数からのみ呼び出されます。

注 : smfi_addrcpt 関数を呼び出すフィルターは、**smfi_register** 関数に渡される **smfiDesc_str** 構造体に **SMFIF_ADDRCPT** フラグを設定する必要があります。

引数

表 23. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
rcpt	新しい受信側のアドレス。

戻り値

smfi_addrcpt 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *rcpt* 引数が NULL である。
- 現行接続状態での受信側の追加は無効である。
- ネットワーク・エラーが発生する。
- **smfi_register** 関数が呼び出されたときに **SMFIF_ADDRCPT** フラグが設定されていなかった。

関連情報

libmilter_xxfi_eom.dita

libmilter_smfi_register.dita

smfi_addrcpt_par 関数

目的

smfi_addrcpt_par 関数は、拡張 Simple Mail Transfer Protocol (ESMTP) 引数を含めて、現行メッセージに受信側を追加します。

構文

```
#include <libmilter/mfapi.h>
int smfi_addrcpt_par(
SMFICTX *ctx,
char *rcpt,
char *args
);
```

説明

smfi_addrcpt_par 関数は、メッセージ・エンベロープに受信側を追加するために、**xxfi_eom** 関数から呼び出されます。

引数

表 24. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
rcpt	新しい受信側のアドレス。
args	新しい受信側の ESMTP パラメーター。

戻り値

smfi_addrcpt 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *rcpt* 引数が NULL である。
- 現行接続状態での受信側の追加は無効である。
- ネットワーク・エラーが発生する。
- **smfi_register** 関数が呼び出されたときに **SMFIF_ADDRCPT_PAR** フラグが設定されていなかった。

関連情報

libmilter_smfi_addrcpt.dita

libmilter_smfi_register.dita

smfi_delrcpt 関数

目的

smfi_delrcpt 関数は、現行メッセージのエンベロープから受信側を削除します。

構文

```
#include <libmilter/mfapi.h>
int smfi_delrcpt(
    SMFICTX *ctx;
    char *rcpt;
);
```

説明

smfi_delrcpt 関数は、現行メッセージ・エンベロープから指定された受信側を削除するために、**xxfi_eom** コールバック関数から呼び出されます。

注 : 削除されるアドレスは、正確に一致していなければなりません。 例えば、アドレスとその拡張形式は一致しません。

引数

表 *25.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
rcpt	削除される受信側アドレス。NULL 以外のヌル終了文字列。

戻り値

smfi_delrcpt 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *rcpt* 引数が NULL である。
- 現行接続状態での受信側の削除は無効である。
- ネットワーク・エラーが発生する。
- smfi_register 関数が呼び出されたときに **SMFIF_DELRCPT** フラグが設定されていなかった。

関連情報

smfi_register

xxfi_eom

smfi_replacebody 関数

目的

smfi_replacebody 関数は、メッセージの本文を置き換えます。

構文

```
#include <libmilter/mfapi.h>
int smfi_replacebody(
    SMFICTX *ctx,
    unsigned char *bodyp,
    int bodylen
);
```

説明

smfi_replacebody 関数は、現行メッセージの本文を置き換えます。 関数が複数回呼び出された場合、後続の呼び出しではデータが新しい本文に付加されます。 この関数は、複数回呼び出されることがあります。

メッセージの本文は大きくなる可能性があるため、SMFIF_CHGBODY フラグを設定すると、フィルターのパフォーマンスに大きな影響を与えることがあります。

フィルターが SMFIF_CHGBODY フラグを設定したが **smfi_replacebody** 関数を呼び出さない場合、元の本文が変更されないままになります。

smfi_replacebody 関数の場合、フィルターの順序が重要です。 新しい本文の内容は、古いフィルターによって作成され、新しいフィルター・ファイルに入れられます。

引数

表 *26.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
bodyp	新しい本文データ (これは、ヌル終了である必要はありません) の先頭へのポインター。 *bodyp* が NULL の場合、長さが 0 として扱われます。本文データは CR 形式または LF 形式でなければなりません。
bodylen	*bodyp* によって指し示されたデータ・バイト数。

戻り値

smfi_replacebody 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *bodyp* が NULL で、*bodylen* > 0 である。
- 現行接続状態での本文の変更は無効である。
- ネットワーク・エラーが発生する。
- smfi_register 関数が呼び出されたときに **SMFIF_CHGBODY** フラグが設定されていなかった。

関連情報

smfi_register

メッセージ処理関数

メッセージ処理関数は、メッセージの内容や状況を変更せずに、**milter** パラメーターまたはメール転送エージェント (MTA) に関する特殊なケースの処理命令を提供します。 メッセージ処理関数は、**xxfi_eom** 関数からのみ呼び出すことができます。 **xxfi_eom** 関数は、MTA との追加の通信を呼び出し、操作の状況を示すために MI_SUCCESS 値または MI_FAILURE 値のどちらかを返すことができます。

注 : 関数によって返される状況は、フィルター・メッセージが MTA に正常に送信されたかどうかを示します。 この状況は、MTA が要求された操作を実行したかどうかを示すものではありません。

表 *27.* メッセージ処理関数

項目	説明
smfi_progress	**smfi_progress** 関数は、進行中の操作を報告します。
smfi_quarantine	**smfi_quarantine** 関数は、メッセージを検疫します。

smfi_progress 関数

目的

smfi_progress 関数は、操作の進行状況を報告します。

構文

```
#include <libmilter/mfapi.h>
int smfi_progress(
    SMFICTX *ctx;
);
```

説明

smfi_progress 関数は、**xxfi_eom** コールバック関数から呼び出され、フィルターがまだメッセージを 処理中であることをメール転送エージェント (MTA) に通知します。 この関数により、MTA はタイムアウトを再始動します。

引数

表 *28.* 引数

項目	説明
ctx	**libmilter** パラメーター内に維持されている、不透明なコンテキスト構造体。

戻り値

smfi_progress 関数は、ネットワーク・エラーがある場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

関連情報

xxfi_eom

smfi_quarantine 関数

目的

smfi_quarantine 関数は、メッセージを検疫します。

構文

```
#include <libmilter/mfapi.h>
int smfi_quarantine(
    SMFICTX *ctx;
    char *reason;
);
```

説明

smfi_quarantine 関数は、示された理由を使用してメッセージを検疫するために、**xxfi_eom** コールバック関数から呼び出されます。

引数

表 *29.* 引数

項目	説明
ctx	**libmilter** パラメーター内に維持されている、不透明なコンテキスト構造体。
reason	検疫理由を示す、NULL 以外のヌル終了文字列。

戻り値

smfi_quarantine 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- *reason* が NULL または空である。
- ネットワーク・エラーが発生する。
- `smfi_register` 関数が呼び出されたときに **SMFIF_QUARANTINE** フラグが設定されていなかった。

関連情報

smfi_register

xxfi_eom

コールバック関数

sendmail フィルターは、1 つ以上のコールバック関数を実装する必要があり、これは **smfi_register** 関数を介して登録されます。

表 *30.* コールバック関数

項目	説明
xxfi_connect	**xxfi_connect** 関数は、各 SMTP 接続の開始時に 1 回呼び出されます。 関数は SMFIS_CONTINUE 値を返します。
xxfi_hello	**xxfi_hello** 関数は、クライアントが HELO/EHLO コマンドを送信するたびに呼び出されます。
xxfi_envfrom	**xxfi_envfrom** 関数は、メッセージの先頭で呼び出されます。

表 30. コールバック関数 (続き)	
項目	**説明**
xxfi_envrcpt	**xxfi_envrcpt** 関数は、受信側ごとに呼び出されます。
xxfi_data	**xxfi_data** 関数は、DATA コマンドを処理します。
xxfi_unknown	**xxfi_unknown** 関数は、不明の Simple Mail Transfer Protocol (SMTP) コマンドを処理します。
xxfi_header	**xxfi_header** 関数は、1 つのメッセージ・ヘッダーを処理します。
xxfi_eoh	**xxfi_eoh** 関数は、複数のメッセージ・ヘッダーを処理します。
xxfi_body	**xxfi_body** 関数は、メッセージ本文の 1 つのチャンクを処理します。
xxfi_eom	**xxfi_eom** 関数は、メッセージの終了を処理します。
xxfi_abort	**xxfi_abort** 関数は、打ち切られたメッセージを処理します。
xxfi_close	**xxfi_close** 関数は、現行接続を閉じるために呼び出されます。
xxfi_negotiate	**xxfi_negotiate** 関数は、SMTP 接続の開始時に呼び出されます。

コールバック関数は、適切な値を返す必要があります。 コールバック関数が定義済みの値以外の値を返した場合、エラーが継続し、**sendmail** コマンドはフィルターへの接続を終了します。

Milter パラメーターは、**recipient-oriented** ルーチン、**message-oriented** ルーチン、および **connection-oriented** ルーチンを、以下のように区別しています。

- **recipient-oriented** コールバック関数は、単一のメッセージ受信側の処理に影響を与えます。
- **message-oriented** コールバック関数は、単一のメッセージに影響を与えます。
- **connection-oriented** コールバック関数は、接続全体 (この間に、複数のメッセージが複数のセットの受信側に送信される可能性があります) に影響を与えます。
- **xxfi_envrcpt** 関数は、受信者指向 (recipient-oriented) です。 **xxfi_conect** 関数、**xxfi_hello** 関数、および **xxfi_close** 関数は、コネクション指向 (connection-oriented) です。 その他のコールバック関数はすべて、メッセージ指向 (message-oriented) です。

表 31. コールバック関数	
項目	**説明**
SMFIS_CONTINUE	現行の接続、メッセージ、または受信側の処理を続行します。
SMFIS_REJECT	• **connection-oriented** ルーチンの場合、この接続をリジェクトして、**xxfi_close** を呼び出します。 • **message-oriented** ルーチン (**xxfi_eom** 関数と **xxfi_abort** 関数を除く) 場合、このメッセージをリジェクトします。 • **recipient-oriented** ルーチンの場合、現行の受信側をリジェクトします (ただし、現行メッセージの処理は続行します)。

表 *31.* コールバック関数 *(*続き*)*

項目	説明
SMFIS_DISCARD	• **message-oriented** ルーチンまたは **recipient-oriented** ルーチンの場合、このメッセージを受け入れますが、メッセージを破棄します。 • **connection-oriented** ルーチンは、SMFIS_DISCARD を返してはなりません。
SMFIS_ACCEPT	• **connection-oriented** ルーチンの場合、この接続を受け入れ、それ以上のフィルター処理は行わずに、**xxfi_close** 関数を呼び出します。 • **message-oriented** ルーチンまたは **recipient-oriented** ルーチンの場合、このメッセージを受け入れ、それ以上のフィルター処理は行いません。
SMFIS_TEMPFAIL	一時的な障害を返します。つまり、対応する Simple Mail Transfer Protocol (SMTP) コマンドが 4xx 状況コードを返します。 • **message-oriented** ルーチン (**xxfi_envfrom** 関数を除く) の場合、このメッセージは失敗します。 • **connection-oriented** ルーチンの場合、この接続は失敗し、**xxfi_close** 関数を呼び出します。 • **recipient-oriented** ルーチンの場合、現行の受信側のみ失敗し、メッセージの処理を続行します。
SMFIS_SKIP	このトランザクション内の以降の同じタイプのコールバックをスキップします。 現在、この戻り値は **xxfi_body** 関数のみに許可されています。 この戻り値は、**milter** パラメーターが判断を下すのに十分な本文のチャンクを受信済みである場合に使用できます。 ただし、この戻り値が使用されるのは、さらに **xxfi_eom** 関数からの呼び出しのみ許可されているメッセージ変更関数を呼び出したい場合です。 **注 : milter** パラメーターは、この動作をメール転送エージェント (MTA) と交渉する必要があります。 **milter** パラメーターは、プロトコル・アクション SMFIP_SKIP が使用可能であるかどうかを検査します。 プロトコル・アクション SMFIP_SKIP が使用可能な場合、**milter** パラメーターはそのアクションを要求する必要があります。

表 31. コールバック関数 (続き)	
項目	**説明**
SMFIS_NOREPLY	• MTA に応答を送信しません。 **milter** パラメーターは、この動作を MTA と交渉する必要があります。 **milter** パラメーターは、該当するプロトコル・アクション SMFIP_NR_* が使用可能であるかどうかを検査する必要があります。 プロトコル・アクション SMFIP_NR_* が使用可能な場合、 **milter** パラメーターはそのアクションを要求する必要があります。 • コールバックに対して SMFIP_NR_* プロトコル・アクションを設定した場合、そのコールバックは常に SMFIS_NOREPLY で応答する必要があります。 それ以外の応答コードを使用すると、アプリケーション・プログラミング・インターフェース (API) の違反になります。 ご使用のコールバックが、場合によっては別の値を返す可能性がある場合は (何らかのリソース不足により)、SMFIP_NR_* を設定してはならず、SMFIS_CONTINUE をデフォルトの戻り値として使用する必要があります。 代わりに、SMFIP_NR_* が設定されていない以降のコールバックまで、問題の報告を遅らせるようにすることもできます。

xxfi_connect コールバック関数

目的

xxfi_connect コールバック関数は、接続情報を提供します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_connect)(
        SMFICTX    *ctx,
        char       *hostname,
        _SOCK_ADDR *hostaddr);
```

説明

xxfi_connect コールバック関数は、各 Simple Mail Transfer Protocol (SMTP) 接続の開始時に 1 回呼び出され、SMFIS_CONTINUE フラグを返します。

注 : 以前のフィルターが **xxfi_connect** コールバック関数ルーチン内の接続をリジェクトしている場合、そのフィルターの **xxfi_connect** コールバック関数は呼び出されません。

引数

表 32. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** 内に維持されています。

表 32. 引数 (続き)	
項目	**説明**
hostname	ホスト・アドレスに対する逆検索によって判別された、メッセージ送信側のホスト名。 逆検索に失敗した場合、または解決されたホスト名の IP アドレスのどれも元の IP アドレスに一致しない場合、ホスト名 (hostname) にはメッセージ送信側の IP アドレスが入り、大括弧で囲って示されます (例えば、[a.b.c.d])。 Simple Mail Transfer Protocol (SMTP) 接続が stdin を介して行われている場合、値はローカル・ホスト (localhost) です。
hostaddr	SMTP ソケットに対する **getpeername(2)** 呼び出しによって判別された、ホスト・アドレス。 タイプが現行バージョンでサポートされていない場合、または **stdin** を介して SMTP 接続が行われている場合、値は NULL です。

xxfi_helo コールバック関数

目的

xxfi_helo コールバック関数は、**HELO または EHLO** コマンドを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_helo)(
    SMFICTX *ctx,
    char *helohost
);
```

説明

xxfi_hello コールバック関数は、クライアントが **HELO または EHLO** コマンドを送信するたびに呼び出され、SMFIS_CONTINUE フラグを返します。 したがって、コールバックは複数回呼び出されることも、呼び出されないこともあります。 メール転送エージェント (MTA) 構成によって、いくつかの制限が強制されることがあります。

引数

表 33. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
helohost	**HELO または EHLO** コマンドに渡される値は、送信側ホストのドメイン・ネームでなければなりません

xxfi_envfrom コールバック関数

目的

xxfi_envfrom コールバック関数は、**MAIL** (エンベロープ送信側) コマンドを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_envfrom)(
    SMFICTX *ctx,
    char **argv
);
```

説明

xxfi_envfrom コールバック関数は、クライアントが **DATA** コマンドを使用した場合に呼び出され、SMFIS_CONTINUE フラグを返します。

注 : ESMTP 応答について詳しくは、RFC 1869 を参照してください。

引数

表 *34.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
argv	ヌル終了の SMTP コマンド引数。`argv[0]` は送信側アドレスであることが保証されています。 以後の引数は、拡張 Simple Mail Transfer Protocol (ESMTP) 引数です。

戻り値

表 *35.* 戻り値

項目	説明
SMFIS_TEMPFAIL	送信側およびメッセージは一時エラーでリジェクトされます。新しい送信側 (および新しいメッセージ) が後で指定される可能性があるため、**xxfi_abort** コールバック関数は呼び出されません。
SMFIS_REJECT	送信側およびメッセージはリジェクトされます。新しい送信側およびメッセージが指定される可能性があるため、**xxfi_abort** コールバック関数は呼び出されません。
SMFIS_DISCARD	メッセージは受け入れられて破棄され、**xxfi_abort** コールバック関数は呼び出されません。
SMFIS_ACCEPT	メッセージは受け入れられ、**xxfi_abort** コールバック関数は呼び出されません。

関連情報

xxfi_abort

xxfi_envrcpt コールバック関数

目的

xxfi_envrcpt コールバック関数は、エンベロープ **RCPT** コマンドを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_envrcpt)(
    SMFICTX *ctx,
    char **argv
);
```

説明

xxfi_envrcpt コールバック関数は、受信側ごとに 1 回、さらに **xxfi_envfrom** コールバック関数の直後にメッセージごとに 1 回以上呼び出され、SMFIS_CONTINUE フラグを返します。

注 : 拡張 Simple Mail Transfer Protocol (ESMTP) 応答について詳しくは、 RFC 1869 を参照してください。

引数

表 *36.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
argv	ヌル終了の SMTP コマンド引数。`argv[0]` は受信側アドレスであることが保証されています。 以後の引数は、拡張 Simple Mail Transfer Protocol (ESMTP) 引数です。

戻り値

表 *37.* 戻り値

項目	説明
SMFIS_TEMPFAIL	受信側は一時的に失敗します。以後にまだ受信側が送信される可能性があり、**xxfi_abort** コールバック関数は呼び出されません。
SMFIS_REJECT	受信側はリジェクトされます。以後にまだ受信側が送信される可能性があり、**xxfi_abort** コールバック関数は呼び出されません。
SMFIS_DISCARD	メッセージは受け入れられるか破棄され、**xxfi_abort** コールバック関数が呼び出されます。
SMFIS_ACCEPT	受信側は受け入れられ、**xxfi_abort** コールバック関数は呼び出されません。

関連情報

xxfi_envfrom

xxfi_abort

xxfi_data コールバック関数

目的

xxfi_data コールバック関数は、**DATA** コマンドを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_data)(
    SMFICTX *ctx
);
```

説明

xxfi_data コールバック関数は、クライアントが **DATA** コマンドを使用する場合に呼び出され、SMFIS_CONTINUE フラグを返します。

注 : ESMTP 応答について詳しくは、RFC 1869 を参照してください。

引数

表 *38.* 引数

項目	説明
ctx	**libmilter** パラメーター内に維持されている、不透明なコンテキスト構造体。

戻り値

表 *39.* 戻り値

項目	説明
SMFIS_TEMPFAIL	メッセージは一時エラーでリジェクトされます。
SMFIS_REJECT	メッセージはリジェクトされます。
SMFIS_DISCARD	メッセージは受け入れられて、破棄されます。
SMFIS_ACCEPT	メッセージは受け入れられます。

xxfi_unknown コールバック関数

目的

xxfi_unknown コールバック関数は、不明および未実装の Simple Mail Transfer Protocol (SMTP) コマンドを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_unknown)(
    SMFICTX *ctx,
    const char *arg
);
```

説明

xxfi_unknown コールバック関数は、クライアントが使用した SMTP コマンドが、不明であるか、メール転送エージェント (MTA) によって実装されていない場合に呼び出され、SMFIS_CONTINUE フラグを返します。

注 : サーバーは常にその SMTP コマンドをリジェクトします。 異なるエラー・コードを返す可能性があるだけです。

引数

表 *40.* 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
arg	すべての引数を含めた SMTP コマンド。

戻り値

表 *41.* 戻り値	
項目	**説明**
SMFIS_TEMPFAIL	コマンドは一時エラーでリジェクトされます。
SMFIS_REJECT	コマンドはリジェクトされます。

xxfi_header コールバック関数

目的

xxfi_header コールバック関数は、メッセージ・ヘッダーを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_header)(
    SMFICTX *ctx ,
    char *headerf,
    char *headerv
);
```

説明

xxfi_header コールバック関数は、メッセージ・ヘッダーごとに 1 回呼び出され、SMFIS_CONTINUE フラグを返します。

注 :

- sendmail 8.14 以降、SMFIP_HDR_LEADSPC フラグを使用して要求された場合、ヘッダー・フィールド内のコロンの後のスペースが保持されます。 例えば、以下のヘッダーは、

  ```
  From: sender <f@example.com>
  To:  user <t@example.com>
  Subject:no
  ```

  以下のように milter パラメーターに送信されます。

  ```
  "From", " sender <f@example.com>"
  "To", "  user <t@example.com>"
  "Subject", "no"
  ```

  これに対して、以前は (あるいは、フラグ SMFIP_HDR_LEADSPC がない場合は)、以下のようになります。

  ```
  "From", "sender <f@example.com>"
  "To", "user <t@example.com>"
  "Subject", "no"
  ```

- 古いフィルターは、新しいフィルターに合わせてヘッダーを変更または追加します。
- ヘッダー・フォーマットについて詳しくは、RFC 822 および RFC 2822 を参照してください。

引数

表 42. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
headerf	ヘッダー・フィールド名。
headerv	ヘッダー・フィールド値。 ヘッダーの内容には、折り返された空白、つまり、後に空白を伴う複数の行が含まれることがあります。この場合、行は (CR または LF ではなく) LF で分離されます。 末尾の行終了文字 (CR または LF) は削除されます。

関連情報

RFC 2822

RFC 822

xxfi_eoh コールバック関数

目的

xxfi_eoh コールバック関数は、メッセージ・ヘッダーの終了を処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_eoh)(
    SMFICTX *ctx
);
```

説明

xxfi_eoh コールバック関数は、すべてのヘッダーの送信と処理が完了した後で 1 回呼び出され、SMFIS_CONTINUE フラグを返します。

引数

表 43. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。

xxfi_body コールバック関数

目的

xxfi_body コールバック関数は、メッセージ本文の断片を処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_body)(
    SMFICTX *ctx,
    unsigned char*bodyp ,
    size_t len
);
```

説明

xxfi_body コールバック関数は、**xxfi_eoh** コールバック関数と **xxfi_eom** コールバック関数の間に、呼び出されないことも、何度も呼び出されることもあり、SMFIS_CONTINUE フラグを返します。

注 :

- **bodyp** は、一連のバイトを指します。 これは C ストリング (「¥0」で終了する文字列) ではありません。 したがって、このバイト・ブロックでは、**strlen(3)** のような通常の C ストリング関数は使用しないでください。 バイト・シーケンスには、ブロックの内部に「¥0」文字が含まれている場合があります。 そのため、末尾に「¥0」が追加されても、C ストリング関数は予想どおりに機能しないことがあります。
- メッセージ本文は大きくなる可能性があるため、**xxfi_body** コールバック関数を定義すると、フィルターのパフォーマンスに大きな影響を与えることがあります。
- 行の終わりは、SMTP から受け取った通りに表されます (通常は、CR/LF)。
- 古いフィルターは、新しいフィルターに合わせて本文の変更を行います。
- メッセージ本文は、チャンクごとに **xxfi_body** コールバック関数の 1 つの呼び出しがある、複数のチャンクで送信される場合があります。
- milter パラメーターが判断を下すのに十分な多数の本文のチャンクを受信済みであるが、さらに **xxfi_eom** コールバック関数からの呼び出しのみ許可されるメッセージ変更関数を呼び出したい場合、この関数は SMFIS_SKIP フラグを返します。
- milter パラメーターは、この動作をメール転送エージェント (MTA) と交渉する必要があります。つまり、プロトコル・アクション SMFIP_SKIP フラグが使用可能かどうかを検査し、使用可能であれば、**milter** パラメーターはそのフラグを要求する必要があります。

引数

表 *44.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。
bodyp	本文データのこのブロックの先頭へのポインター。 *bodyp* は、この **xxfi_body** コールバック関数の外部では無効です。
len	*bodyp* によって指し示されるデータの量。

関連情報

xxfi_eoh

xxfi_eom

xxfi_eom コールバック関数

目的

xxfi_eom コールバック関数は、メッセージ終結を処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_eom)(
    SMFICTX *ctx
);
```

説明

xxfi_eom コールバック関数は、特定のメッセージに関するすべての **xxfi_body** コールバック関数呼び出しの後で 1 回呼び出され、SMFIS_CONTINUE フラグを返します。

注 : xxfi_eom コールバック関数内でメッセージ・ヘッダー、本文、およびエンベロープに対するすべての変更を行うには、フィルターが必要です。 変更は **smfi_*** ルーチンを使用して行われます。

引数

表 *45.* 引数

項目	説明
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。

関連情報

xxfi_body

xxfi_abort コールバック関数

目的

xxfi_abort コールバック関数は、アボート中の現行メッセージを処理します。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_abort)(
    SMFICTX *ctx
);
```

説明

xxfi_abort コールバック関数は、メッセージの処理中 (つまり、何らかのメッセージ指向ルーチンと **xxfi_eom** コールバック関数の間) の任意の時点で呼び出され、SMFIS_CONTINUE フラグを返します。

注 :

- **xxfi_abort** コールバック関数は、メッセージ単位で割り当てられたリソースをすべてレクラメーション処理する必要があります。また、任意の 2 つのメッセージ指向コールバック間に呼び出されることを許容できなければなりません。
- **xxfi_abort** コールバック関数と **xxfi_eom** コールバック関数の呼び出しは、相互に排他的であり、同時に使用することはできません。
- **xxfi_abort** コールバック関数は、接続固有のデータをレクラメーション処理する責任はありません。接続を閉じるときは、常に **xxfi_close** コールバック関数が呼び出されるからです。
- 現行メッセージは既にアボート中であるため、戻り値は現時点では無視されます。
- **xxfi_abort** コールバック関数が呼び出されるのは、メッセージがフィルターの 制御の外で打ち切られ、フィルターがそのメッセージ 指向の処理を完了していない場合に限られます。 例えば、フィルターが既にメッセージ指向のルーチンから SMFIS_ACCEPT フラグ、SMFIS_REJECT フラグ、または SMFIS_DISCARD フラグを返している場合は、 以後にそのメッセージが制御の外で打ち切られても、**xxfi_abort** コールバック関数は呼び出されません。

引数

表 46. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** 内に維持されています。

関連情報

xxfi_close

xxfi_eom

xxfi_close コールバック関数

目的

xxfi_close コールバック関数は、現行接続を閉じます。

構文

```
#include <libmilter/mfapi.h>
sfsistat (*xxfi_close)(
    SMFICTX *ctx
);
```

説明

xxfi_close コールバック関数は、常に各接続の最後に 1 回呼び出され、SMFIS_CONTINUE フラグを返します。

xxfi_close コールバック関数は、順不同で呼び出されることがあります。つまり、**xxfi_connect** コールバック関数が呼び出される前に呼び出されることもあります。 メール転送エージェント (MTA) によってフィルターへの接続が確立された後、MTA がこの接続のトラフィックを破棄することに決定した場合 (例えば、access_db 結果により)、クライアントが閉じるまで MTA からフィルターにデータは渡されません。クライアントが閉じた時点で、**xxfi_close** コールバック関数が呼び出されます。 したがって、このコールバックが、ある特定の接続に対して使用される唯一のコールバックになる可能性があり、**xxfi_close** コールバック関数コードの作成時に、この可能性を予測する必要があります。 特に、このコールバックでは、専用コンテキスト・ポインターが NULL 以外の値であると想定するのは誤りです。

xxfi_close コールバック関数は、前のメール・トランザクションが打ち切られた場合でも、クローズ時に呼び出されます。

xxfi_close コールバック関数は、接続単位で割り当てられたリソースをすべて解放する責任があります。

接続は既にクローズ中であるため、戻り値は現時点では無視されます。

引数

表 47. 引数	
項目	**説明**
ctx	この不透明なコンテキスト構造体は、**libmilter** パラメーター内に維持されています。

関連情報

xxfi_connect

xxfi_negotiate コールバック関数

目的

xxfi_negotiate コールバック関数は、ネゴシエーションを処理します。

構文

```
#include <libmilter/mfapi.h>
#include <libmilter/mfdef.h>
sfsistat (*xxfi_negotiate)(
        SMFICTX *ctx,
    unsigned long f0,
    unsigned long f1,
    unsigned long f2,
    unsigned long f3,
    unsigned long *pf0,
    unsigned long *pf1,
    unsigned long *pf2,
    unsigned long *pf3);
```

説明

xxfi_negotiate コールバック関数は、各 Simple Mail Transfer Protocol (SMTP) 接続の開始時に呼び出され、SMFIS_ALL_OPTS フラグを返します。

この関数を使用して、**milter** パラメーターは、始動中に操作やアクションを動的に判別して要求することができます。 旧バージョンでは、アクション (f0) は **smfiDesc** 構造体の「flags」フィールド内に固定されており、プロトコル・ステップ (f1) は、コールバックが定義されているかどうかを検査して暗黙的に導出されていました。 **milter** の新規バージョンの拡張機能により、**milter** パラメーターが旧メール転送エージェント (MTA) との対話時には使用不可である新規アクションを必要とする場合、こうした静的選択は機能しなくなります。 つまり、コールバックはネゴシエーション中にどの操作が使用可能であるかを判別し、必要なコールバック機能を、提供されるものの中から動的に選択できます。 一部の操作が使用不可である場合、**milter** パラメーターは旧モードにフォールバックすることも、セッションを停止してユーザーにアップグレードを要求することもできます。

プロトコル・ステップ

```
(f1, *pf1)
```

:

- SMFIP_RCPT_REJ: このビットを設定することにより、**milter** パラメーターは MTA に対して、ユーザーが不明であるために (あるいは、類似の理由で) リジェクトされた RCPT コマンドも送信する必要があることを要求できます。ただし、構文エラーのためにリジェクトされた関数は送信されません。 **milter** は、このプロトコル・ステップを要求した場合には、マクロ **{rcpt_mailer}:** を検査して、それがエラーに設定されていないか調べる必要があります。エラーに設定されている場合、受信側は MTA によってリジェクトされることになります。 その場合は通常、マクロ **{rcpt_host}** と **{rcpt_addr}** に、拡張された状況コードとエラー・テキストが入ります。
- SMFIP_SKIP は、MTA が SMFIS_SKIP 戻りコードを認識することを示します。
- SMFIP_NR_* は、MTA が SMFIS_NOREPLY 戻りコードを認識することを示します。 以下のように、さまざまなプロトコル・ステージ用のフラグがあります。
  - SMFIP_NR_CONN: 85 ページの『xxfi_connect コールバック関数』
  - SMFIP_NR_HELO: 86 ページの『xxfi_helo コールバック関数』
  - SMFIP_NR_MAIL: 86 ページの『xxfi_envfrom コールバック関数』
  - SMFIP_NR_RCPT: 87 ページの『xxfi_envrcpt コールバック関数』
  - SMFIP_NR_DATA: 88 ページの『xxfi_data コールバック関数』
  - SMFIP_NR_UNKN: 89 ページの『xxfi_unknown コールバック関数』
  - SMFIP_NR_EOH: 91 ページの『xxfi_eoh コールバック関数』

- SMFIP_NR_BODY: 91 ページの『xxfi_body コールバック関数』
- SMFIP_NR_HDR: 90 ページの『xxfi_header コールバック関数』

- SMFIP_HDR_LEADSPC フラグは、MTA が先行スペースをそのままにしてヘッダー値を送信できることを示します。 このプロトコル・ステップが要求されている場合、MTA は、ヘッダーが追加、挿入、または変更されるときにヘッダーに先行スペースを追加しません。
- MTA に対して、さまざまな SMTP ステージに関する情報を送信しないように指示することができます。これらのフラグは、SMFIP_NO* で始まります。
  - SMFIP_NOCONNECT: 85 ページの『xxfi_connect コールバック関数』
  - SMFIP_NOHELO: 90 ページの『xxfi_header コールバック関数』
  - SMFIP_NOMAIL: 86 ページの『xxfi_envfrom コールバック関数』
  - SMFIP_NORCPT: 87 ページの『xxfi_envrcpt コールバック関数』
  - SMFIP_NOBODY: 91 ページの『xxfi_body コールバック関数』
  - SMFIP_NOHDRS: 90 ページの『xxfi_header コールバック関数』
  - SMFIP_NOEOH: 91 ページの『xxfi_eoh コールバック関数』
  - SMFIP_NOUNKNOWN: 89 ページの『xxfi_unknown コールバック関数』
  - SMFIP_NODATA: 88 ページの『xxfi_data コールバック関数』

  `milter` パラメーターで使用されないこれらの **xxfi_*** コールバックについて、対応するフラグを以下に設定する必要があります。

```
*pf1.
```

使用可能なアクション

```
(f0, *pf0)
```

これらは **(xxfi_flags)** に記述されます。

`milter` が SMFIS_CONTINUE フラグを返す場合、`milter` は必要なアクションとプロトコル・ステップを (出力) パラメーター `pf0` と `pf1` (それぞれ、f0 と f1 に対応) を介して設定します。 (出力) パラメーター `pf2` と `pf3` は、将来のバージョンとの互換性のために、0 に設定する必要があります。

引数

表 *48.* 引数

項目	**説明**
ctx	**libmilter** パラメーター内に維持されている、不透明なコンテキスト構造体。
f0	MTA によって提供されるアクション。
f1	MTA によって提供されるプロトコル・ステップ。
f2	将来の拡張用。
f3	将来の拡張用。
pf0	**milter** によって要求されるアクション
pf1	**milter** によって要求されるプロトコル・ステップ。
pf2	将来の拡張用。
pf3	将来の拡張用。

戻り値

表 49. 戻り値	
項目	**説明**
SMFIS_ALL_OPTS	**milter** が使用可能なプロトコル・ステップおよびアクションを検査するだけである場合は、SMFIS_ALL_OPTS フラグを返すことができます。MTA は **milter** に対してすべてのプロトコル・ステップおよびアクションを使用可能にします。 この場合、出力パラメーター pf0 - pf3 は無視されるので、値を割り当てはなりません。
SMFIS_REJECT	**milter** の始動は失敗し、再交渉されることはありません (現行接続に関して)。
SMFIS_CONTINUE	処理を続行します。 この場合、**milter** パラメーターは、すべての出力パラメーター pf0 - pf3 を設定する必要があります。 これらの出力パラメーターの設定方法の説明は、以下を参照してください。

各種関数および定数関数

各種関数および定数関数は、**libmilter** パラメーターのバージョン情報を取り出します。

表 50. 定数関数	
項目	**説明**
smfi_version	**smfi_version** 関数は、**libmilter** パラメーターの (ランタイム) バージョン情報を取り出します。
smfi_setsymlist	**smfi_setsymlist** は、特定のプロトコル・ステージについて、**libmilter** パラメーターがメール転送エージェント (MTA) から受け取る必要があるマクロのリストを設定します。

表 51. 定数関数	
項目	**説明**
SMFI_VERSION	**SMFI_VERSION** は、**libmilter** パラメーターのランタイム・バージョンを取得します。

smfi_version 関数

目的

smfi_version 関数は、**libmilter** (ランタイム) バージョン情報を提供します。

構文

```
#include <libmilter/mfapi.h>
int smfi_version(
    unsigned int *pmajor,
    unsigned int *pminor,
    unsigned int *ppl
);
```

説明

smfi_version コールバック関数は、いつでも呼び出すことができます。

libmilter ライブラリーのコンパイル時バージョンは、SMFI_VERSION マクロ内にあります。 このマクロからメジャー・バージョン、マイナー・バージョン、および現行のパッチ・レベルを取り出すには、SM_LM_VRS_MAJOR(v) マクロ、SM_LM_VRS_MINOR(v) マクロ、および SM_LM_VRS_PLVL(v) マクロを使用できます。 **milter** パラメーターは、SMFI_VERSION マクロを検査して、(コンパイル時に C プリプロセッサー・ステートメントによって) どの関数を使用するかを判別します。 このマクロと **smfi_version** 関数を使用して、**milter** パラメーターは実行時に、それが予期された **libmilter** バージョンに (動的に) リンクされているかどうかを判別できます。 こうした関数は、メジャー・バージョンとマイナー・バージョンのみを比較する必要があり、パッチ・レベルを比較する必要はありません。つまり、**libmilter** ライブラリーはパッチ・レベルの相違に関係なく互換性があります。

引数

表 *52.* 引数

項目	説明
pmajor	メジャー・バージョン番号を保管するための符号なし int 変数へのポインター。
pminor	マイナー・バージョン番号を保管するための符号なし int 変数へのポインター。
ppl	パッチ・レベル番号を保管するための符号なし int 変数へのポインター。

戻り値

smfi_version 関数は、MI_SUCCESS 値を返します。

smfi_setsymlist 関数

目的

smfi_setsymlist 関数は、特定のプロトコル・ステージについて、**milter** パラメーターがメール転送エージェント (MTA) から受け取る必要があるマクロのリストを設定します。

構文

```
#include <libmilter/mfapi.h>
int smfi_setsymlist(
        SMFICTX    *ctx,
    int        stage,
    char     *macros
);
```

説明

smfi_setsymlist コールバック関数は、**xxfi_negotiate** 関数の処理中に呼び出す必要があります。この関数を使用して、**milter** パラメーターがメール転送エージェント (MTA) から受け取る必要があるマクロのリストをオーバーライドできます。

注 : 設定できるマクロの数に関する内部制限があります (現在は 5)。 ただし、この制限は **milter** パラメーターでは適用されず、MTA によってのみ適用されますが、この制限に違反していても **milter** パラメーターには通知されません。

引数

表 53. 引数	
項目	**説明**
ctx	**libmilter** パラメーター内に維持されている、不透明なコンテキスト構造体。
stage	マクロ・リストを使用する必要があるプロトコル・ステージ。 正しい値については、include/libmilter/mfapi.h ファイルを参照して、SMFIM_ prefix を持つ C マクロを見つけてください。 使用可能なプロトコル・ステージとしては、少なくとも、初期接続、HELO または EHLO、MAIL、RCPT、DATA、ヘッダーの終わり、およびメッセージの終わりがあります。
macros	マクロのリスト (スペースで区切られた)。 例えば、「**{rcpt_mailer} {rcpt_host}**」。

戻り値

smfi_setsymlist 関数は、以下の場合、MI_FAILURE 値を返します。 それ以外の場合、関数は MI_SUCCESS を返します。

- マクロ・リストのコピーを作成するための十分な空きメモリーがない。
- *macros* が NULL または空である。
- *stage* が有効なプロトコル・ステージではない。
- 特定のステージのマクロ・リストが前に設定されている。

関連情報

xxfi_negotiate

sendmail のデバッグ・フラグ

sendmail コマンドには多数のデバッグ・フラグが組み込まれています。

それぞれのデバッグ・フラグには、番号とレベルが指定され、レベルが高くなるに従って、出力される情報が多くなります。 慣例的に、9 を超えるレベルは出力される情報の量が非常に多いため、コードの特定の部分をデバッグする場合を除いて、あまり使用されません。 デバッグ・フラグは、次の例に示すように、**-d** フラグを使用して設定します。

```
debug-flag:     -d debug-list
debug-list:     debug-flag[.debug-flag]*
debug-flag:     debug-range[.debug-level]
debug-range:    integer|integer-integer
debug-level:    integer
```

```
-d12            Set flag 12 to level 1
-d12.3          Set flag 12 to level 3
-d3-17          Set flags 3 through 17 to level 1
-d3-17.4        Set flags 3 through 17 to level 4
```

使用可能なデバッグ・フラグは次のとおりです。

項目 **説明**

-d0 一般的なデバッグ。

-d1 送信情報を示す。

-d2 *finis*() で終了。

項目	説明
-d3	負荷平均の表示。
-d4	十分なディスク・スペース。
-d5	イベントを示す。
-d6	失敗したメールを示す。
-d7	キュー・ファイル名。
-d8	DNS ネーム・レゾリューション。
-d9	RFC1413 照会をトレースする。
-d9.1	ホスト名を正規化する。
-d10	受信側送達を示す。
-d11	送達をトレースする。
-d12	相対ホストのマッピングを示す。
-d13	送達を示す。
-d14	ヘッダー・フィールドのコンマを示す。
-d15	ネットワーク取得要求アクティビティーを 示す。
-d16	発信接続。
-d17	MX ホストをリストする。

注: 現在、**sendmail** にはおよそ 200 の定義済みデバッグ・フラグが存在します。

Internet Message Access Protocol と Post Office Protocol

AIX では、次の 2 つのインターネット・ベースのメール・プロトコル・サーバーのインプリメンテーションを使用して、リモートでメールにアクセスできます。

- **Post Office Protocol** (**POP** または **POP3DS**)
- **Internet Message Access Protocol** (**IMAP** または **IMAPDS**)

それぞれのタイプのサーバーは、電子メッセージを保管し、電子メッセージへのアクセスを提供します。サーバー上でこれらのメール・アクセス・プロトコルを使用すると、メールを受信するためにコンピューターを常時稼働させておく必要がなくなります。

POP または **POP3DS** サーバーはオフラインのメール・システムを提供します。これによって、クライアントは、**POP** または **POP3DS** クライアント・ソフトウェアを使用して、リモートでメール・サーバーをアクセスし、メール・メッセージを取り出すことができます。 クライアントは、ダウンロードしたメール・メッセージを直ちにサーバーから削除することも、または、そのメッセージを **POP** または **POP3DS** サーバーに入れたまま残しておくこともできます。 メールがクライアント・マシンへダウンロードされると、すべてのメール処理はそのクライアントのローカル処理として行われます。 **POP** サーバーにより、このユーザーのメールボックスへのアクセスが許可されるのは、一度に 1 つのクライアントだけです。 **POP3DS** バージョンは OpenSSL ライブラリーを使用しますが、これは セキュリティー証明書を必要とします。

IMAP または **IMAPDS** サーバーは **POP** 機能のスーパーセットを提供しますが、異なるインターフェースを備えています。 **IMAP** または **IMAPDS** サーバーは、オンライン・サービスと切断サービスだけでなく、オフライン・サービスも提供します。 プロトコルは、リモート・メールボックスをローカル・メール・ボックスとまったく同じように操作することが許可される設計になっています。 例えば、 クライアントは、「**deleted**」または「**answered**」などの状態フラグを使用してメッセージの検索とマーク付けができます。また、メッセージを明示的に除去するまで、サーバーのデータベースの中に残しておくこともできます。さらに **IMAP** サーバーでは、同時に複数のクライアントがユーザー・メールボックスへ対話式にアクセスできます。 **IMAPDS** バージョンは OpenSSL ライブラリーを使用しますが、これは セキュリティー証明書を必要とします。

各サーバー・タイプは、メール・アクセスにのみ使用されます。 これらのサーバーは、**シンプル・メール転送プロトコル (SMTP)** を使用してメールを送信します。

各プロトコルはオープン・プロトコルであり、RFC に記述された標準を基礎としています。 **IMAP** サーバーは、RFC 2060 と 2061 を基礎とし、**POP** サーバーは RFC 1939 を基礎としています。 どちらも、TCP ソケットを使用したコネクション指向のサーバーです。 **IMAP** サーバーはポート 143 で listen を行い、**IMAPDS** サーバーはポート 993 で listen を行います。 **POP** サーバーはポート 110 で listen を行い、**POP3DS** サーバーはポート 995 で listen を行います。 どのサーバーも、**inetd** デーモンによって処理されます。

要件 : OpenSSL バージョンを使用するには、OpenSSL をインストールする必要があります。 OpenSSL は *AIX Toolbox for Linux Applications* CD に入っています。

IMAP サーバーと POP サーバーの構成

IMAP サーバーと **POP** サーバーを構成するには、以下の手順を使用します。

このタスクを実行するには、root 権限が必要です。

1. /etc/inetd.conf ファイル内の **imapd** または **imapds** および **pop3d** または **pop3ds** 構成エントリーのコメントを外します。

   以下に構成エントリーの例を示します。

   ```
   #imap2  stream  tcp     nowait  root    /usr/sbin/imapd imapd
   #pop3   stream  tcp     nowait  root    /usr/sbin/pop3d pop3d
   #imaps  stream  tcp     nowait  root    /usr/sbin/imapds imapds
   #pop3s   stream  tcp     nowait  root   /usr/sbin/pop3ds pop3ds
   ```

2. /etc/imapd.cf ファイル内の **imapds** サーバーの構成ファイルおよび /etc/pop3d.cf ファイル内の **pop3ds** サーバーの構成ファイルを設定します。

   デフォルトでは、**imapds** サーバーおよび **pop3ds** サーバーに対して、機密保護機能の低い安全保護ハンドシェーク・プロトコルである Secure Sockets Layer バージョン 2 (SSLv2) と SSLv3 が有効になります。 ただし、以下の例に示すように構成ファイルを更新することで、SSLv2 と SSLv3 を無効にできます。 構成ファイルに SSL_CIPHER_LIST ストリングを指定することによって、暗号を有効にしたり無効にしたりすることもできます。 このオプションは、アプリケーションでハードコーディングされているデフォルトの暗号ストリングを上書きします。

   imapds サーバーの構成ファイル (/etc/imapd.cf):

   ```
       #============================================================
       #
       # Sample IMAP Server Configuration File
       #
       #============================================================
       #============================================================
       # Uncomment the line below to Disable SSL v2 for the imap server.
       #
       #   Disable SSL V2  --->  SSL_OP_NO_SSLv2         YES
       #   Allow SSL V2    --->  SSL_OP_NO_SSLv2         NO
       #
       #
       #SSL_OP_NO_SSLv2         YES  <-------------- uncomment this line to disable sslv2
       #============================================================
       # Uncomment the line below to Disable SSL v3 for the imap server.
       #
       #   Disable SSL V3  --->  SSL_OP_NO_SSLv3         YES
       #   Allow SSL V3    --->  SSL_OP_NO_SSLv3         NO
       #
       #
       #SSL_OP_NO_SSLv3         YES  <-------------- uncomment this line to disable sslv3
       #============================================================
       # Uncomment the line below to use the user provided cipher list
       # for the imap server. Parser logic expect Cipher string within " ".
       #
       #
       #SSL_CIPHER_LIST "ALL:!LOW" <--- uncomment this line to customized (enable/disabled)
   ciphers string
       #============================================================
   ```

pop3ds サーバーの構成ファイル (/etc/pop3d.cf):

```
    #====================================================================
    #
    # Sample POP3 Server Configuration File
    #
    #====================================================================
    #====================================================================
    # Uncomment the line below to Disable SSL v2 for the pop3d server.
    #
    #   Disable SSL V2  --->  SSL_OP_NO_SSLv2         YES
    #   Allow SSL V2    --->  SSL_OP_NO_SSLv2         NO
    #
    #
    #SSL_OP_NO_SSLv2         YES  <----------- uncomment this line to disable sslv2
    #====================================================================
    # Uncomment the line below to Disable SSL v3 for the pop3d server.
    #
    #   Disable SSL V3  --->  SSL_OP_NO_SSLv3         YES
    #   Allow SSL V3    --->  SSL_OP_NO_SSLv3         NO
    #
    #
    #SSL_OP_NO_SSLv3         YES  <----------- uncomment this line to disable sslv3
    #====================================================================
    # Uncomment the line below to use the user provided cipher list
    # for the pop3d server. Parser logic expect Cipher string within " ".
    #
    #
    #SSL_CIPHER_LIST "ALL:!LOW" <---- uncomment this line to customized (enable/disabled)
ciphers string
    #====================================================================
```

3. 次のコマンドを実行して、**inetd** デーモンをリフレッシュします。

```
refresh -s inetd
```

構成テストの実行

いくつかのテストを行って、サーバーが作動可能な状態にあるか確認します。

1. 最初に、サーバーがポート上で listen を行っているかどうかを検査します。

   これを行うには、コマンド・プロンプトに応じて次のコマンドを入力します。その際、コマンドを入力するたびに Enter キーを押してください。

   ```
   netstat -a | grep imap
   netstat -a | grep pop
   ```

   netstat コマンドからの出力は次のとおりです。

   ```
   tcp     0       0     *.imap2         *.*        LISTEN
   tcp     0       0     *.imaps         *.*        LISTEN
   tcp     0       0     *.pop3          *.*        LISTEN
   tcp     0       0     *.pop3s         *.*        LISTEN
   ```

2. これと似た出力を受信しなかった場合は、/etc/inetd.conf ファイル内 のエントリーを再検査して、**refresh -s inetd** コマンドを 再実行してください。
3. imapd サーバーの構成をテストするには、Telnet ポート 143 (IMAPDS の場合は、Telnet ポート 993) を使用して imap2 サーバーにアクセスします。

   Telnet を使用して接続すると、imapd プロンプトが表示されます。 この時点で、RFC 1730 で定義されている IMAP バージョン 4 コマンドを入力できます。 コマンドを実行するには、ピリオド (.) を入力し、その後にスペースを 1 つ入れてから、トークン、 コマンド名、およびパラメーターを入力します。 トークンはコマンド名の順序付けに使用します。 例えば、次のとおりです。

   ```
   . token CommandName parameters
   ```

   imapd サーバーに Telnet でログインすると、パスワードがエコーされます。

   次の Telnet の例では、**login** コマンドの *id_password* という場所にパスワードを指定しなければなりません。

ヒント : IMAPDS の場合は、コマンドと出力が少し異なります。

```
telnet e-xbelize 143
Trying...
Connected to e-xbelize.austin.ibm.com.
Escape character is '^]'.
* OK e-xbelize.austin.ibm.com IMAP4 server ready
. 1 login id id_password
. OK
. 2 examine /usr/spool/mail/root
* FLAGS (¥Answered ¥Flagged ¥Draft ¥Deleted ¥Seen)
* OK [PERMANENTFLAGS (¥Answered ¥Flagged ¥Draft ¥Deleted ¥Seen ¥*)]
* 0 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 823888143]
. OK [READ-ONLY] Examine completed
. 3 logout
* BYE Server terminating connection
. OK Logout completed
Connection closed.
```

4. pop3d サーバーの構成をテストするには、Telnet を使用して POP3 ポート 110 (POP3DS の場合は、Telnet ポート 995) にアクセスします。

   Telnet を使用して接続すると、pop3d プロンプトが表示されます。 ここでは、RFC 1725 で定義された POP コマンドを入力できます。 そのいずれかのコマンドを実行するには、ピリオド (.) を入力し、そのあとにスペースを 1 つ入れてからコマンド名を入力します。 例えば、次のとおりです。

   ```
   . CommandName
   ```

   pop3d サーバーに Telnet でログインすると、パスワードがエコーされます。

   次の Telnet の例では、**pass** コマンドの *id_password* という場所にパスワードを指定しなければなりません。

   ヒント : POP3DS の場合は、コマンドと出力が少し異なります。

   ```
   telnet e-xbelize 110
   Trying...
   Connected to e-xbelize.austin.ibm.com.
   Escape character is '^]'.
   +OK e-xbelize.austin.ibm.com POP3 server ready
   user id
   +OK Name is a valid mailbox
   pass id_password
   +OK Maildrop locked and ready
   list
   +OK scan listing follows
   .
   stat
   +OK 0 0
   quit
   +OK
   Connection closed.
   ```

SYSLOG 機能を使用したロギング

IMAP (および **IMAPDS**) と **POP** (および **POP3DS**) サーバー・ソフトウェアは、ログ・メッセージを **SYSLOG** 機能に送信します。

1. **SYSLOG** 機能を使用して **IMAP** と **POP** のロギング用に システムを構成するには、root ユーザーでなければなりません。 `/etc/syslog.conf` ファイルを編集し、次のように ***.debug** のエントリーを追加します。

   ```
   *.debug /usr/adm/imapd.log
   ```

2. **syslogd** デーモンが `/etc/syslog.conf` 構成ファイルを読み取る前に、`usr/adm/imapd.log` ファイルが存在している必要があります。 このファイルを作成するには、コマンド・ライン・プロンプトに応じて次のように入力してから、Enter キーを押します。

   ```
   touch /usr/adm/imapd.log
   ```

3. **syslogd** デーモンをリフレッシュして、その構成ファイルを再び読みます。 コマンド・ライン・プロンプトに応じて次のように入力してから、Enter キーを押します。

```
refresh -s syslogd
```

メール管理コマンド

ここでは、メール管理コマンドを要約します。

項目	説明
bugfiler	特定のメール・ディレクトリーにバグ・レポートを保管します。
comsat	ユーザーにメールが着信したことを通知します (デーモン)。
mailq	メール・キューの内容を表示します。
mailstats	メールのトラフィックに関する統計情報を表示します。
newaliases	/etc/mail/aliases ファイルから別名データベースの新規コピーを構築します。
rmail	基本ネットワーク・ユーティリティー (BNU) の **uucp** コマンドによって受信したリモート・メールを処理します。
sendbug	特定のアドレスへシステム・バグ・レポートをメールします。
sendmail	ローカルな宛先またはネットワーク上の宛先にメールを経路指定します。
smdemon.cleanu	定期的にハウスキーピングを行うために、**sendmail** キューをクリーンアップします。

メールのファイルとディレクトリー

メールのファイルとディレクトリーは、機能ごとに配置されます。

項目	説明
/usr/share/lib/Mail.rc	メール・プログラムのすべてのユーザーに対してローカル・システムのデフォルト 値を設定します。 修正して **mail** コマンドのデフォルトの特性を設定できるテキスト・ファイルです。
$HOME/.mailrc	ユーザーがローカル・システムのメール機能のデフォルト値を変更できるようにします。
$HOME/mbox	個々のユーザーの処理済みメールを保管します。
/usr/bin/Mail、/usr/bin/mail、または /usr/bin/mailx	同一プログラムへリンクされた 3 つの名前を指定します。 メール・プログラムは、メール・システムへのユーザー・インターフェースの *1* つ です。
/var/spool/mail	デフォルトのメール・ドロップ・ディレクトリーを指定します。 デフォルトでは、すべてのメールが /var/spool/mail/UserName ファイルへ送達されます。
/usr/bin/bellmail	ローカルなメール送達を実行します。
/usr/bin/rmail	BNU 用のリモート・メール・インターフェースを 実行します。
/var/spool/clientmqueue	メール・キュー内にありローカル側で生成されたアウトバウンド・メッセージに関連付けられているログ・ファイルおよび一時ファイルが保持されています。

項目	説明
/var/spool/mqueue	メール・キュー内のメッセージに関連するログ・ファイルおよび一時ファイルが入っています。

項目	説明
/usr/sbin/sendmail	**sendmail** コマンド
/usr/ucb/mailq	/usr/sbin/sendmail へリンクします。 mailq の使用は、/usr/sbin/sendmail -bp コマンドの使用と同じ機能を実行できます。
/usr/ucb/newaliases	/usr/sbin/sendmail ファイルへリンクします。 newaliases の使用は、/usr/sbin/sendmail -bi コマンドの使用と同じ機能を実行できます。
/etc/netsvc.conf	特定のネーム・レゾリューション・サービスの順序を指定します。
/usr/sbin/mailstats	/etc/sendmail.st ファイルが存在する場合に、そのファイル内の **sendmail** の統計をフォーマット設定して表示します。 /etc/sendmail.st ファイルがデフォルトのファイルですが、別のファイルを指定することもできます。
/etc/mail/aliases	**sendmail** コマンド用のテキスト・フォーマットの別名ファイルを示します。 このファイルを編集して、システム用の別名を作成、変更、または削除できます。
/etc/aliasesDB	別名データベース・ファイルの DB.dir と DB.pag が入っているディレクトリーを示します。これらのデータベース・ファイルは、**sendmail -bi** コマンドを実行したときに、/etc/mail/aliases ファイルから作成されます。
/etc/mail/sendmail.cf	テキスト・フォーマットの **sendmail** 構成情報が入っています。 このファイルを編集してこの情報を変更できます。
/usr/lib/smdemon.cleanu	シェル・ファイルを示します。このシェル・ファイルはメール・キューを動作させ、**sendmail** ログ・ファイルを /var/spool/mqueue ディレクトリー内で保守します。
/etc/mail/statistics	メールのトラフィックに関する統計情報を収集します。 このファイルは、サイズが大きくなることはありません。 このファイルの内容を表示するには、/usr/sbin/mailstats コマンドを使用します。 この情報を収集したくない場合は、このファイルを削除してください。
/var/spool/clientmqueue	一時ファイルが入ったディレクトリーが示されます。 このディレクトリーには、メール・キュー内にありローカル側で生成されたアウトバウンド E メールが含まれています。 一時ファイルとしては、例えば、システム E メールなどがあります。
/var/spool/mqueue	キュー内の個々のメッセージに関連する一時ファイルが入ったディレクトリーを示します。 このディレクトリーには、ログ・ファイルを入れることもできます。
/var/spool/cron/crontabs	開始するジョブを判別するために **cron** デーモンが読み取るファイルが入っているディレクトリーを示します。 root ファイルには、smdemon.cleanu シェル・スクリプトを始動する行が入っています。

IMAP と POP のコマンド

IMAP と POP には、**imapd** と **pop3d** のメール・コマンドが使用されます。

項目	説明
/usr/sbin/**imapd**	IMAP (インターネット・メッセージ・アクセス・プロトコル) のサーバー・プロセス
/usr/sbin/**pop3d**	POP3 (Post Office Protocol バージョン 3) のサーバー・プロセス

伝送制御プロトコル/インターネット・プロトコル

コンピューターが相互に通信するときは、一定の規則、 つまりプロトコル により、データを規則正しく送受信できます。 世界で最も普及しているプロトコル・セットは、**伝送制御プロトコル/インターネット・プロトコル (TCP/IP)** です。 (ただし、ヨーロッパの大部分では、**X.25** プロトコルが使用されています。) **TCP/IP** を使用する場合の共通の機能には、電子メール、コンピューター間のファイル転送、リモート・ログインがあります。

mail ユーザー・コマンド、メッセージ処理 (MH) のユーザー・コマンド、**sendmail** サーバー・コマンドは、システム間のメール送受信に **TCP/IP** を使用できます。また、基本ネットワーク・ユーティリティー (BNU) は、システム間のファイルとコマンドの送受信に **TCP/IP** を使用できます。

TCP/IP はコンピューター間の通信規格を指定し、ネットワークの経路指定と相互接続に関する規約を詳細に記述するプロトコル群のことです。 TCP/IP はインターネットを通じて広範囲に使用されているため、研究機関、大学、政府、企業の相互間で通信を行えるようにしています。

TCP/IP はネットワーク上で接続された複数のコンピューター (ホスト) 間の通信を可能にします。 各ネットワークは別のネットワークに接続して、 そのネットワーク上のホストと通信できます。 ネットワークのテクノロジーには多くの種類があり、その大部分はパケット交換およびストリーム転送により作動しますが、**TCP/IP** にはハードウェアに依存しないという大きな長所があります。

インターネット・プロトコルは送信単位を定義し、その送信方法を指定するので、**TCP/IP** によりネットワーク・ハードウェアの細部を意識しないで、多くのタイプのネットワーク・テクノロジーによる接続と情報交換が可能になります。 IP アドレスにより、ネットワーク上のすべてのコンピューターが同一ネットワーク上の他のコンピューターと通信できます。 また、**TCP/IP** はユーザーが必要とする多数の通信サービスの規格を提供します。

TCP/IP が提供する機能により、コンピューター・システムはネットワークに接続可能なインターネット・ホストになり、他のインターネット・ホストと通信できるようになります。 **TCP/IP** には次に示す操作を可能にするコマンドと機能が含まれています。

- システム間のファイル転送
- リモート・システムへのログイン
- リモート・システム上でのコマンドの実行
- リモート・システム上でのファイルの印刷
- リモート・ユーザーへの電子メールの送信
- リモート・ユーザーとの会話
- ネットワークの管理

> **注: TCP/IP** は基本的なネットワーク管理機能を提供します。 **シンプル・ネットワーク管理プロトコル (SNMP)** は、その他の多数のネットワーク管理コマンドと機能を提供します。

TCP/IP 用語

TCP/IP に関連して使用されている次のようなインターネット用語を理解しておくと役立ちます。

項目	説明
クライアント	ネットワーク上の別のコンピューターまたはプロセスのデータ、サービス、 リソースにアクセスするコンピューターまたはプロセス。

項目	説明
ホスト	インターネット・ネットワークに接続され、他のインターネット・ホストと通信可能なコンピューター。 ユーザーのローカル・ホスト とは、そのユーザーが作業をしているコンピューターのことです。 外部ホスト とは、ネットワーク上の他のホスト名です。 通信ネットワークから見ると、 ホストはパケットのソースと 宛先を兼ねています。 どのホストもクライアント、サーバー、またはその両方になることができます。 インターネット・ネットワーク上では、ホストはインターネット名とアドレスで識別されます。
ネットワーク	2 つ以上のホストの組み合わせおよびそのホスト間の接続リンク。 物理ネットワーク とは、ネットワークを構成するハードウェアのことです。 論理ネットワーク とは、1 つ以上の物理ネットワークの全体またはその一部にまたがる抽象的な構成のことです。 インターネット・ネットワークは、論理ネットワークの一例です。 インターフェース ・プログラムは、論理ネットワーク操作から物理ネットワーク操作への変換を処理します。
パケット	ホストとそのネットワーク間のトランザクションのための制御情報とデータのブロック。 パケットとは、プロセスがインターネット・ネットワークを通じてデータを送受信するのに使用する交換メディアです。 パケットは、ソース から宛先 に送信されます。
ポート	プロセスの論理的な接続点。 データはポート (またはソケット) を通じてプロセス間で転送されます。 各ポートは、データ送受信用のキューを提供します。 インターフェース ・プログラム・ネットワークでは、 各ポートにその使用法に基づくインターネット・ポート番号があります。 各ポートは、 インターネット・ホスト・アドレスとポート番号の組み合わせであるインターネット・ソケット・アドレス で識別されます。
プロセス	実行中のプログラム。 プロセスとは、コンピューター内のアクティブな 状況のエレメントのことです。 端末、ファイル、および他の入出力装置は、プロセスを通じて相互に通信します。 したがって、ネットワーク通信とはプロセス間通信 (すなわち、プロセス同士の通信) のことです。
プロトコル	物理レベルまたは論理レベルで通信を処理するための一連の規則。 多くの場合、プロトコルは他のプロトコルを使用してサービスを提供します。 例えば、接続レベル・プロトコルは転送レベル・プロトコル を使用して、2 つのホスト間の接続を維持するパケットを転送します。
サーバー	ネットワーク上の他のコンピューターまたはプロセスがアクセスできるデータ、サービス、リソースを提供するコンピューターまたはプロセス。

TCP/IP ネットワークの計画

TCP/IP は非常に柔軟なネットワーク・ツールであるため、ユーザーの特定の用途に合わせて、さまざまな方法で TCP/IP をカスタマイズできます。 ネットワークを計画する際は、このトピックの主要な問題を検討してください。 これらの問題の詳細については、他のトピックで説明しています。 ここでは、事項を単に紹介するだけです。

1. 使用するネットワーク・ハードウェアのタイプ (トークンリング、イーサネット・バージョン 2、IEEE 802.3、FDDI、光シリアル・チャネル (SOC)、またはシリアル・ライン・インターフェース ・プロトコル (SLIP)) を決定します。
2. ネットワークの物理的なレイアウトを計画します。

   各ホスト・マシンがサービスする機能について考慮します。 例えば、ネットワークにケーブルを接続する前に、ゲートウェイとして働くマシンを決定する必要があります。
3. フラット・ネットワーク編成と階層 ネットワーク編成のどちらがユーザーの用途に適しているかを決定します。

   1 つの物理ネットワークで構成され、サイトが 1 つの小規模なネットワークでは、多くの場合、フラット・ネットワークが適しています。 複数の物理ネットワークで構成され、サイトが複数の、非常に大規模なまたは複雑なネットワークでは、階層ネットワークの方が効率的な場合があります。
4. ネットワークを他のネットワークに接続する場合は、ゲートウェイをどのようにセットアップし、構成するかを計画する必要があります。

   次のことを考慮してください。

a. ゲートウェイとして働くマシンを決定します。
b. 静的と動的のどちらの経路指定を使用するか、または両方を組み合わせて使用するのかを決定します。 動的経路指定を選択する場合は、サポートする通信プロトコルのタイプに合わせて、各ゲートウェイが使用する経路指定デーモンを決定します。

5. アドレッシング方式を決定します。

   そのネットワークが大規模なインターネットワークの一部でない場合は、用途に最適のアドレッシング方式を選択します。 インターネットのような大規模なインターネットワークに接続するネットワークの場合は、インターネット・サービス・プロバイダー (ISP) から公認アドレス・セットを入手する必要があります。
6. システムをサブネットに分割する必要があるかどうかを決定します。 分割が必要な場合は、サブネット・マスクの割り当て方法を決定します。
7. 命名方式を決定します。 ネットワーク上の個々のマシンには、固有なホスト名が必要です。
8. ネーム・レゾリューションのためのネーム・サーバーがネットワークに必要かどうか、あるいは `/etc/hosts` ファイルを使用するだけで十分かどうかを決定します。

   ネーム・サーバーを使用する場合は、ネットワークを効率的に機能させるために必要なネーム・サーバーのタイプと数を考慮します。
9. ネットワークがリモート・ユーザーに提供するサービスのタイプを決定します。例えば、メール・サービス、印刷サービス、ファイル共用、リモート・ログイン、リモート・コマンドの実行などです。

TCP/IP のインストール

TCP/IP ネットワークを構成するために必要なソフトウェアは、基本オペレーティング・システムのインストール時に、一緒にインストールされています。 TCP/IP ネットワークは、特定のパッケージのインストールをさらに必要とすることはありません。

基本オペレーティング・システム のインストールについて詳しくは、『インストールおよび移行』を参照してください。

TCP/IP の構成

AIX の基本オペレーティング・システムをインストールした後、システム用の TCP/IP ネットワークの構成を開始できます。

いくつもの **TCP/IP** 構成タスクを、以下の複数の方法で実行できます。

- システム管理インターフェース・ツール (SMIT) を使用する。
- ファイルの形式を編集する。
- シェル・プロンプトからコマンドを入力する。

例えば、`rc.net` シェル・スクリプトはシステム始動プロセス中に、**TCP/IP** に必要な最小限のホスト構成を実行します (`rc.net` スクリプトはブートの第 2 フェーズで構成マネージャー・プログラムによって実行されます)。 SMIT を使用してホスト構成を実行すると、`rc.net` ファイルが自動的に構成されます。

他の方法としては、標準テキスト・エディターを使用して `/etc/rc.bsdnet` ファイルを構成できます。この方法で、従来の UNIX **TCP/IP** 構成コマンド (**ifconfig**、**hostname**、**route** など) を指定できます。ファイル編集の方法を使用する場合は、`smit configtcp` 高速パスを入力してから、「**BSD Style rc Configuration (BSD スタイルの rc 構成)**」を選択する必要があります。 **TCP/IP** ファイルおよびファイル・フォーマットについては、*Communications Programming Concepts* の List of TCP/IP Programming References を参照してください。

ネーム・サーバーの構成など、一部のタスクは SMIT では実行できません。

ネットワークを構成する際は、以下の手順をガイドとして使用してください。 必ず、適切な資料を読んで理解しておくようにしてください。

以下の手順を開始する前に、次の前提条件を満たしていることを確認してください。

1. ネットワーク・ハードウェアがインストールされ、ケーブルで接続されていること。 ハードウェアのインストールとケーブル接続の詳細については、166 ページの『TCP/IP ローカル・エリア・ネットワーク・アダプター・カード』を参照してください。
2. **TCP/IP** ソフトウェアがインストールされていること。 TCP/IP ソフトウェアのインストールの詳細については、インストールおよび移行を参照してください。

ネットワークが正常に稼働している場合でも、このチェックリストはすべてのネットワークに起こりうる問題をデバッグするための参考として役立ちます。

TCP/IP ネットワークを構成する手順は、次のとおりです。

1. **TCP/IP** の基本編成について理解するために、127 ページの『TCP/IP プロトコル』を読みます。
   以下について理解する必要があります。
   - **TCP/IP** の階層構造 (すなわち、種々のプロトコルが種々の層に存在すること)。
   - 各層の間をデータがどのように流れるか。
2. ネットワーク上の各ホスト・マシンを最小限の構成にします。
   すなわち、個々のホストにネットワーク・アダプターを追加し、IP アドレスとホスト名を割り当てるほか、ネットワークへのデフォルトの経路を定義します。 これらのタスクの背景情報については、169 ページの『TCP/IP ネットワーク・インターフェース』、175 ページの『TCP/IP アドレッシング』、および 181 ページの『ネットワーク内のホストの命名』を参照してください。

   注 : ネットワーク上の個々のマシンには、そのマシンがエンド・ユーザー・ホスト、ファイル・サーバー、ゲートウェイ、ネーム・サーバーのいずれであっても、この基本構成操作が必要です。
3. ネットワーク上の個々のホスト・マシン上で **inetd** デーモンを構成し、始動します。 407 ページの『TCP/IP デーモン』を読んでから、408 ページの『inetd デーモンの構成』の指示に従ってください。
4. 各ホスト・マシンを、ローカル名レゾリューションを実行するよう、あるいは、ネーム・サーバーを使用するように構成します。
   階層状のドメイン名ネットワークをセットアップする場合は、少なくとも 1 つのホストがネーム・サーバーとして機能するよう構成してください。 184 ページの『ネーム・レゾリューション』 を読んで、その方法に従ってください。
5. リモート・ネットワークと通信するネットワークの場合は、最低 1 つのホストをゲートウェイとして機能するよう構成してください。
   ゲートウェイは、静的経路を使用するか経路指定デーモンを使用してインターネットワーク経路指定を実行できます。 410 ページの『TCP/IP 経路指定』 を読んで、その方法に従ってください。
6. ネットワーク上の各ホスト・マシンが使用するサービスを決定します。
   デフォルトでは、すべてのサービスを使用できます。 特定のサービスを使用不可にしたい場合は、409 ページの『クライアント・ネットワーク・サービス』の方法に従ってください。
7. ネットワーク上でサーバーとして使用するホストと、特定のサーバーが提供するサービスを決定します。
   409 ページの『サーバー・ネットワーク・サービス』の手順に従って、実行したいサーバー・デーモンを始動してください。
8. 必要となるリモート印刷サーバーを構成します。
   詳しくは、Printing administration を参照してください。
9. **オプション :** 必要に応じて、ネットワーク用のマスター・タイム・サーバーとして使用するか、あるいはマスター・タイム・サーバーとして機能するようにホストを構成します。
   詳しくは、**timed** デーモンを参照してください。

ホストの構成

ネットワーク上の各ホスト・マシンは、エンド・ユーザーおよびネットワーク全体のニーズに応じて機能するように構成する必要があります。

ネットワーク上にある個々のホストについて、ネットワーク・インターフェースを 構成し、IP アドレスとホスト名を設定しなければなりません。 また、ゲートウェイやその他のホストへの静的経路をセットアップし、デフォルトで始動するデーモンを指定して、ネーム・レゾリューションのための /etc/hosts ファ

イルをセットアップする (または、ネーム・レゾリューションのためにネーム・サーバーを使用するホストをセットアップする) ことも必要です。

サーバー構成のホスト
ホスト・マシンにゲートウェイ、ファイル・サーバー、またはネーム・サーバーとしてサービスさせるなど、特定の機能を実行させたい場合は、基本構成の完了後に必要な構成タスクを実行する必要があります。

例えば、階層状に編成されたネットワークで、**ドメイン名**プロトコルを使用して名前をインターネット・アドレスに変換したい場合は、ネットワークにこの機能を提供するために、少なくとも 1 つのネーム・サーバーを構成する必要があります。

サーバー・ホストは専用マシンである必要はなく、他の用途にも使用できます。 例えば、ネットワークのネーム・サーバー機能がかなり小規模なら、そのマシンをワークステーションとして使用したり、ネットワーク用のファイル・サーバーとして使用することもできます。

注 : ご使用のシステムに NIS がインストールされている場合は、これらのサービスが提供するネーム・レゾリューションも使用することができます。

ゲートウェイの構成
他のネットワークと通信するネットワークの場合は、最低 1 つのゲートウェイ・ホストを構成する必要があります。

サポートする通信プロトコルを考慮した上で、それらのプロトコルをサポートする経路指定デーモン (**routed** デーモンまたは **gated** デーモン) を使用する必要があります。

TCP/IP 構成コマンドと管理コマンド
さまざまなコマンドを使用して、**TCP/IP** ネットワークの構成と管理を行うことができます。 この表では、これらのコマンドについて説明します。

項目	説明
arp	**アドレス解決**プロトコルが使用するハードウェア・アドレス変換テーブルに合わせて IP アドレスを表示または変更します。
finger	指定したホスト上のユーザーに関する情報を戻します。
host	指定したホストの IP アドレスか、指定した IP アドレスのホスト名を表示します。
hostname	ローカル・ホストのインターネット名とアドレスを表示または設定します。
ifconfig	ネットワーク・インターフェースとその 特性を構成します。
netstat	ローカル・アドレスと外部アドレス、ルーティング・テーブル、ハードウェアに関する統計、転送されるパケットの合計を表示します。
no	現行のネットワーク・カーネル・オプションを設定または表示します。
ping	ホストへ到達可能かどうかを判別します。
route	ルーティング・テーブルを手作業で操作できるようにします。
ruptime	ローカル物理ネットワークに接続され、**rwhod** サーバーを実行中であるホストに関する状況情報を表示します。
rwho	ローカル物理ネットワークに接続され、**rwhod** サーバーを実行中であるホスト上のユーザーに関する状況情報を表示します。
setclock	ネットワーク・タイム・サービスを読み取り、それに応じてローカル・ホストの時刻と日付を設定します。
timedc	**timed** デーモンに関する情報を戻します。

項目	説明
trpt	TCP ソケット上のプロトコル・トレースを報告します。
whois	インターネット・ネーム・ディレクトリー・サービスを提供します。

認証とセキュア rcmds

これらのコマンドの機能は拡張され、 現在使用されている認証方法に加えて、 追加の認証方法を提供するようになりました。

セキュア rcmd は、**rlogin**、**rcp**、**rsh**、**telnet**、 および **ftp** です。 デフォルトでは、これらのコマンドは標準 *AIX* 方式の認証を使用します。 この追加の方法は、Kerberos V.5 および Kerberos V.4 の 2 つの方法です。

Kerberos V.5 認証方式を使用する場合、クライアントは Kerberos V.5 チケットを DCE セキュリティー・サーバーまたは Native Kerberos サーバーから入手します。 このチケットは、接続しようとしている **TCP/IP** サーバーに対する暗号化された、ユーザーの現在の DCE または Native 信用証明情報の一部です。 **TCP/IP** サーバーのデーモンは、このチケットを暗号化解除します。 これにより、**TCP/IP** サーバーはユーザーを完全に識別できます。 チケットに記述された DCE または Native プリンシパルがオペレーティング ・システムのユーザー・アカウントへのアクセスを許可されている場合、接続が進行します。

> **注:** DCE バージョン 2.2 以降では、DCE セキュリティー・サーバーは Kerberos V.5 チケットを戻すことができます。 AIX オペレーティング・システムのセキュア rcmds は、Kerberos V.5 ライブラリーと NAS (Network Authentication Service) バージョン 1.3 提供の GSSAPI ライブラリーを使用します。

クライアントを認証することに加えて、Kerberos V.5 は現在のユーザーの証明書を **TCP/IP** サーバーに転送します。 証明書が転送可能としてマークされている場合、クライアントはそれらを Kerberos TGT (発券許可証) としてサーバーに送信します。 **TCP/IP** サーバー側では、いずれかのサーバーが DCE セキュリティー・サーバーと通信する場合は、デーモンが **k5dcecreds** コマンドを使用して、TGT を完全な DCE 証明書にアップグレードします。

ftp コマンドは、他のコマンドとは異なる認証方式を使用します。 このコマンドは、GSSAPI セキュリティー・メカニズムを使用して、**ftp** コマンドと **ftpd** デーモンの間で認証が渡されます。 **ftp** クライアントは、**clear**/**safe**/**private** サブコマンドを使用してデータの暗号化を行います。

オペレーティング・システムのクライアントとサーバーの 間では、**ftp** が拡張され、暗号化されたデータ接続のために複数バイト転送を行うことができます。 標準では、暗号化データ接続には単一バイト転送だけが定義されています。 第三者のマシンへの接続でデータの暗号化を使用する場合には、**ftp** は単一バイト転送の制限に従います。

注 : rlogin、**rsh**、および **telnet** のセキュアな rcmds コマンドを **klogin** および **kshell** Kerberos V.5 認証方式で使用すると、リモート・ホストへの接続がクローズされるまでに 3 回試行できます。

セキュア rcmds のシステム構成

すべてのセキュア rcmd に関して、そのシステムでどのような認証方式が使用できるかを決定するシステム・レベルの構成メカニズムがあります。 この構成によって、発信と着信の両方の接続を制御します。

認証構成は、ライブラリー `libauthm.a` と 2 つのコマンド **lsauthent** および **chauthent** から構成されます。この 2 つのコマンドは、ライブラリーの 2 つのルーチン `get_auth_methods` と `set_auth_methods` に対するコマンド・ライン・アクセスを提供します。

システムは、3 つの異なる認証方式、すなわち、Kerberos V.5、Kerberos V.4、および標準 *AIX* をサポートします。 認証方式は、ネットワークでユーザーの認証に使用する方法を定義します。

- Kerberos V.5 は DCE の基本であり、最も一般的な方法です。 オペレーティング・システムは、着信 Kerberos V.5 チケットを完全な DCE 証明書にアップグレードするか、または着信 Native Kerberos V.5 チケットを使用します。

- Kerberos V.4 は、セキュア rcmd の 2 つコマンド、**rsh** と **rcp** によってのみ使用されます。 これは、SP システムで以前のバージョンとの互換性をサポートするために提供されており、SP システムでのみ機能します。Kerberos V.4 チケットは DCE 証明書にはアップグレードされません。
- 標準 *AIX* 認証方式という用語は、AIX オペレーティング・システムで使用される認証方式を指します。

複数の認証方式を構成するとフォールバック 設定になります。 最初の方式の接続に失敗した場合は、クライアントは、 構成されている次の認証方式を使用して認証を試みます。

認証方式は、任意の順序で構成できます。 唯一の例外は、標準 *AIX* にフォールバック ・オプションがないので、これを最後の認証方式として構成する必要があることです。標準 *AIX* が認証方式として構成されていない場合には、パスワードによる認証は試みられず、この方式を使用した接続の試みはすべて拒否されます。

認証方式のないシステムを構成することも可能です。 そのような場合、そのシステムは、セキュア rcmd を使用するどのような端末との間での接続も拒否します。 また、Kerberos V.4 は、**rsh** および **rcp** コマンドによってのみサポートされるので、Kerberos V.4 だけの使用を構成されているシステムでは、**telnet**、**ftp**、または **rlogin** を使用した接続は許されません。

関連情報

get_auth_method サブルーチン
set_auth_method サブルーチン
lsauthent コマンド
chauthent コマンド

セキュア rcmds の Kerberos V.5 ユーザー検証

Kerberos V.5 認証方式を使用する場合、**TCP/IP** クライアントは **TCP/IP** サーバーに対する暗号化されたサービス・チケットを入手します。 サーバーはこのチケットを暗号化解除するにあたって、(DCE または Native プリンシパルによる) ユーザーを識別する確実な方式を持っています。

ただし、それでも、この DCE または Native プリンシパルがローカル・アカウントに対するアクセスを許可されているかどうかを判別する必要があります。 DCE または Native プリンシパルをローカル・オペレーティング・システムのアカウントにマッピングする 処理は、単一のサブルーチン `kvalid_user` を含む共用ライブラリー `libvaliduser.a` によって実行されます。 別のマッピング方式を使用したい場合は、システム管理者は `libvaliduser.a` ライブラリーに代わるものを用意する必要があります。

セキュア rcmds の DCE 構成

セキュア rcmd を使用するには、接続できるネットワーク・インターフェースごとに 2 つの DCE プリンシパルが必要です。

それらは、次のとおりです。

```
host/FullInterfaceName
ftp/FullInterfaceName
```

FullInterfaceName は、1 次 *HostName.DomainName* のインターフェース 名とドメイン名です。

セキュア rcmds のネイティブ構成

セキュア rcmd を使用するには、接続できるネットワーク・インターフェースごとに 2 つのプリンシパルが必要です。

それらは、次のとおりです。

```
host/FullInterfaceName@Realmname
ftp/FullInterfaceName@Realmname
```

FullInterfaceName は、1 次 `HostName.DomainName` のインターフェース 名とドメイン名です。
Realmname は、Native Kerberos V レルムの名前です。

TCP/IP のカスタマイズ

TCP/IP をカスタマイズする場合は、.netrc ファイルを作成します。

.netrc ファイルは、**ftp** と **rexec** コマンドのための自動ログイン情報を指定します。 また、新しい ftp マクロを作成することもできます。このマクロは $HOME/.netrc ファイル内で定義されます。 キー機能またはキー・シーケンスをカスタマイズするには、$HOME/.3270keys ファイルを作成して編集します。 さらに、.k5login ファイルは、どのセルにあるどのような DCE プリンシパルがユーザーのアカウントにアクセスできるかを指定します。

.netrc ファイルの作成

$HOME/.netrc ファイルを作成および編集する手順を次に説明します。

1. /usr/samples/tcpip/netrc ファイルのコピーがあること。
2. システム上で **securetcpip** コマンドが作動中でないこと。

.netrc ファイルを作成する場合:

1. 次のコマンドを入力して、/usr/samples/tcpip/netrc ファイルを $HOME ディレクトリーにコピーします。

   ```
   cp /usr/samples/tcpip/netrc $HOME
   ```

2. $HOME/netrc ファイルを編集して、 適切な *HostName*、*LoginName*、*Password* 変数を与えます。 例えば、次のとおりです。

   ```
   machine host1.austin.century.com login fred password bluebonnet
   ```

3. **chmod** コマンドを使用して、$HOME/netrc ファイルの許可を 600 に設定します。コマンド・ライン・プロンプト ($) で次のように入力します。

   ```
   chmod 600 $HOME/netrc
   ```

4. $HOME/netrc ファイルの名前を $HOME/.netrc ファイルに変更します。 先頭のピリオド (.) により、ファイルが隠されます。

   ```
   mv $HOME/netrc $HOME/.netrc
   ```

$HOME/.netrc ファイルには複数のログイン定義と、 ログイン定義ごとに最高 16 個のマクロを入れることができます。

ftp マクロの作成

ここでは、**ftp** マクロの作成手順を説明します。

$HOME/.netrc ファイルを作成しておくこと。

ftp マクロを作成する場合:

1. $HOME/.netrc ファイルを編集して、次の命令を組み込みます。

   ```
   macdef init
   put schedule
   ```

   ftp マクロの終わりに必ずブランク行を挿入してください。 ブランク行は **ftp** マクロを終了させます。 上記の例では、**macdef** サブコマンドはサブコマンド・マクロ init を定義します。 この行の次にマクロが指定するコマンド (この場合は put *schedule*) が続きます。*schedule* はファイル名です。

2. **ftp** マクロの作成が終わったら、コマンド・ライン・プロンプトから、次のように入力します。

   ```
   ftp hostname
   ```

   ここで、*hostname* は接続するホスト名です。

ftp は $HOME/.netrc ファイルをスキャンして、ホスト名と一致するログイン定義を見つけ、そのログイン定義を使用してログインします。

3. ログイン後、コマンド・ラインのプロンプトから次のように入力します。

```
ftp init
```

この例では、**ftp** は init というマクロをスキャンし、マクロが指定する 1 つ以上のコマンドを実行します。

ftp マクロは直前のログイン・エントリーに関連しています。 **ftp** マクロは $HOME/.netrc ファイルに対してグローバルではありません。 マクロ init はログイン時に自動的に実行されます。 次のように入力すると、他のマクロを **ftp** プロンプト (ftp>) から実行できます。

```
$getit
```

この例で、$ は **ftp** マクロ getit を実行します。

キー・セットの割り当ての変更

TCP/IP をカスタマイズする場合は、次の手順を使用してキー機能とシーケンスを変更することができます。

1. **vi** エディターで作業するための知識があること。
2. vi エディターがシステム 上にあること。

$HOME/.3270keys ファイルを作成および編集して、キー機能またはキー・シーケンスをカスタマイズする手順を次に説明します。

1. 次のコマンドを使用して、/etc/3270.keys ファイルを $HOME ディレクトリーにコピーし、.3270keys という名前に変更します。

```
cp /etc/3270.keys $HOME/.3270keys
```

2. 次の手順で、$HOME/.3270keys ファイルのバインド・ステートメントを変更し、キー・セットの割り当てを変更します。

   a) 新しいファイル上で vi エディターを始動し、挿入モードに入ります。

   b) Ctrl-V キー・シーケンスを押してから、マップしたいキーを押します。
   これにより、押したキーの値が表示されます。

   c) 表示された値を $HOME/.3270keys ファイルの Sequence 列の該当する行に配置します。

例えば、vi エディターを始動して挿入モードに入り、Ctrl-V を押してから Alt-Insert を押します。 これによって、[[141q が表示されます。 最初の [は Sequence 列では ¥e に置き換えられ、構成済みの行は次のようになります。

```
   3270 Function Sequence  Key
   bind  pa1      "¥e[141q" #a_insert
```

.k5login ファイル

.k5login ファイルは、セキュア rcmd に対して Kerberos V.5 認証が使用される場合に使用されます。 このファイルは、どのセルにあるどのような DCE プリンシパルがユーザーのアカウントにアクセスできるかを指定します。

このファイルは、$HOME/.k5login に入っています。 これはローカル・ユーザーが所有し、オーナーはこのファイルに対する読み取り許可を持っている必要があります。 このファイルに対する最小の許可設定は 400 です。

.k5login ファイルには、そのアカウントにアクセスできる DCE プリンシパル/セルの対がリストされています。 プリンシパル/セルの対は、Kerberos フォーマット (DCE フォーマットではなく) で保持されています。 例えば、ファイルに次のものが入っている場合、

```
    UserA@Cell1
```

DCE セル Cell1 の DCE プリンシパル UserA は、該当のアカウントにアクセスできます。

DCE プリンシパルがユーザーのアカウント名と同じで、そのアカウントに対する $HOME/.k5login ファイルがない場合には、その DCE プリンシパルでアカウントにアクセスすることができます (ただし、Kerberos V.5 認証が構成されている場合)。

Kerberos V.5 認証の詳細は、111 ページの『認証とセキュア rcmds』を参照してください。

他のシステムおよびユーザーとの通信方法

他のシステムおよびユーザーと通信するには、いくつかの方法があります。このセクションでは、そのうちの 2 つの方法について説明します。1 つ目の方法は、ローカル・ホストをリモート・ホストに接続する方法です。2 つ目の方法は、リモート・ユーザーと会話する方法です。

ローカル・ホストとリモート・ホストの接続

リモート・ログインとコマンド実行に使用する **TCP/IP** ホスト接続コマンドを次に示します。

いくつかの理由で、他のコンピューターにアクセスする必要が生じることがあります。例えば、現在作業中の重要なファイルへの許可をシステム管理者が再び割り当てる必要がある場合や、他のユーザーのワークステーションからの個人用ファイルにアクセスしたい場合などです。別のユーザーのコンピューター端末から自分のコンピューターに接続するということも考えられます。**rlogin**、**rexec**、**telnet** コマンドなどのリモート・ログイン機能を使用すると、ローカル・ホストを入出力端末ホストとして機能させることができます。キー入力した内容はリモート・ホストに送信され、結果はローカル・モニターに表示されます。リモート・ログイン・セッションを終了すると、すべての機能はローカル・ホストに戻ります。

TCP/IP には、リモート・ログインとコマンド実行に使用する以下のコマンドが用意されています。

項目	説明
rexec	**rexec** コマンドは、**rlogin** コマンドを使用してリモート・ホストにログインしたときに、別の外部ホスト上でコマンドを対話式に実行できるようにします。ネットワークに特別なセキュリティーが必要な場合は、システム管理者がこのコマンドを使用不可にします。**rexec** コマンドを出すと、ローカル・ホストはユーザー名とパスワードを得るために、リモート・ホストの $HOME/.netrc ファイルをローカル・ホストから検索します。ユーザー名とパスワードが見つかると、ローカル・ホスト上で実行するよう要求を出したコマンドが実行されます。見つからない場合には、要求を認める前にログイン名とパスワードの入力を求めるプロンプトが表示されます。

項目	説明
rlogin	**rlogin** コマンドは、類似する外部ホストへのログインを可能にします。 さまざまなリモート・ホストで使用できる **telnet** とは異なり、**rlogin** コマンドは UNIX ホストでのみ使用できます。 ネットワークに特別なセキュリティーが必要な場合は、システム管理者がこのコマンドを使用不可にします。 **rlogin** コマンドは、 ローカル・ホストをリモート・ホストに接続できるようにするという点で **telnet** コマンドに似ています。 唯一の違いは、**rlogin** コマンドはトラステッド・コマンドではないので、 特別なセキュリティーが必要なシステムでは使用不可とされる点です。 **rlogin** コマンドがトラステッド・コマンドでない理由は、ローカル・ユーザーが所有する $HOME/.rhosts ファイルとシステム管理者が所有する /etc/hosts.equiv ファイルの両方に、ローカル・ホストにアクセスするリモート・ホストのリストが保持されるためです。 したがって、席を離れるときに端末をオンのままにしておくと、権限をもたないユーザーがこれらのファイルに入っている名前やパスワードを調べたり、何らかの方法でリモート・ホストに損傷を与える可能性があります。 リモート・ユーザーは、**rlogin** コマンドを出した後パスワードを入力することが理想的ですが、これは多くの場合バイパスされます。 $HOME/.rhosts ファイルと /etc/hosts.equiv ファイルの両方に、 ログインしようとするリモート・ホスト名が入っていない場合、 ローカル・ホストはパスワードの入力を求めるプロンプトを表示します。 最初にリモート・パスワード・ファイルが検査され、 入力されたパスワードが検査されます。 パスワードが正しくない場合は、ログイン・プロンプトが再度表示されます。 ログイン・プロンプトで、ティルドとピリオド (~.) を押すと、リモート・ログインの試行を終了します。 **rlogin** コマンドは構成することによって、 ユーザーを認証する目的で Kerberos V.5 を使用できます。 このオプションにより、$HOME/.rhosts ファイルを使用したり、ネットワークを介してパスワードを渡したりしなくても、ユーザーの識別が可能になります。 **rlogin** コマンドの、この使用法の詳細は、111 ページの『認証とセキュア rcmds』を参照してください。

項目	説明
rsh および **remsh**	**rsh** コマンドおよび **remsh** コマンドは、類似する外部ホスト上でコマンドを実行できるようにします。 必要な入力はすべてリモート・ホストが実行しなければなりません。 ネットワークに特別なセキュリティーが必要な場合は、 システム管理者が **rsh** コマンドおよび **remsh** コマンドを使用不可にします。 **rsh** コマンドには、次の 2 つの使用法があります。 • コマンド名が指定されている場合には、リモート・ホスト上で単一のコマンドを実行する。 • コマンド名が指定されていない場合は、**rlogin** コマンドを実行する。 **rsh** コマンドを出すと、ローカル・ホストはリモート・ホストの /etc/hosts.equiv ファイルを検索して、ログイン許可を探します。 この検索が失敗した場合には、$HOME/.rhosts ファイルが検索されます。 この 2 つのファイルはいずれも、ログイン許可をもつリモート・ホストのリストです。 リモート・ユーザーは **rsh** コマンドを出した後、パスワードを入力しなければなりません。 **rlogin** コマンドを出す必要がないようにできます。 **rsh** コマンドはリモート・ホスト上でのコマンドの実行を許可しますが、パスワードの入力が不要になるわけではありません。リモート・ホストへのアクセスにパスワードが必要な場合は、**rsh** コマンドの使用にもパスワードが必要です。これは、いずれのコマンドも $HOME/.rhosts ファイルと /etc/hosts.equiv ファイルにアクセスするためです。 **rsh** コマンドは構成することにより、 ユーザーを認証する目的で Kerberos V.5 を使用できます。 このオプションにより、$HOME/.rhosts ファイルを使用したり、ネットワークを介してパスワードを渡したりしなくても、ユーザーの識別が可能になります。 **rsh** コマンドの、この使用法の詳細は、111 ページの『認証とセキュア rcmds』を参照してください。

項目	説明
telnet、**tn**、および **tn3270**	**telnet** コマンドは TELNET プロトコルをインプリメントし、類似しているかどうかに関係なく外部ホストにログインできるようにする端末エミュレーション・プログラムです。 それは TCP/IP を使用してネットワーク内の他のホストと通信します。 **注:** 便宜上、ここでは **telnet** を **telnet**、**tn**、**tn3270** コマンドの総称として使用します。 **telnet** コマンドは、リモート・ホストにログインする方法の 1 つです。 **telnet** コマンドの最も重要な特徴は、このコマンドがトラステッド ・コマンドである点です。 これに対して、同じくリモート・ログインを可能にする **rlogin** コマンドは、 トラステッド・コマンドとは見なされません。 許可されていないユーザーによるファイルへのアクセス、重要データの盗難、ファイルの削除、システムへのウィルスやワームの 侵入などを防止するために、システムが特別なセキュリティーを必要とする場合があります。 TCP/IP のセキュリティー機能は、これらの発生を防止できるように設計されています。 **telnet** コマンドを使用してリモート・ホストにログインしたいユーザーは、 そのコンピューターにおいて承認されたユーザーのユーザー名とパスワードを入力する必要があります。 これは、ローカル・ホストへのログインに使用する手順と同様です。 リモート・ホストに正常にログインすると、ユーザーの端末はホストに直接接続されているかのように作動します。 **telnet** コマンドは、端末ネゴシエーション と呼ばれるオプションをサポートします。 リモート・ホストが端末ネゴシエーションをサポートする場合、**telnet** コマンドはリモート・ホストにローカル端末のタイプを送信します。 リモート・ホストがローカル端末のタイプを受け入れなければ、**telnet** コマンドは 3270 端末と DEC VT100 端末をエミュレートしようとします。 エミュレートする端末を指定すると、**telnet** コマンドは端末のタイプのネゴシエーションを行いません。 ローカル・ホストとリモート・ホストが端末のタイプについて同意できなければ、ローカル・ホストはデフォルトの **none** になります。 **telnet** コマンドは 3277-1、3278-1、3278-2、3278-3、3278-4、3278-5 の 3270 端末のタイプをサポートします。 カラー・ディスプレイ 上で 3270 モードで **telnet** コマンドを使用すると、 デフォルトでは 3279 ディスプレイ 上と同様の色とフィールドが示されます。 上記のリストの端末のタイプのキーボード・マッピング・ファイルの 1 つを編集すると、 他の色を選択できます。 **telnet** セッションが終了すると、ディスプレイはセッション開始前に使用していた色にリセットされます。 また、**telnet** コマンドは、ユーザーを認証するために Kerberos V.5 を使用するように構成することもできます。 このオプションにより、`$HOME/.rhosts` ファイルを使用したり、ネットワークを介してパスワードを渡したりしなくても、ユーザーの識別が可能になります。 この **telnet** コマンド使用に関する詳細については、111 ページの『認証とセキュア rcmds』を参照してください。

注 : rsh コマンドと **rexec** コマンドを使用してリモート・ホスト上でコマンドを実行できますが、いずれもトラステッド・コマンドではないため、コンピューターに設定されたすべてのセキュリティー・レベルに対応できない場合があります。 システムに特別なセキュリティーが必要な場合、これらのコマンドは使用不可にされることがあります。

リモート・ホストへのログイン

telnet コマンドを使用してリモート・ホストにログインできます。

このためには、リモート・ホストに対する有効なユーザー ID およびパスワードがなければなりません。

リモート・ホスト (この例では `host1`) にログインするには、次のように入力します。

```
telnet host1
```

次のような情報が画面上に表示されます。

```
Trying . . .
Connected to host1
Escape character is '^T'.

AIX telnet (host1)

AIX Operating System
Version 7.1
(/dev/pts0)
login:_
```

ログインした後で、コマンドを発行できます。 システムからログアウトして接続をクローズするには、Ctrl-D キー・シーケンスを使用します。

ログインできない場合は、Ctrl-T キー・シーケンスを使用して接続を取り消します。

リモート・ユーザーとの会話

talk コマンドを使用して、リモート・ホスト上の別のユーザーとリアルタイムで会話します。

1. **talkd** デーモンは、ローカル・ホストとリモート・ホストの両方で活動状態でなければなりません。
2. リモート・ホスト上のユーザーにログインする必要があります。

talk コマンドには、バインド先の有効なアドレスが必要です。 リモート端末のホスト名は、他のネットワーク・コマンド (**ping** コマンドなど) で使用可能な作業用ネットワーク・インターフェースにバインドする必要があります。 スタンドアロン端末装置であるネットワーク・インターフェースがマシンにない 場合、マシンは、**talk** コマンドが機能するようにそのホスト名をループバック・アドレス (127.0.0.1) にバインドする必要があります。

電子メールを使用して、テキスト・メッセージをローカル・ネットワーク上の他のユーザーに送信し、同様に他のユーザーからメールを受信することができます。 コンピューター・システムが適切に構成され、該当する電子アドレスがわかっている場合は、世界中どこでもリモート・システム上のユーザーに電子メール・メッセージを送信することができます。

TCP/IP には、遠隔通信に使用する以下のコマンドが用意されています。

項目	説明
mail	電子メモおよびメールを送受信する
talk	リモート・ホスト上のユーザーと対話式に会話することができる

1. リモート・ホスト上でログインされたリモート・ユーザー dale@host2 と会話するには、jane@host1 側で次のように入力します。

   ```
   talk dale@host2
   ```

   次のようなメッセージが dale@host2 の画面上に表示されます。

   ```
   Message from TalkDaemon@host1 at 15:16...
   talk: connection requested by jane@host1.
   talk: respond with: talk jane@host1
   ```

   このメッセージは、jane@host1 が dale@host2 との会話を試みていることを dale@host2 に通知します。
2. 送信勧誘を受け入れるには、dale@host2 側で次のように入力します。

   ```
   talk jane@host1
   ```

   ユーザー dale@host2 と jane@host1 は、これで対話式会話を行うことができます。
3. 任意の時点で会話を終了するには、Ctrl-C キー・シーケンスを押します。
   これにより、コマンド・ラインのプロンプトに戻ります。

ファイル転送

電子メールを使用すると比較的短いファイルを送信できますが、 大きなファイルを転送するには、より効率のよい方法があります。

一般に、電子メール・プログラムは比較的少量のテキストを送信するように設計されているので、 大きなファイルを効率よく転送するには他の手段が必要です。 **ftp**、**rcp**、**tftp** コマンドは、**TCP/IP** によってローカル・ホストとリモート・ホストの直接接続を確立します。 基本ネットワーク・ユーティリティー (BNU) も、**TCP/IP** を使用して外部ホストとの直接接続を行うことができます。

ftp コマンドおよび rcp コマンドを使用したファイルの転送

ftp コマンドは、リモート・ホストからファイルをコピーするために使用します。 **ftp** コマンドを使用した場合、ファイル属性は保存されず、サブディレクトリーはコピーされません。 この 2 つの条件のうちのいずれかが必要な場合は、**rcp** コマンドを使用します。

項目	説明
ftp	**ファイル転送プロトコル (FTP)** を使用して、ファイルシステムまたは EBCDIC や ASCII などの互いに相異なる文字表現を使用するホスト間でファイルを転送します。 リモート・ホストにパスワードを送信してセキュリティーを提供し、自動ログイン、ファイル転送、ログオフを可能にします。
rcp	ローカル・ホストとリモート・ホスト間、2 つの別個のリモート・ホスト間、 または同じリモート・ホスト上のファイル間で、1 つ以上のファイルをコピーします。 このコマンドは **cp** コマンドに似ていますが、リモート・ファイル操作にしか機能しない点で異なります。 ネットワークに特別なセキュリティーが必要な場合は、システム管理者がこのコマンドを使用不可にします。

ftp および **rcp** コマンドを使用してファイル転送を試みる前に、以下の条件を満たしていることを確認してください。

1. 自動ログイン機能を使用する場合は、 リモート・ホストの `$HOME/.netrc` ファイル内で指定されたリモート・ログイン許可があること。 自動ログイン許可がない場合は、リモート・ホスト用のログイン名とパスワードを知っていること。 `.netrc` ファイルの詳細は、113 ページの『.netrc ファイルの作成』を参照してください。

   代わりに、Kerberos V.5 認証を使用できるようにシステムを構成することも可能です。 これは、`.netrc` または `$HOME/.rhosts` ファイルの代わりに使用されます。 111 ページの『認証とセキュア rcmds』を参照してください。
2. リモート・ホストからファイルをコピーしたい場合は、 そのファイルに対する読み取り許可があること。

   注: リモート・ホスト上のファイルとディレクトリーに対する読み取り許可と書き込み許可は、使用したログイン名によって決まります。
3. ローカル・ホストからリモート・ホストにファイルをコピーしたい場合は、 ファイルのコピー先となるディレクトリーに書き込み許可があること。 また、リモート・ホスト上のディレクトリーに、そこに入れたいファイルと同じ名前をもつファイルが入っている場合は、 リモート・ホスト上でファイルを付加するための書き込み許可があること。

リモート・ホストへの直接ログイン

ファイル転送に **TCP/IP** を使用している場合は、次の手順を使用してリモート・ホストに直接ログインすることができます。

1. **cd** コマンドを使用して、送信したいファイルが入っているディレクトリー (ファイル送信) または転送したファイルを入れたいディレクトリー (ファイル受信) に移動します。
2. リモート・ホストに直接ログインするには、次のように入力します。

   ```
   ftp HostName
   ```

   自動ログイン許可がある場合は、次のような情報がローカル・ホストに表示されます。

```
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server
(Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
331 Password required for dee.
230 User dee logged in.
ftp>
```

自動ログイン許可がない場合は、次のような情報がローカル・ホストに表示されます。

```
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server
(Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
Name (canopus:eric): dee
331 Password required for dee.
Password:
230 User dee logged in.
ftp>
```

3. システム・プロンプトが表示されたら、自分のログイン名とパスワードを入力します。

これで、2 つのホスト間でファイルをコピーする準備ができました。

リモート・ホストへの間接ログイン

ファイル転送に **TCP/IP** を使用している場合は、次の手順を使用してリモート・ホストに間接的にログインすることができます。

1. **cd** コマンドを使用して、送信したいファイルが入っているディレクトリー (ファイル送信) または転送したファイルを入れたいディレクトリー (ファイル受信) に移動します。
2. リモート・ホストに間接的にログインするには、次のように入力します。

```
ftp
```

3. ftp> プロンプトが表示されたら、次のように入力します。

```
open HostName
```

自動ログイン許可がある場合は、次のような情報がローカル・ホストに表示されます。

```
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server
(Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
331 Password required for dee.
230 User dee logged in.
ftp>
```

自動ログイン許可がない場合は、次のような情報がローカル・ホストに表示されます。

```
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server
(Version 4.1 Sat Nov 23 12:52:09 CST 1995) ready.
Name (canopus:eric): dee
331 Password required for dee.
Password:
230 User dee logged in.
ftp>
```

4. システム・プロンプトが表示されたら、自分のログイン名とパスワードを入力します。

リモート・ホストからローカル・ホストへのファイルのコピー

リモート・ホストからローカル・ホストにファイルをコピーするには、**ftp** コマンドを使用します。

ftp コマンドを使用してリモート・ホストからローカル・ホストにファイルをコピーするには、まずリモート・システムに直接または間接的にログインする必要があります。 説明については、リモート・ホストへの直接ログインまたはリモート・ホストへの間接ログインを参照してください。

注 : ftp コマンドは、ASCII デフォルト転送タイプを使用してファイルをコピーします。

リモート・ホストからローカル・ホストにファイルをコピーするには、次のようにします。

1. **dir** サブコマンドを実行して、コピーするファイルが現行ディレクトリー内にあるかどうかを判別します。
   (**ftp** コマンドの **dir** サブコマンドは、**ls -l** コマンドと同じ方法で機能します。) ファイルがこのディレクトリー内にない場合は、**cd** サブコマンドを使用して適切なディレクトリーに移動します。
2. バイナリー・イメージを使用してローカル・ファイルをコピーするには、次のように入力します。

   ```
   binary
   ```

3. ファイルをご使用のホストにコピーするには、次のように入力します。

   ```
   get FileName
   ```

   ファイルは、**ftp** コマンドの発行元のディレクトリーに入れられます。
4. セッションを終了するには、Ctrl-D キー・シーケンスを使用するか、または quit と入力します。

ローカル・ホストからリモート・ホストへのファイルのコピー

ローカル・ホストからリモート・ホストにファイルをコピーするには、**ftp** コマンドを使用します。

ftp コマンドを使用してローカル・ホストからリモート・ホストにファイルをコピーするには、まずリモート・システムに直接または間接的にログインする必要があります。 説明については、リモート・ホストへの直接ログインまたはリモート・ホストへの間接ログインを参照してください。

注 : ftp コマンドは、ASCII デフォルト転送タイプを使用してファイルをコピーします。

ローカル・ホストからリモート・ホストにファイルをコピーするには、次のようにします。

1. $HOME ディレクトリー以外のディレクトリーにファイルを 入れる場合は、**cd** サブコマンドを使用して任意のディレクトリーに移動します。
2. バイナリー・イメージを使用してローカル・ファイルをコピーするには、次のように入力します。

   ```
   binary
   ```

3. ファイルをリモート・ホストにコピーするには、次のように入力します。

   ```
   put FileName
   ```

4. セッションを終了するには、Ctrl-D キー・シーケンスを使用するか、または quit と入力します。

tftp コマンドおよび utftp コマンドを使用したファイルの転送

tftp および **utftp** コマンドは、**トリビアル・ファイル転送プロトコル** (**TFTP**) によりホスト間でファイルを転送するために使用します。

TFTP は単一ファイル転送プロトコルなので、**tftp** と **utftp** コマンドは、**ftp** コマンドのすべての機能を備えているわけではありません。 ネットワークに特別なセキュリティーが必要な場合は、システム管理者がこのコマンドを使用不可にすることがあります。

> **注:** ホストが高水準のセキュリティーで作動している場合は、**tftp** コマンドを使用することはできません。

tftp および **utftp** コマンドを使用してファイル転送を試みる前に、以下の条件を満たしていることを確認してください。

1. リモート・ホストから ファイルをコピーしたい場合は、必要なファイルが入っているディレクトリーに対する読み取り 許可があること。
2. リモート・ホストに ファイルをコピーしたい場合は、ファイルのコピー先となるディレクトリーに対する書き込み 許可があること。

リモート・ホストからのファイルのコピー

ファイルのコピーに **TCP/IP** を使用している場合は、次の手順を使用してリモート・ホストからファイルをコピーすることができます。

1. リモート・ホストへの接続を確立するには、次のように入力します。

```
tftp host1
```

この例では、host1 が接続先となるホスト名です。

tftp> プロンプトが表示されます。

2. 接続が確立されたかどうかを判別するには、次のように入力します。

```
状況
```

次のようなメッセージが表示されます。

```
Connected to host1
Mode: netascii Verbose: off Tracing: off
Remxt-interval: 5 seconds, Max-timeout: 25 seconds
tftp>
```

3. **get** サブコマンド、転送したいファイル名、リモート・システム上でファイルに割り当てたい名前を、それぞれ次のように入力します。

```
get /home/alice/update update
```

リモート・ホスト上の /home/alice ディレクトリーには、他のユーザーに対する読み取り許可を設定する必要があります。 この例では、/home/alice/update ファイルが、host1 からローカル・システム上の現行ディレクトリー内の update ファイルに転送されます。

4. セッションを終了するには、次のように入力するか、

```
quit
```

または、Ctrl-D キー・シーケンスを押します。

リモート・ホストへのファイルのコピー

ファイルのコピーに **TCP/IP** を使用している場合は、次の手順を使用してリモート・ホストにファイルをコピーすることができます。

1. リモート・ホストへの接続を確立するには、次のように入力します。

```
tftp host1
```

この例では、host1 が接続先となるホスト名です。

tftp> プロンプトが表示されます。

2. 接続が確立されたかどうかを判別するには、次のように入力します。

```
状況
```

次のようなメッセージが表示されます。

```
Connected to host1
Mode: netascii Verbose: off Tracing: off
Remxt-interval: 5 seconds, Max-timeout: 25 seconds
tftp>
```

3. **put** サブコマンド、ローカル・ホストから転送したいファイル名、およびリモート・システム上のファイルのパスとファイル名を、それぞれ次のように入力します。

```
put myfile /home/alice/yourfile
```

リモート・ホスト上の /home/alice ディレクトリーには、他のユーザーに対する書き込み許可を設定する必要があります。

ユーザーの現行の作業ディレクトリー内に置かれている myfile ファイルが host1 に転送されます。デフォルト・パスが確立されていない場合は、パス名を指定する必要があります。 myfile ファイルは、リモート・ホスト上では yourfile という名前になります。

4. セッションを終了するには、次のように入力するか、

```
quit
```

または、Ctrl-D キー・シーケンスを押します。

リモート・システムへのファイルの印刷

ここでは、ユーザーのホストにローカル・プリンターが接続されている場合のリモート・プリンターへの印刷について説明します。 また、ローカル・プリンターが接続されていない場合のデフォルト以外のリモート・プリンターへの印刷についても説明します。

1. ホスト名がリモート・ホストの /etc/hosts.lpd ファイル内に記述されていること。

   注 : キューイング・システムでは、マルチバイトのホスト名はサポートされません。

   システムを再始動せずに /etc/hosts.lpd ファイルに対する変更を行うには、システム・リソース・コントローラー (SRC) の **refresh** コマンドを使用します。
2. ローカルの /usr/lib/lpd/qconfig ファイルで、キュー名とリモート・プリンター名を判別できること。

enq コマンドまたはシステム管理インターフェース・ツール (SMIT) を使用して、このタスクを実行できます。

> **注:** このセクションでは、リモート・ホストへの印刷の方法として、最も単純な方法を説明します。 リモート印刷の詳細と考え方は、**enq** コマンドの説明を参照してください。

リモート印刷キューへの印刷ジョブの挿入

ファイルの印刷に **TCP/IP** を使用している場合は、次の手順を使用してリモート印刷キューにジョブを入れることができます。

リモート印刷キューへジョブを入れるには、リモート・ホストの /etc/hosts.lpd ファイルにホスト名が入っている必要があります (キューイング・システムでは、マルチバイトのホスト名はサポートされません)。 システムを再始動せずに /etc/hosts.lpd ファイルに対する変更を行うには、システム・リソース・コントローラー (SRC) の **refresh** コマンドを使用します。 また、ローカルの /usr/lib/lpd/qconfig ファイルに入っている、キュー名とリモート・プリンター名を判別できる必要があります。

1. 該当するキュー名とリモート・デバイス名を探します。 一般にキュー名は文字 rp で始まり、後に 1 つ以上の数字が続きます。 一般にリモート・プリンター名は文字 drp で始まり、後に 1 つ以上の数字が続きます。
2. 次のコマンドを入力します。

   ```
   enq -P QueueName:PrinterName FileName
   ```

   ここで、*QueueName* はキュー名 (rp1 など)、*PrinterName* は /usr/lib/lpd/qconfig ファイル内にあるプリンター名 (drp1 など) です。 *QueueName* と *PrinterName* の間のコロン (:) を省略しないでください。 *FileName* は印刷したいファイル名です。

次に、**enq** コマンドの使用例を示します。

- デフォルト・プリンターでファイル memo を印刷するには、次のように入力します。

  ```
  enq memo
  ```

- ファイル prog.c をページ番号を付けて印刷するには、次のように入力します。

  ```
  pr prog.c | enq
  ```

  pr コマンドでは、ファイルの最終更新日、ファイル名、およびページ番号などを含む見出しを各ページの一番上に挿入します。 次に、**enq** コマンドで、このファイルを印刷します。

- fred キュー用に構成された次に使用可能なプリンターでファイル report を印刷するには、次のように入力します。

```
enq -P fred report
```

- fred キュー用に構成された次に使用可能なプリンターで、プレフィックス sam で始まる複数のファイルを印刷するには、次のように入力します。

```
enq -P fred sam*
```

プレフィックス sam で始まるすべてのファイルが、1 つの印刷ジョブに入れられます。 通常の状況コマンドでは、**-T** フラグを使用して異なる値が指定されていない限り、印刷ジョブのタイトルだけ (この場合はキューに入っている最初のファイルの名前だけ) が表示されます。 印刷ジョブに含まれているすべてのファイルの名前をリストするには、長状況コマンド **enq -A -L** を使用します。

SMIT を使用したジョブのエンキュー

ファイルのエンキューに **TCP/IP** を使用している場合は、**smit** コマンドを使用することができます。

1. SMIT を使用してジョブをエンキューするには、次のコマンドを入力します。

```
smit
```

2.「**Spooler (スプーラー)**」を選択し、印刷ジョブ・メニューを開始します。
3.「**File to Print (印刷するファイル)**」オプションを選択し、印刷したいファイルの名前を入力します。
4. **Print Queue** オプションを選択し、次に印刷したいリモート・プリンター名を選択します。

これで、リモート・プリンターに印刷する準備ができました。

リモート・システムからのファイルの印刷

リモート・ホスト上にあるファイルを印刷しなければならない場合があります。 印刷出力のロケーションは、リモート・ホストに使用できるリモート・プリンターによって異なります。

1. **rlogin** または **telnet** コマンドを使用して、リモート・システムにログインできること。
2. ローカル・プリンター上で印刷したいリモート・ファイルに対する読み取り許可があること。

> **注:** ここでは、リモート・ホストへの印刷の方法として、最も単純な方法を説明します。 リモート印刷の詳細と考え方は、**enq** コマンドの説明を参照してください。

リモート・システムから印刷するには、

1. **rlogin** または **telnet** コマンドを使用して、リモート・システムにログインできること。
2. 該当するキュー名とリモート・デバイス名を探します。 一般にキュー名は文字 rp で始まり、後に 1 つ以上の数字が続きます。 一般にリモート・プリンター名は文字 drp で始まり、後に 1 つ以上の数字が続きます。
3. 次のコマンドを入力します。

```
enq -P QueueName:PrinterName FileName
```

ここで、*QueueName* はキュー名 (rp1 など)、*PrinterName* は /usr/lib/lpd/qconfig ファイル内にあるプリンター名 (drp1 など) です。 *QueueName* と *PrinterName* の間の : (コロン) を省略しないでください。 *FileName* は、印刷したいファイルの名前です。

4. Ctrl-D シーケンスを押すか、または quit と入力して、リモート・ホストへの接続を終了します。

状況情報の表示

TCP/IP コマンドを使用して、ネットワークの状況を判別し、ユーザーに関する情報を表示し、別のホストまたはユーザーとの通信に必要なホスト情報を解析できます。

TCP/IP 状況コマンド

TCP/IP には、ローカル・ホスト、リモート・ホスト、およびそのネットワークの状況を判別するための状況コマンドがあります。

項目	説明
finger または **f**	指定されたホスト上の現行ユーザーに関する情報を表示します。 この情報には、ユーザーのログイン名、完全名、端末名、およびログインの日時などが含まれます。
host	ホスト名を IP アドレスに変えるか、または IP アドレスをホスト名に変えます。
ping	ネットワークまたはホストの状況を判別しやすくします。 一般に、ネットワークまたはホストが現在稼働しているかどうかを検査するときに使用します。
rwho	ローカル・ネットワーク上のホストにログインしているユーザーを表示します。 このコマンドにより、ローカル・ネットワーク上のユーザーごとにユーザー名、ホスト名、およびログイン日時が表示されます。
whois	ユーザー ID またはニックネームがどのユーザーのものであるかを識別します。 このコマンドは、ローカル・ネットワークがインターネットに接続されている場合にのみ使用できます。

ホストにログインしたすべてのユーザーに 関する情報の表示

以下の手順を使用して、リモート・ホストにログインしたすべての ユーザーに関する情報を表示します。

リモート・ホストにログインしたすべてのユーザーに関する情報を表示するには、次のようにします。

1. 通信相手のリモート・ホストにログインします。
2. ホスト alcatraz にログインしたすべてのユーザーに関する情報を表示するには、次のように入力します。

```
finger @alcatraz
```

次のような情報が表示されます。

```
brown    console  Mar 15 13:19
smith    pts0     Mar 15 13:01
jones    tty0    Mar 15 13:01
```

ユーザー brown はコンソールで、ユーザー smith は疑似テレタイプ・ライン pts0 から、ユーザー jones は tty0 からログインしています。 システム管理者は、**finger** コマンドが別の方法で機能するようにご使用のシステムをセットアップすることができます。 **finger** コマンドの使用時に問題が生じた場合は、システム管理者に連絡してください。

ホストにログインしたユーザーに関する情報の表示

以下の手順を使用して、リモート・ホストにログインした特定の ユーザーに関する情報を表示します。

リモート・ホストにログインした単一ユーザーに関する情報を表示するには、次のようにします。

1. 通信相手のリモート・ホストにログインします。
2. ホスト alcatraz 上のユーザー brown に関する情報を表示するには、次のように入力します。

```
finger brown@alcatraz
```

次のような情報が表示されます。

```
Login name:  brown
Directory:  /home/brown     Shell:  /home/bin/xinit -L -n Startup
On since May 8 07:13:49 on console
No Plan.
```

システム管理者は、**finger** コマンドが別の方法で機能するようにご使用のシステムをセットアップすることができます。 **finger** コマンドの使用時に問題が生じた場合は、システム管理者に連絡してください。

TCP/IP プロトコル

プロトコルとは、メッセージのフォーマットと 手順に関する一連の規則であり、マシンやアプリケーション・プログラムは、それらの規則を使用して情報を交換することができます。 通信に関与する個々のマシンは、受信側ホストがメッセージを理解できるよう、これらの規則に従わなければなりません。 TCP/IP プロトコル群 は、階層 (またはレベル) として理解することができます。

この図は、**TCP/IP** プロトコルの階層を示します。1 番上から、アプリケーション層、トランスポート層、ネットワーク層、ネットワーク・インターフェース 層、およびハードウェアの順に並んでいます。

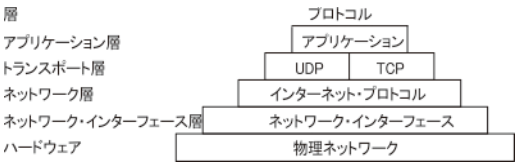


図 *4. TCP/IP* プロトコル群

TCP/IP では、情報が送信側から受信側に移動する方法が入念に定義されます。 最初に、アプリケーション・プログラムがメッセージやデータ・ストリームをいずれかのインターネット・トランスポート層プロトコル、つまり、**ユーザー・データグラム・プロトコル** (**UDP**) か**伝送制御プロトコル** (**TCP**) に渡します。これらのプロトコルはアプリケーションからデータを 受信し、そのデータをパケット と呼ばれる小さな単位に分割し、宛先アドレスを追加してから、次のプロトコル層、つまりインターネット・ネットワーク層に渡します。

インターネット・ネットワーク層は、パケットを**インターネット・プロトコル** (**IP**) データグラムに入れ、データグラム・ヘッダーとデータグラム・トレーラーを付け、データグラムの送信先 (直接宛先に送信するか、あるいは、ゲートウェイに送信するか) を決定してから、そのデータグラムをネットワーク・インターフェース層に渡します。

ネットワーク・インターフェース 層は **IP** データグラムを受け取り、それらのデータグラムをフレーム として特定のネットワーク・ハードウェア (イーサネット・ネットワークやトークンリング・ネットワークなど) を介して送信します。

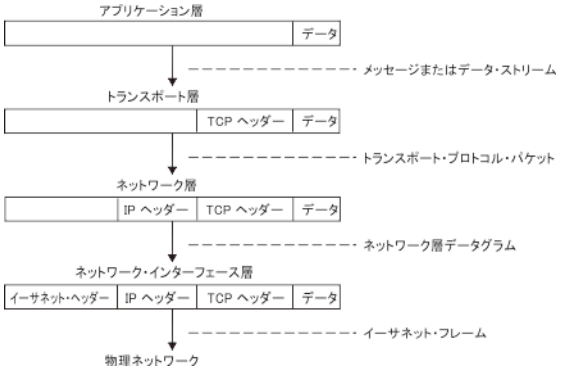


図 *5.* 送信側のアプリケーションから受信側のホストへの情報の移動

この図では、送信側からホストへ TCP/IP プロトコル層を下る情報の流れを示します。

ホストが受信したフレームは、プロトコル層を逆に移動します。 各層は対応するヘッダー情報を除去し、データは最後にアプリケーション層に戻ります。

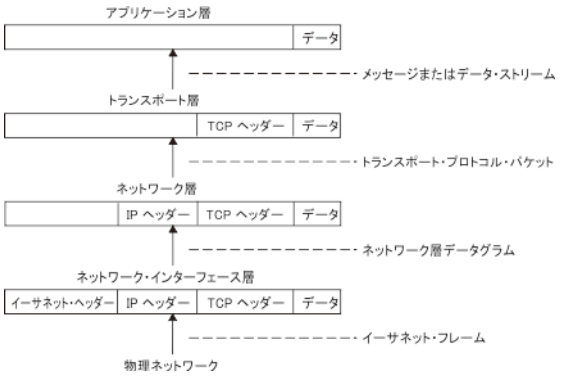
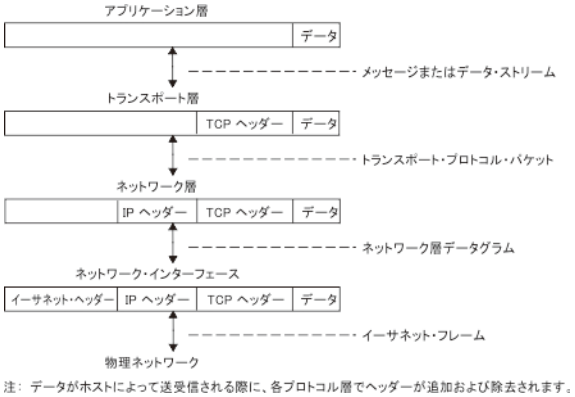


図 6. ホストからアプリケーションへの情報の移動

この図では、ホストから送信側へ **TCP/IP** プロトコル層を昇る情報の流れを示します。

フレームはネットワーク・インターフェース 層 (この場合はイーサネット・アダプター) によって受信されます。 ネットワーク・インターフェース 層は、イーサネット・ヘッダーを除去し、データグラムをネットワーク層へ送信します。 ネットワーク層では、インターネット・プロトコルが IP ヘッダーを除去し、パケットをトランスポート層へ送信します。 トランスポート層では、(この場合は) **TCP** が **TCP** ヘッダーを除去し、アプリケーション層へデータを送信します。

ネットワーク上の複数のホストは、情報の送信と受信を同時に行います。128 ページの図 7 は、通信時のホストをさらに正確に表しています。



注: データがホストによって送受信される際に、各プロトコル層でヘッダーが追加および除去されます。

図 7. ホストのデータ送信と受信

この図では、**TCP/IP** 層の両方向のデータ・フローを示します。

インターネット・プロトコル (IP) バージョン 6

インターネット・プロトコル (**IP**) バージョン 6 (**IPv6** または IP*ng*) は、次世代の **IP** であり、**IP** バージョン 4 (**IPv4**) からの進化段階となるように設計されています。

IPv4 では、グローバルなインターネットの 開発が可能でしたが、将来的にも通用する能力はありません。これはアドレス・スペースの制限と複雑な経路指定という 2 つの基本的な要因によるものです。 **IPv4** の 32 ビットによるアドレスでは、グローバルなインターネットの 経路指定を柔軟に行うことはできません。Classless InterDomain Routing (CIDR) を導入することで、**IPv4** による経路指定は数年間存続を延長できましたが、経路指定の管理作業はさらなる向上を求め続けられることになります。 **IPv4** の経路指定を拡大することが可能としても、最終的にインターネットはネットワーク数の限界を超えることになります。

IETF (Internet Engineering Task Force) は、驚異的に増加するインターネットを **IPv4** がサポートすることは不可能と認識し、IETF IP*ng* 作業グループを結成しました。 作成されたプロポーザルの中で、**Simple Internet Protocol Plus** (**SIPP**) が IP 開発の進化ステップとして選択されました。 これは IP*ng* に名前変更され、RFC1883 が 1995 年 12 月に最終 決定されました。

IPv6 は、IP アドレスの最大値を拡大し、増加一方のインターネットのユーザー数に対処します。 **IPv4** から進化した **IPv6** には、同一ネットワーク上で新旧が共存できるという利点があります。 この共存により、

IPv4 (32 ビット・アドレッシング) から **IPv6** (128 ビット・アドレッシング) への移行を作動ネットワーク上で順序よく行うことができます。

この概要の目的は、IP*ng* プロトコルの全般的な理解を与えることです。 詳しくは、RFC 2460、2373、2465、1886、2461、2462、および 2553 を参照してください。

セキュリティー では、**IPv6** を含む **TCP/IP** プロトコル群のセキュリティーについて説明しています。 IP セキュリティー、バージョン 4 および 6 の詳しくは、インターネット・プロトコル (IP) のセキュリティーを参照してください。

IPv6 自動構成

ノードを始動させ、**IPv4** ネットワークを介して他のノードと通信できるようにする基本的なメカニズムには、ハードコーディング、**BOOTP**、および **DHCP** があります。

IPv6 では、有効範囲 という概念が **IP** アドレスに取り入れられていますが、その 1 つにリンク内があります。 リンク内有効範囲によって、ホストは定義済みのリンク内接頭部とローカル ID から有効なアドレスを作成することが可能となります。 このローカル ID は一般に、構成するインターフェースのメディア・アクセス制御 (MAC) アドレスから派生します。 このアドレスを使うと、ノードは、同じサブネット上の他のホストと通信することができ、また完全に分離されたサブネットでは、他のアドレス構成の必要性をなくすことができます。

IPv6 の意味のあるアドレス

IPv4 では、アドレス内で一般的に意味を理解できるのは、ブロードキャスト (通常、すべて 1 かすべて 0) とクラス (例えば、クラス D はマルチキャスト) だけです。 **IPv6** では、接頭部を簡単に見分けて、有効範囲 (リンク内など)、マルチキャストとユニキャスト、また割り当てのメカニズム (プロバイダー・ベースまたは地域ベース) を判別することができます。

同様に経路指定情報もアドレスの上位ビットに明示的にロードされる可能性がありますが、IETF で最終決定していません (プロバイダー・ベース・アドレスの場合、経路指定情報がアドレスの中に暗黙的に入ります)。

IPv6 重複アドレスの検出

インターフェースが 初期化または再度初期化されると、自動構成を使って一時的にリンク内アドレスをこのインターフェースと 関連付けます (従来の考えでは、アドレスはこのインターフェースにまだ 割り当てられません)。 この時点で、インターフェースは 全ノードのマルチキャスト・グループと送信請求ノードのマルチキャスト・グループを結合し、これらのグループに隣接ディスカバリー・メッセージを送信します。 マルチキャスト・アドレスを使用することにより、ノードは特定のリンク内アドレスが既に割り当て済みか判別し、代わりのアドレスを選ぶことができます。

このようにして、同一アドレスを同じリンク内で、2 つの異なるインターフェースに 割り当てるエラーを減少させることができます (ただし、同一リンクにないノードにグローバル有効範囲のアドレスを、重複して作成することは可能です)。

隣接ディスカバリー/ステートレス・アドレス自動構成

IPv6 の**隣接ディスカバリー・プロトコル** (**NDP**) は、ノード (ホストとルーター) が使用して、接続されたリンク上にあることが分かっている隣接のリンク層アドレスを判別し、アクティブ接続のための宛先別ルーティング・テーブルを維持管理します。 **IPv6** は、ステートフルとステートレスの 両方のアドレス自動構成メカニズムを定義します。 ステートレス自動構成 では、ホストをマニュアルで構成する必要はありません。最小限必要がある場合でも、ルーターの構成のみで、他のサーバーは必要ありません。

ホストも、**NDP** を使用して、ホストに代わってパケットを転送しようとしている隣接ルーターを見つけ、変更されたリンク層アドレスを検出します。 **NDP** は、独自の固有メッセージ・タイプと一緒に、**インターネット制御メッセージ・プロトコル** (**ICMP**) バージョン 6 を使用します。 通常、**IPv6** の隣接ディスカバリー・プロトコルは、**IPv4** の**アドレス解決プロトコル** (**ARP**)、ICMP Router Discovery (RDISC)、および **ICMP** リダイレクト (ICMPv4) の組み合わせに相当しますが、これらの **IPv4** プロトコルと比べて多くの改善が加えられています。

ステートレスなメカニズムの場合、ホストは、ローカルに使用できる情報とルーターが公示する情報を組み合わせて使い、ホスト自体のアドレスを生成することができます。 ルーターは、リンクに関連したサブネットを識別する接頭部を公示し、一方、ホストはサブネット上のインターフェースを 固有に識別するインターフェース・トークンを生成します。 アドレスは両者が結合して成立します。 ルーターがない場合は、ホストはリンク内アドレスだけ生成することができます。 ただし、リンク内アドレスでも、同一リンクに接続されたノード間で通信を行うことは十分に可能です。

IPv6 拡張経路指定とアドレッシング

IPv6 では、**IP** アドレスのサイズが 32 ビットから 128 ビットに拡大されるため、アドレスを指定する階層のレベルが拡大し、アドレッシングが可能なノード数が増加し、またアドレスの自動構成が単純化します。

IPv6 のアドレスには、3 つのタイプがあります。

項目	説明
unicast (ユニキャスト)	ユニキャスト・アドレスに送信されるパケットは、そのアドレスで識別されるインターフェースに 送達されます。 ユニキャスト・アドレスには、特定の有効範囲として、リンク内、サイト内、グローバルがあります。 また、次の 2 つの特別なユニキャスト・アドレスもあります。 • ::/128 (指定なしアドレス) • ::1/128 (ループバック ・アドレス)
multicast (マルチキャスト)	マルチキャスト・アドレスに送信されるパケットは、そのアドレスで識別されるすべてのインターフェースに 送達されます。 マルチキャスト・アドレスは、接頭部を ff::/8 として識別します。ユニキャスト・アドレスの有効範囲と同様に、マルチキャスト・アドレスの有効範囲にも、ノード内、リンク内、サイト内、および組織内があります。
anycast (エニーキャスト)	エニーキャスト・アドレスには、送信側が 1 つで、複数のリスナーがあり、また応答側は 1 つです (通常、応答側は経路指定プロトコルによって「計測された距離」が「最短のもの」になります)。 この例として、エニーキャスト・アドレス上で listen 中の複数の Web サーバーがあります。 要求がエニーキャスト・アドレスに送信されると、1 つだけが応答します。 エニーキャスト・アドレスは、ユニキャスト・アドレスと区別することができません。 ユニキャスト・アドレスは、複数のインターフェースがそのアドレスによって構成されている 場合、エニーキャスト・アドレスになります。

注 : IPv6 には、ブロードキャスト・アドレスはありません。 ブロードキャスト・アドレスの機能は、マルチキャスト・アドレスが代わって行います。

IPv6 自動構成

ノードを始動させ、**IPv4** ネットワークを介して他のノードと通信できるようにする基本的なメカニズムには、ハードコーディング、**BOOTP**、および **DHCP** があります。

IPv6 では、有効範囲 という概念が **IP** アドレスに取り入れられていますが、その 1 つにリンク内があります。 リンク内有効範囲によって、ホストは定義済みのリンク内接頭部とローカル ID から有効なアドレスを作成することが可能となります。 このローカル ID は一般に、構成するインターフェースのメディア ・アクセス制御 (MAC) アドレスから派生します。 このアドレスを使うと、ノードは、同じサブネット上の他のホストと通信することができ、また完全に分離されたサブネットでは、他のアドレス構成の必要性をなくすことができます。

IPv6 の意味のあるアドレス

IPv4 では、アドレス内で一般的に意味を理解できるのは、ブロードキャスト (通常、すべて 1 かすべて 0) とクラス (例えば、クラス D はマルチキャスト) だけです。 **IPv6** では、接頭部を簡単に見分けて、有効範囲 (リンク内など)、マルチキャストとユニキャスト、また割り当てのメカニズム (プロバイダー・ベースまたは地域ベース) を判別することができます。

同様に経路指定情報もアドレスの上位ビットに明示的にロードされる可能性がありますが、IETF で最終決定していません (プロバイダー・ベース・アドレスの場合、経路指定情報がアドレスの中に暗黙的に入ります)。

IPv6 重複アドレスの検出

インターフェースが 初期化または再度初期化されると、自動構成を使って一時的にリンク内アドレスをこのインターフェースと 関連付けます (従来の考えでは、アドレスはこのインターフェースにまだ 割り当てられません)。 この時点で、インターフェースは 全ノードのマルチキャスト・グループと送信請求ノードのマルチキャスト・グループを結合し、これらのグループに隣接ディスカバリー・メッセージを送信します。 マルチキャスト・アドレスを使用することにより、ノードは特定のリンク内アドレスが既に割り当て済みか判別し、代わりのアドレスを選ぶことができます。

このようにして、同一アドレスを同じリンク内で、2 つの異なるインターフェースに 割り当てるエラーを減少させることができます (ただし、同一リンクにないノードにグローバル有効範囲のアドレスを、重複して作成することは可能です)。

隣接ディスカバリー/ステートレス・アドレス自動構成

IPv6 の**隣接ディスカバリー・プロトコル** (**NDP**) は、ノード (ホストとルーター) が使用して、接続されたリンク上にあることが分かっている隣接のリンク層アドレスを判別し、アクティブ接続のための宛先別ルーティング・テーブルを維持管理します。 **IPv6** は、ステートフルとステートレスの 両方のアドレス自動構成メカニズムを定義します。 ステートレス自動構成 では、ホストをマニュアルで構成する必要はありません。最小限必要がある場合でも、ルーターの構成のみで、他のサーバーは必要ありません。

ホストも、**NDP** を使用して、ホストに代わってパケットを転送しようとしている隣接ルーターを見つけ、変更されたリンク層アドレスを検出します。 **NDP** は、独自の固有メッセージ・タイプと一緒に、**インターネット制御メッセージ・プロトコル** (**ICMP**) バージョン 6 を使用します。 通常、**IPv6** の隣接ディスカバリー・プロトコルは、**IPv4** の**アドレス解決プロトコル** (**ARP**)、ICMP Router Discovery (RDISC)、および **ICMP** リダイレクト (ICMPv4) の組み合わせに相当しますが、これらの **IPv4** プロトコルと比べて多くの改善が加えられています。

ステートレスなメカニズムの場合、ホストは、ローカルに使用できる情報とルーターが公示する情報を組み合わせて使い、ホスト自体のアドレスを生成することができます。 ルーターは、リンクに関連したサブネットを識別する接頭部を公示し、一方、ホストはサブネット上のインターフェースを 固有に識別するインターフェース・トークンを生成します。 アドレスは両者が結合して成立します。 ルーターがない場合は、ホストはリンク内アドレスだけ生成することができます。 ただし、リンク内アドレスでも、同一リンクに接続されたノード間で通信を行うことは十分に可能です。

経路指定の単純化

経路指定作業を単純化するため、**IPv6** のアドレスは接頭部と ID の 2 つの部分から考えられています。 これは **IPv4** のネット-ホスト・アドレスの細分化と同じように見えますが、次に示す 2 つの利点があります。

項目	説明
クラスなし	接頭部や ID のビット数が固定されていません。したがって、過剰なビット数を割り当てることによる無駄を減少できます。
ネスティング	接頭部に異なるビット数を考えることによって、何分割でも任意の数で分割することが可能になります。

ケース 1

128 ビット
ノード・アドレス

ケース 2

項目	**説明**
n ビット	128-*n* ビット

項目	説明
サブネット接頭部	インターフェース ID

ケース 3:

項目	説明	
n ビット	80-*n* ビット	48 ビット
サブスクライバー接頭部	サブネット ID	インターフェース ID

ケース 4:

項目	説明		
s ビット	*n* ビット	*m* ビット	128-*s*-*n*-*m* ビット
サブスクライブ接頭部	領域 ID	サブネット ID	インターフェース ID

通常、IPv4 は、可変長サブネット・マスク (VLSM は、いくつかの一般的なネットワーク全体の規則ではなく、個々の必要性に応じて **IP** アドレッシング・リソースをサブネットに割り振る手段) を指定しても、ケース 3 までです。 可変長接頭部の定義をできる限り工夫してアドレス長を短くした場合ですが、それでも何の価値もありません。

ヘッダー・フォーマットの単純化

IPv6 では、ヘッダー全体を除去するか、**IPv4** ヘッダーにあるフィールドの一部を拡張ヘッダーに移動することで、**IP** ヘッダーを単純化します。 これは、オプションの情報 (拡張ヘッダー) に対して柔軟なフォーマットを定義します。

特に、次の項目がないことに注意してください。

- header length (長さは固定)
- identification
- flags
- fragment offset (フラグメント化拡張ヘッダーに移動)
- header checksum (上位層のプロトコルまたはセキュリティー拡張ヘッダーがデータ保全性を処理)

表 *54. IPv4* ヘッダー

項目	説明	説明	説明	説明
Version	IHL	Type of Service	Total Length	
Identification	Identification	Identification	Flags	Fragment Offset
Time to Live	Time to Live	Protocol	Header Checksum	Header Checksum
Source Address	Source Address	Source Address	Source Address	Source Address
Destination Address	Destination Address	Destination Address	Destination Address	Destination Address
Options	Options	Options	Options	Padding

表 55. IPv6 ヘッダー				
項目	**説明**	**説明**	**説明**	**説明**
Version	Prio		Flow Label	
Payload Length	Payload Length	Payload Length	Next Header	Hop Limit
Source Address	Source Address	Source Address	Source Address	Source Address
Destination Address	Destination Address	Destination Address	Destination Address	Destination Address

IP*ng* のオプション・メカニズムは **IPv4** に比べて改善されています。 **IPv6** のオプションは、パケットの **IPv6** ヘッダーとトランスポート層ヘッダーの間に置かれた別々の拡張ヘッダーに入れられます。 ほとんどの拡張ヘッダーは、最終宛先に到達するまで、パケットの送達パス上で、どのルーターからも検査を受けず、処理もされません。 このようなメカニズムによって、オプションを含むパケットを処理するルーターのパフォーマンスを、大幅に改善することができます。 **IPv4** では、どのようなオプションもすべてルーターによる検査を必要とします。

また別な改善点として、**IPv4** のオプションとは異なり、**IPv6** の拡張ヘッダーは長さが不定で、パケットで運ばれるオプションの総数が 40 バイトに限定されません。 この機能とオプションの処理方法によって、**IPv6** のオプションは **IPv4** では実用的でない機能に使用することができます。例えば、**IPv6** の認証オプションやセキュリティーのカプセル化オプションなどです。

後続するオプション・ヘッダー、およびトランスポート・プロトコルの処理時のパフォーマンスを向上するために、IPv6 のオプションは必ず長さを 8 オクテットの整数倍にして、後続するヘッダーのためにこの位置合わせを保ちます。

プロトコル指定子とオプション・フィールドの代わりに拡張ヘッダーを使用することにより、新しく定義した拡張機能の組み込みがさらに容易になります。

現行の仕様では、拡張ヘッダーは次のように定義されます。

- パス上の各ホップ (ルーター) に適用する hop-by-hop オプション。
- 疎/密送信元経路指定に対する経路指定ヘッダー (まれに使用)。
- フラグメントは、パケットをフラグメントとして定義し、フラグメントの情報を含みます (**IPv6** ルーターはフラグメント化を行いません)。
- 認証 (セキュリティーの『**TCP/IP** のセキュリティー』を参照)
- 暗号化 (セキュリティーの『**TCP/IP** のセキュリティー』を参照)
- 宛先ノード用の宛先オプション (ルーターは無視)。

サービス品質の改善/トラフィック・コントロール

サービスの品質は **RSVP** などコントロール・プロトコルを使用して制御できますが、**IPv6** は **IP** ヘッダーの優先順位フィールドを使用してパケットの優先順位を明示的に定義します。

ノードはこの値を設定し、特定のパケットまたはパケットのグループに 相対的な優先順位を示すことができます。次に、この値をノード、1 つ以上のルーター、あるいは宛先が使用し、パケットに関する選択を行います (すなわちパケットをドロップするか否か)。

IPv6 では、過密制御トラフィックと非過密制御トラフィックの 2 種類の優先順位が指定されます。 この 2 種類の優先順位間に相対的な順位はありません。

過密制御トラフィック は、過密に対してある種の「バックオフ」または他の制限アルゴリズムを用いて応答するトラフィックとして定義されます。 過密制御トラフィックの優先順位は、次のとおりです。

項目	**説明**
0	非特性トラフィック
1	「フィラー」トラフィック (例えば netnews)
2	不在データ転送 (例えば、メール)

項目	説明
3	(予約済み)
4	在席バルク転送 (例えば、**FTP**)
5	(予約済み)
6	対話式トラフィック (例えば、Telnet)
7	コントロール・トラフィック (例えば、経路指定プロトコル)

非過密制御トラフィック は、ビデオ、オーディオ、その他のリアルタイム・トラフィックのように、パケットをドロップする (あるいは単に再送しない) ことによって、過密に応答するトラフィックとして定義されます。 明示的なレベルは、例をあげて定義していませんが、順序は次に示すように過密制御トラフィックの場合と同様です。

- 送信元が廃棄に対して最も積極的な最低値をトラフィックに使用する必要があります。
- 送信元が廃棄に対して最も消極的な最高値をトラフィックに使用する必要があります。

このような優先順位のコントロールは、特定の発信元アドレスからのトラフィックに対してだけ適用できます。 あるアドレスからのトラフィックのコントロールは、別なアドレスからの在席バルク転送より明示的に高位の優先順位にはなりません。

フローのラベル

トラフィックの基本的な優先順位付け以外に、**IPv6** ではパケットの特定のフローを指定するメカニズムを定義します。 **IPv6** の用語でフロー とは、特定の送信元から、ルーターに介入して特別な処理をすることを送信元が求める特定の (ユニキャストまたはマルチキャスト) 宛先へ送信された一連のパケットとして定義されます。

このフロー ID は、優先順位のコントロールに使うことができますが、またいくつもの他のコントロールにも使うことができます。

フロー・ラベルはランダムに選ばれ、したがって、トラフィックが属すフローを識別する以外、トラフィックのどんな特性も識別することはありません。 したがって、ルーターがフロー・ラベルを検査して、パケットが特定のタイプであることを判別することはできません。 ただし、同じラベルを持つ最後のパケットが属す一連のパケットの一部であることは、判別できます。

注 : IPv6 が一般的になるまで、ほとんどの場合、フロー・ラベルは実験的なものです。 フロー・ラベルの使用とコントロールについては、現在まで定義も標準化も行われていません。

IPv6 トンネル伝送

トンネル伝送は、既存の **IPv4** 経路指定インフラストラクチャーを使用して **IPv6** トラフィックを転送する手段を提供します。

IPv6 への移行のキーは、既存の **IPv4** ホストとルーターのインストール・ベースとの互換性にあります。 **IPv4** との互換性を保ちながら、**IPv6** を取り入れることによって、インターネットの **IPv6** への移行タスクがスムーズになります。 **IPv6** のインフラストラクチャーを導入しながら、既存の **IPv4** の経路指定インフラストラクチャーの機能を残し、**IPv6** トラフィックの転送に使用することができます。

IPv6 や **IPv4** のホストとルーターは、**IPv6** のデータグラムを **IPv4** パケットにカプセル化することで、**IPv4** 経路指定トポロジーの領域をトンネル伝送させることができます。 トンネル伝送には、次のようにいろいろな使用方法があります。

項目	説明
ルーター・ツー・ルーター	**IPv4** のインフラストラクチャーによって相互接続された **IPv6** や **IPv4** のルーターは、**IPv6** のパケットを IPv6/IPv4 のルーター間でトンネル伝送することができます。 この場合、トンネルは **IPv6** のパケットが渡る終端間パスの 1 セグメントにまたがります。

項目	説明
ホスト・ツー・ルーター	**IPv6** や **IPv4** ホストは、**IPv4** インフラストラクチャーを介して到達できる中間の **IPv6** や **IPv4** ルーターに **IPv6** のパケットをトンネル伝送することができます。 このタイプのトンネルは、パケットの終端間パスの最初のセグメントにまたがります。
ホスト・ツー・ホスト	**IPv4** のインフラストラクチャーによって相互接続された **IPv6** や **IPv4** のホストは、**IPv6** のパケットを IPv6/IPv4 のホスト間でトンネル伝送することができます。 この場合、トンネルはパケットが渡る終端間パス全体にまたがります。
ルーター・ツー・ホスト	**IPv6/IPv4** ルーターは、それらの最終宛先である **IPv6** や **IPv4** ホストに **IPv6** パケットをトンネル伝送することができます。 このトンネルは、終端間パスの最後のセグメントだけにまたがります。

トンネル伝送の技法は、通常、カプセル化を行うノードがトンネルの最後になるノード・アドレスの判別に使用するメカニズム によって分類されます。 ルーター・ツー・ルーターまたはホスト・ツー・ルーター方式の場合、**IPv6** のパケットはルーターにトンネル伝送されます。 ホスト・ツー・ホストまたはルーター・ツー・ホスト方式の場合、**IPv6** のパケットは最終的な宛先まで全経路をトンネル伝送されます。

トンネルのエントリー・ノード (カプセル化ノード) は、カプセル化 **IPv4** ヘッダーを作成し、カプセル化されたパケットを送信します。 トンネルの終了ノード (カプセル化解除ノード) は、カプセル化されたパケットを受信し、**IPv4** のヘッダーを除去し、**IPv6** のヘッダーを更新し、受信された **IPv6** のパケットを処理します。 ただし、カプセル化ノードは、トンネルの最大伝送単位 (MTU) など、各トンネルのソフト状態情報を維持して、トンネルに転送する **IPv6** パケットを処理する必要があります。

IPv6 には、2 つのタイプのトンネルがあります。

自動トンネル
自動トンネルは、**IPv6** アドレスに組み込まれた **IPv4** アドレス情報を使用して構成されます。宛先ホストの **IPv6** アドレスには、パケットのトンネル伝送先の **IPv4** アドレスについての情報が含まれます。

構成済みトンネル
構成済みトンネルは、手動で構成する必要があります。 これらのトンネルは、組み込み **IPv4** 情報がない **IPv6** アドレスの使用時に使用されます。 トンネルのエンドポイントの **IPv6** アドレスと **IPv4** アドレスを指定する必要があります。

自動および構成済みトンネルの情報については、143 ページの『IPv6 でのトンネリングのセットアップ』を参照してください。

IPv6 マルチホーム・リンク内サポートとサイト内サポート

1 つのホストに複数のインターフェースを 定義することができます。 複数のアクティブなインターフェースを持つホストをマルチホームと呼びます。 各インターフェースには、関連するリン・ローカル・アドレスがあります。

リンク内アドレスだけでも、同一リンクに接続したノード間で通信を行うことが可能です。

マルチホーム・ホストには、2 つ以上の関連のリンク内アドレスが指定されます。 この AIX **IPv6** のインプリメンテーションでは、マルチホーム・ホストでリンク層アドレスを解決するための対処方法として、4 つのオプションが用意されています。 オプション 1 がデフォルトです。

項目	説明
オプション 0	マルチホーム・アクションは行われません。 送信は最初のリンク内インターフェースを使用します。 **隣接ディスカバリー・プロトコル** (**NDP**) がアドレス解決を実行する必要がある場合には、NDP がリンク内ローカル・アドレスが定義されているそれぞれのインターフェースを 使用して Neighbor Solicitation メッセージをマルチキャストします。 NDP は、最初の Neighbor Advertisement メッセージを受信するまで、データ・パケットをキューの中に保持します。 受信後、データ・パケットはこのリンクで送信されます。

項目	説明
オプション 1	**NDP** がアドレス解決を実行する必要がある場合、すなわち、データ・パケットを宛先に送信し、ネクスト・ホップに関するリンク層情報が Neighbor Cache にない場合は、NDP はリンク内アドレスが定義されている各インターフェースを使用して Neighbor Solicitation メッセージをマルチキャストします。 そして、**NDP** は、リンク層情報を入手するまで、データ・パケットをキューの中に保持します。 **NDP** は、インターフェースごとに応答を受信するまで待機します。 これにより、データ・パケットは適切な出力インターフェースを使用して送信されるようになります。 **NDP** が待機しないで、最初に受信した Neighbor Advertisement に応答すると、リンクで送信するデータ・パケットが、パケットの発信元アドレスと関連していないリンクで送信される可能性があります。 **NDP** は待機する必要があるので、送信中の最初のパケット内で遅延が発生します。 ただし、いずれにしても、最初の応答を待機する間に遅延が発生します。
オプション 2	マルチホーム操作が可能ですが、データ・パケットのディスパッチングは、main_if6 によって指定されたインターフェースに限定されます。 **NDP** がアドレス解決を実行する必要がある場合には、NDP はリンク内アドレスが定義されている各インターフェースを 使用して、Neighbor Solicitation メッセージをマルチキャストします。 そして、main_if6 によって指定されたインターフェースからの Neighbor Advertisement メッセージを待機します (**no** コマンドを参照)。 このインターフェースからの 応答を受信後、データ・パケットがこのリンクで送信されます。
オプション 3	マルチホーム操作が可能ですが、データ・パケットのディスパッチングは、main_if6 によって指定されたインターフェースに限定されます。また、サイト内アドレスは、main_site6 によって指定されたインターフェースに関してのみ経路指定されます (**no** コマンドを参照)。 NDP は、オプション 2 の場合と同様に機能します。マルチホーム・ホストでサイト内アドレスを使用してデータ・パケットを経路指定するアプリケーションの場合は、main_site6 によって指定されたサイト内アドレスのみが使用されます。

IPv4 が構成されている場合の *IPv6* へのアップグレード

このシナリオでは、**IPv4** から **IPv6** への手動アップグレードについて説明しています。

この例で使用されるネットワークは、1 つのルーターおよび 2 つのサブネットから構成されます。 各サブネット上に 2 つのホスト (すなわち、ルーターと別のホスト) があります。 このネットワーク上の各マシンを **IPv6** にアップグレードします。 このシナリオが終了するまでに、ルーターは、ネットワーク・インターフェース `en0` 上で接頭部 `3ffe:0:0:aaaa::/64` を、ネットワーク・インターフェース `en1` 上で接頭部 `3ffe:0:0:bbbb::/64` を通知します。 最初に、**IPv6** を一時的にサポートするためのマシンを構成して、マシンをテストできるようにします。 続いて、ブート時に **IPv6** 用の準備が整うようにマシンを構成します。

AIX オペレーティング・システムを実行していて、**IPv4** の設定値が構成されていない場合は、138 ページの『IPv4 が構成されていない場合の IPv6 へのアップグレード』を参照してください。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ステップ 1: IPv6 用のホストのセットアップ

両方のサブネット上のホストで、以下を行います。

1. 次のコマンドを入力して、**IPv4** が構成済みであることを確認します。

   ```
   netstat -ni
   ```

   次のような結果が表示されます。

```
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#2      0.6.29.4.55.ec      279393     0      2510     0     0
en0   1500  9.3.230.64  9.3.230.117         279393     0      2510     0     0
lo0   16896 link#1                            913     0       919     0     0
lo0   16896 127         127.0.0.1              913     0       919     0     0
lo0   16896 ::1                                913     0       919     0     0
```

2. root 権限を使用し、次のコマンドを入力して **IPv6** の設定値を構成します。

```
autoconf6
```

3. 次のコマンドを再実行します。

```
netstat -ni
```

次のような結果が表示されます。

```
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#2      0.6.29.4.55.ec      279679     0      2658     0     0
en0   1500  9.3.230.64  9.3.230.117         279679     0      2658     0     0
en0   1500  fe80::206:29ff:fe04:55ec        279679     0      2658     0     0
sit0  1480  link#3      9.3.230.117              0     0         0     0     0
sit0  1480  ::9.3.230.117                        0     0         0     0     0
lo0   16896 link#1                            2343     0      2350     0     0
lo0   16896 127         127.0.0.1              2343     0      2350     0     0
lo0   16896 ::1                                2343     0      2350     0     0
```

4. 次のコマンドを入力して、**ndpd-host** デーモンを開始します。

```
startsrc -s ndpd-host
```

ステップ 2: IPv6 用のルーターのセットアップ

1. 次のコマンドを入力して、**IPv4** の設定値が構成済みであることを確認します。

```
netstat -ni
```

2. root 権限を使用して、次のコマンドを入力します。

```
autoconf6
```

3. 次のコマンドを入力して、2 つの各サブネットに属するルーターのインターフェース 上でグローバル・アドレスを手動構成します。

```
# ifconfig en0 inet6 3ffe:0:0:aaaa::/64 eui64 alias
# ifconfig en1 inet6 3ffe:0:0:bbbb::/64 eui64 alias
```

ルーターがパケットを送信する先のサブネットごとにこの構成を行う必要があります。

4. IPv6 転送を活動化するには、次のコマンドを入力します。

```
no -o ip6forwarding=1
```

5. **ndpd-router** デーモンを開始するには、次のコマンドを入力します。

```
startsrc -s ndpd-router
```

ndpd-router デーモンは、ルーター上で構成したグローバル・アドレスに対応する接頭部を通知します。 この場合、ndpd-router は、en0 上で接頭部 3ffe:0:0:aaaa::/64 を、en1 上で接頭部 3ffe:0:0:bbbb::/64 を通知します。

ステップ 3. ブート時にホスト上で IPv6 が構成されるようにセットアップする

新たに構成された **IPv6** は、マシンのリブート時に削除されます。 リブートするたびに **IPv6** ホスト機能を使用可能にするには、以下を行います。

1. 任意のテキスト・エディターを使用して /etc/rc.tcpip ファイルを開きます。

2. そのファイルの以下の行のコメントを外します。

```
# Start up autoconf6 process
start /usr/sbin/autoconf6 ""
```

```
# Start up ndpd-host daemon
start /usr/sbin/ndpd-host "$src_running"
```

3. **-A** フラグを start /usr/sbin/autoconf6 "" に追加します。

```
start /usr/sbin/autoconf6 "" -A
```

リブート時に、**IPv6** 構成が設定されます。 ホストごとにこのプロセスを繰り返します。

ステップ 4: ブート時にルーター上で構成される IPv6 のセットアップ

新たに構成された **IPv6** は、リブート時に削除されます。 リブートするたびに **IPv6** ルーター機能を使用可能にするには、以下を行います。

1. 任意のテキスト・エディターを使用して /etc/rc.tcpip ファイルを開きます。
2. そのファイルの以下の行のコメントを外します。

```
# Start up autoconf6 process
start /usr/sbin/autoconf6 ""
```

3. 前のステップでコメントを外した行の直後に以下の行を追加します。

```
# Configure global addresses for router
ifconfig en0 inet6 3ffe:0:0:aaaa::/64 eui64 alias
ifconfig en1 inet6 3ffe:0:0:bbbb::/64 eui64 alias
```

このシナリオでは、ネットワークには en0 および en1 という 2 つのサブネットしかありません。 ルーターがパケットを送信する先のサブネットごとに、行をこのファイルに追加する必要があります。

4. そのファイルの以下の行のコメントを外します。

```
# Start up ndpd-router daemon
start /usr/sbin/ndpd- router "$src_running"
```

リブート時に、**IPv6** が自動的に開始されます。

IPv4 が構成されていない場合の *IPv6* へのアップグレード

このシナリオでは、**IPv4** の設定値を構成せずに **IPv6** のホストおよびルーターをセットアップする方法を示します。

この例で使用されるネットワークは、1 つのルーターおよび 2 つのサブネットから構成されます。 各サブネット上に 2 つのホスト (すなわち、ルーターと別のホスト) があります。 このシナリオが終了するまでに、ルーターは、ネットワーク・インターフェース en0 上で接頭部 3ffe:0:0:aaaa::/64 を、ネットワーク・インターフェース en1 上で接頭部 3ffe:0:0:bbbb::/64 を通知します。 最初に、**IPv6** を一時的にサポートするためのマシンを構成して、マシンをテストできるようにします。 続いて、ブート時に IPv6 用の準備が整うようにマシンを構成します。

このシナリオでは、bos.net.tcp.client ファイルセットがインストールされることを想定しています。

構成済みの **IPv4** を使用して **IPv6** にアップグレードするには、136 ページの『IPv4 が構成されている場合の IPv6 へのアップグレード』を参照してください。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ステップ 1: IPv6 用のホストのセットアップ

1. root 権限を使用して、サブネット上の各ホストに対して次のコマンドを入力します。

```
autoconf6 -A
```

これにより、システム上のすべての **IPv6** 対応インターフェースが起動します。

注: 一部のインターフェースを起動させるには、**-i** フラグを使用します。例えば、`autoconf6 -i en0 en1` と入力すると、インターフェース `en0` と `en1` が起動します。

2. インターフェースを表示するには、次のコマンドを入力します。

```
netstat -ni
```

次のような結果が表示されます。

```
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en0   1500  link#3      0.4.ac.17.b4.11        7     0       17     0     0
en0   1500  fe80::204:acff:fe17:b411             7     0       17     0     0
lo0   16896 link#1                            436     0      481     0     0
lo0   16896 127         127.0.0.1              436     0      481     0     0
lo0   16896 ::1                                436     0      481     0     0
```

3. 次のコマンドを入力して、**ndpd-host** デーモンを開始します。

```
startsrc -s ndpd-host
```

ステップ 2: IPv6 用のルーターのセットアップ

1. root 権限を使用して、ルーター・ホストに対して次のコマンドを入力します。

```
autoconf6 -A
```

これにより、システム上のすべての **IPv6** 対応インターフェースが起動します。

注: 一部のインターフェースを起動させるには、`-i` フラグを使用します。例えば、`autoconf6 -i en0 en1` と入力すると、インターフェース `en0` と `en1` が起動します。

次のような結果が表示されます。

```
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
en1   1500  link#2      0.6.29.dc.15.45        0     0        7     0     0
en1   1500  fe80::206:29ff:fedc:1545             0     0        7     0     0
en0   1500  link#3      0.4.ac.17.b4.11        7     0       17     0     0
en0   1500  fe80::204:acff:fe17:b411             7     0       17     0     0
lo0   16896 link#1                            436     0      481     0     0
lo0   16896 127         127.0.0.1              436     0      481     0     0
lo0   16896 ::1                                436     0      481     0     0
```

2. 次のコマンドを入力して、2 つの各サブネットに属するルーターのインターフェース上でグローバル・アドレスを手動構成します。

```
# ifconfig en0 inet6 3ffe:0:0:aaaa::/64 eui64 alias
# ifconfig en1 inet6 3ffe:0:0:bbbb::/64 eui64 alias
```

注: ルーターがパケットを送信する先のサブネットごとにこの構成を行う必要があります。

3. **IPv6** 転送を活動化するには、次のコマンドを入力します。

```
no -o ip6forwarding=1
```

4. **ndpd-router** デーモンを開始するには、次のコマンドを入力します。

```
startsrc -s ndpd-router
```

ndpd-router デーモンは、ルーター上で構成したグローバル・アドレスに対応する接頭部を通知します。この場合、ndpd-router は、`en0` 上で接頭部 `3ffe:0:0:aaaa::/64` を、`en1` 上で接頭部 `3ffe:0:0:bbbb::/64` を通知します。

5. 続行するには、Enter キーを押してください。

6. もう一度 Enter キーを押して、決定したことを確認し、ソフトウェア・バンドルのインストールを開始します。

ステップ 3. ブート時にホスト上で IPv6 が構成されるようにセットアップする

ホストごとにステップ 1 を完了した後、**IPv6** は、マシンのリブート時に削除されます。 リブートするたびに **IPv6** ホスト機能を使用可能にするには、以下を行います。

1. 任意のテキスト・エディターを使用して /etc/rc.tcpip ファイルを開きます。
2. そのファイルの以下の行のコメントを外します。

```
# Start up autoconf6 process
start /usr/sbin/autoconf6 ""
```

```
# Start up ndpd-host daemon
start /usr/sbin/ndpd-host "$src_running"
```

3. **-A** フラグを start /usr/sbin/autoconf6 "" に追加します。

```
start /usr/sbin/autoconf6 "" -A
```

4. ホストごとにこのプロセスを繰り返します。

リブート時に、**IPv6** が自動的に開始されます。

ステップ 4: ブート時にルーター上で構成される IPv6 のセットアップ

ルーターに対してステップ 2 を完了した後、**IPv6** はリブート時に削除されます。 リブートするたびに **IPv6** ルーター機能を使用可能にするには、以下を行います。

1. 任意のテキスト・エディターを使用して /etc/rc.tcpip ファイルを開きます。
2. そのファイルの以下の行のコメントを外します。

```
# Start up autoconf6 process
start /usr/sbin/autoconf6 ""
```

3. **-A** フラグをその行に追加します。

```
start /usr/sbin/autoconf6 "" -A
```

4. 前のステップでコメントを外した行の直後に以下の行を追加します。

```
# Configure global addresses for router
ifconfig en0 inet6 3ffe:0:0:aaaa::/64 eui64 alias
ifconfig en1 inet6 3ffe:0:0:bbbb::/64 eui64 alias
```

このシナリオでは、ネットワークには en0 および en1 という 2 つのサブネットしかありません。 ルーターがパケットを送信する先のサブネットごとに、行をこのファイルに追加する必要があります。

5. そのファイルの以下の行のコメントを外します。

```
# Start up ndpd-router daemon
start /usr/sbin/ndpd-router "$src_running"
```

6. ブート時に IP 転送を使用可能にするために、次のコマンドを実行します。

```
no -r -o ip6forwarding=1
```

リブート時に、**IPv6** が自動的に開始されます。

IPv6 インターフェースをホストでセットアップ

このシナリオでは、静的な IP および経路を使用した、ノードのランタイム構成の手順について説明します。

この例で使用されるネットワークは、1 つのホストと 1 つのルーターで構成されています。シナリオの終わりまでに、ホスト上に IPv6 インターフェースがセットアップされます 。最初に、IPv6 を一時的にサポートするためのマシンを構成して、マシンをテストできるようにします。 次に、ブート時に IPv6 に対応するよう、マシンを構成します。

考慮事項

- この操作手順のシナリオは、特定のバージョンの AIX を使用してテストされました。得られる結果は、ご使用の AIX のバージョンおよびレベルによって大幅に異なる場合があります。
- この例では、**2001:1:2::/48** を、Internet Assigned Numbers Authority (IANA) がプロバイダーに割り当てた、IPv6 インターフェース用の集約可能グローバル・ユニキャスト・アドレスと想定しています。また、**2001:1:2:3:4::/64** は、ネットワーク管理者によって割り当てられた、ビット 49 - 64 を使用するサブネットです。
- IPv6 グローバル・ユニキャスト・アドレス形式を理解するには RFC 3587 を参照する必要があります。

ステップ 1. IPv6 用のホストのセットアップ

IPv6 用のホストをセットアップするには、この手順を実行します。

1. root 権限を使用し、次のコマンドを入力して IPv6 の設定値を構成します。

   ```
   # autoconf6
   ```

2. 次のコマンドを再実行します。

   ```
   # netstat -ni
   ```

   次の出力に似た結果が表示される必要があります。

   ```
   Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
   en0   1500  link#2      0.6.29.4.55.ec      279679     0     2658     0     0
   en0   1500  9.3.230.64  9.3.230.117         279679     0     2658     0     0
   en0   1500  fe80::206:29ff:fe04:55ec        279679     0     2658     0     0
   sit0  1480  link#3      9.3.230.117              0     0        0     0     0
   sit0  1480  ::9.3.230.117                        0     0        0     0     0
   lo0   16896 link#1                           2343     0     2350     0     0
   lo0   16896 127         127.0.0.1           2343     0     2350     0     0
   lo0   16896 ::1                             2343     0     2350     0     0
   ```

3. **chdev** コマンドを使用して、ホスト・インターフェースに IPv6 アドレスを追加します。この例では、下位 64 ビットは、インターフェース **en0** 上の **autoconf6** によって生成されたリンク・ローカル IP の下位 64 ビットから取得されます。

   ```
   # chdev -l en0 -a netaddr6='2001:2:3:4:206:29ff:fe04:55ec' -a prefixlen=64
   ```

4. 次の接頭部に既存の接頭部リンク経路があれば、それらの経路を削除します。

   ```
   # route delete -inet6 2001:2:3:4::/64
   ```

5. ルーターへの到達可能性を追加するために、ホスト上に接頭部静的経路を構成します。ここで、**fe80::206:29ff:fe04:66e** はルーター、またはルーターへの接続性を備えたゲートウェイです。

   ```
   # route add -inet6 -net 2001:2:3:4::/64 fe80::206:29ff:fe04:66e -static
   ```

   注 : デフォルトの経路の変更が必要な場合は、必ず、**-R** オプションを指定して **autoconf6** を実行してください。このオプションにより、ノード上のデフォルトの経路の追加や上書きが防止されます。その後、ステップ 3 から 5 を繰り返します。

ステップ 2. IPv6 用のルーターのセットアップ

IPv6 用のルーターをセットアップするには、この手順を実行します。

1. 次のコマンドを入力して、IPv4 の設定値が構成済みであることを確認します。

```
# netstat -ni
```

2. root 権限を使用して、次のコマンドを入力します。

```
# autoconf6
```

3. IPv6 転送を活動化するには、次のコマンドを入力します。

```
# no -o ip6forwarding=1
```

4. 次のコマンドを入力して、ルーター・インターフェース 上にグローバル IP を構成します。

```
# chdev –l en0 –a netaddr6='2001:4:5:6:207:30ff:fe05:66ec' –a prefixlen=64
```

5. パケットの正確な配信を可能にするには、ルーター上に手動で経路を構成します。例えば、**fe80::3ca6:70ff:fe00:3004/64** が接頭部 **2001:2:3:4::/64** のゲートウェイである場合は、次のようにして接頭部経路を追加します。

```
# route add –inet6 –net 2001:2:3:4::/64 fe80::3ca6:70ff:fe00:3004 –static
```

ステップ 3. 再始動時のたびにホスト上に構成される IPv6 のセットアップ

『**ステップ 1. IPv6 用のホストのセットアップ**』で構成した IPv6 ホストの設定は、マシンを再始動すると削除されます。マシンを再始動するたびに IPv6 ホスト機能が使用可能になるようにするには、この手順を実行します。

1. テキスト・エディターで **/etc/rc.tcpip** ファイルを開きます。
2. **/etc/rc.tcpip** ファイルの以下の行のコメントを外します。

```
# Start up autoconf6 process
start /usr/sbin/autoconf6 ""
```

   注 : 上記の行が **/etc/rc.tcpip** ファイル内に存在しない場合は、ファイルに追加してください。
3. **-A** フラグを **start /usr/sbin/autoconf6 ""** に追加します。

```
start /usr/sbin/autoconf6 "" –A
```

4. **/etc/rc.tcpip** ファイル内の、コメントを外した (または追加した) 行の後に、次の行を追加します。

```
chdev –l en0 –a netaddr6='2001:2:3:4:206:29ff:fe04:55ec' –a prefixlen=64
```

5. 以前の既存の接頭部経路があれば、次のコマンドを入力して削除します。

```
chdev -l inet0 –a delrout6='–net, 2001:2:3:4::/64'
```

6. 次のコマンドを入力して、経路をセットアップします。

```
chdev -l inet0 –a rout6='–net, 2001:2:3:4::/64 ,fe80::206:29ff:fe04:66e,–static'
```

ステップ 4. 再始動時のたびにルーター上に構成される IPv6 のセットアップ

『**ステップ 2. IPv6 用のルーターのセットアップ**』で構成した IPv6 ルーター設定は、マシンを再始動すると削除されます。 マシンを再始動するたびに IPv6 ルーター機能が使用可能になるようにするには、この手順を実行します。

1. テキスト・エディターで **/etc/rc.tcpip** ファイルを開きます。
2. **/etc/rc.tcpip** ファイルの以下の行のコメントを外します。

```
# Start up autoconf6 process
start /usr/sbin/autoconf6 ""
```

   注 : 上記の行が **/etc/rc.tcpip** ファイル内に存在しない場合は、ファイルに追加してください。

3. **-A** フラグを **start /usr/sbin/autoconf6 ""** に追加します。

```
start /usr/sbin/autoconf6 "" -A
```

4. ルーター上にグローバル IP を構成し、接頭部経路を構成するために、ステップ 2 でコメントを外した (または追加した) 行の後に以下の行を追加します。

```
chdev -l en0 -a netaddr6='2001:4:5:6:207:30ff:fe05:66ec' -a prefixlen=64
chdev -l inet0 -a rout6='-net,2001:2:3:4::/64,fe80::3ca6:70ff:fe00:3004,-static'
```

このシナリオでは、ネットワークが持つサブネットは **en0** の 1 つだけです。 ルーターがパケットを送信する先のすべてのサブネットについて、このファイルに 1 行を追加する必要があります。

IPv6 でのトンネリングのセットアップ

2 つの方法のいずれかを用いて IPv6 でトンネリングをセットアップすることができます。 1 番目の方法では、自動トンネルをセットアップします。 2 番目の方法では、構成済みトンネルをセットアップします。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

IPv6 での自動トンネルのセットアップ

このシナリオでは、**autoconf6** コマンドを使用して、IPv6 を構成し、1 次インターフェース en2 を介して自動トンネルをセットアップします。 次に、**autoconf6** コマンドを使用して、2 次インターフェース en0 を介してトンネルを構成します。

以下は、**netstat -ni** コマンドの結果です。このコマンドにより、システムの現在のネットワーク構成が表示されます。

```
en0   1500  link#2      MAC address here         0     0       33     0     0
en0   1500  1.1         1.1.1.3                  0     0       33     0     0
en2   1500  link#3      MAC address here     79428     0      409     0     0
en2   1500  10.1        10.1.1.1             79428     0      409     0     0
```

- IPv6 および 1 つの自動トンネルを使用可能にするには、以下のコマンドを入力します。

```
autoconf6
```

netstat -ni コマンドを実行すると、次の結果が出力されます。

```
# netstat -in
en0   1500  link#2      MAC address here         0     0       33     0     0
en0   1500  1.1         1.1.1.3                  0     0       33     0     0
en0   1500  fe80::204:acff:fe49:4910              0     0       33     0     0
en2   1500  link#3      MAC address here     79428     0      409     0     0
en2   1500  10.1        10.1.1.1             79428     0      409     0     0
en2   1500  fe80::220:35ff:fe12:3ae8
sit0  1480  link#7      10.1.1.1                 0     0        0     0     0
sit0  1480  ::10.1.1.1
```

en2 (IP アドレス 10.1.1.1) が 1 次インターフェースの 場合は、アドレス ::10.1.1.1 が現在、en2 インターフェースによる 自動トンネリングに使用できます。

- インターフェース en0 を介して自動トンネルを使用可能にするには、次のコマンドを入力します。

```
autoconf6 -s -i en0
```

netstat -ni コマンドを実行すると、次の結果が出力されます。

```
# netstat -in
en0   1500  link#2      MAC address here         0     0       33     0     0
en0   1500  1.1         1.1.1.3                  0     0       33     0     0
en0   1500  fe80::204:acff:fe49:4910              0     0       33     0     0
en2   1500  link#3      MAC address here     79428     0      409     0     0
en2   1500  10.1        10.1.1.1             79428     0      409     0     0
en2   1500  fe80::220:35ff:fe12:3ae8
```

```
sit0  1480  link#7      1.1.1.3                 0     0        3     0     0
sit0  1480  ::10.1.1.1                          0     0        3     0     0
sit0  1480  ::1.1.1.3                           0     0        3     0     0
```

このアクションにより、IPv4 と互換性のある IPv6 アドレスが既存の SIT インターフェース sit0 に追加されます。 これで、トンネリングはアドレス ::1.1.1.3 を使用してインターフェース en0 にも使用できます。 同じインターフェース sit0 は、両方のトンネルに使用されます。

注: 自動トンネルは、システム再起動時に削除されます。 ブート時に自動トンネルが存在するようにするには、/etc/rc.tcpip ファイル内の **autoconf6** コマンドに必要な引数を追加します。

構成済みトンネルのセットアップ

このシナリオでは、SMIT を使用して構成済みトンネルをセットアップします。 このトンネルは、ODM に保管されるため、システム再起動時に使用可能になります。 トンネルは、システム alpha と beta の間に構成されます。 alpha の IPv4 アドレスは 10.1.1.1、beta の IPv4 アドレスは 10.1.1.2 です。

構成済みトンネルをセットアップするには、次のステップを実行します。

1. alpha と beta の間のトンネルを構成するには、両方のシステムで次のコマンドを入力します。

   ```
   smit ctinet6
   ```

2. 両方のシステムで「**IPv4 トンネル・インターフェースに IPv6 を追加**」を選択します。

   ```
   autoconf6
   ```

3. このシナリオでは、IPv4 アドレスに基づいて、alpha に次のような値を入れました。

   ```
   * IPV4 SOURCE ADDRESS (dotted decimal)              [10.1.1.1]
   * IPV4 DESTINATION ADDRESS (dotted decimal)         [10.1.1.2]
     IPV6 SOURCE ADDRESS (colon separated)             []
     IPV6 DESTINATION ADDRESS (colon separated)        []
   ```

   beta には、以下の値が入力されています。

   ```
   * IPV4 SOURCE ADDRESS (dotted decimal)              [10.1.1.2]
   * IPV4 DESTINATION ADDRESS (dotted decimal)         [10.1.1.1]
     IPV6 SOURCE ADDRESS (colon separated)             []
     IPV6 DESTINATION ADDRESS (colon separated)        []
   ```

4. 構成済みインターフェースを 表示するには、次のコマンドを入力します。

   ```
   ifconfig cti X
   ```

   ここで、*X* はインターフェースの 番号です。 このシナリオでは、以下の結果が戻されました。 alpha の場合:

   ```
   cti0: flags=8080051<UP,POINTOPOINT,RUNNING,MULTICAST>
           inet6 fe80::a01:101/128 --> fe80::a01:102
   ```

   beta の場合:

   ```
   cti0: flags=8080051 <UP,POINTOPOINT,RUNNING,MULTICAST>
           inet6 fe80::a01:102/128 --> fe80::a01:101
   ```

SMIT は、次の方法を使用してトンネルの両端に IPv6 アドレスを自動的に作成します。

- 下位 32 ビットには IPv4 アドレスが入ります。
- 上位 96 ビットには接頭部 fe80::/96 が入ります。

必要に応じて、特定の IPv6 アドレスを入れることができます。

パケット・トレース

パケット・トレースとは、パケットが各層を通って宛先に到達するまでのパスを検査するプロセスのことです。

iptrace コマンドは、ネットワーク・インターフェース・レベルのパケット・トレースを実行します。 **ipreport** コマンドは、パケット・トレースに関する情報を 16 進および ASCII フォーマットで出力します。 **trpt** コマンドは、**TCP** のトランスポート・プロトコル・レベルのパケット・トラッキングを実行します。 **trpt** コマンドを使用すると、時刻、**TCP** の状態、およびパケットの順序付けといった詳しい情報が出力されます。

ネットワーク・インターフェースのパケット・ヘッダー

ネットワーク・インターフェース 層では、発信データにパケット・ヘッダーが付加されます。

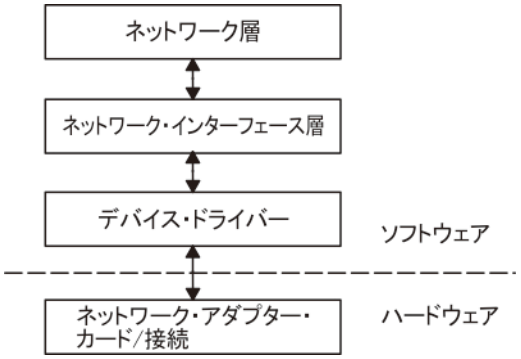


図 *8.* ネットワーク・インターフェースにおけるパケットのフロー

この図は、ネットワーク・インターフェース 構造の層を流れる両方向のデータ・フローを示します。 1 番上 (ソフトウェア) から、ネットワーク層、ネットワーク・インターフェース 層、デバイス・ドライバー、および (ハードウェア) ネットワーク・アダプター・カードまたは接続の順に並べられています。

その後、パケットはネットワーク・アダプターを介して適切なネットワークへ送信されます。 パケットは、宛先に到達するまでに多数のゲートウェイを通る場合があります。 宛先ネットワークでは、パケットからヘッダーが除去され、適切なホストへデータが送信されます。

一般的なネットワーク・インターフェースのパケット ・ヘッダー情報について次に説明します。

イーサネット・アダプターのフレーム・ヘッダー

インターネット・プロトコル (IP) や**アドレス解決プロトコル (ARP)** のイーサネット・アダプター用のフレーム・ヘッダーは、以下の 3 つのフィールドから構成されます。

表 *56.* イーサネット・アダプターのフレーム・ヘッダー

フィールド	長さ	定義
DA	6 バイト	宛先アドレス
SA	6 バイト	発信元アドレス。 このフィールドのビット 0 が 1 に設定された場合、それは経路指定情報 (RI) が存在することを示しています。
タイプ	2 バイト	パケットが **IP** か **ARP** かを指定します。 タイプ番号の値を次に示します。

Type フィールドの番号

項目	説明
IP	0800
ARP	0806

トークンリングのフレーム・ヘッダー

トークンリング・アダプターのメディア・アクセス制御 (MAC) ヘッダーは、5 つのフィールドから構成されます。

表 *57.* トークンリング *MAC* ヘッダー

フィールド	長さ	定義
AC	1 バイト	アクセス制御。 このフィールドの値が x`00' の場合、ヘッダーの優先順位は 0 となります。
FC	1 バイト	フィールド制御。 このフィールドの値が x`40' の場合は、論理リンク制御フレームが指定されます。
DA	6 バイト	宛先アドレス
SA	6 バイト	発信元アドレス。 このフィールドのビット 0 が 1 に設定された場合、それは経路指定情報 (RI) が存在することを示しています。
RI	18 バイト	経路指定情報。 有効なフィールドは、次の説明のとおりです。

MAC ヘッダーは、それぞれ 2 バイトからなる 2 つの経路指定情報フィールド、つまり経路指定制御 (RC) フィールドとセグメント番号のフィールドから構成されます。 最大 8 つの「セグメント番号」フィールドは、限定ブロードキャストの受信側を指定するために使用できます。 RC 情報は、RI フィールドのバイト 0 とバイト 1 に入っています。 RC フィールドの最初の 2 ビットの設定には、次の意味があります。

項目	説明
bit (0) = 0	RI フィールド内で指定した非ブロードキャスト経路を使用します。
bit (0) = 1	RI フィールドを作成し、すべてのリングにブロードキャストします。
bit (1) = 0	すべてのブリッジを介してブロードキャストします。
bit (1) = 1	限定ブリッジを介してブロードキャストします。

論理リンク制御 (LLC) ヘッダーは、次の LLC ヘッダー・テーブルに示すように、5 つのフィールドで構成されています。

表 *58. 802.3 LLC* ヘッダー

フィールド	長さ	定義
DSAP	1 バイト	宛先サービス・アクセス・ポイント (DSAP)。 このフィールドの値は x`aa' です。
SSAP	1 バイト	ソース・サービス・アクセス・ポイント (SSAP)。 このフィールドの値は x`aa' です。
CONTROL	1 バイト	LLC のコマンドと応答を決定します。 このフィールドに入る 3 つの値については、このあとに説明があります。

表 58. 802.3 LLC ヘッダー (続き)		
フィールド	**長さ**	**定義**
PROT_ID	3 バイト	プロトコル ID。 このフィールドは予約済みです。 このフィールドの値は x`0' です。
TYPE	2 バイト	パケットが **IP** か **ARP** かを指定します。

制御フィールドの値
トークンリング制御フィールドには、無番号情報フレーム、交換 ID フレーム、およびテスト・フレームがあります。 ここでは、これらの値について説明します。

項目	**説明**
x`03'	無番号情報 (UI) フレーム。 これはネットワークを介してトークンリング・アダプターのデータを送信する通常の (つまり順序付けを行わない) 方法です。 **TCP/IP** はデータの順序付けを行います。
x`AF'	交換 ID (XID) フレーム。 このフレームは、送信側ホストの特性を伝えます。
x`E3'	テスト・フレーム。 このフレームは送信パスのテストをサポートし、受信したデータをエコー・バックします。

802.3 フレームのヘッダー
802.3 アダプターの MAC ヘッダーは、この MAC ヘッダー・テーブルに示すように、2 つのフィールドで構成されています。

表 59. 802.3 MAC ヘッダー		
フィールド	**長さ**	**定義**
DA	6 バイト	宛先アドレス
SA	6 バイト	発信元アドレス。 このフィールドのビット 0 が 1 に設定された場合、それは経路指定情報 (RI) が存在することを示しています。

802.3 の論理リンク制御 (LLC) ヘッダーは、トークンリング MAC ヘッダーと同じです。

インターネット・ネットワーク・レベル・プロトコル
インターネット・ネットワーク・レベル・プロトコルは、マシン間の通信を処理します。

言い換えれば、この層は **TCP/IP** 経路指定をインプリメントします。 これらのプロトコルは、トランスポート層からパケット送信要求を (宛先のマシンのネットワーク・アドレスと一緒に) 受け入れ、パケットをデータグラム・フォーマットに変換して次の処理のためにネットワーク・インターフェース 層へ送信します。

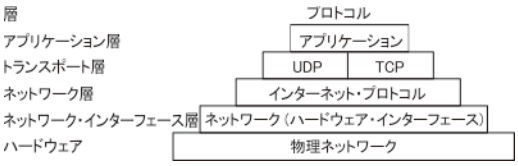


図 9. *TCP/IP* プロトコル群のネットワーク層

この図は、**TCP/IP** プロトコル群のさまざまな層を示します。 1 番上のアプリケーション層は、アプリケーションで構成されます。 トランスポート層には、**UDP** と **TCP** が含まれます。 ネットワーク層には、ネットワーク (ハードウェア) インターフェースが 含まれます。 1 番下のハードウェア層には、物理ネットワークが含まれます。

TCP/IP は、RFC 1100 に準拠しなければならない公認インターネット・プロトコル と、インターネット・コミュニティーのホストが一般的に使用する、その他のプロトコルを提供します。

注 : TCP/IP では、インターネット・ネットワークの使用法、バージョン番号、ソケット番号、サービス番号、およびプロトコル番号も RFC 1010 の *Assigned Numbers* に準拠しています。

アドレス解決プロトコル *(ARP)*

1 番目のネットワーク・レベル・プロトコルは、**アドレス解決プロトコル** (**ARP**) です。 **ARP** は IP アドレスを、ローカル・エリア・ネットワーク上の固有なハードウェア・アドレスに動的に変換します。

ARP がどのように機能するかを示すために、X と Y という 2 つのノードがあると想定して説明します。 ノード X が Y と通信する必要が生じたときに、X と Y が異なるローカル・エリア・ネットワーク (LAN) 上にある場合、X と Y は IP アドレスを使用して、ブリッジ 、ルーター 、またはゲートウェイ を介して通信します。 LAN の中では、ノードは低レベルのハードウェア・アドレスを使用して通信します。

同じ LAN の同じセグメント上にあるノードは、**ARP** を使用して他のノードのハードウェア・アドレスを判別します。 最初に、ノード X がノード Y のハードウェア・アドレスを求める **ARP** 要求をブロードキャストします。 この **ARP** 要求には、X の **IP** アドレスとハードウェア・アドレスのほか、Y の **IP** アドレスが入っています。 Y は、この **ARP** 要求を受信すると、**ARP** キャッシュ (IP アドレスからハードウェア・アドレスへの高速マップに使用される) の中に X のエントリーを入れ、Y の **IP** アドレスとハードウェア・アドレスが入っている **ARP** 応答で直接 X に応答します。 ノード X は、Y の **ARP** 応答を受信すると、Y のエントリーを **ARP** キャッシュの中に入れます。

Y に関する **ARP** キャッシュ・エントリーがいったん X にできると、ノード X は **ARP** を使用しないで、直接 Y にパケットを送信できるようになります (ただし、Y の **ARP** キャッシュ・エントリーが削除されていない場合に限ります。削除されていると、**ARP** を再使用して Y に連絡します)。

ほとんどのプロトコルと異なり、**ARP** パケットには固定フォーマットのヘッダーがありません。 その代わり、メッセージが各種のネットワーク・テクノロジーに使用できるように設計されています。そのようなネットワーク・テクノロジーの例を次に示します。

- イーサネット LAN アダプター (イーサネット・プロトコルと 802.3 プロトコルの両方をサポートします)
- トークンリング・ネットワーク・アダプター
- ファイバー分散データ・インターフェース (FDDI) ネットワーク・アダプター

ただし、**シリアル・ライン・インターフェース・プロトコル** (**SLIP**) や**光シリアル・チャネル変換機構** (**SOC**) のアドレスは Point-to-Point 接続であるために、ARP はこれらを変換しません。

カーネルは変換テーブルを保守しますが、ユーザーやアプリケーションは直接 **ARP** を使用することはできません。 アプリケーションがインターフェース ・ドライバーの 1 つにインターネット・パケットを送信すると、ドライバーは適切なアドレス・マッピングを要求します。 マッピングがテーブル内にない場合は、要求側のインターフェース ・ドライバーを介して **ARP** ブロードキャスト・パケットがローカル・エリア・ネットワーク上のホストへ送信されます。

ARP マッピング・テーブル内のエントリーは 20 分後に削除され、不完全なエントリーは 3 分後に削除されます。 **ARP** マッピング・テーブル内に永続的なエントリーを作成するには、次のように **arp** コマンドに *pub* パラメーターを指定します。

```
arp -s 802.3 host2 0:dd:0:a:8s:0 pub
```

ARP をサポートするホストが **ARP** 要求パケットを受信した場合、ホストは要求側のシステムの **IP** アドレスとハードウェア・アドレスを検出し、必要であればマッピング・テーブルを更新します。 受信側ホストの **IP** アドレスが要求されたアドレスに一致しない場合、ホストは要求パケットを破棄します。 **IP** アドレスが一致する場合、受信側ホストは要求側システムへ応答パケットを送信します。 要求側システムは新しいマッピングを格納し、そのマッピングを使用して、保留中の同様のインターネット・パケットを送信します。

インターネット制御メッセージ・プロトコル

2 番目のネットワーク・レベル・プロトコルは、**インターネット制御メッセージ・プロトコル (ICMP)** です。 **ICMP** は、すべての **IP** インプリメンテーションの必須部分です。 **ICMP** は、**IP** に関するエラー・メッセージと制御メッセージを処理します。

このプロトコルを使用すると、ゲートウェイとホストはパケット送信側のマシンに問題レポートを送信できます。 **ICMP** の機能には次のようなものがあります。

- 宛先が作動中で、しかも到達可能であるかどうかをテストします。
- データグラム・ヘッダーに関するパラメーターの問題を報告します。
- クロックの同期と通過時間を見積もります。
- IP アドレスとサブネット・マスクを入手します。

注 : ICMP は、上位レベルのプロトコルであるかのように、**IP** の基本サポートを使用します。 しかし、**ICMP** は実際には **IP** の必須部分であり、すべての **IP** モジュールにインプリメントされていなければなりません。

ICMP は通信環境の問題に関するフィードバックを行いますが、**IP** の信頼性を保証するわけではありません。 つまり、**ICMP** は **IP** パケットが確実に送達されることを保証するものではなく、また、**IP** パケットが送達されなかったり誤って送達されたときに、**ICMP** メッセージが送信元ホストに戻ることを保証するものでもありません。

ICMP メッセージは、次のいずれかの状況のときに送信される場合があります。

- パケットを宛先に送達できないとき
- ゲートウェイ・ホストにパケットを転送するためのバッファー容量がないとき
- ゲートウェイが、ホストに対して短い経路でトラフィックを送信するよう指示できるとき

TCP/IP は複数の ICMP メッセージ・タイプを送受信します (149 ページの『インターネット制御メッセージ・プロトコルのメッセージ・タイプ』のセクションを参照してください)。 **ICMP** はカーネルに組み込まれ、このプロトコルへの API は提供されません。

インターネット制御メッセージ・プロトコルのメッセージ・タイプ

ICMP は、以下のタイプのメッセージを送受信します。

項目	説明
エコー要求	宛先が作動中で到達可能であるかどうかをテストするため、ホストおよびゲートウェイから送信されます。
情報要求	接続しているネットワークの IP アドレスを入手するため、ホストおよびゲートウェイから送信されます。 このメッセージ・タイプは、宛先の **IP** アドレスのネットワーク部分が、値 0 に設定されて送信されます。
タイム・スタンプ要求	宛先マシンの時刻の現在値を宛先マシンから戻すよう要求するために送信されます。
アドレス・マスク要求	サブネット・マスクを学習するためにホストから送信されます。 ホストは、ゲートウェイへ送信するか、ゲートウェイ・アドレスが分かっている場合は、ブロードキャスト・メッセージを送信することができます。
宛先到達不能	ゲートウェイが **IP** データグラムを送達できないときに送信されます。
送信元の抑制	ゲートウェイやホストの処理能力を超えた速度でデータグラムが送達されたとき、送信元に対してデータグラム送信の速度を落とすように要求するために破棄側のマシンから送信されます。
メッセージのリダイレクト	あるホストが最適でない経路を使用していることをゲートウェイが検出したときに送信されます。

項目	説明
エコー応答	エコー要求を受信したマシンから、その要求を送信したマシンへの応答として送信されます。
情報応答	ゲートウェイから、ネットワーク・アドレスを求める要求への応答として送信されます。これには、指定された **IP** データグラムの送信元フィールドと宛先フィールドの両方が付いています。
タイム・スタンプ応答	時刻の現在値と一緒に送信されます。
アドレス・マスク応答	サブネット・マスクを要求しているマシンへ送信されます。
パラメーターの問題	ホストやゲートウェイがデータグラム・ヘッダーから問題を検出したときに送信されます。
時間超過	次の場合に送信されます。 • 各 **IP** データグラムに存続時間カウンター (ホップ・カウント) が入っており、そのカウンターが 1 ゲートウェイごとに減少する場合 • ホップ・カウントの値が 0 になったために、ゲートウェイがデータグラムを破棄した場合
インターネット・タイム・スタンプ	経路を通過する間のタイム・スタンプを記録するために使用されます。

インターネット・プロトコル

3 番目のネットワーク・レベル・プロトコルは**インターネット・プロトコル** (**IP**) で、インターネットに信頼性が低いコネクションレスのパケット送達を提供します。

IP がコネクションレスであるのは、情報のパケットを個別に処理するからです。 また、信頼性が低い理由は、送達を保証しない、つまり、送信側ホスト、受信側ホスト、または中間ホストからの肯定応答を必要としないからです。

IP は、ネットワーク・インターフェース・レベルのプロトコルへのインターフェースを 提供します。 ネットワークの物理接続により、情報はヘッダーとデータが付いたフレーム単位で転送されます。 ヘッダーには発信元アドレスと宛先アドレスが入っています。 **IP** は、物理フレームによく似た情報が入っているインターネット・データグラムを使用します。 データグラムにもヘッダーがあり、このヘッダーにデータの送信元と宛先の両方の IP アドレスが入っています。

IP は、インターネットを介して送信されるデータのすべてのフォーマットを定義します。

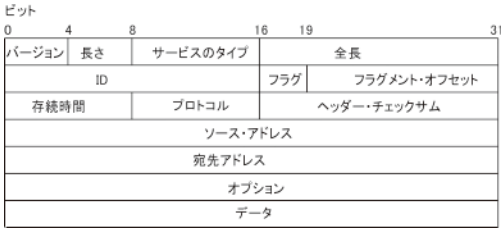


図 *10.* インターネット・プロトコル・パケット・ヘッダー

この図は、通常の IP パケット・ヘッダーの最初の 32 ビットを示します。 それぞれの内容については次の表を参照してください。

IP ヘッダーのフィールド定義

項目	説明
Version (バージョン)	使用する **IP** のバージョンを指定します。 **IP** プロトコルの現行のバージョンは 4 です。
Length (長さ)	データグラム・ヘッダーの長さを 32 ビットのワード単位で指定します。

項目	説明
Type of Service	そのパケットに必要な優先順位、遅延、スループット、および信頼性のタイプを指定する 5 つのサブフィールドから成ります。 (インターネットは、この要求を保証しません。) これら 5 つのサブフィールドのデフォルト設定は、ルーチンの優先順位、通常の遅延、通常のスループット、および通常の信頼性です。 このフィールドは、現時点のインターネットでは一般的には使用されません。 この **IP** のインプリメンテーションは、**IP** 仕様 RFC 791、インターネット・プロトコル の要件に準拠しています。
Total Length (全長)	ヘッダーとデータの両方が入ったデータグラムの長さを、オクテット単位で指定します。 ゲートウェイでのパケットのフラグメント化と、それに伴う宛先での再アセンブリーが提供されています。 **IP** パケットの全長は、インターフェースごとに、**ifconfig** コマンド、またはシステム管理インターフェース・ツール (SMIT) 高速パスの `smit chinet` を使用して構成できます。構成データベースに永続的に値を設定するには SMIT を使用し、実行中のシステム内で値を設定または変更するには、**ifconfig** コマンドを使用します。
Identification (ID)	データグラムを識別する固有な整数が入っています。
Flags (フラグ)	「Identification」フィールドとともに、データグラムのフラグメント化を制御します。 この Fragment Flags は、データグラムがフラグメント化されているかどうかと、現行のフラグメントが最後のものであるかどうかを指定します。
Fragment Offset	元のデータグラム内のフラグメントのオフセットを、8 オクテット単位で指定します。
Time to Live (存続時間)	データグラムがインターネット上に止まる時間を指定します。 これにより、誤って経路指定されたデータグラムが永久にインターネット上に残っていることがなくなります。 デフォルトの存続時間は 255 秒です。
Protocol	高水準プロトコルのタイプを指定します。
Header Checksum	ヘッダーの値の整合性を確認するために計算された数値を示します。
Source Address	送信側ホストの IP アドレスを指定します。
Destination Address	受信側ホストの IP アドレスを指定します。

項目	説明
Options (オプション)	ネットワークのテストとデバッグに使用されます。 このフィールドは、すべてのデータグラムに必要というわけではありません。 **End of Option List** オプション・リストの終わりを示します。 これは、個々のオプションの終わりでなく、最終オプションの終わりに使用されます。 このオプションを使用する必要が生じるのは、これを使用しないと、オプションの終わりが **IP** ヘッダーの終わりと一致しなくなる場合のみです。 End of Option List は、オプションがデータグラムの長さを超える場合に使用されます。 **No Operation** これ以外のオプション間の位置合わせのため、例えば、後続のオプションの開始位置を 32 ビット境界に合わせるために使用されます。 **Loose Source and Record Route** インターネット・データグラムの送信元が、宛先へのデータグラムの転送と経路情報の記録のためにゲートウェイによって使用される経路指定情報を提供する手段となります。 これは疎 送信元経路であり、ゲートウェイまたはホストの **IP** は、任意の数の中間ゲートウェイからなる任意の経路を使用して、経路内の次のアドレスに到達できます。 **Strict Source and Record Route** インターネット・データグラムの送信元が、宛先へのデータグラムの転送と経路情報の記録のためにゲートウェイによって使用される経路指定情報を提供する手段となります。 これは、完全 送信元経路です。つまり、経路内に指定されている次のゲートウェイやホストに到達するには、そのゲートウェイやホストの **IP** は、この送信元経路内にある次のアドレスにデータグラムを直接送信する必要があり、しかも、次のアドレス内に指定されている直接接続されたネットワークだけにしか送信することができません。 **Record Route** このオプションを使用すると、インターネット・データグラムの経路を記録できます。 **Stream Identifier** このオプションを使用すると、ストリームの概念をサポートしないネットワーク内でもストリーム ID を搬送できます。 **インターネット・タイム・スタンプ** 経路内を通過する間のタイム・スタンプを記録できます。

発信パケットの前には、自動的に **IP** ヘッダーが付加されます。 着信パケットからは、上位レベルのプロトコルに送信される前に、**IP** ヘッダーが除去されます。 **IP** プロトコルでは、インターネット・ネットワーク内のホストの汎用アドレッシングが可能です。

インターネットのトランスポート・レベル・プロトコル

TCP/IP トランスポート・レベルのプロトコルにより、アプリケーション・プログラムは、他のアプリケーション・プログラムと通信できるようになります。

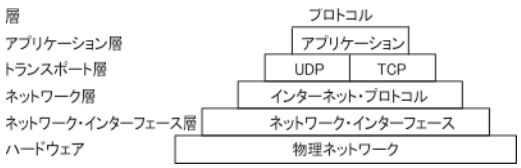


図 *11. TCP/IP* プロトコル群のトランスポート層

この図は、**TCP/IP** プロトコル群のさまざまな層を示します。 1 番上のアプリケーション層は、アプリケーションで構成されます。 トランスポート層には、**UDP** と **TCP** が含まれます。 ネットワーク層には、ネットワーク (ハードウェア) インターフェースが 含まれます。 1 番下のハードウェア層には、物理ネットワークが含まれます。

ユーザー・データグラム・プロトコル (**UDP**) と **TCP** は、 インターネット・ホスト同士を接続するための基本的なトランスポート・レベルのプロトコルです。 **TCP** と **UDP** のどちらを使用しても、プログラムは他のホスト上のアプリケーションへメッセージを送信したり、 他のホスト上のアプリケーションからメッセージを受信したりできます。 アプリケーションからトランスポート層へメッセージ送信要求が送信されると、**TCP** と **UDP** は 情報をいくつかのパケットに分割し、宛先アドレスが入ったパケット・ヘッダーを付加し、 その情報を次の処理のためにネットワーク層へ送信します。 **TCP** と **UDP** はどちらも、メッセージの特定の宛先を識別するためにホスト上のプロトコル・ポートを使用します。

上位レベルのプロトコルとアプリケーションは、**UDP** を使用してデータグラム接続を確立し、**TCP** を使用してストリーム接続を確立します。 オペレーティング・システムのソケット ・インターフェースは、これらのプロトコルをインプリメントします。

ユーザー・データグラム・プロトコル

ネットワーク上のアプリケーションは、他のネットワーク上にある特定のアプリケーションまたはプロセスへメッセージを送信しなければならない場合があります。 **UDP** は、インターネット・ホスト上にあるアプリケーション相互間のデータグラム方式の通信を可能にします。

送信側は、指定された時点でどのプロセスがアクティブになっているかを識別できないので、**UDP** は正の整数で識別される宛先プロトコル・ポート (つまりマシン内の抽象的な宛先ポイント) を使用してホスト上の複数の宛先のいずれかにメッセージを送信します。 プロトコル・ポートはメッセージを受信し、そのメッセージを、受信側ネットワーク上のアプリケーションが取り出せるようになるまで、キューの中に保持します。

UDP は、基礎となる **IP** に依存してデータグラムを送信します。そのため、**UDP** は、**IP** と同じコネクションレスのメッセージ送達を提供します。 データグラム送達や重複保護は保証されません。 ただし、**UDP** により、送信側がメッセージの送信元と宛先ポート番号を指定できるようになるほか、データとヘッダーのチェックサムが計算されます。 これらの 2 つの機能によって、送信側アプリケーションと受信側アプリケーションは、正確なメッセージ送達を行うことができます。

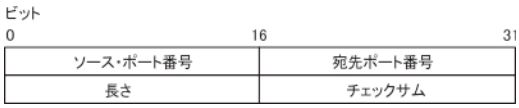


図 *12.* ユーザー・データグラム・プロトコル *(UDP)* パケット・ヘッダー

この図は、**UDP** パケット・ヘッダーの最初の 32 ビットを示します。 最初の 16 ビットには、送信元のポート番号と長さが含まれます。 次の 16 ビットには、宛先のポート番号とチェックサムが含まれます。

信頼できるデータグラム送達を必要とするアプリケーションでは、**UDP** を使用する場合にそれ自体の信頼性検査機能を含めてください。 データ・ストリームの信頼できる送達を必要とするアプリケーションでは、**TCP** を使用してください。

UDP ヘッダーのフィールド定義

項目	説明
Source Port Number (発信元ポート番号)	情報の送信元となるプロトコル・ポートのアドレス。
Destination Port Number (宛先ポート番号)	情報を受信するプロトコル・ポートのアドレス。
Length (長さ)	オクテット単位で示した **UDP** データグラムの長さ。
Checksum	**IP** と同じアルゴリズムを使用して、**UDP** データグラムを検査します。

UDP へのアプリケーション・プログラミング・インターフェース (API) は、ソケット・インターフェースが提供する一連のライブラリー・サブルーチンです。

InfiniBand および RoCE 上の Reliable Datagram Sockets

Reliable Datagram Sockets (RDS) はコネクションレスのレコード単位のプロトコルであり、InfiniBand および RDMA over Converged Ethernet (RoCE) 上で順繰りに重複のないサービスを提供します。RDS は、ソケット API のユーザー・データグラム・プロトコル (UDP) サブセットを公開します。

RDS は、カーネル TCP/IP スタックをバイパスするプロトコルに使用される **AF_BYPASS** ドメインの一部です。

AIX オペレーティング・システムは、RDSv2 と RDSv3 という 2 つのバージョンの RDS を提供します。RDSv3 は最新バージョンであり、リモート・ダイレクト・メモリー・アクセス (RDMA) のサポートを含んでいます。AIX 7.2 以降の RDSv3 は、Open Fabrics Enterprise Distribution (OFED) ベースの RDMA over Converged Ethernet (RoCE) をサポートします。

RDS ソケットの作成

RDS ソケットを作成するには、アプリケーション・プログラムに以下の行を追加して、**socket()** システム・コールを呼び出します。

```
#include <sys/bypass.h>
include <net/rds_rdma.h>           /* for RDSv3 only */
sock = socket (AF_BYPASS, SOCK_SEQPACKET,BYPASSPROTO_RDS);
```

BYPASSPROTO_RDS プロトコルが、**AF_BYPASS** ファミリー内でサポートされている唯一の信頼できるデータグラム・プロトコルである場合は、以下のようにして **socket()** システム・コールを呼び出すこともできます。

```
sock = socket (AF_BYPASS, SOCK_SEQPACKET,0);
```

システム・コール

RDS は、以下のシステム・コールもサポートします。

- blind()
- close()
- getsockopt()
- recvform()
- recvmsg()
- sendmsg()
- sendto()
- setsockopt()

さらに、RDSv3 は、以下のシステム・コールもサポートします。

- connect()
- read()
- recv()
- send()
- write()

注 : RDS ソケットはコネクションレスですが、**connect()** システム・コールは RDSv3 によってサポートされています。ただし、その場合、**connect()** は、2 つの RDS エンドポイント間にソケット・レベル接続エンティティーを作成しません。 単に、デフォルト宛先エンドポイントとソケットを関連付けるだけです。 このため、**listen()**、**accept()**、および **shutdown()** の各システム・コールは、RDS ソケットについてはサポートされません。

RDSv2 の rdsctrl ユーティリティー

チューナブル、および RDS 統計情報の診断を変更するには、**rdsctrl** ユーティリティー (`/usr/sbin/rdsctrl`) を使用します。RDSv2 の場合、このユーティリティーは RDS をロードした後に使用できます

(**bypassctrl load rds**)。 このユーティリティーについて詳しくは、rdsctrl コマンドを引数なしで実行してください。

統計

各種の RDS 統計情報を表示するには、`# rdsctrl stats` コマンドを実行します。

統計情報をリセットするには、`# rdsctrl stats reset` コマンドを実行します。

チューニング・パラメーター

以下の RDS パラメーターは、RDS のロード後、ただし RDS アプリケーションの実行前に調整することができます。

rds_sendspace
フローごとの送信バッファーの最高水準点を指定します。1 つのソケットに複数のフローが存在する場合もあります。デフォルト値は 524288 バイト (512 KB) です。この値は、次のコマンドを使用して設定します。`# rdsctrl set rds_sendspace= <value in bytes>`。

rds_recvspace
ソケットごとの受信バッファーの、フローごとの最高水準点を指定します。このソケットへの追加フローごとに、**receive high-water** マークはこの値ずつ増加します。デフォルト値は 524288 バイト (512 KB) です。この値は、次のコマンドを使用して設定します。`# rdsctrl set rds_recvspace= <value in bytes>`。

注 : RDS ストリーミングのパフォーマンスを向上させるには、**rds_sendspace** パラメーターと **rds_recvspace** パラメーターの値を、RDS の **sendmsg()** の最大サイズに 4 を乗算した値以上にする必要があります。RDS は、受信した 4 つのメッセージを 1 セットとして、1 セットごとに ACK を送信します。**rds_recvspace** がメッセージ・サイズの 4 倍以上大きくない場合、スループットが非常に低下します。

rds_mclustsize
個々のメモリー・クラスターのサイズを指定します (これはメッセージ・フラグメント・サイズでもあります。デフォルト・サイズは 16384 バイト (16 KB) です。この値 (常に 4096 の倍数) は、次のコマンドを使用して設定します。`# rdsctrl set rds_mclustsize= <multiple of 4096, in bytes>`。

重要 : rds_mclustsize 値は、クラスター内のすべてのシステム (ノード) で同じ値であることが必要です。この値を変更すると、パフォーマンスにも影響します。

上記の各パラメーターの現行値は、`# rdsctrl get <parameter>` コマンドを使用して取得できます。

すべてのチューナブルとその値のリストを取得するには、`# rdsctrl get` コマンドを実行します。

RDSv3 の rdsctrl ユーティリティー

RDSv3 の場合、**rdsctrl** コマンドは RDSv3 のオプションをサポートします。 以下は、それらのオプションのリストです。

項目	説明
help [<*tunable name*>]	**help** オプションは、指定された RDSv3 チューナブルを説明するメッセージを表示します。チューナブルを指定しなかった場合、このオプションは RDSv3 用にサポートされているすべてのチューナブルのリストを、各チューナブルの説明を付けて表示します。
set [-p] {<*tunable name*> = <*value*>}	**set** オプションは、指定された RDSv3 チューナブルの値を設定します。このオプションは、ユーザーが、権限のないユーザーによる RDS チューナブルの変更を防止するために必要な特権を持っているかどうかを確認します。また、新しいチューナブル値の範囲の検証も行います。 **-p** フラグを使用すると、リブート操作を実行しても割り当てが持続されます。

項目	説明
get [<*tunable name*>]	**get** オプションは、照会されたチューナブルの現行値を取得します。名前フィールドを指定しなかった場合、このコマンドは使用可能なすべての RDS チューナブルの現行値を返します。
default [-p] [<*tunable name*>]	**default** オプションは、チューナブルをデフォルト値にリセットするために使用します。名前フィールドを指定した場合、そのチューナブルだけがリセットされます。名前フィールドを指定しなかった場合、このコマンドはすべてのチューナブルをデフォルト値にリセットします。 また、このオプションは、**-p** フラグを使用することにより、リブートが実行されても変更が持続される手段を提供します。
load [ofed | aixib]	**load** オプションは、RDSv3 カーネル・エクステンションをロードします (まだロードされていない場合)。 ofed 引数は、RoCE モードで OFED verb の RDSv3 にカーネル・エクステンションをロードします。aixib 引数は、InfiniBand モードで RDSv3 にカーネル・エクステンションをロードします。**load** オプションに引数を指定するのはオプションです。引数が指定されない場合、**load** オプションはデフォルトで aixib 引数に設定されます。 デフォルトでは、コマンド・ラインで新しい属性 (ofed) が指定されない限り、**rdsctrl** ユーティリティーは InfiniBand デバイスをロードします。
unload	**unload** オプションは、RDSv3 カーネル・エクステンションをアンロードするために使用します。
ras [-p] <*minimal* | *normal* | *detail* | *maximal*>	**ras** オプションは、RDSv3 に関する AIX オペレーティング・システムの RAS トレースおよびエラー・チェックの設定を、指定されたレベルに設定します。このコマンドは、内部で AIX オペレーティング・システムの **errctrl** コマンドおよび **ctctrl** コマンドを呼び出します。 **-p** フラグを使用すると、リブート操作を実行しても設定が持続されます。
ras extract	**ras extract** オプションは、RDS の RAS エラーおよび非エラー・トレース・バッファーの内容を標準出力にダンプします。
info [<*flags*>]	**info** オプションは、**rds-info** コマンドの別名です。
ping [<*IP v4 address*>]	**ping** オプションは、**rds-ping** コマンドの別名です。
conn <*restart* | *kill*> <*source IP address*> <*destination IP address*>	**conn** オプションは、指定された RDS 接続を再始動するか (**restart** サブオプション)、指定された RDS 接続を永続的に終了します (**kill** サブオプション)。 再始動するか終了する RDS 接続は、接続のローカル・ノードとリモート・ノードの IP アドレスを渡すことによって指定します。 接続を再始動すると、下層の InfiniBand 接続がドロップし、再び接続の確立が試みられます。それとは対照的に、接続を終了すると (kill サブオプション)、下層の InfiniBand 接続がドロップし、対応する RDS 接続に関連付けられているすべてのリソースが割り振り解除されます。

項目	説明
trace start <*trace file path*> <*maximum data captured per RDS fragment*>	**trace start** オプションは、RDSv3 プロトコル用の保護されたトラフィックをキャプチャーするために、トレース・セッションを開始します。RDSv3 メッセージは、フラグメントとして送信されます。送受信される各 RDS フラグメントは、トレース・パケットとして、指定されたトレース・ファイルにキャプチャーされます。RDS フラグメントごとに、そのペイロードが最大 *<maximum data captured per RDS fragment>* バイトまでキャプチャーされます。 特権ユーザーだけが RDS トラフィックをトレースでき、一度にアクティブにできるトレース・セッションは 1 つだけです。
trace stop	**trace stop** オプションは、前に **trace start** コマンドによって開始されたトレース・セッションを終了します。 このオプションは、トレース・セッションに関連付けられているトレース・ファイルを閉じます。 このコマンドの後、**trace report** コマンドを使用して、トレース・ファイルのテキスト・レポートを生成できます。
trace report <*trace file path*>	**trace report** オプションは、前にキャプチャーした RDS プロトコルのトレース・ファイルから、テキスト・レポートを標準出力に出力します。
version	**version** オプションは、現在システムにロードされている RDS プロトコルのバージョンを出力します。

RDSv3 チューナブル

RDSv3 用にサポートされているチューナブルのリストを表示するには、**rdsctrl help** コマンドを引数なしで実行します。

RDMA API (RDSv3 のみ)

RDS ソケットを使用した RDMA での作業のプログラミング・モデルは、クライアント/サーバー・モデルに基づいています。RDMA クライアントは、指定された RDMA サーバーからの RDMA 読み取り操作または書き込み操作を開始するアプリケーションです。 RDMA サーバーは、RDMA データ転送を処理するアプリケーションです。RDMA 読み取り操作がクライアントのアドレス・スペースからサーバーのアドレス・スペースへのデータ転送であるのに対し、RDMA 書き込み操作はサーバーのアドレス・スペースからクライアントのアドレス・スペースへのデータ転送です。 いずれの場合も、データは両方のサイドのユーザー・スペース・メモリーの間で直接転送され、どちらのサイドのカーネル・スペース・メモリーにもコピーされません。

RDMA クライアント・アプリケーションは、アプリケーション・レベルの要求を RDMA Cookie と一緒に RDMA サーバー・アプリケーションに送信することにより、 RDMA 読み取り操作または書き込み操作を開始することができます。アプリケーション・レベルの要求では、RDMA サーバーがリモート側で読み取りまたは書き込みを行うクライアントのメモリー領域のアドレスと長さだけでなく、操作が RDMA の読み取り操作か書き込み操作かも指定する必要があります。

RDMA クライアントから RDMA サーバーに RDMA 要求を送信するには、2 つの方法があります。

最初の方法は、RDS ソケット上で **sendmsg()** システム・コールを使用して、**RDS_CMSG_RDMA_MAP** 制御メッセージ (**rds_get_mr_args** 構造を搬送する) をアプリケーション・レベルの RDMA 要求と一緒に送信することです。クライアント・サイドの AIX オペレーティング・システム・カーネルは、指定された (クライアント・アプリケーションのアドレス・スペースからの) ローカル・メモリー領域を DMA アクセス用にマップし、RDMA Cookie を生成することにより、**RDS_CMSG_RDMA_MAP** 制御メッセージを処理します。その後、アプリケーション・レベルの要求は RDMA Cookie と一緒にサーバーへ送信されます。

2 番目の方法は、2 つのステップで構成されます。最初のステップは、**RDS_GET_MR** ソケット・オプションを指定して **setsockopt()** システム・コールを呼び出し、**rds_get_mr_args** 構造を引き渡すことです。この呼び出しは、DMA アクセス用の指定されたローカル・メモリー領域をマップし、RDMA Cookie を返します。2 番目のステップは、**sendmsg()** システム・コールを使用して、**RDS_CMSG_RDMA_DEST** 制御メッセージ (最初のステップで取得した RDMA Cookie を搬送する) をアプリケーション・レベルの RDMA 要求と一緒に送信することです。

最初の方法は 1 回のシステム・コールで済むため、2 回のシステム・コールを必要とする 2 番目の方法よりも好まれます。

RDMA サーバー・アプリケーションは、クライアントからのアプリケーション・レベルの **RDMA read** 要求を受け取るとき、**RDS_CMSG_RDMA_DEST** 制御メッセージ (クライアントからの RDMA Cookie を搬送する) も受け取ります。次に、サーバーはクライアントにアプリケーション・レベルの応答を **RDS_CMSG_RDMA_ARGS** 制御メッセージ (**rds_rdma_args** 構造を搬送する) と一緒に送信することにより、**RDMA read** 操作を開始します。サーバー・サイドの AIX オペレーティング・システム・カーネルは、指定された (サーバー・アプリケーションのアドレス・スペースからの) ローカル・メモリー領域を DMA アクセス用にマップし、RDMA 読み取り操作を物理的に開始することにより、**RDS_CMSG_RDMA_ARGS** 制御メッセージを処理します。RDMA 読み取り操作は、サーバー・サイドの InfiniBand アダプターによって行われます。このアダプターはクライアント・サイドの InfiniBand アダプターと対話して、クライアント・アプリケーションのメモリーからサーバー・アプリケーションのメモリーへのデータ転送を直接実行し、それ以上のソフトウェアの介入を必要としません。RDMA 読み取り操作が完了した後、サーバー・サイドのアダプターは、アプリケーション・レベルの応答をクライアントに送信します。これにより、クライアント・アプリケーションは、その RDMA 読み取り操作が完了したことを認識します。

注 : クライアントが RDMA 操作を要求する際、**RDS_RDMA_USE_ONCE** フラグがセットされた **RDS_CMSG_RDMA_MAP** 制御が使用されます。この要求では、クライアントのメモリーのアドレス・スペース内に DMA 用にマップされたメモリー領域は、クライアントがサーバーからアプリケーション・レベルの応答を受信した時点で、DMA 用のマッピングが自動的に解除されます。

この暗黙の DMA マッピングまたはマップ解除のメカニズムは、RDMA アプリケーションの作成を単純化しますが、開発者は、AIX オペレーティング・システムでの DMA 用のメモリーの登録がコストの大きい操作であることを認識しておく必要があります。このため、RDMA を使用して同じメモリー領域に複数回アクセスする場合は、初回にだけ DMA 登録を行う方が効率的です。このアクティビティーを 実行するには、クライアント・アプリケーションでサーバーに RDMA 要求を送信するときに、**RDS_RDMA_USE_ONCE** フラグをセットせずに **RDS_CMSG_RDMA_MAP** 制御メッセージを使用する必要があります。その場合、同じクライアント・メモリー領域への後続の RDMA 転送は、クライアントがサーバーへ別の要求を送信しなくても、RDMA サーバー・アプリケーションによって開始することができます。最後に、クライアント・アプリケーションは **RDS_FREE_MR** ソケット・オプションを指定した **setsockopt()** システム・コールを使用して、DMA 用にマップされたメモリーを明示的にマップ解除する必要があります。

RDS 固有のソケット・オプションは、**setsockopt()** システム・コールまたは **getsockopt()** システム・コールに **SOL_RDS** をレベル・パラメーターとして使用することによって指定します。

伝送制御プロトコル

TCP は、インターネット・ホスト相互間においてデータの高信頼性ストリーム伝送の送達を提供します。

UDP と同様に、**TCP** は、基礎となるプロトコルのインターネット・プロトコルを使用してデータグラムを転送し、プロセス・ポート相互間における連続したデータグラム・ストリームのブロック送信をサポートします。 **UDP** とは異なって、**TCP** のメッセージ送達には信頼性があります。 **TCP** はデータの損傷、損失、重複、および受信側プロセスへの無秩序な送達が起こらないことを保証します。 このように転送の信頼性が保証されているので、アプリケーション・プログラマーはソフトウェアの中に通信保護機能を組み込む必要がありません。

TCP の動作特性を次に示します。

項目	説明
基本データ転送	**TCP** は、8 ビットからなるオクテットの連続ストリームをユーザー間の各方向へ転送できます。その際、いくつかのバイトがセグメントにパックされ、インターネット・システムを通じて送信されます。 **TCP** をインプリメントすると、1024 バイト以上のセグメント・サイズを使用できます。 一般に、**TCP** は自己の都合に合わせて、パケットをブロック化する時期とパケットを転送する時期を決定します。

項目	説明
信頼性	**TCP** は、インターネットによって損傷したり、失われたり、重複したり、無秩序に送達されたデータを回復したりしなければなりません。 **TCP** は、この信頼性を達成するために、送信する各オクテットにシーケンス番号を割り当て、受信側 **TCP** に対して肯定応答 (ACK) を要求します。 タイムアウト・インターバル内に ACK が受信されなかった場合、データは再送されます。 **TCP** の再送タイムアウト値は、往復時間を基礎に個々の接続について動的に決定されます。 受信側では、無秩序に受信した可能性があるセグメントを、シーケンス番号を使用して正しく順序付け、重複がないようにします。 データの損傷に対処するため、送信される個々のセグメントにチェックサムが付加され、受信側はそのチェックサムを検査し、損傷があるセグメントを破棄します。
フロー制御	**TCP** は、送信されるデータの量を制御するために、1 回の ACK ごとにウィンドウを戻し、正常に受信された最後のセグメント以後の受け入れ可能なシーケンス番号の範囲を示します。 このウィンドウは、送信側が次の許可を受信する前に送信できるオクテットの数を示します。
多重化	**TCP** では、1 つのホスト内の多数のプロセスが同時に **TCP** 通信機能を使用できます。 **TCP** は、各ホスト内のポートのアドレス・セットを受信します。 **TCP** はこのポート番号をネットワーク・アドレスおよびホスト・アドレスと組み合わせて、各ソケットを固有に識別します。 1 対のソケットは、1 つの接続を固有に識別します。
接続	**TCP** は個々のデータ・ストリームについて、ある特定の状況情報を初期化し、保守しなければなりません。 この情報の組み合わせを、ソケット、シーケンス番号、ウィンドウ・サイズも含めて、接続と呼びます。 個々の接続は、2 つの側を識別する 1 対のソケットによって固有に指定されます。
優先順位とセキュリティー	**TCP** のユーザーは、通信のセキュリティーと優先順位を指示できます。 これらの機能が必要ない場合は、デフォルト値が使用されます。

TCP パケット・ヘッダーの図は、上記の特性を表します。

図 *13.* 伝送制御プロトコル *(TCP)* パケット・ヘッダー

この図は、**TCP** パケット・ヘッダーに含まれる情報を示します。 個々の説明については、以下を参照してください。

TCP ヘッダーのフィールド定義

ここでは、それぞれの**伝送制御プロトコル** (**TCP**) フィールドについて簡潔に説明します。

項目	説明
Source Port (発信元ポート)	送信元アプリケーション・プログラムのポート番号を識別します。
Destination Port (宛先ポート)	宛先アプリケーション・プログラムのポート番号を識別します。

項目	説明
Sequence Number (シーケンス番号)	このセグメント内にあるデータの最初のバイトのシーケンス番号を指定します。
Acknowledgment Number (肯定応答番号)	受信したデータの最上位バイトの位置を識別します。
Data Offset (データ・オフセット)	セグメントのデータ部分のオフセットを指定します。
Reserved (予約済み)	将来の使用に備えて予約されています。
Code (コード)	次のようにセグメントの目的を識別するための制御ビットです。 **URG** 「Urgent pointer」フィールドが有効です。 **ACK** 「Acknowledgement」フィールドが有効です。 **PSH** セグメントが PUSH を要求します。 **RTS** 接続をリセットします。 **SYN** シーケンス番号を同期化します。 **FIN** 送信側がバイト・ストリームの終わりに到達しました。
Window (ウィンドウ)	宛先が受け入れるデータ量を指定します。
Checksum	セグメントのヘッダーとデータの整合性を検査します。
Urgent Pointer (緊急ポインター)	可能な限り急いで送達するデータを示します。 このポインターは、緊急データの終了位置を指定します。
Options (オプション)	**End of Option List** オプション・リストの終わりを示します。 これは、個々のオプションの終わりでなく、最終オプションで使用されます。 このオプションを使用する必要が生じるのは、これを使用しないと、オプションの終わりが **TCP** ヘッダーの終わりと一致しなくなる場合のみです。 **No Operation** オプション間の境界を示します。 これ以外のオプションの間に使用できます。例えば、後続のオプションの開始位置をワード境界に合わせるために使用できます。 送信側がこのオプションを使用する保証はないので、受信側はオプションがワード境界から始まらない場合でも、そのオプションを処理できるように準備しておかなければなりません。 **Maximum Segment Size** **TCP** が受信できるセグメントの最大サイズを示します。 これは、初期接続要求でのみ送信されます。

TCP へのアプリケーション・プログラミング・インターフェース (API) は、ソケット・インターフェースが提供する一連のライブラリー・サブルーチンから構成されています。

インターネットのアプリケーション・レベル・プロトコル

TCP/IP は、アプリケーション・プログラム・レベルに上位レベルのインターネット・プロトコルをインプリメントしています。

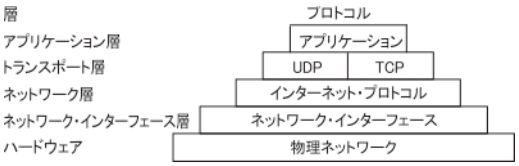


図 *14. TCP/IP* プロトコル群のアプリケーション層

この図は、**TCP/IP** プロトコル群のさまざまな層を示します。 1 番上のアプリケーション層は、アプリケーションで構成されます。 トランスポート層には、**UDP** と **TCP** が含まれます。 ネットワーク層には、ネットワーク (ハードウェア) インターフェースが 含まれます。 1 番下のハードウェア層には、物理ネットワークが含まれます。

アプリケーションが別のホスト上の別のアプリケーションへデータを送信する必要がある場合、アプリケーションは送信に備えてトランスポート・レベルのプロトコルへ情報を送信します。

インターネットのアプリケーション ・レベルの公認プロトコルは次のとおりです。

- **ドメイン名プロトコル**
- **外部ゲートウェイ・プロトコル**
- **ファイル転送プロトコル**
- **Name/Finger プロトコル**
- **Telnet プロトコル**
- **トリビアル・ファイル転送プロトコル**

TCP/IP がインプリメントするプロトコルには、上記以外にも、公認のインターネット・プロトコルでないにもかかわらず、インターネット・コミュニティーにおいてアプリケーション ・プログラム・レベルで一般的に使用される、上位レベルのプロトコルがあります。 それらのプロトコルは次のとおりです。

- **分散コンピューター・ネットワーク (DCN) ローカル・ネットワーク・プロトコル**
- **リモート・コマンド実行プロトコル**
- **リモート・ログイン・プロトコル**
- **リモート・シェル・プロトコル**
- **Wake On LAN プロトコル**
- **RIP (Routing Information Protocol)**
- **タイム・サーバー・プロトコル**

TCP/IP では、これらのアプリケーション・レベル・プロトコルに対するアプリケーション・プログラミング・インターフェース (API) を提供しません。

ドメイン名プロトコル

ドメイン名プロトコル (**DOMAIN**) を使用すれば、ドメイン内のある 1 つのホストを、そのドメイン内の他のホストのネーム・サーバー として機能させることが可能になります。

DOMAIN は、基礎となるプロトコルとして **UDP** や **TCP** を使用して、ローカル・ネットワークが、他のドメインとは関係なく、そのドメイン内でホスト名を割り当てられるようにします。 通常、**DOMAIN** プロトコルでは **UDP** を使用します。 ただし、**UDP** 応答が切り捨てられる場合は **TCP** を使用できます。 **TCP/IP** 内の **DOMAIN** プロトコルは両方をサポートします。

DOMAIN 階層状の命名システムでは、ローカルの resolver ルーチンは、**named** デーモンが維持管理するローカル名レゾリューション・データベースを使用して、インターネット名とアドレスを解決することができます。 ホストが要求した名前がローカル・データベースにない場合、resolver ルーチンはリモート **DOMAIN** ネーム・サーバーに照会します。 どちらの場合も、ネーム・レゾリューション情報が使用できなければ、resolver ルーチンは `/etc/hosts` ファイルを使用してネーム・レゾリューションを試みます。

注 : ローカル・ファイル /etc/resolv.conf が存在する場合、**TCP/IP** は、**DOMAIN** プロトコル用のローカル resolver ルーチンを構成します。 このファイルが存在しない場合、**TCP/IP** は、/etc/hosts データベースを使用するローカル resolver ルーチンを構成します。

TCP/IP は、**named** デーモン内と resolver ルーチン内に **DOMAIN** プロトコルをインプリメントしており、このプロトコルへのアプリケーション・プログラミング・インターフェース (API) は提供しません。

外部ゲートウェイ・プロトコル

外部ゲートウェイ・プロトコル (**EGP**) は、ある自律システム 上の外部ゲートウェイが、他の自律システム上の外部ゲートウェイと経路指定情報を共用できるようにする、メカニズムです。

自律システム

ゲートウェイは、同じ自律システムに常駐する場合は内部隣接 であり、異なる自律システムに常駐する場合は 外部隣接 です。 **EGP** を使用して経路指定情報を交換するゲートウェイは、**EGP** ピア または隣接 と呼ばれます。 自律システムのゲートウェイは、**EGP** を使用して **EGP** 隣接にアクセス情報を提供します。

EGP は、外部ゲートウェイが他の外部ゲートウェイに対して、アクセス情報の交換に同意するよう依頼することを許可します。また、**EGP** 隣接が確実に応答しているかどうかを絶えず検査し、さらに、経路指定更新メッセージを渡すことによって **EGP** 隣接がアクセス情報を交換するのを助けます。

EGP は外部ゲートウェイに対して、そのゲートウェイの属する自律システム内でなければ到達不可能な宛先ネットワークを公示させないように制限します。 したがって、**EGP** を使用している外部ゲートウェイは **EGP** 隣接に情報を渡しますが、その自律システムの外部にある **EGP** 隣接に関するアクセス情報は公示しません。

EGP は他のプロトコルからの経路指定更新メッセージ内の距離メトリックは解釈しません。 **EGP** は、距離フィールドを使用してパスが存在するかどうかを指定します (値 255 は、そのネットワークが到達不可能であることを意味します)。 2 つの経路が両方とも同じ単一の自律システムの中に含まれていなければ、この値を使用して、短い方の経路を計算することはできません。 したがって、**EGP** を経路指定アルゴリズムとして使用することはできません。 その結果、どのネットワークに対しても、外部ゲートウェイからのパスは 1 つしか存在しないことになります。

EGP 経路は、経路を動的に再構成するインターネット・ネットワークの自律システム内で使用できる **Routing Information Protocol** (**RIP**) とは異なり、**/etc/gated.conf** ファイル内にあらかじめ設定されています。 **EGP** は、**IP** が基礎をなすプロトコルであると想定します。

EGP メッセージのタイプ

ここでは、さまざまな外部ゲートウェイ・プロトコル (EGP) メッセージ・タイプを定義します。

項目	説明
Neighbor Acquisition Request	互いに隣接になることを要求するために、外部ゲートウェイによって使用されます。
Neighbor Acquisition Reply	隣接になる要求を受け入れるために、外部ゲートウェイによって使用されます。
Neighbor Acquisition Refusal	隣接になる要求を拒絶するために、外部ゲートウェイによって使用されます。 この拒絶メッセージには、例えば「out of table space」のように拒絶の理由が含まれています。
Neighbor Cease	隣接関係を終了する要求に肯定応答するために、外部ゲートウェイによって使用されます。 この終了メッセージには、例えば「going down」のように終了の理由が含まれています。
Neighbor Cease Acknowledgment	隣接関係を終了する要求に肯定応答するために、外部ゲートウェイによって使用されます。

項目	説明
Neighbor Hello	接続を判別するために、外部ゲートウェイによって使用されます。 あるゲートウェイが「Hello」メッセージを発行し、別のゲートウェイが「I Heard You」メッセージを発行します。
I Heard You	Hello メッセージに応答するために、外部ゲートウェイによって使用されます。 I Heard You メッセージには応答側ゲートウェイのアクセス権が含まれており、そのゲートウェイが到達不能の場合は、例えば、「You are unreachable because of problems with my network interface」のように、アクセス権がない理由も含まれます。
NR Poll	外部ゲートウェイが、他のゲートウェイに到達できるかどうかを隣接ゲートウェイに照会するために使用します。
Network Reachability	NR Poll メッセージに応答するために、外部ゲートウェイによって使用されます。 「Network Reachability」メッセージには、このメッセージ内の各ゲートウェイについて、そのゲートウェイが隣接を介して到達できるアドレスに関する情報が入っています。
EGP Error	正しくないチェックサムが入った EGP メッセージや間違った値が入ったフィールドのある EGP メッセージに応答するために、外部ゲートウェイによって使用されます。

TCP/IP は **gated** サーバー・コマンドに **EGP** プロトコルをインプリメントしており、このプロトコルへのアプリケーション・プログラミング・インターフェース (API) は提供しません。

ファイル転送プロトコル

ファイル転送プロトコル (**FTP**) を使用すると、ホストは機種の異なるホストとの間でデータを転送でき、また、2 つの外部ホスト間で間接的にファイルを転送できます。

FTP は、リモート・ディレクトリーの一覧表示、現行のリモート・ディレクトリーの変更、リモート・ディレクトリーの作成と除去、1 つの要求での複数ファイルの転送などのタスクを提供します。 **FTP** は、外部ホストにユーザー・パスワードとアカウント・パスワードを渡すことによって、転送のセキュリティーを確保します。 **FTP** は主にアプリケーションから使用するよう設計されていますが、対話式のユーザー指向セッションでも使用できます。

FTP は高信頼性ストリーム伝送 (**TCP/IP**) を使用してファイルを送信し、Telnet 接続を使用してコマンドおよび応答を転送します。 **FTP** は、NETASCII、IMAGE、Local 8 などの基本的なファイル・フォーマットも認識できます。

TCP/IP は ftp ユーザー・コマンドと ftpd サーバー・コマンドに **FTP** をインプリメントしており、このプロトコルへのアプリケーション・プログラミング・インターフェース (API) は提供しません。

匿名の ftp ユーザーおよびディレクトリーを作成するときは、匿名 ftp ユーザーのホーム・ディレクトリー (例えば /u/ftp) は root が所有し、書き込み許可を与えないようにしてください (dr-xr-xr-x のように)。スクリプト /usr/samples/tcpip/anon.ftp を使用すると、そのようなアカウント、ファイル、およびディレクトリーを作成できます。

トリビアル・ファイル転送プロトコル

トリビアル・ファイル転送プロトコル (**TFTP**) は、外部ホストからのファイルの読み取りと、外部ホストへのファイルの書き込みを行うことができます。

TFTP は、信頼性のない**ユーザー・データグラム・プロトコル**を使用してファイルを転送するので、通常は **FTP** よりも高速で処理します。 **FTP** と同様に、**TFTP** はファイルを NETASCII 文字として、または 8 ビットのバイナリー・データとして、転送することができます。 **FTP** とは異なり、**TFTP** を使用して、外部ホストのディレクトリーをリストまたは変更することはできません。また、パスワード保護のようなセキュリティー機能も備わっていません。 さらに、パブリック・ディレクトリーの中でだけ、データの書き込みや検索ができます。

TCP/IP は、**tftp** ユーザー・コマンドまたは **utftp** ユーザー・コマンド、および **tftpd** サーバー・コマンドに **TFTP** を実装します。 **utftp** コマンドは、パイプで使用できる形式の **tftpd** コマンドです。 **TCP/IP** は、このプロトコルへの API を提供しません。

Name/Finger プロトコル

Name/Finger プロトコル (**FINGER**) は、アプリケーション・レベルのインターネット・プロトコルで、**finger** コマンドと **fingerd** デーモンの間のインターフェースを 提供します。

fingerd デーモンは、現在ログインしているユーザーの情報を、指定されたリモート・ホストへ戻します。 finger コマンドに特定のホストのユーザーを指定して実行すると、そのユーザーに関する詳細情報を入手できます。 **FINGER** プロトコルは、リモート・ホスト上と要求側ホストに用意しておく必要があります。 **FINGER** は、**伝送制御プロトコル** (158 ページの『伝送制御プロトコル』) をその基礎となるプロトコルとして使用します。

注 : TCP/IP は、このプロトコルへの API を提供しません。

Telnet プロトコル

Telnet プロトコル (**TELNET**) は、端末デバイスと端末向けプロセス間のインターフェースのための 標準的な手段を提供します。

TELNET は、通常、リモート・ホストへログインできるようにする端末エミュレーション・プログラムによって使用されます。 ただし、**TELNET** は、端末間通信やプロセス間通信にも使用できます。 **TELNET** は、プロトコル制御チャネルを確立するために他のプロトコル (例: **FTP**) によっても使用されます。

TCP/IP は、**tn**、**telnet**、または **tn3270** の各ユーザー・コマンドに **TELNET** をインプリメントしています。 **telnetd** デーモンは、**TELNET** への API を提供しません。

TCP/IP は、クライアントとサーバーとの 間で折衝される次の **TELNET** オプションをサポートします。

項目	説明
BINARY TRANSMISSION (**tn3270** セッションで使用)	文字をバイナリー・データとして送信します。
SUPPRESS GO_AHEAD (オペレーティング・システムは GO-AHEAD オプションを抑止)	データの送信側と受信側の事実上の接続時に、送信側が GO_AHEAD オプションを送信する必要がないことを示します。 GO_AHEAD オプションが不要な場合、接続中の当事者はどちらの方向にもそのオプションを抑止すると考えられます。 この処置は、両方向で別個に実行されなければなりません。
TIMING MARK (認識されるが否定応答がある)	以前に送信したデータの処理が完了したことを確認します。
EXTENDED OPTIONS LIST	**TELNET** オプション・リストに別の 256 個のオプションを追加します。 このオプションがない場合、**TELNET** オプションを 256 個だけ使用できます。
ECHO (ユーザーによる変更が可能なコマンド)	既に受信したエコー・データ文字を元の送信側へ送信します。
TERM TYPE	ユーザーの **TELNET** プログラムへ接続された端末のタイプをサーバーが判別できるようにします。
SAK (セキュア・アテンション・キー)	ユーザーとシステムの間の安全な通信に必要な環境を確立します。

項目	説明
NAWS (ウィンドウ・サイズに関して折衝する)	クライアントとサーバーがウィンドウ・サイズについて動的に取り決めるようにします。 このオプションは、ウィンドウ・サイズの変更をサポートするアプリケーションによって使用されます。

注 : TELNET は、バイナリー・モードでない場合でも、ISO 8859 Latin コード・ページをインプリメントするために 8 ビット文字を送信できなければなりません。

分散コンピューター・ネットワーク・ローカル・ネットワーク・プロトコル

自律システムは、1 つの管理部門が管理する、ネットワークとゲートウェイのグループです。

ローカル・ネットワーク・プロトコル (**HELLO**) は、自律システム内で使用するように設計された内部ゲートウェイ・プロトコルです。 (詳しくは、162 ページの『外部ゲートウェイ・プロトコル』を参照してください。) **HELLO** は、接続、経路指定、時間管理に関する情報を保守します。 これを使用すると、ネットワーク内の各コンピューターは、時間遅延に基づいて宛先への最短パスを判別でき、その宛先への経路指定情報を動的に更新できます。

詳しくは、**gated** デーモンを参照してください。

リモート・コマンド実行プロトコル

rexec ユーザー・コマンドと **rexecd** デーモンは、リモート・コマンド実行プロトコルを提供し、ユーザーが互換性のあるリモート・ホスト上でコマンドを実行できるようにします。

詳しくは、**rexec** コマンドおよび **rexecd** デーモンを参照してください。

リモート・ログイン・プロトコル

rlogin ユーザー・コマンドと **rlogind** デーモンは**リモート・ログイン・プロトコル**を提供します。このプロトコルを使用すると、リモート・ホストへログインでき、端末をそのリモート・ホストへ直接接続しているかのように使用できます。

詳しくは、**rlogin** コマンドおよび **rlogind** デーモンを参照してください。

リモート・シェル・プロトコル

rsh ユーザー・コマンドと **rshd** デーモンは、**リモート・コマンド・シェル・プロトコル**を提供します。このプロトコルを使用すると、互換性のある外部ホスト上のシェルをオープンして、コマンドを実行できます。

詳しくは、**rsh** コマンドおよび **rshd** デーモンを参照してください。

Wake On LAN プロトコル

Wake On LAN (WOL) を使用すると、指定したサブネット上の指定した 1 つ以上のアドレスに Magic Packet を送信することにより、中断モードでネットワークに接続している 1 つ以上のホストをウェイクアップできます。

WOL の使用法の詳細については、**wol** コマンドを参照してください。

Routing Information Protocol

Routing Information Protocol (**RIP**) と、これをインプリメントした **routed** デーモンおよび **gated** デーモンは、ゲートウェイ・ホップに基づいて経路指定情報を追跡し、カーネル・ルーティング・テーブル・エントリーを保守します。

詳しくは、**routed** デーモンと **gated** デーモンを参照してください。

タイム・サーバー・プロトコル

timed デーモンは、ホストの時刻を他のホストの時刻と同期化させるために使用されます。

このデーモンは、クライアント/サーバーの概念に基づいています。 詳しくは、**timedc** コマンドおよび **timed** デーモンを参照してください。

割り当て番号

一般のネットワーク環境との互換性のために、インターネットのバージョン、ネットワーク、ポート、プロトコル、およびプロトコル・オプションには、既知の番号が割り当てられます。 また、マシン、ネットワーク、オペレーティング・システム、プロトコル、サービス、および端末にも既知の名前が割り当てられます。

TCP/IP は RFC 1010 の *Assigned Numbers* で定義されている、割り当て番号と割り当て名に準拠しています。

インターネット・プロトコル (**IP**) は、**IP** ヘッダー内に 4 ビットからなるフィールドを定義し、このフィールドが、使用される汎用インターネットワーク・プロトコルのバージョンを識別します。 **IP** の場合、このバージョン番号は 10 進数の 4 です。**TCP/IP** が使用する割り当て番号と割り当て名の詳細については、**TCP/IP** に添付されている /etc/protocols ファイルと /etc/services ファイルを参照してください。 さらに詳細については、RFC 1010 と /etc/services ファイルを参照してください。

TCP/IP ローカル・エリア・ネットワーク・アダプター・カード

ネットワーク・アダプター・カードはネットワーク回線へ物理的に接続されるハードウェアです。 このハードウェアは、物理レベルでのデータの送受信を担当します。

ネットワーク・アダプター・カードは、ネットワーク・アダプターのデバイス・ドライバーによって制御されます。

マシンは、接続したネットワークごとに (ネットワーク・タイプごとでなく) 1 つずつネットワーク・アダプター・カード (または接続) を備えていなければなりません。 例えば、2 つのトークンリング・ネットワークに接続したホストには、2 つのネットワーク・アダプター・カードがなければなりません。

TCP/IP は、次のネットワーク・アダプター・カードと接続を使用します。

- 標準イーサネット・バージョン 2
- IEEE 802.3
- トークンリング
- 非同期通信アダプターとネイティブ・シリアル・ポート
- ファイバー分散データ・インターフェース (FDDI)
- 光シリアル・チャネル変換機構 (*Kernel Extensions and Device Support Programming Concepts* を参照)
- ファイバー・チャネル

イーサネットおよび 802.3 ネットワークのテクノロジーは、同じタイプのアダプターを 使用しています。

各コンピューターには限られた数の拡張スロットがあり、その一部または全部を通信アダプター用に使用できます。 また、各コンピューターは特定のタイプの通信アダプターを限られた数だけサポートします。 これらの制限 (ソフトウェア制限) 内であれば、ユーザーのコンピューターで使用できる拡張スロットの合計数 (ハードウェア制限) まで、アダプターを自由に組み合わせてインストールできます。

システムがサポートする 光シリアル・チャネル変換機構の数に関係なく、**TCP/IP (伝送制御プロトコル/インターネット・プロトコル)** インターフェースは 1 つだけ構成できます。 光シリアル・デバイス・ドライバーは、論理 **TCP/IP** インターフェースが 1 つだけ構成されている場合でも、両方のチャネル変換機構を利用します。

ネットワーク・アダプターのインストール

ネットワーク・アダプターをインストールするには、以下の手順を使用します。

ネットワーク・アダプターをインストールするには、次のようにします。

1. コンピューターをシャットダウンします。 システムをシャットダウンする方法については、**shutdown** コマンドのセクションを参照してください。
2. コンピューターの電源を切ります。
3. コンピューターのカバーを外します。
4. 空きスロットを見付けて、ネットワーク・アダプターを挿入します。 アダプターがスロットに正しく装着されるように慎重に作業します。
5. コンピューターのカバーを取り付けます。
6. 測定エポックを再始動します。

アダプターの管理および構成
トークンリング・アダプターまたはイーサネット ・アダプターを構成し管理するには、次のテーブルに示すタスクを使います。

表 *60.* アダプター・タスクの構成と管理

タスク	SMIT 高速パス	コマンドまたはファイル
アダプターの構成	`smit chgtok` (トークンリング) `smit chgenet` (イーサネット)	1. アダプター名を判別する:[1] `lsdev -C -c adapter -t tokenring -H` または `lsdev -C -c adapter -t ethernet -H` 2. 必要ならリング・スピード (トークンリング) またはコネクター・タイプ (イーサネット) をリセットする。 例: `chdev -l tok0 -a ring_speed=16 -P` または `chdev -l ent0 -a bnc_select=dix -P`
ネットワーク・アダプターのハードウェア・アドレスの判別	`smit chgtok` (トークンリング) `smit chgenet` (イーサネット)	`lscfg -l tok0 -v` (トークンリング)[2] `lscfg -l ent0 -v` (イーサネット)[2]

代替ハードウェア・アドレスの設定	`smit chgtok` (トークンリング) `smit chgenet` (イーサネット)	1. 代替ハードウェア・アドレスを定義する。 次に例を示します。トークンリングの場合:[2、3]
		`chdev -l tok0 -a alt_addr=0X10005A4F1B7F`
		イーサネットの場合:[2、3]
		`chdev -l ent0 -a alt_addr=0X10005A4F1B7F -p`
		2. 代替アドレスの使用を開始する。トークンリングの場合:[4]
		`chdev -l tok0 -a use_alt_addr=yes`
		イーサネットの場合:[4]
		`chdev -l ent0 -a use_alt_addr=yes`

注 :

1. ネットワーク・アダプターの名前は、アダプターをあるスロットから別のスロットへ移動したり、システムから除去したりすると変更される場合があります。 そのような操作を行った場合は、**diag -a** コマンドを入力して構成データベースを更新してください。
2. `tok0` と `ent0` を使用しているアダプター名に置き換えてください。

3. 0X10005A4F1B7F を使用しているハードウェアのアドレスに置き換えてください。
4. この手順を実行したあと、他のホストがアドレス解決プロトコル (ARP) キャッシュをフラッシュして、こちらのホストの新しいハードウェア・アドレスを入手するまで、他のホストとの通信が中断する場合があります。

仮想ローカル・エリア・ネットワーク

VLAN (仮想ローカル・エリア・ネットワーク) は、論理的ブロードキャスト・ドメインと見なすことができます。 VLAN は、実際の物理ネットワーク上のネットワーク・ユーザーのグループを、論理ネットワークのセグメントに分割します。

このインプリメンテーションは、IEEE 802.1Q VLAN タグ付け標準をサポートし、イーサネット・アダプター上で複数の VLAN ID をサポートする機能があります。 それぞれの VLAN ID は、上位層 (IP など) に対する別個のイーサネット・インターフェースと 関連付けられており、ent1、ent2 などのように、VLAN ごとに固有の論理イーサネット・アダプター・インスタンスを作成します。

IEEE 802.1Q VLAN サポートは、サポートされるいずれのイーサネット・アダプターを 介しても構成することができます。 アダプターは、IEEE 802.1Q VLAN をサポートする交換機に接続されている必要があります。

単一システム上で複数の VLAN 論理デバイスを構成することができます。 各 VLAN 論理デバイスは、追加のイーサネット・アダプター・インスタンスを構成します。 これらの論理デバイスを使用することにより、物理イーサネット・アダプターで 使用されるものと同じイーサネット IP インターフェースを 構成することができます。 この場合、**no** オプションの *ifsize* (デフォルトは 8) を増やして、アダプターごとのイーサネット・インターフェースだけでなく、構成するすべての VLAN 論理デバイスも組み込まれるようにする必要があります。 **no** コマンドのドキュメンテーションを参照してください。

各 VLAN は、単一の物理イーサネット・アダプターを 共有する場合でも、異なる最大伝送単位 (MTU) 値をとることができます。

VLAN サポートは、SMIT を使用して管理します。 コマンド・ラインから smit vlan 高速パスを入力して、メインの VLAN メニューから項目を選択してください。 オンライン・ヘルプを使用することができます。

VLAN を構成したら、SMIT またはコマンドを使用して、標準イーサネットには en1、IEEE 802.3 には et1 というように、IP インターフェースを 構成します。

AIX 5.3 以降は、仮想入出力スイッチをメソッドとして 使用する仮想イーサネットをサポートし、POWER5 システムの区画間のメモリー内通信を実行します。 また、スイッチは、IEEE 802.1Q タグ付けをサポートするので、仮想イーサネット・アダプターはスイッチ 上の異なる VLAN に所属することが可能です。 仮想イーサネット・アダプターは、ハードウェア管理コンソール (HMC) を使用して、区画上で作成および構成されます。 区画が一度作成されると、デバイスのスキャン時に、オープン・ファームウェア・ツリーに仮想イーサネット・アダプターが 表示されます。 検出後、仮想イーサネット・アダプターは、物理イーサネット・アダプターと 同じように構成され、使用されます。 詳しくは、POWER5 システムのハードウェアに関する資料を参照してください。

注 :

1. 指定アダプターで 既に使用されている VLAN ID 値を構成すると、次のエラーを示して構成が失敗します。

```
Method error (/usr/lib/methods/chgvlan):
  0514-018 The values specified for the following attributes
           are not valid:
           vlan_tag_id    VLAN Tag ID
```

2. ユーザー (例: IP インターフェース) が現在 VLAN 論理デバイスを使用していると、VLAN 論理デバイスを除去しようとしてもすべて失敗します。 次のようなメッセージが表示されます。

```
Method error (/usr/lib/methods/ucfgcommo):
        0514-062 Cannot perform the requested function because the
                 specified device is busy.
```

論理 VLAN デバイスを除去するには、最初にそのユーザーを切り離します。 例えば、ユーザーが IP インターフェース en1 の場合は、次のコマンドを使用することができます。

```
ifconfig en1 detach
```

次に、SMIT TCP/IP メニューを使用してネットワーク・インターフェースを 除去します。

3. ユーザー (例えば IP インターフェース) が現在 VLAN 論理デバイスを使用していると、VLAN 特性 (VLAN タグ ID や基本アダプター) を変更しようとしても、すべて失敗します。 次のようなメッセージが表示されます。

```
Method error (/usr/lib/methods/chgvlan):
        0514-062 Cannot perform the requested function because the
                 specified device is busy.
```

論理 VLAN デバイスを変更するには、最初にそのユーザーを切り離します。 例えば、ユーザーが IP インターフェース en1 の場合は、次のコマンドを使用することができます。

```
ifconfig en1 detach
```

続いて、SMIT TCP/IP メニューを使用して再度ネットワーク・インターフェースを 追加します。

VLAN トラブルシューティング

tcpdump および **trace** を使用して、VLAN のトラブルシューティングを 行うことができます。

転送パケットのタイプごとのトレース・フック ID が示されます。

項目	説明
送信パケット	3FD
受信パケット	3FE
その他のイベント	3FF

entstat コマンドは、VLAN が構成されている物理アダプターについての 集約統計を提供します。 特定の VLAN 論理デバイスについての 個々の統計は提供しません。

VLAN 制約事項

VLAN ではリモート・ダンプはサポートされません。 また、Cisco Systems の Etherchannel の作成に VLAN 論理デバイスを使用することはできません。

TCP/IP ネットワーク・インターフェース

TCP/IP ネットワーク・インターフェース 層は、ネットワーク層の IP データグラムをフォーマット 設定して、特定のネットワーク・テクノロジーが解釈して送信できるパケットにします。

ネットワーク・インターフェースは、ネットワーク固有のデバイス・ドライバーおよび IP 層と通信するネットワーク固有のソフトウェアであり、その目的は、存在する可能性があるすべてのネットワーク・アダプターへの一貫したインターフェースを IP 層に提供することにあります。

IP 層は、送信するパケットの宛先アドレスに基づいて、適切なネットワーク・インターフェースを 選択します。 個々のネットワーク・インターフェースには、ネットワーク・アドレスがあります。 ネットワーク・インターフェース 層は、メッセージを宛先に送達するのに必要なリンク層プロトコル・ヘッダーを追加したり除去したりします。 **ネットワーク・アダプター** のデバイス・ドライバーは、ネットワーク・アダプター・カードを制御します。

通常、ネットワーク・インターフェースはネットワーク ・アダプターへ関連付けられていますが、関連付けられていなくてもかまいません。 例えば、ループバック・インターフェースにはネットワーク ・アダプターが関連付けられていません。 マシンは、接続したネットワークごとに (ネットワーク・タイプごとでなく) 1 つずつネットワーク・アダプター・カードを備えていなければなりません。 しかし、マシンに必要なネットワーク・インターフェース・ソフトウェアのコピーは、使用するネットワーク・アダプターごとに 1 つだけです。 例えば、2 つのトークンリング・ネットワークに接続したホストには、2 つのネットワーク・アダプター・カードがなければなりません。 ただし、**トークンリング**・ネットワーク・インターフェ

ース・ソフトウェアのコピーとトークンリング・デバイス・ドライバーのコピーは、どちらも 1 つずつあれば十分です。

TCP/IP は、次のタイプのネットワーク・インターフェースをサポートします。

- 標準イーサネット・バージョン 2 (en)
- IEEE 802.3 (et)
- トークンリング (tr)
- **シリアル回線インターネット・プロトコル (SLIP)**
- ループバック (lo)
- FDDI
- 光シリアル (so)
- **Point-to-Point プロトコル (PPP)**
- 仮想 IP アドレス (vi)

イーサネット、802.3、トークンリングの各インターフェースは、ローカル・エリア・ネットワーク (LAN) で使用します。 **SLIP** インターフェースは、シリアル接続で使用します。 ループバック・インターフェースは、ホストがそれ自体にメッセージを返送するために使用します。 光シリアル・インターフェースは、光シリアル・リンク・デバイス・ハンドラーを使用する光学式 Point-to-Point ネットワーク用です。 **Point-to-Point プロトコル**は、モデムを介して別のコンピューターやネットワークに接続している場合に最も頻繁に使用されます。 仮想 IP アドレス・インターフェース (仮想インターフェース とも呼ばれる) は、特定のネットワーク・アダプターには関連付けられていません。 仮想インターフェースの 複数インスタンスをホスト上で構成できます。 仮想インターフェースを 構成する場合、アプリケーションが異なるインターフェースを選択していない限り、最初の仮想インターフェースのアドレスが 発信元アドレスになります。 仮想 IP アドレスを発信元アドレスとして使用するプロセスは、宛先への最善の経路を提供するネットワーク・インターフェースを 介してパケットを送信できます。 仮想 IP アドレス宛ての着信パケットは、着信時に使用されたインターフェースに 関係なく、プロセスに送られます。

ネットワーク・インターフェースの自動構成

新しいネットワーク・アダプターをシステムに 物理的にインストールするときにオペレーティング・システムはそのアダプター 用の適切なネットワーク・インターフェースを 自動的に追加します。

例えば、トークンリング・アダプターをシステムにインストールする 場合、オペレーティング・システムはそのアダプターに `tok0` という名前を割り当て、`tr0` という名前のトークンリング・ネットワーク・インターフェースを 追加します。 イーサネット・アダプターをシステムにインストールする 場合には、オペレーティング・システムはそのアダプターに `ent0` という名前を割り当て、`en0` という名前のイーサネット・バージョン 2 インターフェースと `et0` という名前の IEEE 802.3 インターフェースを 追加します。

ほとんどの場合、アダプター名とネットワーク・インターフェース 名は 1 対 1 で対応しています。 例えば、トークンリング・アダプター `tok0` はインターフェース `tr0` に対応し、アダプター `tok1` はインターフェース `tr1` に対応します。 同様に、イーサネット・アダプター `ent0` はインターフェース `en0` (イーサネット・バージョン 2 の場合) と `et0` (IEEE 802.3 の場合) に対応し、アダプター `ent1` はインターフェース `en1` (イーサネット・バージョン 2 の場合)、`et1` (IEEE 802.3 の場合) に対応します。

注 : 通常の環境では、ネットワーク・インターフェースを 手動で削除したり追加したりする必要は生じません。 しかし、一部の問題判別手順でそうしなければならない場合もあります。 その場合は、SMIT 高速パス `smit inet` を使用してインターフェースを 削除し、適切なインターフェースを 再度追加してください。

TCP/IP 構成のデフォルト値

各システムの始動時に、オペレーティング・システムは ODM データベース内の情報に基づいて自動的にネットワーク・インターフェース・ソフトウェアを構成します。 ネットワーク・インターフェースは、当初はデフォルト値で構成されています。

特定のネットワーク・インターフェースを 介して通信するには、IP アドレスを設定しなければなりません。 ユーザーが設定しなければならない属性はこれだけです。 その他の必要なすべての属性には、デフォルト値を使用できます。 個々のネットワーク・インターフェースのデフォルト 値は、次のとおりです。

TCP/IP イーサネットのデフォルト値

有効なイーサネット・ネットワーク・アダプター属性の値は、SMIT の「ネットワーク・インターフェースの選択」メニューを使用して変更できます。

属性	デフォルト値	設定可能な値
netaddr		
state	down	up、down、detach
arp	yes	yes、no
netmask		
broadcast		

次に示すのは、SMIT の「ネットワーク・インターフェース ・ドライバー」メニューを使用して変更できる、有効なイーサネット・ネットワーク・デバイス・ドライバーの属性とそのデフォルト値です。

属性	デフォルト値	設定可能な値
mtu	1500	60 から 1500

TCP/IP 802.3 のデフォルト値

有効な 802.3 ネットワーク・アダプター属性の値は、SMIT の「ネットワーク・インターフェースの 選択」メニューを使用して変更できます。

属性	デフォルト値	設定可能な値
netaddr		
state	down	up、down、detach
arp	yes	yes、no
netmask		
broadcast		

次に示すのは、SMIT の「ネットワーク・インターフェース ・ドライバー」メニューを使用して変更できる、有効な 802.3 ネットワーク・デバイス・ドライバーの属性とそのデフォルト値です。

属性	デフォルト値	設定可能な値
mtu	1492	60 から 1492

TCP/IP トークンリングのデフォルト値

有効なトークンリング・ネットワーク・アダプター属性の値は、SMIT の「ネットワーク・インターフェースの選択」メニューを使用して変更できます。

属性	デフォルト値	設定可能な値
netaddr		
netmask		
state	down	up、down、detach
arp	yes	yes、no
hwloop	no	yes、no
netmask		
broadcast		

属性	デフォルト値	設定可能な値
allcast	no	yes、no

次に示すのは、SMIT の「ネットワーク・インターフェース・ドライバー」メニューを使用して変更できる、有効なトークンリング・ネットワーク・デバイス・ドライバーの属性とそのデフォルト値です。

属性	デフォルト値	設定可能な値
mtu (4Mbps)	1500	60 から 4056
mtu (16Mbps)	1500	60 から 17960

注 : ブリッジを介して運用する場合は、最大伝送単位 (MTU) のデフォルト値 1500 を、経路指定制御フィールド内のブリッジによって公示される最大情報フィールド (最大 I フレーム) の値よりも 8 だけ小さい値に変更しなければなりません。 例えば、最大 I フレーム値が経路指定制御フィールド内で 1500 である場合、MTU サイズは 1492 に設定しなければなりません。 これは、トークンリング・ネットワーク・インターフェースの場合だけです。 詳しくは、482 ページの『トークンリング/トークンリングのブリッジに関する TCP/IP 問題』を参照してください。

IBM® 16/4 PowerPC トークンリング・アダプター (ISA) を使用する場合、MTU は 2000 に制限されます。

TCP/IP SLIP のデフォルト値

有効な SLIP ネットワーク・アダプター属性の値は、SMIT の「ネットワーク・インターフェースの 選択」メニューを使用して変更できます。

属性	デフォルト値	設定可能な値
netaddr		
dest		
state	up	up、down、detach
netmask		

次に示すのは、SMIT の「ネットワーク・インターフェース・ドライバー」メニューで表示される、有効な SLIP ネットワーク・デバイス・ドライバーの属性とそのデフォルト値です。

属性	デフォルト値	設定可能な値
mtu	1006	60 から 4096

TCP/IP 光シリアルのデフォルト値

有効な光シリアル・ネットワーク・チャネル・コンバーターの値は、SMIT の「ネットワーク・インターフェースの選択」メニューを使用して変更できます。

属性	デフォルト値	設定可能な値
netaddr		
state	down	up、down、detach
netmask		

次に示すのは、SMIT の「ネットワーク・インターフェース・ドライバー」メニューで表示される、有効な光シリアル・ネットワーク・デバイス・ハンドラーの属性とそのデフォルト値です。

属性	デフォルト値	設定可能な値
mtu	61428	1 から 61428

同一ネットワーク上の複数ネットワーク・インターフェースの意味

複数ネットワーク・インターフェースが 1 つのネットワークに接続されている場合、それぞれのインターフェースは固有の IP アドレスを持っている必要があります。

マルチパス経路指定フィーチャーによって、同じサブネット上のマルチパス・インターフェースの IP ルーティング・テーブルに、経路指定を追加できます。 これによって、発信トラフィックは、1 つのインターフェースのみを通るのではなく、インターフェースを 交互に代えることができます。

ネットワーク・インターフェースの管理

ネットワーク・インターフェースを 管理するには、WSM ネットワーク、高速パス (アプリケーション)、またはこの表に示すタスクを使用します。

表 *61.* ネットワーク・インターフェースの管理のタスク

タスク	SMIT 高速パス	コマンドまたはファイル
すべてのネットワーク・デバイスをリストする	`smit lsinet`	`lsdev -C -c if`
ネットワーク・デバイスを構成する	`smit chinet`	**ifconfig** コマンドおよび `rc.net` ファイルを参照
ネットワーク・インターフェース情報をリモート・マウントの `/usr` に変更する	`smit chdev`[1,2]	`chgif`[1,2]
ネットワーク・インターフェースの統計を入手する		`netstat -v`

注 :

1. リモート・マウントの `/usr` からの変更は、ネットワークが再始動されるか、**ifconfig** コマンドを使用して直ちに変更を有効にするまで、情報データベース (ODM) に影響するだけです。
2. リモート・マウントの `/usr` を使用する場合、使用中のインターフェースは、ライブラリー、コマンド、およびカーネルの場所となるので、そのインターフェースを 修正しないよう注意が必要です。

インターフェース固有のネットワーク・オプション

TCP/IP インターフェースは、良好で高速のネットワーク・パフォーマンス (100 Mb 以上) を提供できるように特別に調整する必要があります。 この作業には、複数のネットワーク・インターフェース、ならびに、従来 **TCP/IP** インターフェースと 高速 TCP/IP インターフェースの 組み合わせを単一システム 上で使用できるということから、複雑です。

AIX オペレーティング・システムでは、インターフェース 固有のネットワーク・オプション (ISNO) を使用すれば、システム管理者は、最高のパフォーマンスとなるよう、それぞれの **TCP/IP** インターフェースを 個別に調整できます。

サポートされているインターフェースごとに 5 つの ISNO パラメーター、つまり、**rfc1323**、**tcp_nodelay**、**tcp_sendspace**、**tcp_recvspace**、および **tcp_mssdflt** が用意されています。 設定されると、これらのパラメーターの値は、**no** コマンドで設定した同じ名前のシステム・ワイド・パラメーターをオーバーライドします。 ISNO オプションが特定インターフェースに 対して設定されていない場合は、システム・ワイド・オプションが使用されます。 アプリケーションが **setsockopt** サブルーチンを使用してオプションを特定ソケットに対して設定していた場合、それらのオプションは ISNO をオーバーライドします。

no コマンドで設定されたネットワーク・オプション **use_isno** を有効にするためには、ISNO に対して値 1 を指定する必要があります。 **use_isno** のデフォルト値は 1 です。

一部の高速アダプターは、デフォルトによって ISNO パラメーターが ODM データベース内に設定されています。

ギガビット・イーサネット・インターフェースは、9000 という MTU を使用するよう構成されている場合、デフォルトの設定により、次の ISNO 値を使用します。

名前	AIX 4.3.3 値	AIX 4.3.3 (4330-08) 値	AIX 5.1 (以降の) 値
tcp_sendspace	131072	262144	262144
tcp_recvspace	92160	131072	131072
rfc1323	1	1	1

ギガビット・イーサネット・インターフェースは、1500 という MTU を使用するよう構成されている場合、デフォルトの設定により、次の ISNO 値を使用します。

名前	AIX 4.3.3 値	AIX 4.3.3 (4330-08) 値	AIX 5.1 (以降の) 値
tcp_sendspace	65536	131072	131072
tcp_recvspace	16384	65536	65536
rfc1323	0	未設定	未設定

FDDI インターフェースは、4352 という MTU を使用するよう構成されている場合、デフォルトの設定により、次の ISNO 値を使用します。

名前	値
tcp_sendspace	45046
tcp_recvspace	45046

ISNO パラメーターは、SMIT を使用して表示または変更できません。 これらは、**chdev** コマンドや **ifconfig** コマンドを使用して設定できます。 **ifconfig** コマンドは、次のリブートまで値だけを変更します。 **chdev** コマンドは、後続のリブートで使用されるように、ODM データベース内の値を変更します。 **lsattr** コマンドや **ifconfig** コマンドを使用して、現在の値を表示することができます。

次の例のコマンドを使用して、まずシステムとインターフェースのサポートを 検査してから、新しい値を設定および検証することができます。

1. **no** コマンドと **lsattr** コマンドを使用して、汎用システムとインターフェース ・サポートを検査します。
   - 次のようなコマンドを使用して、**use_isno** オプションが使用可能であることを確認します。

     ```
     $ no -a | grep isno
              use_isno=1
     ```

   - 次のような **lsattr -El** コマンドを使用して、インターフェースが 新しい 5 つの ISNO をサポートすることを確認します。

     ```
     $ lsattr -E -l en0 -H
               attribute   value  description
               rfc1323                     N/A
               tcp_nodelay                 N/A
               tcp_sendspace               N/A
               tcp_recvspace               N/A
               tcp_mssdflt                 N/A
     ```

2. **ifconfig** コマンドか **chdev** コマンドを使用して、インターフェース 固有の値を設定します。 **ifconfig** コマンドは、一時的に値を設定します。テストにはお勧めの方法です。 **chdev** コマンドは ODM を変更するので、カスタマイズ済みの値はリブート後も有効です。
   - 次のいずれかを使用して、**tcp_recvspace** と **tcp_sendspace** を 64K に設定し、**tcp_nodelay** を使用可能にします。

     ```
     $ ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1
     $ chdev -l en0 -a tcp_recvspace=65536 -a tcp_sendspace=65536 -a tcp_nodelay=1
     ```

- 別の方法として、**no** コマンドが rfc1323=1 グローバル値を報告すると想定した場合、root ユーザーは、次のコマンドを使用して、en0 を介するすべての接続に関して、**rfc1323** をオフにすることができます。

```
$ ifconfig en0 rfc1323 0
$ chdev -l en0 -a rfc1323=0
```

3. 次の例で示すように、**ifconfig** コマンドか **lsattr** コマンドを使用して、設定を検証します。

```
$ ifconfig en0 <UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT>
          en0: flags=e080863
              inet 9.19.161.100 netmask 0xffffff00 broadcast 9.19.161.255
              tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1 rfc1323 0
$ lsattr -El en0
          rfc1323         0           N/A               True
          tcp_nodelay     1           N/A               True
          tcp_sendspace   65536       N/A               True
          tcp_recvspace   65536       N/A               True
          tcp_mssdflt                 N/A               True
```

TCP/IP アドレッシング

TCP/IP には IP アドレッシング方式が組み込まれており、ユーザーとアプリケーションはこの方式を使用して通信相手である特定のネットワークまたはホストを識別できます。

IP アドレスは郵便の住所のように機能し、これを使用すると選択した宛先までデータを経路指定できます。**TCP/IP** は、ネットワーク、サブネットワーク、ホスト、ソケットなどにアドレスを割り当てる規格や、ブロードキャストやローカル・ループバック用に特別のアドレスを使用する規格を備えています。

IP アドレスは、ネットワーク・アドレスとホスト (またはローカル) アドレスから形成されます。 この 2 つの部分からなるアドレスを使用すると、送信側はネットワークを指定できるほかに、そのネットワーク上にある特定のホストも指定できます。 1 つのネットワークが他のインターネット・ネットワークへ接続されると、そのネットワークに 1 つの固有な公認ネットワーク・アドレスが割り当てられます。 ただし、他のインターネット・ネットワークへ接続しないローカル・ネットワークには、ローカルでの使用に便利な任意のネットワーク・アドレスを割り当てることができます。

IP アドレッシング方式は、インターネット・プロトコル (IP) アドレスと 2 つの特殊な IP アドレス、つまりブロードキャスト・アドレスとループバック・アドレスから構成されます。

インターネット・アドレス

インターネット・プロトコル (IP) は、2 つの部分からなる 32 ビットのアドレス・フィールドを使用します。

この 32 ビットは、次のような 4 つのオクテット に分割されます。

```
01111101 00001101 01001001 00001111
```

これらの 2 進数は、次のように変換できます。

```
125 13 73 15
```

IP アドレスの 2 つの部分は、ネットワーク・アドレス部分とホスト・アドレス部分です。 このため、リモート・ホストは情報を送信するときに、リモート・ネットワークとリモート・ネットワーク上のホストの両方を指定できます。 慣例として、ホスト番号 0 は、そのネットワーク自体を表すために使用されます。

TCP/IP は、クラス A、クラス B、クラス C という 3 クラスの IP アドレスをサポートします。これらの IP アドレス・クラスは、32 ビットのアドレスの割り当て方によって区別されます。 ネットワークへ割り当てられるアドレス・クラスは、そのネットワークのサイズによって決まります。

クラス *A* アドレス

クラス A アドレスは、8 ビットのネットワーク・アドレスと 24 ビットのローカルまたはホスト・アドレスで構成されます。

ネットワーク・アドレスの最初のビットは、ネットワーク・クラスを示すために専用に使用され、残りの 7 ビットが実際のネットワーク・アドレスとして使用されます。 2 進数の 7 ビットで表現できる最高の数値は 128 なので、128 個のクラス A のネットワーク・アドレスが使用できます。 この 128 個のネットワ

ーク・アドレスのうち、2 つは特別な用途に予約されています。つまり、ネットワーク・アドレス 127 はローカル・ループバック・アドレスで、すべてが 1 のネットワーク・アドレスはブロードキャスト・アドレスを示します。

したがって、126 個のクラス A ネットワーク・アドレスと、16,777,216 個のローカル・ホスト・アドレスが使用できます。 クラス A アドレスでは、最上位ビットは 0 に設定されます。

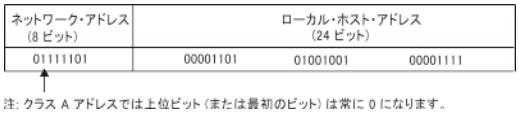


図 *15.* クラス *A* アドレス

この図は、典型的なクラス A アドレスの構造を示します。 最初の 8 ビットでは、ネットワーク・アドレス (先頭は常にゼロ) を指定します。 残りの 24 ビットでは、ローカル・ホスト・アドレスを指定します。

クラス A アドレスの第 1 オクテットは 1 から 126 の範囲内にあります。

クラス *B* アドレス

クラス B アドレスは、16 ビットのネットワーク・アドレスと 16 ビットのローカルまたはホスト・アドレスで構成されます。

ネットワーク・アドレスの最初の 2 ビットは、ネットワーク・クラスを示すために専用に使用され、残りの 14 ビットが実際のネットワーク・アドレスに使用されます。 したがって、16,384 個のネットワーク・アドレスと 65,536 個のローカル・ホスト・アドレスが使用できます。 クラス B アドレスでは、最上位ビットは 1 と 0 に設定されます。

図 *16.* クラス *B* アドレス

この図は、典型的なクラス B アドレスの構造を示します。 最初の 16 ビットでは、ネットワーク・アドレスを指定します。 2 つの最上位ビットは、常に、1 とゼロです。 残りの 16 ビットでは、ローカル・ホスト・アドレスを指定します。

クラス B アドレスの第 1 オクテットは 128 から 191 の範囲内にあります。

クラス *C* アドレス

クラス C アドレスは、24 ビットのネットワーク・アドレスと 8 ビットのローカル・ホスト・アドレスで構成されます。

ネットワーク・アドレスの最初の 3 ビットはネットワーク・クラスを示し、残りの 21 ビットが実際のネットワーク・アドレス用です。 したがって、2,097,152 個のネットワーク・アドレスと、256 個のローカル・ホスト・アドレスが使用できます。 クラス C アドレスでは、最上位ビットは 1-1-0 にセットされます。

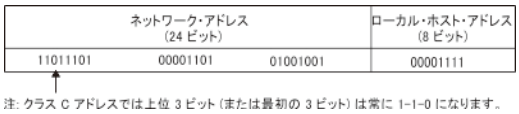


図 *17.* クラス *C* アドレス

この図は、典型的なクラス C アドレスの構造を示します。最初の 24 ビットにネットワーク・アドレスが格納されます (3 つの最上位ビットは常に 1-1-0 になります)。残りの 8 ビットでは、ローカル・ホスト・アドレスを指定します。

言い換えれば、クラス C アドレスの第 1 オクテットは 192 から 223 の範囲内にあります。

使用するネットワーク・アドレス・クラスを決定する場合は、ネットワーク上のローカル・ホストの予定数と、その組織内のサブネットワークの予定数を考慮に入れる必要があります。 組織が小規模で、そのネットワークのホスト数が 256 未満の場合は、多くの場合、クラス C アドレスで十分です。 組織が大規模の場合は、クラス B またはクラス A のアドレスのほうが適切な場合があります。

注 : クラス D (最上位ビットが 1-1-1-0) アドレスは、マルチキャスト・アドレス用で、このオペレーティング・システムの UDP/IP によってサポートされます。

マシンはアドレスをバイナリー・コードで読み取ります。 インターネット・ホスト・アドレスの慣例的な表記法は小数点付き *10* 進数 で、32 ビットのアドレスが 4 つの 8 ビット・フィールドに分割されます。 これを 2 進数で表すと、次のようになります。

```
0001010 00000010 00000000 00110100
```

上記の 2 進値は、次のように表現できます。

`010.002.000.052` または `10.2.0.52`

ここで、個々のフィールドの値は 10 進数として指定されており、各フィールドはピリオドで区切られています。

注 : **hostent** コマンドは、.08、.008、.09、.009 の各アドレスを認識できます。 先行ゼロが付いたアドレスは 8 進数として解釈され、8 進の数表示には 8 または 9 という数字は含まれません。

TCP/IP では、ネットワーク上にある個々のネットワーク・インターフェース (アダプター) ごとに 1 つずつ固有の IP アドレスが必要です。 これらのアドレスは、構成データベース内のエントリーによって決定されます。このエントリーは、`/etc/hosts` ファイル内のエントリーに適合しなければならず、ネットワークがネーム・サーバーを使用する場合は **named** データベース内のエントリーに適合しなければなりません。

ゼロを使用したインターネット・アドレス

C クラスの IP アドレスにホスト・アドレス部分として 0 が入っている (例えば、192.9.200.0) 場合、TCP/IP はネットワーク上にワイルドカード・アドレスを送信します。

クラス C アドレスが 192.9.200.*X* (*X* は 0 から 254 の値を表す) であるすべてのマシンは、この要求に応答します。 その結果、存在しないマシンへの要求がネットワーク上にあふれます。

同様に、129.5.0.0 のようなクラス B アドレスでも問題が発生します。 クラス B アドレスが 129.5.*X*.*X* (*X* は 0 から 254 の値を表す) であるすべてのマシンは、その要求に応答する義務があります。 その場合、クラス B アドレスはクラス C アドレスより大きなネットワークで使用されるので、クラス C ネットワークの場合よりずっと多くの存在しないマシンへの要求がネットワーク上にあふれます。

サブネット・アドレス

サブネット・アドレッシングを使用すると、複数ネットワークからなる自律システム は同じ IP アドレスを共用できます。

また、TCP/IP のサブネットワーク機能により、1 つのネットワークを複数の論理ネットワーク (サブネット) に分割できます。 例えば、ある組織が 1 つのインターネット・ネットワーク・アドレスを使用し、そのアドレスを組織の外部のユーザーに知らせておく一方、内部ではその組織のネットワークを部門別に 複数のサブネットに構成することもできます。 いずれにしても、必要なインターネット・ネットワーク・アドレスが少なくて済む一方、ローカル経路指定機能は強化されます。

標準インターネット・プロトコル (IP) アドレスには、ネットワーク・アドレスとローカル・アドレスの 2 つの部分があります。 サブネットを使用できるようにするため、IP アドレスのローカル・アドレス部分はサブネット番号とホスト番号に分割されます。 各サブネットが識別されるので、ローカルな自律システムからメッセージを正しい経路で確実に送ることができます。

8 ビットのネットワーク・アドレスと 24 ビットのローカル・アドレスからなる基本的なクラス A の IP アドレスでは、ローカル・アドレスはネットワーク上にある特定のホスト・マシンを識別します。

ネットワーク・アドレス (8 ビット)	ローカル・ホスト・アドレス (24 ビット)		
01111101	00001101	01001001	00001111

図 *18.* クラス *A* アドレス

この図は、典型的なクラス A アドレスの構造を示します。 最初の 8 ビットでは、ネットワーク・アドレス (先頭は常にゼロ) を指定します。 残りの 24 ビットでは、ローカル・ホスト・アドレスを指定します。

このクラス A の IP アドレスのサブネット・アドレスを作成するため、ローカル・アドレスは物理ネットワーク (またはサブネット) を識別する番号と、サブネット上のホストを識別する番号に分割できます。 送信側は公示されたネットワーク・アドレスへメッセージを 経路指定し、ローカル・システムはサブネットと

ホストへメッセージを経路指定することに 責任を負います。 ローカル・アドレスをサブネット・アドレスとホスト・アドレスに分割する方法を 決めるときには、サブネットの数とサブネット上のホストの数を考慮に入れなければなりません。

次の図では、ローカル・アドレスは 12 ビットのサブネット・アドレスと、12 ビットのホスト・アドレスに分割されています。

ネットワーク・アドレス (8 ビット)	ローカル・ホスト・アドレス (24 ビット)	
ネットワーク・アドレス	サブネット・アドレス	ホスト・アドレス
01111101	00001101 0100	1001 00001111

注: クラス A アドレスでは上位ビット (または最初のビット) は常に 0 になります。

図 *19.* クラス *A* アドレスと、対応するサブネット・アドレス

この図は、典型的なクラス A アドレスの構造を示します。 最初の 8 ビットでは、ネットワーク・アドレス (先頭は常にゼロ) を指定します。 残りの 24 ビットでは、ローカル・ホスト・アドレスを指定しますが、この最初の 8 ビットはサブネット・アドレスを指定し、最後の 8 ビットはホスト・アドレスを指定します。

サブネット・アドレスとホスト・アドレスの割り当て方法は柔軟です。 ローカル・アドレスのビットは、編成とそのネットワーク構造のニーズおよび将来の成長予測に合わせて分割できます。 制限事項は次の点だけです。

- `network_address` はネットワークの IP アドレスです。
- `subnet_address` は、各ネットワークを示す一定幅のフィールドです。
- `host_address` は、最低の幅が 1 ビットのフィールドです。

「`subnet_address`」フィールドの幅が 0 の場合、そのネットワークはサブネットに組み込まれず、そのネットワークへのアドレッシングはインターネット・ネットワーク・アドレスを使用して行われます。

サブネットを識別するビットはビット・マスクによって指定されます。このため、それらのビットがアドレス内で隣接する必要はありません。 しかし、一般的には、サブネット・ビットは隣接しているのが望ましく、ローカル・アドレスの最上位ビットとして配置されているのが望ましいものです。

サブネット・マスク

ホストが宛先にメッセージを送信する場合、システムは宛先が送信元と同じネットワーク上にあるかどうか、つまり、いずれかのローカル・インターフェースを 介して宛先に直接到達できるかどうかを判別しなければなりません。 システムはサブネット・マスク を使用して宛先アドレスとホスト・アドレスを比較します。

宛先がローカルでない場合、システムはゲートウェイにメッセージを 送信します。 ゲートウェイは同じ比較を行って宛先アドレスがローカルに到達できるネットワーク上にあるかどうかを調べます。

サブネット・マスクは、サブネット分割方式をシステムに知らせます。 このビット・マスクは、IP アドレスのネットワーク・アドレス部分とサブネット・アドレス部分で構成されます。

ネットワーク・アドレス (8 ビット)	ローカル・ホスト・アドレス (24 ビット)	
ネットワーク・アドレス	サブネット・アドレス	ホスト・アドレス
01111101	00001101 0100	1001 00001111

クラス A アドレスと、対応するサブネット・アドレス

ネットワーク・アドレス (8 ビット)	ローカル・ホスト・アドレス (24 ビット)	
ネットワーク・アドレス	サブネット・アドレス	ホスト・アドレス
サブネット・マスク		ホスト・アドレス
01111101	00001101 0100	1001 00001111

クラス A アドレスと、対応するサブネット・マスク

図 *20.* クラス *A* アドレスと、対応するサブネット・アドレス

この図は、典型的なクラス A アドレスの構造を示します。 最初の 8 ビットでは、ネットワーク・アドレス (先頭は常にゼロ) を指定します。 残りの 24 ビットでは、ローカル・ホスト・アドレスを指定しますが、この最初の 8 ビットはサブネット・アドレスを指定し、最後の 8 ビットはホスト・アドレスを指定します。

例えば、上記の例で定義された分割方式を備えたクラス A アドレスのサブネット・マスクは、この図のように表されます。

サブネット・マスクは、IP アドレスと同じように 4 バイトで 1 セットになっています。 サブネット・マスクは、ネットワークおよびサブネットワーク・アドレスのビット位置に対応する上位ビット (1 の部分) と、ホスト・アドレスのビット位置に対応する下位ビット (0 の部分) から構成されます。 上記のアドレスのサブネット・マスクは、次の図のように表されます。

ネットワーク・アドレス (8 ビット)	ローカル・ホスト・アドレス (24 ビット)	
ネットワーク・アドレス	サブネット・アドレス	ホスト・アドレス
11111111	11111111 1111	0000 00000000

図 *21.* サブネット・マスクの例

この図は、サブネット・マスク構造の例を示します。 最初の 8 ビットでは、ネットワーク・アドレスを指定します。 残りの 24 ビットでは、ローカル・ホスト・アドレスを指定しますが、この最初の 8 ビットはサブネット・アドレスを指定し、最後の 8 ビットはホスト・アドレスを指定します。

アドレスの比較

宛先アドレスとローカル・ネットワーク・アドレスは、送信元ホストのサブネット・マスク上で論理 AND と排他 OR を実行することによって比較されます。

比較プロセスの概要を次に示します。

1. 宛先アドレスとローカル・サブネット・アドレスのマスクとの論理 AND を実行します。
2. 前記操作の結果とローカル・インターフェースのローカル ・ネット・アドレスとの排他 OR を実行します。 結果がすべて 0 である場合、その宛先は、ローカル・インターフェースを 介して直接到達できると見なされます。
3. 自律システムに複数のインターフェースがある 場合 (複数の IP アドレスがある場合) は、各ローカル・インターフェースごとにこの 比較プロセスが繰り返されます。

例えば、T125 というホスト・ネットワーク用に 2 つのローカル・インターフェースが 定義されているとします。 その IP アドレスと、そのアドレスの 2 進表記は、次の例のようになります。

```
CLASS A   73.1.5.2    =  01001001 00000001 00000101 00000010

CLASS B   145.21.6.3  =  10010001 00010101 00000110 00000011
```

これらのローカル・ネットワーク・インターフェースに 対応するサブネット・マスクは、次の例に示すとおりです。

```
CLASS A   73.1.5.2    =  11111111 11111111 11100000 00000000

CLASS B   145.21.6.3  =  11111111 11111111 11111111 11000000
```

送信元ネットワーク T125 が、ホスト・アドレス 114.16.23.8 (2 進表記では 01110010 00010000 00010111 00001000) の宛先ネットワークにメッセージを送信するように要求されると、システムはローカル・インターフェースを 介してその宛先に到達できるかどうかを検査します。

注 : サブネットをサポートする予定の各ホストの構成データベースには、subnetmask キーワードが設定されていなければなりません。 サブネットワーク機能を使用する前にすべてのホストがサブネットワーク機能をサポートしていなければなりません。 SMIT の「ネットワーク・インターフェースの 選択」メニューを使用して構成データベース内にサブネット・マスクを永続的に設定してください。このサブネット・マスクは、実行中のシステム内で **ifconfig** コマンドを使用して設定することもできます。 ただし、**ifconfig** を使用したサブネット・マスクの設定は、永続的な変更ではありません。

ブロードキャスト・アドレス

TCP/IP を使用すると、あるローカル・ネットワーク上のすべてのホストへデータを送信したり、直接接続されたすべてのネットワーク上のすべてのホストへデータを送信したりできます。 このような送信をブロードキャスト・メッセージ と呼びます。

例えば、**routed** 経路指定デーモンは、ブロードキャスト・メッセージを使用して経路を照会し、また、経路の照会へ応答します。

直接接続されたすべてのネットワーク上のすべてのホストにデータをブロードキャストする場合は、データの送信にユーザー・データグラム・プロトコル (UDP) とインターネット・プロトコル (IP) が使用され、IP ヘッダー内のホスト宛先アドレスのビットはすべて 1 に設定されます。特定ネットワーク上のすべてのホストにデータを ブロードキャストする場合は、IP アドレスのローカル・アドレス部分のビットはすべて 0 に設定されます。ブロードキャスト機能を使用するユーザー・コマンドはありません。ただし、このようなコマンドやプログラムを開発することは可能です。

ifconfig コマンドの *broadcast* パラメーターを変更することにより、ブロードキャスト・アドレスを一時的に変更できます。 ブロードキャスト・アドレスを永続的に変更するには、SMIT 高速パス `smit chinet` を使用します。異なるブロードキャスト・アドレスを使用する旧バージョンのソフトウェアとの互換性を保つ必要があるとき、例えば、ホスト ID をすべて 0 に設定する必要があるときなどは、ブロードキャスト・アドレスを変換すると便利な場合があります。

ローカル・ループバック・アドレス

インターネット・プロトコル (IP) では、特殊なネットワーク・アドレス 127.0.0.1 をローカル・ループバック・アドレスとして定義します。

ホストは、ローカル・ループバック・アドレスを使用してホスト自身へメッセージを送信します。 ローカル・ループバック・アドレスは、システム始動プロセスのときに構成マネージャーによって設定されます。ローカル・ループバックはカーネル内にインプリメントされるので **ifconfig** コマンドを使用して設定することもできます。 ループバックは、システムの始動時に始動されます。

TCP/IP ネーム・レゾリューション

32 ビット IP アドレスによって、マシンはインターネットワークを介して送信されるデータグラムの送信元と宛先を効率的に識別できますが、ユーザーはもっと意味のある覚えやすい名前を好みます。 **TCP/IP (伝送制御プロトコル/インターネット・プロトコル)** は、フラット・ネットワーク編成と階層ネットワーク編成の両方をサポートする命名システムを提供します。

フラット・ネットワークでの命名はとても簡単です。ホスト名は 1 つの文字セットから構成され、通常はローカルに管理されます。 フラットな **TCP/IP** ネットワークの場合、ネットワーク上の各マシンには、ネットワーク上にあるすべてのホストの名前から IP アドレスへのマッピング情報が入ったファイル (`/etc/hosts`) があります。 **TCP/IP** ネットワークが拡張するにつれて、各マシンの命名ファイルを常に最新状態にするという管理上の負担が大きくなります。 **TCP/IP** ネットワークがインターネットのように非常に大規模になると、命名は階層に分割されます。 一般に、分割はネットワークの編成に従って行われます。 **TCP/IP** では、階層型の命名はドメイン・ネーム・システム (DNS) と呼ばれ、DOMAIN プロトコルを使用します。 DOMAIN プロトコルは、**TCP/IP** 内で **named** デーモンによって実装されます。

フラット・ネットワークでの命名の場合と同様に、ドメイン名階層ではユーザーに分かりやすく覚えやすいシンボル名をネットワークとホストに割り当てることができます。 しかし、ネットワーク上の各マシンが、そのネットワーク上にある他のすべてのホストの名前からアドレスへのマッピングが入っているファイルを保持する代わりに、1 つ以上のホストがネーム・サーバー として機能するよう選択されます。 ネーム・サーバーは、ネットワークとホストへ割り当てられたシンボル名を、マシンによって効率的に使用される IP アドレスに変換 (解決) します。 ネーム・サーバーはゾーン と呼ばれるドメインの一部についての完全な情報を備えており、そのゾーンについての権限 も備えています。

命名機関

フラット・ネットワークでは、ネットワーク内のすべてのホストは 1 つの中央権限によって管理されます。このフォーマットのネットワークでは、ネットワーク内のすべてのホストに固有のホスト名が付いていなければなりません。 大規模なネットワークでは、この要件のために中央権限上に大きな管理上の負担が発生します。

ドメイン・ネットワークでは、ホストのグループはドメインとサブドメインからなるツリー構造の階層内で個別に管理されます。 この場合、ホスト名はローカル・ドメイン内でだけ固有であればよくルート・ドメイン だけが中央権限によって管理されます。 この構造によって、サブドメインをローカルに管理することができ、中央権限の負担が軽減されます。 例えば、インターネットのルート・ドメインは、`com` (商業組織)、`edu` (教育機関)、`gov` (政府機関)、`mil` (軍事組織) などのドメインから構成されています。 新しい第 1 レベル・ドメインを追加できるのは中央権限だけです。 第 2 レベルでの命名は、それぞれのドメイン内の指定されたエージェントに委任されます。 例えば次の図では、`COM` はその下にあるすべての商業組織について命名権限を備えています。 同様に、第 3 レベルは (それ以下のレベルも)、そのレベル内のエージェン

トに委任されます。 例えば、「インターネットにおけるドメイン構造」の図では、Century ドメインに、そのサブドメイン Austin、Hopkins および Charlotte に対する命名権限があります。

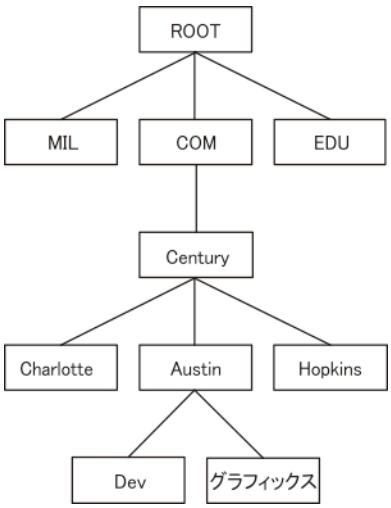


図 *22.* インターネットのドメイン構造

この図は、インターネットの階層構造を示します。 1 番上の root から始まり、そこから mil、com、および edu という次のレベルのドメインに分岐しています。 com ドメインの下に、Charlotte、Austin、および Hopkins を含む、別のレベルがあります。 Austin の下に Dev と Graphics があります。

Century の Austin サブドメインを、さらに Dev や Graphics といったゾーンに分割することもできます。 その場合、ゾーン `austin.century.com` には、Dev と Graphics に委任されたものを除いて、ドメイン `austin.century.com` に入っているすべてのデータが入っています。 例えば、ゾーン `dev.century.com` には Dev に委任されたデータだけが入っており、このゾーンは Graphics に関しては何も認識しません。 ゾーン `austin.century.com` には、(同じ名前のドメインとは逆に) その他のゾーンへ委任されなかったデータだけが入っています。

命名規則
階層型ドメイン・ネーム・システムでは、名前は大文字と小文字を区別しないサブ名を並べたもので、それぞれのサブ名は、間に空白を入れずにピリオドで区切られます。

DOMAIN プロトコルの仕様では、ローカル・ドメイン名は 64 文字未満で、ホスト名は 32 文字未満の長さでなければなりません。 最初にホスト名で、その後にピリオド (.) が続き、さらにピリオドで区切られた一連のローカル・ドメイン名が続き、最後がルート・ドメインです。ホストの完全指定ドメイン名は、ピリオドを含めた長さが 255 文字未満で、次のフォーマットになっていなければなりません。

```
host.subdomain1.[subdomain2 . . subdomain].rootdomain
```

ホスト名は 1 つのドメイン内で固有でなければならないため、同じドメイン内のホストにメッセージを送信する場合は省略名を使用できます。 例えば、`eng` ドメイン内のホストは、`smith.eng.lsu.edu` へメッセージを送信する代わりに `smith` へメッセージを送信できます。 さらに、各ホストには、他のホストがメッセージを送信する場合にそれらの別名を使用できるように複数の別名を付けることもできます。

ネットワーク内のホストの命名
ホストに名前を使用する目的は、ネットワーク内のコンピューターを迅速で簡単かつ確実に参照する方法を提供することです。 インターネットのシステム 管理者たちによって、ホスト名として選択してよい名前と悪い名前が発見されています。 次のヒントを参考に、ホスト名を選択する際に陥りやすい誤りを回避してください。

次に固有で覚えやすいホスト名を選択するためのヒントを示します。

- `sphinx` または `eclipse` などのようなまれにしか使用されない用語を使用する。
- 色、元素 (`helium`、`argon`、`zinc` など)、花、魚など、テーマ別の名前を使用する。
- 実在する (文字を無作為に並べたものでない) 言葉を使用する。

次に不適切な選択の例を示します。 これらを選択すべきでない理由は、一般に覚えにくいか、(人間またはコンピューターにとって) 混同しやすいからです。

- up、down、crash など、既に一般的に使用されている名前。
- 数字だけからなる名前。
- 句読記号が入っている名前。
- Orange と orange など、大文字と小文字の区別に依存する名前。
- システムの 1 次ユーザーの名前またはイニシャル。
- 8 文字を超える名前。
- czek (これは「check」または「czech」と混同される恐れがある) など、一般的でないか意図的に誤ったスペルの語。
- yale.edu など、ドメイン名そのものか、ドメイン名に類似した名前。

ネーム・サーバー

フラットなネーム・スペースでは、すべての名前がネットワーク上の各ホストの /etc/hosts ファイル内に保持されていなければなりません。 ネットワークの規模が非常に大きい場合に、各マシンのリソースに大きな負担となる可能性があります。 階層ネットワークでは、ネーム・サーバー として指定された特定のホストが、その他のホストのために名前を IP アドレスに解決します。

この方法には、フラットなネーム・スペースを上回る 2 つの利点があります。つまり、ネットワーク上にある個々のホストのリソースがネーム・レゾリューションのために占有されることがなく、システム管理者はネットワーク上にある個々のマシンのネーム・レゾリューション・ファイルを保持する必要がなくなります。 1 つのネーム・サーバーによって管理されるネーム・セットは、そのネーム・サーバーの権限ゾーン と呼ばれます。

注 : ある権限ゾーン用にネーム・レゾリューション機能を実行するホスト・マシンは一般にネーム・サーバー ・ホストと呼ばれますが、その機能を制御するプロセスである **named** デーモンが実際のネーム・サーバー・プロセスです。

不要なネットワークのアクティビティーをさらに 減らすため、すべてのネーム・サーバーは、名前からアドレスへのマッピングをキャッシュ (一定の期間だけ格納) します。 クライアントがサーバーに 名前の解決を依頼した場合、サーバーは最初にキャッシュを検査して、その名前が最近解決されたかどうかを調べます。 ドメイン名とホスト・ネームは変化するので、個々の項目はレコードの TTL で指定された時間だけキャッシュ内に止まります。 このため、権限を持つユーザーは、ネーム・レゾリューションが正確である期間を指定できます。

どのような自律システム内にも、複数のネーム・サーバーを入れることができます。 通常、ネーム・サーバーは階層状に編成され、ネットワーク編成に対応しています。 「インターネットのドメイン 構造」という図では、各ドメインに、そのドメイン内にあるすべてのサブドメインについて責任を負うネーム・サーバーがあると考えられます。 個々のサブドメイン・ネーム・サーバーは、他のサブドメインのネーム・サーバーと通信するだけでなく、ドメインにあるネーム・サーバー (親 ネーム・サーバーと呼ばれます) とも通信します。

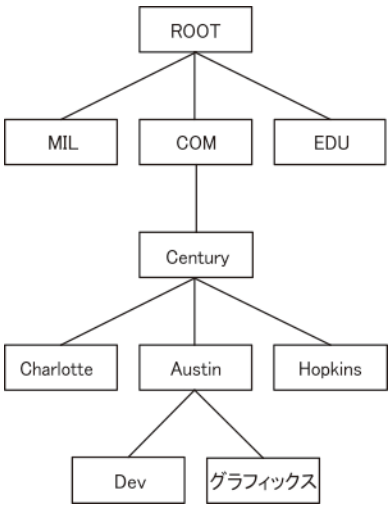


図 *23.* インターネットのドメイン構造

この図は、インターネットの階層構造を示します。 1 番上の root から始まり、そこから mil、com、および edu という次のレベルのドメインに分岐しています。 com ドメインの下に、Charlotte、Austin、および Hopkins を含む、別のレベルがあります。 Austin の下に Dev と Graphics があります。

例えば、「インターネットのドメイン構造」の図では、Austin、Hopkins、Charlotte はすべて、Century ドメインのサブドメインです。 ネットワークの設計がこのツリー階層に従っていれば、Austin ネーム・サーバーは Charlotte および Hopkins の各ネーム・サーバーと通信するほか、親の Century ネーム・サーバーとも通信します。 Austin ネーム・サーバーは、そのサブドメインに対する責任を負うネーム・サーバーとも通信します。

ネーム・サーバーには、次のように複数のタイプがあります。

項目	説明
マスター・ネーム・サーバー	ファイルまたはディスクからデータをロードし、また、ドメイン内にある他のサーバーに権限を委任できます。
スレーブ・ネーム・サーバー	システムの始動時に、マスター・ネーム・サーバーから与えられた権限ゾーンに関する情報を受信し、その後、定期的にその情報の更新をマスター・サーバーに要求します。 スレーブ・ネーム・サーバー上の権限開始 (SOA) リソース・レコードに入っているリフレッシュ値が満了するか、マスター・ネーム・サーバーから通知メッセージを受信した場合に、マスター・ネーム・サーバー上にあるデータベースのシリアル番号が、スレーブ・ネーム・サーバー上の現行データベースのシリアル番号より大きければ、スレーブはマスター・ネーム・サーバーのデータベースを再ロードします。 マスター・サーバーから新しいゾーンを強制的に転送する必要が生じた場合には、既存のスレーブ・データベースを除去して、スレーブ・ネーム・サーバー上で **named** デーモンをリフレッシュするだけです。
スタブ・ネーム・サーバー	データベース複製のメソッドはスレーブ・ネーム・サーバーのメソッドと似ていますが、スタブ・ネーム・サーバーはデータベース全体ではなくマスター・データベースのネーム・サーバー・レコードのみを複製します。

項目	説明
ヒント・サーバー	他のネーム・サーバーに対する以前の照会から作成したヒントにのみ基づくネーム・サーバーを示します。 ヒント・ネーム・サーバーのキャッシュ内に名前からアドレスへのマッピングがない場合、ヒント・ネーム・サーバーは、必要な情報を提供する権限を備えた他のサーバーに問い合わせることによって照会に応答します。
フォワーダーまたはクライアント・サーバー	ローカルでは対応できない照会を、転送サーバーの固定リストへ転送します。 転送専用サーバー (情報を入手して他のクライアントへ渡すが、実際にはサーバーでないフォワーダー) は、ルート・ドメインや他のドメインのマスター・ネーム・サーバーとは対話しません。 転送サーバーへの照会は、再帰的です。 転送サーバーは 1 つだけの場合も複数存在する場合もあり、リストの最後に至るまで順に試行されます。 一般的に、クライアントとフォワーダーの 構成は、特定のインストール先の一部のサーバーだけが残りのインターネット・サーバーと対話するようにしたい場合、または特定の数のネーム・サーバー上に大きなキャッシュを作成したい場合に使用します。
リモート・サーバー	リモート・サーバー・ローカル・ホスト上でネーム・サーバー・プロセスが実行されていなくても、ネーム・サーバーを使用するすべてのネットワーク・プログラムを実行します。 すべての照会は、ネットワーク上にある別のマシン上で実行されているネーム・サーバーによって処理されます。

1 つのネーム・サーバー・ホストが、異なる権限ゾーンで異なる機能を果たす場合もあります。 例えば、1 つのネーム・サーバー・ホストが、あるゾーンのホスト・ネーム・サーバーになり、別のゾーンのスレーブ・ネーム・サーバーになることもできます。

ネーム・レゾリューション

ホスト名から IP アドレスを入手するプロセスはネーム・レゾリューションと呼ばれ `gethostbyname` サブルーチンによって実行されます。

IP アドレスをホスト名に変換するプロセスは逆ネーム・レゾリューションと呼ばれ `gethostbyaddr` サブルーチンによって実行されます。 これらのルーチンは、基本的には *resolver* という名前変換ルーチンのライブラリーへのアクセス機能です。

通常、**TCP/IP** を実行しているホスト上の resolver ルーチンは、以下のソースを使用して名前を解決しようとします。

1. BIND/DNS (名前を指定されたもの)
2. ネットワーク情報サービス (NIS)
3. ローカル `/etc/hosts` ファイル

ドメイン・ネットワーク内の名前を解決するために、resolver ルーチンは最初にドメイン・ネーム・サーバーのデータベースに照会します。このデータベースはローカルの場合 (ホストがドメイン・ネーム・サーバーであれば) もあり、外部ホスト上にある場合もあります。 ネーム・サーバーは、ドメイン名を IP アドレスに変換します。 あるネーム・サーバーが責任を持つ名前のグループがそのサーバーの 権限ゾーンとなります。 resolver ルーチンがリモート・ネーム・サーバーを使用している場合は、そのルーチンはマッピングの照会にドメイン・ネーム・プロトコル (DOMAIN) を使用します。 resolver ルーチンは、フラット・ネットワークの名前を解決するためにローカル `/etc/hosts` ファイルの中にエントリーがあるかどうかを検査します。 NIS が使用されている場合は、マスター・サーバー上の `/etc/hosts` ファイルが検査されます。

デフォルトでは、resolver ルーチンは上記のリソースを使用して名前を解決しようとします。 最初に、BIND/DNS が使用されます。 `/etc/resolv.conf` ファイルが存在しない場合、または BIND/DNS が当該

の名前を検出できなかった場合は、NIS が稼働中であれば、これに対して照会が行われます。 NIS はローカルの /etc/hosts に対する権限を持っているため、NIS が稼働中であれば、ここで検索は終了します。 NIS が稼働していないときは、ローカルの /etc/hosts ファイルが検索されます。 これらのどのサービスでも名前を見つけることができなかった場合、resolver ルーチンは HOST_NOT_FOUND 値を伴って戻ります。 すべてのサービスが使用不可だった場合、resolver ルーチンは SERVICE_UNAVAILABLE 値を伴って戻ります。

上記のデフォルトの順序は、構成ファイル /etc/irs.conf を作成し、求める順序を指定することによって上書きできます。 さらに、デフォルトと /etc/irs.conf の順序は、どちらも環境変数 **NSORDER** を使用して上書きできます。 /etc/irs.conf ファイルまたは **NSORDER** 環境変数のいずれかを定義した場合は、このオプションと一緒に最低 1 つの値を指定しなければなりません。

/etc/irs.conf ファイルを使用してホストの順序付けを指定するには、次のようにします。

```
hosts value [ continue ]
```

順序は、行上に示されている各メソッドを使用してそのまま指定されます。 *value* はリストされているメソッドの中の 1 つであり、continue キーワードは次の行に別の resolver ルーチン・メソッドが続くことを示します。

NSORDER 環境変数を使用してホストの順序付けを指定するには、次のようにします。

```
NSORDER=value,value,value
```

順序は、コンマで区切られた値を 1 行で指定します。 コンマと等号の間には、ホワイト・スペースが使用できます。

例えば、ローカル・ネットワークをフラット・ネットワークとして編成する場合は、 /etc/hosts ファイルだけが必要になります。 この例では、/etc/irs.conf ファイルに以下の行が入っています。

```
hosts local
```

別の方法として、**NSORDER** 環境変数を次のように設定することもできます。

```
NSORDER=local
```

ローカル・ネットワークが、ネーム・レゾリューション用にネーム・サーバーを使用し、バックアップ用に /etc/hosts ファイルを使用するドメイン・ネットワークである場合は、両方のサービスを指定する必要があります。 この例では、/etc/irs.conf ファイルに以下の行が入っています。

```
hosts dns continue
hosts local
```

NSORDER 環境変数は、次のように設定します。

```
NSORDER=bind,local
```

注 : リストする値は小文字でなければなりません。

定義済みまたはデフォルトのいずれかの resolver ルーチンの順序付けに従う時、検索アルゴリズムは、以下の場合にのみ、ある resolver ルーチンから次の resolver ルーチンまで継続します。

- 現行サービスが実行されていないので、使用不可である場合。
- 現行サービスがその名前を見つけることができないか、現行サービスに権限がない場合。

/etc/resolv.conf ファイルが存在しない場合は、BIND/DNS がセットアップされていないか、あるいは実行されていないと見なされるため、使用不能です。 サブルーチン getdomainname と yp_bind が失敗した場合は、NIS サービスがセットアップされていないか、あるいは実行されていないと見なされるため、これは使用できません。 /etc/hosts ファイルをオープンすることができなかった場合は、ローカル検索が不可能であるために、ファイルおよびサービスが使用不可となります。

権限があるもの として登録されたサービスは、そのサービスが後続のサービスのエキスパートであり、すべての適切な名前とアドレスを持つことを意味します。 resolver ルーチンは、後続のサービスを実行しようとしません。これは、後続のサービスが持つことができるのは、権限のあるサービスの情報のサブセットだけだからです。 ネーム・レゾリューションは、権限があるものとして登録されたサービスで終了し、名前を検出できなかった場合でも同様です (この場合、resolver ルーチンは HOST_NOT_FOUND を戻します)。 権限があるサービスが使用不可能な場合は、次に指定されたサービスに対して照会が行われます。

権限のある送信元は、値の直後に =auth という文字列を付けて指定されます。 authoritative という単語全体を入力できますが、auth という文字列だけが使用されます。 例えば、**NSORDER** 環境変数に次の値が入っているとします。

```
hosts = nis=auth,dns,local
```

NIS が実行中の場合は、名前が検出されたかどうかに関係なく、NIS への照会後に検索は終了します。 NIS が実行中でない場合は、次のソース、つまり DNS に対して照会が行われます。

TCP/IP のネーム・サーバーは、リモート・ネットワーク上のホストの名前を検索するコストを軽減するためにキャッシュを使用します。 要求が出されるたびにホスト名を検索する代わりに、ネーム・サーバーはそのホスト名が最近解決されたかどうかを確認するために、キャッシュの中を探します。 ドメイン名とホスト名は変化するため、個々の項目はレコードの存続時間 (TTL) 値で指定された時間だけキャッシュ内に残ります。 このようにして、ネーム・サーバーは、応答の有効期間を指定することができます。

ネーム・サーバーと *sendmail* の間でホスト名が矛盾する可能性

DNS 環境では、コマンド・ラインからの **hostname** コマンドで設定されるか、rc.net ファイル・フォーマットで設定されるホスト名は、ネーム・サーバーから戻されたとおりの正式なホスト名でなければなりません。

一般に、この名前は次のフォーマットになったホストの完全なドメイン名です。

```
host.subdomain.subdomain.rootdomain
```

注 : resolver ルーチンを使用するには、デフォルトのドメインが設定されていなければなりません。 デフォルト・ドメインは、**hostname** コマンド内に設定されていない場合は、/etc/resolv.conf ファイル内に設定されていなければなりません。

ホスト名が完全修飾ドメイン名としてセットアップされていない場合、かつ **sendmail** プログラムと一緒にドメイン・ネーム・サーバーを使用するようにシステムがセットアップされている 場合は、**sendmail** 構成ファイル (/etc/sendmail.cf) を編集して、その公認ホスト名が反映されるようにしなければなりません。 また、この構成ファイル内のドメイン・ネーム・マクロを、**sendmail** プログラムが正しく動作するように、設定しなければなりません。

注 : /etc/sendmail.cf ファイル内で指定したドメインは、すべての **sendmail** 機能について **hostname** コマンドで設定したドメインより優先します。

ネーム・サーバーと *sendmail* の間でドメイン名が矛盾する可能性

ローカル・ドメイン名とドメイン・ネーム・サーバーは、ホストがドメイン・ネーム・サーバーであるかどうかによって、別々のファイルに指定されます。

ドメイン・ネットワーク内にあって、ネーム・サーバーでないホストの場合、ローカル・ドメイン名およびドメイン・ネーム・サーバーは /etc/resolv.conf ファイルの中で指定されます。 ドメイン・ネーム・サーバー・ホストでは、ローカル・ドメインおよびその他のネーム・サーバーは、**named** デーモンが始動時に読み取るファイルの中で定義されます。

逆アドレス解決プロトコル

逆アドレス解決プロトコル (RARP) は、固有のハードウェア・アドレスをイーサネット LAN アダプター上でインターネット・アドレスに変換します (イーサネット・プロトコルのみ)。

標準イーサネット・プロトコルは、次の制限付きでサポートされます。

- サーバーは、**RARP** 要求にのみ応答します。
- サーバーは、永続的な **ARP** テーブル・エントリーのみを使用します。

- サーバーは、動的 **ARP** テーブル・エントリーを使用しません。
- サーバーは、自ら自動的に応答することはありません。

システム管理者は、**arp** コマンドを使用して永続的な **ARP** エントリーのテーブルを手作業で構築し、保守しなければなりません。 権限がある送信元からの **RARP** 応答を必要とする個々のホストについて、サーバー上で特定の **ARP** テーブル・エントリーを追加しなければなりません。

ローカル・ネーム・レゾリューション (/etc/hosts) のタスク

使用するネットワークが小規模で、フラットな命名方式を使用する場合には、/etc/hosts ファイルを構成してください。

ネーム・サーバーを持つ階層状の (またはドメイン) 命名方式を使用する場合であっても、ネーム・サーバーにとって未知のホストを識別するために /etc/hosts ファイルを構成できます。

ローカル・ホスト・レゾリューション用にシステムを構成するには、システム管理インターフェース・ツール (SMIT)、またはコマンドを使用します。コマンド方式を選択する場合は、必ず /etc/hosts ファイルのフォーマットが保存されるようにしてください (ファイル参照 の Hosts File Format for TCP/IP を参照)。

表 *62.* ローカル・ネーム・レゾリューションのタスク

タスク	SMIT 高速パス	コマンドまたはファイル
すべてのホストのリスト	**smit lshostent**	**hostent** コマンドまたは **view** /etc/hosts を使用する
ホストの追加	**smit mkhostent**	**hostent** コマンドまたは **edit** /etc/hosts を使用する
ホストの特性の変更/表示	**smit chhostent**	**hostent** コマンドまたは **edit** /etc/hosts を使用する
ホストの除去	**smit rmhostent**	**hostent** コマンドまたは **edit** /etc/hosts を使用する

ドメイン・ネーム・レゾリューションの計画

DOMAIN ネーム・レゾリューション・システムを計画する際は、次の推奨事項を参考にしてください。

大規模なインターネットワークの一部である場合は、ドメインおよびネーム・サーバーの設定を中央権限に合わせる必要があります。

- アーキテクチャーと 構成には数多くの可能性があるので、どの計画を固める場合も、事前に **TCP/IP**、DNS、および BIND について理解しておいてください。 ネットワーク情報サービスの使用を予定している場合は、NFS と NIS についても理解しておいてください。 これらのトピックに関するさまざまなマニュアルが入手可能です。
- 計画を立てます。

  名前の変更は、最初に名前をセットアップするよりもさらに 困難です。 ファイルをセットアップする 前に、編成内でネットワーク、ゲートウェイ、ネーム・サーバー、ホスト名についての合意を得てください。
- ネーム・サーバーは余分にセットアップします。

  余分なネーム・サーバーをセットアップできない場合は、スレーブおよびヒント・ネーム・サーバーをセットアップして、何らかのタイプのバックアップを確保します。
- ネーム・サーバーを選択する場合、次の点に気を付けます。
  - 外部のシステムに物理的に近いマシンを選択してください。
  - ネーム・サーバーは、できるだけ独立したものにしてください。 電源を別にし、独立した配線になるようにしてください。
  - ネーム・レゾリューション・サービスをバックアップする別のネットワークを見つけ、他のネットワークについても同じことをしてください。
- サーバーをテストします。

- 通常のネーム・レゾリューションと逆ネーム・レゾリューションの両方をテストしてください。
- マスターからスレーブ・ネーム・サーバーへのゾーン転送をテストしてください。
- システム・クラッシュとリブートのあと、各ネーム・サーバーをテストしてください。

- ネーム・レゾリューション要求を外部のネーム・サーバーへ送信する前に、フォワーダー・サーバーへ送信します。 これにより、ネーム・サーバーがキャッシュを共用でき、マスター・ネーム・サーバーの負荷が軽減されるので、パフォーマンスが改善されます。

```
objectclass container
        requires
                objectclass,
                cn
objectclass hosts
        requires
                objectclass,
                hname
        allows
                addr
                halias,
                comment
```

ネーム・サーバー・レゾリューション
階層ネットワークでは、特定のホストがネーム・サーバー として指定されます。 それらのホストは、他のホストの名前を IP アドレスとして解決します。

named デーモンはネーム・サーバー機能を管理します。したがって、これはネーム・サーバー・ホスト上で実行されなければなりません。

ネーム・サーバーを構成する前に、そのサーバーがサービスするネットワークに最適なタイプがどれかを判断してください。 ネーム・サーバーには、次のように複数のタイプがあります。

マスター・ネーム・サーバー は、名前からアドレスへのマッピング情報が入っているデータベースを実際に保管しています。 このサーバーは、データをファイルまたはディスクからロードし、ドメイン内にある他のサーバーに権限を代行させることが できます。 スレーブ・ネーム・サーバー またはスタブ・ネーム・サーバー は、システムの始動時に、マスター・ネーム・サーバーから、特定の権限ゾーンに関する情報を受信し、その後、定期的にその情報の更新をマスター・サーバーに要求します。ヒント・ネーム・サーバー は、必要な情報を提供する権限がある他のサーバーに照会することにより、名前を解決する要求に応答します。

注 : 前の世代の **named** ネーム・サーバーは、マスター・ネーム・サーバーを 1 次ネーム・サーバーとして、スレーブ・ネーム・サーバーを 2 次ネーム・サーバーとして、ヒント・ネーム・サーバーをキャッシュ専用ネーム・サーバーとして指定しました。

ネーム・サーバーは、異なる権限ゾーンの異なる容量の環境で機能する可能性があるので注意してください。 例えば、1 つのネーム・サーバー・ホストが、あるゾーンのホスト・ネーム・サーバーになり、別のゾーンのスレーブ・ネーム・サーバーになることもできます。 ご使用のシステムに NIS がインストールされている場合は、これらのサービスが提供するネーム・レゾリューションも使用することができます。

ネーム・サーバーを構成する場合、いくつかのファイルが関係してきます。

項目	説明
conf	このファイルは、**named** デーモンの始動時に読み取られます。 conf ファイル内のレコードは、ネーム・サーバーのタイプ、そのネーム・サーバーが権限を持つドメイン (ネーム・サーバーの権限ゾーン)、および最初にネーム・サーバーのデータベースをセットアップする 場合のデータの取得場所を **named** デーモンに知らせます。 このファイルのデフォルトの名前は、/etc/named.conf です。 ただし、このファイルの名前は、**named** デーモンの始動時にコマンド・ラインでファイルの名前とパスを指定すれば変更できます。 /etc/named.conf を conf ファイルとして使用しようとして、このファイルが存在しない場合は、syslog ファイル内にメッセージが生成され、**named** が終了します。 ただし、代替の conf ファイルが指定されており、その代替ファイルが存在しない場合は、エラー・メッセージは生成されず、**named** は続行します。

項目	説明
cache	ローカル・キャッシュに関する情報が入っています。 このローカル・キャッシュ・ファイルには、ネットワーク内の最高権限を備えたネーム・サーバーの名前とアドレスが入っています。 このキャッシュ・ファイルは標準リソース・レコード・フォーマットを使用します。 キャッシュ・ファイルの名前は、conf ファイルの中で設定されます。

domain data	3 つの一般的なドメイン・データ・ファイルが存在し、**named** データ・ファイルとも呼ばれます。 **named** ローカル・ファイルには、ローカル・ループバック用のアドレス解決情報が入っています。 **named** データ・ファイルには、ネーム・サーバーの権限ゾーン内にあるすべてのマシンのアドレス解決データが入っています。 **named** 逆データ・ファイルには、ネーム・サーバーの権限ゾーンにあるすべてのマシンの逆アドレス解決データが入っています。 これらのドメイン・データ・ファイルは標準リソース・レコード・フォーマットを使用します。 これらのファイルの名前はユーザーが定義でき、conf ファイル内で設定されます。 規約上、これらのファイルの名前に通常はデーモンの名前 (named) を組み込み、拡張子にファイルのタイプとドメイン名を組み込みます。 例えば、ドメイン abc のネーム・サーバーは次のようなファイルを備えています。
	
	`named.abc.data`
	`named.abc.rev`
	`named.abc.local`
	
	named データ・ファイルを修正する場合は、SOA リソース・レコード内のシリアル番号を 1 つ増やして、スレーブ・ネーム・サーバーが新しいゾーン変更を正しく認識できるようにしてください。

resolv.conf	このファイルが存在する場合、ホストは最初にネーム・サーバーで名前を解決します。 resolv.conf が存在しない場合、ホストは /etc/hosts ファイルを探してネーム・レゾリューションを行います。 ネーム・サーバー上には resolv.conf ファイルが存在しなければならず、このファイルにはローカル・ホストのアドレスかループバック・アドレス (127.0.0.1) を入れることができますが、空でもかまいません。 **注 :** resolver ルーチンを使用するには、デフォルトのドメインが設定されていなければなりません。 デフォルト・ドメインは、/etc/resolv.conf ファイルの中に設定されていない場合は、hostname の中に設定されていなければなりません。

存続時間 (TTL) はリソース・レコードの中で指定されます。 レコード内に TTL が指定されていないと、この時間の長さは、デフォルトではそのゾーンの権限の開始 (SOA) レコード内で定義されている最小値フィールドの値になります。 TTL は、ゾーンの外部 (キャッシュ内) にデータを保管するとき、データが無期限に残ることがないようにするために使用されます。

ドメイン・ネーム・サーバーの構成

このシナリオでは、マスター・ネーム・サーバー、スレーブ・ネーム・サーバー、およびヒント・ネーム・サーバーは、ネーム・レゾリューションを実行するために構成されます。 各ネーム・サーバーは個別のマシンとなり、それぞれには /etc/named.conf ファイルが構成されます。ただし、各ファイルの情報は異なります。 /etc/named.conf は、**named** デーモンが開始されるたびに読み取られ、どのタイプのサーバー (マスター、スレーブ、またはヒント) であるか、ならびにそのネーム・レゾリューション・データの取得場所を示します。 これらのネーム・サーバーのそれぞれは BIND 8 を実行することになります。

マスター・ネーム・サーバーは、ネーム・レゾリューションを abc.aus.century.com ゾーンに提供するように構成されます。 このシナリオでは、マスター・ネーム・サーバーの IP アドレスは 192.9.201.1 であり、そのホスト名は venus.abc.aus.century.com です。 マスター・ネーム・サーバーは、 venus、 earth、 mars、 および jupiter の各ホスト名のネーム・レゾリューションを可能にします。 /etc/named.conf ファイルは、**named** デーモンが /usr/local/domain ディレクトリーからそのデータ・ファイルを探索する必要があることを指定するために構成されます。 マスター・ネーム・サーバー用に構成

されるデータ・ファイルは、named.ca、named.abc.local、named.abc.data、および named.abc.rev です。

スレーブ・ネーム・サーバーが次に構成されます。 スレーブ・ネーム・サーバーのホスト名は earth.abc.aus.century.com となり、その IP アドレスは 192.9.201.5 となります。 スレーブ・ネーム・サーバーの /etc/named.conf ファイルでは、スレーブ・ネーム・サーバーがマスター・ネーム・サーバーの named.abc.data および named.abc.rev ファイルを複製できるように、マスター・ネーム・サーバーのアドレスを指定します。 さらに、named.ca および named.abc.local データ・ファイルは、このサーバー用に構成されます。

ヒント・ネーム・サーバーが次に構成されます。 ヒント・ネーム・サーバーは、ホスト名およびアドレス・マッピングのローカル・キャッシュを保管します。 要求されたアドレスまたはホスト名がそのキャッシュ内にない場合は、ヒント・ネーム・サーバーはマスター・ネーム・サーバーと交信し、レゾリューション情報を取得し、その情報をそのキャッシュに追加します。 さらに、named.ca および named.abc.local データ・ファイルは、このサーバー用に構成されます。

ネーム・サーバー上の named データ・ファイル (/etc/named.conf ファイルではない) 内のすべての情報は、Standard Resource Record Format でなければなりません。 named データ・ファイルの情報については、ファイル参照 の TCP/IP の Standard Resource Record Format を参照してください。

各ネーム・サーバーの管理者は gail.zeus.abc.aus.century.com になります。 これは、各ネーム・サーバー上のローカル・データ・ファイル内に指定されます。 さらに、このシナリオでは、ルート・ネーム・サーバーは、IP アドレス 129.114.1.2 を指定した relay.century.com です。

このシナリオの最後で、ネーム・レゾリューションはホスト venus、earth、mars、および jupiter に提供されます。 さらに、逆ネーム・レゾリューション (IP アドレスからホスト名へ) も提供されます。 解決できない要求を受信した場合、マスター・ネーム・サーバーは、必要な情報を見つけるために relay.century.com と交信します。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ステップ 1. マスター・ネーム・サーバーの構成

1. マスター・ネーム・サーバーで、/etc/named.conf ファイルを開きます。 /etc ディレクトリーの中に /etc/named.conf ファイルがない場合は、次のコマンドを実行してこのファイルを作成します。

   ```
   touch /etc/named.conf
   ```

   /etc/named.conf ファイルを構成するには、次のようにします。

   a. options スタンザにディレクトリー節を指定します。 これにより、named データ・ファイルが /usr/local/domain ディレクトリーからの相対パスを使用できるようになります。 このシナリオでは、以下の内容が追加されました。

      ```
      options {
          directory "/usr/local/domain";
      };
      ```

      ここでディレクトリーを指定しないことを選択した場合は、/etc ディレクトリーから必要なデータ・ファイルが検索されます。

   b. レコード・データを定義済みゾーンの外でキャッシュできるようにするには、ヒント・ゾーン・ファイルの名前を指定します。 このシナリオでは、以下の内容が追加されました。

      ```
      zone "." IN {
          type hint;
          file "named.ca";
      };
      ```

c. 各ゾーン、構成するネーム・サーバーのタイプ、およびネーム・サーバーのドメイン・データ・ファイルを指定するために、以下のスタンザを追加します。 このシナリオでは、正引きゾーンと逆引きゾーンの両方のマスター・サーバーは次のとおりです。

```
zone "abc.aus.century.com" in {
    type master;
    file "named.abc.data";
};
zone "201.9.192.in-addr.arpa" in {
    type master;
    file "named.abc.rev";
};
```

d. named ローカル・ファイルの名前を定義します。 以下に例を示します。

```
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "named.abc.local";
};
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

2. /usr/local/domain/named.ca ファイルを開きます。 ドメイン用のルート・ネーム・サーバーのアドレスを追加します。 このシナリオでは、以下の内容が追加されました。

```
; root name servers.
.          IN    NS    relay.century.com.
relay.century.com.   3600000    IN    A     129.114.1.2
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

3. /usr/local/domain/named.abc.local ファイルを開きます。 次の情報を追加します。

- ゾーンの権限の開始 (SOA) とデフォルトの存続時間に関する情報。 このシナリオでは、以下の内容が追加されました。

```
$TTL 3h    ;3 hour

@ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                           1       ;serial
                           3600    ;refresh
                           600     ;retry
                           3600000 ;expire
                           3600    ;negative caching TTL
)
```

- ネーム・サーバー (NS) レコード。 行の先頭にタブ・スペースを挿入します。**named** デーモンがそのタブ・スペースをゾーン名と置き換えます。

```
<tab>    IN    NS     venus.abc.aus.century.com.
```

- ポインター (PTR) レコード。

```
1      IN    PTR    localhost.
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

4. /usr/local/domain/named.abc.data ファイルを開きます。 次の情報を追加します。

- ゾーンの権限の開始とデフォルトの存続時間に関する情報。 このレコードは、ゾーンの開始を指定します。 1 ゾーン当たり許可されている権限の開始レコードは 1 つだけです。 このシナリオでは、以下の内容が追加されました。

```
$TTL 3h    ;3 hour

@ IN    SOA     venus.abc.aus.century.com. gail.zeus.abc.aus.century.com. (
                1       ;serial
                3600    ;refresh
                600     ;retry
                3600000 ;expire
```

```
                3600    ;negative caching TTL
)
```

- ゾーン内のすべてのマスター・ネーム・サーバーのネーム・サーバー・レコード。 行の先頭にタブ・スペースを挿入します。**named** デーモンがそのタブ・スペースをゾーン名と置き換えます。

```
<tab>    IN    NS       venus.abc.aus.century.com.
```

- 権限のネーム・サーバー・ゾーン内のすべてのホストに関する、名前からアドレスへのレゾリューション情報。

```
venus        IN    A        192.9.201.1
earth        IN    A        192.9.201.5
mars         IN    A        192.9.201.3
jupiter      IN    A        192.9.201.7
```

必要に応じて、その他のタイプのエントリー (正規のネーム・レコードやメール・エクスチェンジャー・レコードなど) を組み込みます。

ファイルを編集した後で、そのファイルを保管し、閉じます。

5. /usr/local/domain/named.abc.rev ファイルを開きます。 次の情報を追加します。

- ゾーンの権限の開始とデフォルトの存続時間に関する情報。 このレコードは、ゾーンの開始を指定します。 1 ゾーン当たり許可されている権限の開始レコードは 1 つだけです。

```
$TTL 3h    ;3 hour

@  IN  SOA  venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (
                          1       ;serial
                          3600    ;refresh
                          600     ;retry
                          3600000 ;expire
                          3600    ;negative caching TTL
)
```

- その他のタイプのエントリー (ネーム・サーバー・レコードなど)。 これらのレコードを組み込む場合は、行の先頭にタブ・スペースを挿入します。**named** デーモンがそのタブ・スペースをゾーン名と置き換えます。 このシナリオでは、以下の内容が追加されました。

```
<tab>    IN      NS      venus.abc.aus.century.com.
```

- 権限のネーム・サーバー・ゾーン内のすべてのホストに関する、アドレスからネームへのレゾリューション情報。

```
1                    IN    PTR    venus.abc.aus.century.com.
5                    IN    PTR    earth.abc.aus.century.com.
3                    IN    PTR    mars.abc.aus.century.com.
7                    IN    PTR    jupiter.abc.aus.century.com.
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

6. 次のコマンドを実行して、/etc/resolv.conf ファイルを作成します。

```
touch /etc/resolv.conf
```

このファイルが存在する場合、ホストはネーム・レゾリューションにネーム・サーバーを使用します。

7. 次のエントリーを /etc/resolv.conf ファイルに追加します。

```
nameserver 127.0.0.1
```

この 127.0.0.1 というアドレスはループバック・アドレスであり、これによってホストはネーム・サーバーとしてのそれ自体にアクセスします。 /etc/resolv.conf ファイルには、次のようなエントリーを入れることもできます。

```
domain abc.aus.century.com
```

この場合、abc.aus.century.com はドメイン名です。

ファイルを編集した後で、そのファイルを保管し、閉じます。

8. smit stnamed SMIT 高速パスを使用して、**named** デーモンを使用可能にします。 これによって、このデーモンはシステムが始動するたびに初期化されます。 現時点で **named** デーモンを始動するか、次回のシステム再始動時に始動するか、それともその両方で始動するかを指示します。

ステップ 2. スレーブ・ネーム・サーバーの構成

スレーブ・ネーム・サーバーを構成するには、以下の手順を使用します。 一連のファイルを編集してから、SMIT を使用して **named** デーモンを開始します。

1. スレーブ・ネーム・サーバーで、/etc/named.conf ファイルを開きます。 /etc ディレクトリーの中に /etc/named.conf ファイルがない場合は、次のコマンドを実行してこのファイルを作成します。

```
touch /etc/named.conf
```

/etc/named.conf ファイルを構成するには、次のようにします。

a. options スタンザにディレクトリー節を指定します。 これにより、named データ・ファイルが /usr/local/domain ディレクトリーからの相対パスを使用できるようになります。 このシナリオでは、以下の内容が追加されました。

```
options {
    directory "/usr/local/domain";
};
```

ここでディレクトリーを指定しないことを選択した場合は、**named** デーモンは、/etc ディレクトリーから必要なデータ・ファイルを検索します。

b. レコード・データを定義済みゾーンの外でキャッシュできるようにするには、ネーム・サーバー用のヒント・ゾーン・ファイルの名前を指定します。

```
zone "." IN {
    type hint;
    file "named.ca";
};
```

c. スレーブ・ゾーン節を指定します。 各スタンザには、ゾーン・タイプ、ネーム・サーバーがそのデータをバックアップするためのファイル名、およびマスター・ネーム・サーバーの IP アドレスが組み込まれます。スレーブ・ネーム・サーバーは、このマスター・ネーム・サーバーからそのデータ・ファイルを複製します。 このシナリオでは、以下のスレーブ・ゾーン節が追加されました。

```
zone "abc.aus.century.com" IN {
    type slave;
    file "named.abc.data.bak";
    masters { 192.9.201.1; };
};
```

```
zone "201.9.192.in-addr.arpa" IN {
    type slave;
    file "named.abc.rev.bak";
    masters { 192.9.201.1; };
};
```

d. ループバック・ネットワーク・アドレスの解決をサポートするために、named.abc.local というソースを指定したタイプ *master* のゾーン、ならびにネーム・サーバーが受け持つドメインを指定します。

```
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "named.abc.local";
};
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

2. /usr/local/domain/named.ca ファイルを編集します。

   このファイルには、ネットワークのルート・ドメイン・サーバーであるアドレス・サーバーが含まれます。 このシナリオでは、以下の内容が追加されました。

   ```
   ; root name servers.
   .          IN    NS    relay.century.com.
   relay.century.com.   3600000    IN    A     129.114.1.2
   ```

   ファイルを編集した後で、そのファイルを保管し、閉じます。
3. /usr/local/domain/named.abc.local ファイルを開きます。 このシナリオでは、以下の内容が追加されました。

   - ゾーンの権限の開始 (SOA) とデフォルトの存続時間に関する情報。

     ```
     $TTL 3h    ;3 hour

     @ IN SOA earth.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                                1       ;serial
                                3600    ;refresh
                                600     ;retry
                                3600000 ;expire
                                3600    ;negative caching TTL
     )
     ```

   - ネーム・サーバー (NS) レコード。 行の先頭にタブ・スペースを挿入します。**named** デーモンがそのタブ・スペースをゾーン名と置き換えます。 以下に例を示します。

     ```
     <tab>    IN    NS     earth.abc.aus.century.com.
     ```

   - ポインター (PTR) レコード。

     ```
     1      IN    PTR    localhost.
     ```

   ファイルを編集した後で、そのファイルを保管し、閉じます。
4. 次のコマンドを実行して、/etc/resolv.conf ファイルを作成します。

   ```
   touch /etc/resolv.conf
   ```

5. そのファイルに次のエントリーを追加します。

   ```
   nameserver 127.0.0.1
   domain abc.aus.century.com
   ```

   ファイルを編集した後で、そのファイルを保管し、閉じます。
6. smit stnamed SMIT 高速パスを使用して、**named** デーモンを使用可能にします。 これによって、このデーモンはシステムが始動するたびに初期化されます。 現時点で **named** デーモンを始動するか、次回のシステム再始動時に始動するか、それともその両方で始動するかを指示します。

ステップ 3. ヒント・ネーム・サーバーの構成

ヒント (またはキャッシュ専用) ネーム・サーバーを構成するには、一連のファイルを編集してから、SMIT またはコマンド・ラインを使用して **named** デーモンを始動する次の手順を使用します。

1. ヒント・ネーム・サーバーで、/etc/named.conf ファイルを編集します。 /etc ディレクトリーの中に /etc/named.conf ファイルがない場合は、次のコマンドを実行してこのファイルを作成します。

   ```
   touch /etc/named.conf
   ```

   /etc/named.conf ファイルを構成するには、次のようにします。

a. options スタンザにディレクトリー節を指定します。 これにより、named データ・ファイルが /usr/local/domain ディレクトリーからの相対パスを使用できるようになります。 このシナリオでは、以下の内容が追加されました。

```
options {
    directory "/usr/local/domain";
};
```

b. ループバック・ネットワーク・アドレスの解決をサポートするために、named.abc.local というソースを指定したタイプ *master* のゾーン、ならびにネーム・サーバーが受け持つドメインを指定します。 この例では、options ディレクトリー・キーワードが /etc/named.conf ファイルに指定されています。

```
zone "0.0.127.in-addr.arpa" IN {
    type master;
      file "named.abc.local";
};
```

c. キャッシュ・ゾーン・ファイルの名前を指定します。 以下に例を示します。

```
zone "." IN {
    type hint;
    file "named.ca";
};
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

2. /usr/local/domain/named.ca ファイルを編集します。

このファイルには、ネットワークのルート・ドメインに対する権限ネーム・サーバーのアドレスが含まれています。 例えば、次のとおりです。

```
; root name servers.
.          IN    NS    relay.century.com.
relay.century.com.   3600000    IN    A     129.114.1.2
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

3. /usr/local/domain/named.local ファイルを編集します。 このシナリオでは、以下の情報がこのファイルに追加されました。

- ゾーンの権限の開始 (SOA) とデフォルトの存続時間に関する情報。

```
$TTL 3h    ;3 hour

@ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                            1       ;serial
                            3600    ;refresh
                            600     ;retry
                            3600000 ;expire
                            3600    ;negative caching TTL
)
```

- ネーム・サーバー (NS) レコード。 行の先頭にタブ・スペースを挿入します。**named** デーモンがそのタブ・スペースをゾーン名と置き換えます。

```
<tab>    IN    NS     venus.abc.aus.century.com.
```

- ポインター (PTR) レコード。

```
1      IN    PTR    localhost.
```

ファイルを編集した後で、そのファイルを保管し、閉じます。

4. 次のコマンドを実行して、/etc/resolv.conf ファイルを作成します。

```
touch /etc/resolv.conf
```

5. そのファイルに次のエントリーを追加します。

```
nameserver 127.0.0.1
domain abc.aus.century.com
```

   ファイルを編集した後で、そのファイルを保管し、閉じます。

6. `smit stnamed` SMIT 高速パスを使用して、**named** デーモンを使用可能にします。 これによって、このデーモンはシステムが始動するたびに初期化されます。 現時点で **named** デーモンを始動するか、次回のシステム再始動時に始動するか、それともその両方で始動するかを指示します。

リブート時に、IPv6 構成が設定されます。 ホストごとにこのプロセスを繰り返します。

ドメイン・メール・サーバーの構成

ドメイン・メール・サーバーを構成すると、ある組織の外部ユーザーはその組織内のユーザーにメールを簡単に送ることができます。 つまり、ドメイン・メール・サーバーがない場合は、メール・アドレスでその組織内の特定のホストを指定しなければなりません。

例えば、`sam@orange.widget.com` の場合、`widget.com` はその組織のドメイン名であり、`orange` は `sam` が使用しているホストです。 しかし、ドメイン・メール・サーバーがあれば、その組織の外部ユーザーは、相手がどのホストを使用しているかを知らなくても、例えば `sam@widget.com` のように、簡単にユーザー名とドメイン名を指定できます。

ドメイン・メール・サーバーを構成するには、以下の手順を使用します。

1. メール・サーバー `black.widget.com` 用に次のようなメール・エクスチェンジャー (MX) レコードとアドレス (A) レコードを作成します。

```
widget.com           IN    MX     10 black.widget.com
widget.com           IN     A     192.10.143.9
black.widget.com     IN     A     192.10.143.9
```

2. メール・サーバー (`black.widget.com`) 上で `sendmail.cf` を編集し、次のようにドメイン別名 (**w** クラス) を追加します。

```
Cw $w $?D$w.$D$. widget.com
```

3. メール・クライアントはローカル・メールではないメールの送信先を知っていなければならないので、次のように個々のクライアント上にある `sendmail.cf` を編集して、メール・サーバーを指示します (**S** マクロ)。

```
DRblack.widget.com
```

4. すべてのユーザーが、ネーム・サーバー `brown.widget.com` 内で定義されている MX レコードを使用できるようにするために、**NameServOpt** オプションを使用して **sendmail** デーモンを構成します。
5. 別名ファイルを使用して、メール・サーバー上にアカウントを持たないユーザーの別名を、次のようにドメインに追加します。

```
sam:sam@orange.widget.com
david:david@green.widget.com
judy:judy@red.widget.com
```

   注: メールボックス (MB) レコードも同じ機能を実行できます。

6. データベースを修正したため、SOA リソース・レコード内のシリアル番号を増やす必要があります。
7. `refresh -s named` コマンドを入力してネーム・サーバー・データベースをリフレッシュします。
8. クライアント上で、`refresh -s sendmail` コマンドを実行し、変更を有効にします。

ドメイン・メール・サーバーを構成するには、その他にもいくつか方法があります。 以下の手順は、メールボックス (MB)、メール名変更 (MR)、メール・グループ (MG) の各レコードの使用に関連しています。

メールボックス・レコードを使用したドメイン・メール・サーバーの構成

メールボックス・レコードを使用してドメイン・メール・サーバーを構成するには、以下の手順を使用します。

1. ドメイン内の各ユーザーのメールボックス (MB) レコードを定義します。 次のようなエントリー、

   ```
   sam IN MB orange.widget.com.
   ```

   を、ホスト brown.widget.com の /usr/local/domain/named.abc.data ファイルに追加してください。 これらのエントリーは、メール・サーバー black.widget.com に対し、そのドメインに属する各ユーザー宛てのメールをどこへ送信すればよいかを示します。
2. ネーム・サーバー brown.widget.com 内に定義された MB レコードを使用するために、メール・サーバー black.widget.com 上の **sendmail** デーモンを構成します。

   NameServOpt オプションを使用します。
3. データベースを修正したため、SOA リソース・レコード内のシリアル番号を増やします。
4. refresh -s named コマンドを実行して、ネーム・サーバー・データベースをリフレッシュします。
5. refresh -s sendmail コマンドを入力し、変更を有効にします。

ユーザーのメール名変更レコードの定義

メール名変更レコードを定義するには、以下の手順を使用します。

1. ドメイン・ネーム・サーバー上の /usr/local/domain/named.abc.data ファイルを編集します。
2. 1 つの別名につき 1 つのメール名変更レコードを追加します。
   例えば、ユーザー sam の別名が sammy の場合、メール名変更レコードは次のようになります。

   ```
   sammy IN MR sam
   ```

   このレコードにより、sammy 宛てにアドレッシングされたすべてのメールが sam へ送達されます。 各 MR レコードは単独で 1 行に入力してください。
3. データベースを修正したため、SOA リソース・レコード内のシリアル番号を増やす必要があります。
4. refresh -s named コマンドを入力して、ネーム・サーバー・データベースをリフレッシュします。
5. refresh -s sendmail コマンドを入力し、変更を有効にします。

メール・グループ・メンバー・レコードの定義

メール・グループ・メンバー・レコードを定義するには、以下の手順を使用します。

1. ドメイン・ネーム・サーバー上の /usr/local/domain/named.abc.data ファイルを編集します。
2. 各メール・グループ (MG) について MG レコードを追加します。 MG レコードは、ネーム・サーバー上で保守される別名を使用して、/etc/aliases ファイルのように機能します。 例えば、次のとおりです。

   ```
   users IN HINFO users-request widget.com
   users IN MG sam
   users IN MG david
   users IN MG judy
   ```

   この例では、users@widget.com 宛てにアドレッシングされたすべてのメールが、sam、david、judy へ送達されるようにします。 それぞれの MG レコードは単独で 1 行に入力にしてください。

   注: ユーザー sam、david、judy の MB レコードが定義されていなければなりません。
3. データベースを修正したため、SOA リソース・レコード内のシリアル番号を増やす必要があります。
4. refresh -s named コマンドを入力して、ネーム・サーバー・データベースをリフレッシュします。
5. refresh -s sendmail コマンドを入力し、変更を有効にします。

メール・エクスチェンジャー・レコードの定義

メール・エクスチェンジャー・レコードを定義するには、以下の手順を使用します。

1. ドメイン・ネーム・サーバー上の /usr/local/domain/named.abc.data ファイルを編集します。

2. ネットワークに直接接続しておらず、メールの転送先としたい各マシンのメール・エクスチェンジャー (MX) レコードを追加します。
   例えば、purple.widget.com 上のユーザー宛てにアドレッシングされたメールを post.office.widget へ転送する場合、MX レコードは次のようになります。

   ```
   purple.widget.com IN MX 0 post.office.widget.
   ```

   MX レコードを使用する場合は、ホスト名とマシン名の両方を指定しなければなりません。 それぞれの MG レコードは単独で 1 行に入力にしてください。 次のように、ワイルドカードを使用することもできます。

   ```
   *.widget.com IN MX 0 post.office.widget.
   ```

   この例では、widget.com ドメイン内の未知のホスト (明示的な MX レコードがないホスト) 宛てのメールが post.office.widget へ転送されるようにします。

   注: ワイルドカード MX レコードをインターネット上で使用するのは適切ではありません。
3. データベースを修正したため、SOA リソース・レコード内のシリアル番号を増やす必要があります。
4. refresh -s named コマンドを入力して、ネーム・サーバー・データベースをリフレッシュします。
5. refresh -s sendmail コマンドを入力し、変更を有効にします。

フォワーダーの構成

フォワーダー・サーバーを構成するには、以下の手順を使用します。 一連のファイルを編集してから、SMIT またはコマンド・ラインを使用して **named** デーモンを始動します。

1. /etc/named.conf ファイルを編集します。

   /etc ディレクトリーの中に named.conf ファイルがない場合は、/usr/samples/tcpip/named.conf というサンプル・ファイルを /etc ディレクトリーにコピーし 編集します。 詳細および conf ファイルの詳細な例については、ファイル参照 の『named.conf File Format for TCP/IP』を参照してください。

   - /etc/named.conf ファイルの options スタンザの中に、転送された要求を受信するネーム・サーバーの IP アドレスをリストした forwarders 行を指定してください。 以下に例を示します。

     ```
     options {
         ...
         directory "/usr/local/domain";
         forwarders { 192.100.61.1; 129.35.128.222; };
         ...
     };
     ```

   - ループバック・ゾーンを指定します。 以下に例を示します。

     ```
     zone "0.0.127.in-addr.arpa" in {
         type master;
         file "named.abc.local";
     };
     ```

   - ヒント・ゾーンを指定します。 以下に例を示します。

     ```
     zone "." IN {
         type hint;
         file "named.ca";
     };
     ```

2. /usr/local/domain/named.ca ファイルを編集します。 詳細およびキャッシュ・ファイルの詳細な例については、ファイル参照 の『DOMAIN Cache File Format for TCP/IP』を参照してください。

   このファイルには、ネットワークのルート・ドメインに対する権限ネーム・サーバーのアドレスが含まれています。 例えば、次のとおりです。

```
; root name servers.
.          IN    NS    relay.century.com.
relay.century.com.   3600000    IN    A     129.114.1.2
```

注: このファイル内のすべての行は、標準リソース・レコード・フォーマットになっていなければなりません。

3. /usr/local/domain/named.abc.local ファイルを編集します。

   詳細およびローカル・データ・ファイルの詳細な例については、ファイル参照 の『DOMAIN Local Data File Format for TCP/IP』を参照してください。

   a) ゾーンの権限の開始 (SOA) とデフォルトの存続時間に関する情報を指定します。 例えば、次のとおりです。

   ```
   $TTL 3h    ;3 hour

   @ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                              1       ;serial
                              3600    ;refresh
                              600     ;retry
                              3600000 ;expire
                              86400   ;negative caching TTL
   )
   ```

   b) ネーム・サーバー (NS) レコードを指定します。 例えば、次のとおりです。

   ```
   <tab>    IN    NS     venus.abc.aus.century.com.
   ```

   c) ポインター (PTR) レコードを指定します。

   ```
   1      IN    PTR    localhost.
   ```

   注: このファイル内のすべての行は、標準リソース・レコード・フォーマットになっていなければなりません。

4. 次のコマンドを入力して /etc/resolv.conf ファイルを作成します。

   ```
   touch /etc/resolv.conf
   ```

   このファイルが存在する場合、ホストはネーム・レゾリューションに、/etc/hosts ファイルでなく、ネーム・サーバーを使用します。

   それに代わる方法として、/etc/resolv.conf ファイルに次のエントリーを入れることもできます。

   ```
   nameserver 127.0.0.1
   ```

   この 127.0.0.1 というアドレスはループバック・アドレスであり、これによってホストはネーム・サーバーとしてのそれ自体にアクセスします。 /etc/resolv.conf ファイルには、次のようなエントリーを入れることもできます。

   ```
   domain domainname
   ```

   前の例では、*domainname* の値は austin.century.com です。

5. 次のいずれかの手順を実行します。

   - smit stnamed SMIT 高速パスを使用して、**named** デーモンを使用可能にします。 これによって、このデーモンはシステムが始動するたびに初期化されます。 現時点で **named** デーモンを始動するか、次回のシステム再始動時に始動するか、それともその両方で始動するかを指示します。
   - /etc/rc.tcpip ファイルを編集します。 次に示す行からコメント記号 (#) を除去することにより、この **named** デーモンの行がコメントとして扱われないようにします。

     ```
     #start /etc/named "$src_running"
     ```

これによって、このデーモンはシステムが始動するたびに初期化されます。

6. SMIT によって named デーモンを初期化しない場合は、次のコマンドを入力することにより、このセッションについてデーモンを始動します。

```
startsrc -s named
```

転送専用ネーム・サーバーの構成

転送専用ネーム・サーバーを構成するには、以下の手順を使用します。 一連のファイルを編集してから、SMIT またはコマンド・ラインを使用して **named** デーモンを始動します。

注 : 転送専用ネーム・サーバーを実行しなくても同様の構成を行うことができます。 代わりに、使用したいフォワーダーを指すネーム・サーバー行が入った /etc/resolv.conf ファイルを作成します。

1. /etc/named.conf ファイルを編集します。

   /etc ディレクトリーの中に named.conf ファイルがない場合は、/usr/samples/tcpip/named.conf というサンプル・ファイルを /etc ディレクトリーにコピーし編集します。 詳細および conf ファイルの詳細な例については、ファイル参照 の『named.conf File Format for TCP/IP』を参照してください。

   - /etc/named.conf ファイルの options スタンザの中に、転送された要求を受信するネーム・サーバーの IP アドレスをリストした forwarders 行および forward only 行を指定してください。 以下に例を示します。

     ```
     options {
         ...
         directory "/usr/local/domain";
         forwarders { 192.100.61.1;  129.35.128.222; };
         forward only;
         ...
     };
     ```

   - ループバック・ゾーンを指定します。 以下に例を示します。

     ```
     zone "0.0.127.in-addr.arpa" in {
         type master;
         file "named.abc.local";
     };
     ```

   - ヒント・ゾーンを指定します。 以下に例を示します。

     ```
     zone "." IN {
         type hint;
         file "named.ca";
     };
     ```

2. /usr/local/domain/named.ca ファイルを編集します。 例えば、次のとおりです。

   詳細およびキャッシュ・ファイルの詳しい例については、ファイル参照 の『DOMAIN Cache File Format for TCP/IP』 を参照してください。 このファイルには、ネットワークのルート・ドメインに対する権限ネーム・サーバーのアドレスが含まれています。

   ```
   ; root name servers.
   .          IN    NS    relay.century.com.
   relay.century.com.   3600000    IN    A     129.114.1.2
   ```

   注 : このファイル内のすべての行は、標準リソース・レコード・フォーマットになっていなければなりません。

3. /usr/local/domain/named.abc.local ファイルを編集します。 詳細およびローカル・データ・ファイルの詳細な例については、ファイル参照 の『DOMAIN Local Data File Format for TCP/IP』を参照してください。

   a) ゾーンの権限の開始 (SOA) とデフォルトの存続時間に関する情報を指定します。 以下に例を示します。

```
$TTL 3h    ;3 hour

@ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                            1       ;serial
                            3600    ;refresh
                            600     ;retry
                            3600000 ;expire
                            86400   ;negative caching TTL
)
```

b) ネーム・サーバー (NS) レコードを指定します。 以下に例を示します。

```
<tab>    IN    NS     venus.abc.aus.century.com.
```

c) ポインター (PTR) レコードを指定します。

```
1      IN    PTR    localhost.
```

注 : このファイル内のすべての行は、標準リソース・レコード・フォーマットになっていなければなりません。

4. 次のコマンドを入力して /etc/resolv.conf ファイルを作成します。

```
touch /etc/resolv.conf
```

このファイルが存在する場合、ホストはネーム・レゾリューションに、/etc/hosts ファイルでなく、ネーム・サーバーを使用します。

それに代わる方法として、/etc/resolv.conf ファイルに次のエントリーを入れることもできます。

```
nameserver 127.0.0.1
```

この 127.0.0.1 というアドレスはループバック・アドレスであり、これによってホストはネーム・サーバーとしてのそれ自体にアクセスします。 /etc/resolv.conf ファイルには、次のようなエントリーを入れることもできます。

```
domain domainname
```

前の例では、*domainname* の値は austin.century.com です。

5. 次のいずれかの手順を実行します。

- smit stnamed SMIT 高速パスを使用して、**named** デーモンを使用可能にします。 これによって、このデーモンはシステムが始動するたびに初期化されます。 現時点で **named** デーモンを始動するか、次回のシステム再始動時に始動するか、それともその両方で始動するかを指示します。
- /etc/rc.tcpip ファイルを編集します。 次に示す行からコメント記号 (#) を除去することにより、この **named** デーモンの行がコメントとして扱われないようにします。

```
#start /etc/named "$src_running"
```

これによって、このデーモンはシステムが始動するたびに初期化されます。

6. SMIT によって **named** デーモンを初期化しない場合は、次のコマンドを入力することにより、このセッションについてデーモンを始動します。

```
startsrc -s named
```

ネーム・サーバーを使用するためのホストの構成

ネーム・サーバーを使用するためにホストを構成するには、次の手順を使用します。

1. 次のコマンドを実行して、/etc/resolv.conf ファイルを作成します。

```
touch /etc/resolv.conf
```

2. /etc/resolv.conf ファイルの 1 行目に、domain というワードとそのあとにこのホストが入っているドメインのフルネームを入力します。 例えば、次のとおりです。

```
domain abc.aus.century.com
```

3. domain 行のあとの任意のブランク行に、nameserver というワードを入力し、それに続けて最低 1 スペースを空けて、さらにそのあとに、このホストが使用する予定のネーム・サーバー (このネーム・サーバーは domain ステートメントで示されたドメインに機能しなければなりません) の IP アドレスを小数点付き 10 進数のフォーマットで入力します。

   ネーム・サーバー・エントリーは最大 3 個まで入力できます。 例えば、/etc/resolv.conf ファイルに次のようなエントリーを入れることができます。

```
nameserver 192.9.201.1
nameserver 192.9.201.2
```

   システムは、このリストに示された順序でネーム・サーバーに照会します。

```
search domainname_list
```

   一方、検索キーワードを、リゾルバーがドメイン・リスト内を照会する順序を指定するために使用することもできます。 この場合、domainname_list 値は「abc.aus.century.com」および「aus.century.com」です。 domainname_list には、それぞれスペースで区切って、最大 1024 の文字ストリングを入れることができます。

4. ネーム・サーバーが作動可能であれば、次のコマンドを入力して、ホストとネーム・サーバーの間の通信をテストできます。

```
host hostname
```

   そのネーム・サーバーによって解決されなければならないホストの名前を使用して、プロセスが機能しているかどうかを調べてください。 表示される出力は次のようになるはずです。

```
brown.abc.aus.century.com is 129.35.145.95
```

その他の構成タスクを、次のテーブルに示します。

表 63. ネーム・サーバーを使用するためのホストの構成のタスク

タスク	SMIT 高速パス	コマンドまたはファイル
/etc/resolv.conf ファイルを作成します。	smit stnamerslv2	/etc/resolv.conf[1] を作成して編集する
ホストが使用するすべてのネーム・サーバーのリスト	smit lsnamerslv	/etc/resolv.conf を表示する
ネーム・サーバーの追加	smit mknamerslv	/etc/resolv.conf[2] を編集する
ネーム・サーバーの除去	smit rmnamerslv	/etc/resolv.conf を編集する
ドメイン・ネーム・レゾリューションを使用した始動と再始動	smit stnamerslv	
ドメイン・ネーム・レゾリューションを使用した停止	smit spnamerslv	
ドメインの変更または表示	smit mkdomain	/etc/resolv.conf を編集する
ドメインの除去	smit rmdomain	/etc/resolv.conf を編集する

関連情報

netsvc.conf ファイル

DNS ネーム・サーバー上の動的ゾーン

named コマンドでは、動的な更新も考慮されています。 named データベース・ファイルおよび構成ファイルは、クライアント・マシンが更新を発行できるように構成されていなければなりません。 ゾーンは、動的にも静的にも設定できます。 デフォルトの「ゾーン」は静的です。

ゾーンを動的にするには、`allow-update` キーワードを、`/etc/named.conf` ファイル内のそのゾーンのスタンザに追加する必要があります。 `allow-update` キーワードは、更新の実行依頼を許可されたホストを定義する IP アドレス適合リストを指定します。 詳細および `conf` ファイルの詳細な例については、ファイル参照 の『named.conf File Format for TCP/IP』を参照してください。 以下の例では、すべてのホストが動的ゾーンの更新を許可されています。

```
zone "abc.aus.century.com" IN {
    type master;
    file "named.abc.data";
    allow-update { any; };
};
```

ゾーンに dynamic のマークを付けると、次の 3 つのモードのセキュリティーを 開始できます。

項目	説明
非セキュア (Unsecured)	すべてのユーザーが任意の時点で、ゾーン内の任意の情報を更新できます。 **重要 :** このモードの使用は、お勧めできません。 このモードでは、データが失われたり、代行受信されたりする恐れがあります。 少なくとも、非セキュア・ゾーンは特定の IP アドレスからの更新のみに限定する必要があります。
制御 (Controlled)	新しい情報を作成したり、既存の情報を置換したりできます。 多くの場合、セキュア状態遷移環境にとっては、これが最も使用しやすいモードです。 このモードでは、すべての着信更新にタイム・スタンプを付け、キー付きの署名を付ける必要があります。
事前保護 (Presecured)	既存の情報に対するすべての更新は、類似情報によって置き換えられなければなりません。 新規情報は作成できません。 このモードでは、すべての着信更新にタイム・スタンプを付け、キー付きの署名を付ける必要があります。

動的ゾーンは、デフォルトでは非セキュア・モードになります。 それ以外のモードを使用するには、`/etc/named.conf` ファイルのゾーン・スタンザ内の `update-security` キーワードのあとに `controlled` または `presecured` を入力します。 これらのキーワードによって、**named** サーバーはそのゾーンで使用するセキュリティー・レベルを知らされます。 以下に例を示します。

```
zone "abc.aus.century.com" IN {
    type master;
    file "named.abc.data";
    allow-update { any; };
    update-security controlled;
};
```

モードを選択したあと、実際のデータ・ファイルをセキュリティー・レベルに合わせて修正する必要があります。 非セキュア・モードでは、データ・ファイルは「現状のまま」使用されます。 制御モードまたはセキュア・モードでは、一連のマスター・サーバー/ホスト名のキーのペアを、ゾーン内の名前ごとに生成する必要があります。 これは、**nsupdate** コマンドに **-g** オプションを使用して行ってください。 このコマンドはキーのペア (秘密鍵と公開鍵) を生成します。 これらのキーは、更新に対して認証署名を行うために必要です。 ゾーン名のリストにすべてのキーを生成したあと、それらのキーをデータ・ファイルに追加する必要があります。 KEY のフォーマットは次のとおりです。

```
Index    ttl    Class    Type    KeyFlags    Protocol    Algorithm    KeyData
```

この場合、

項目	説明
Index	ゾーン内のデータを参照するために使用する名前を指定します。
ttl	このデータの存続時間 (TTL) を指定します。 これはオプション・フィールドです。
Class	データのクラスを指定します。 これはゾーンによって異なりますが、通常は IN です。
Type	レコードのタイプを示します。 この場合は KEY です。
KeyFlags	キーに関する named 情報を与えます。 0x0000 はホストに使用される一般的なキー・レコードを定義します。 0x0100 はゾーン名に関連するキー・レコードを定義します。
Protocol	使用するプロトコルを指定します。 現時点では 1 つだけあり、0 です。
Algorithm	キーのアルゴリズムを指定します。 現時点では 1 つだけあり、1 です。これは、MD5 Private/Public 認証メソッドです。
KeyData	base64 表記でキーを示します。 **nsupdate** コマンドは、公開鍵と秘密鍵の両方を base64 表記で生成します。 公開鍵は、出力ファイルの最後に出力されます。

例えば、動的ゾーン内のホスト名のセキュリティーを保証するには、ホスト名を含むゾーンのゾーン・ファイルに次のような行を追加する必要があります。

```
bears    4660    IN    KEY    0x0000    0    1    AQOtg......
```

上記の例は、bears に KEY レコードが定義されていることを示しています。 bears を更新したいユーザーは、データベース内の公開鍵に適合する秘密鍵を使用して、更新にサインする必要があります。 **nsupdate** コマンドを正常に実行するには、秘密鍵がクライアント上のキー・ファイル (デフォルトは /etc/keyfile) に入っている必要があります。 次のフォーマットを使用してください。

```
 hostname          mastername          base64          key
```

同様の KEY エントリーが、ゾーン定義セクションにも必要です。 ゾーン・キーは、セキュア・モードと制御モードのどちらにも必要です。ゾーン・キーがない場合には、非セキュア・モードと見なされます。 キーの指定は、前記の bears の例のようにしますが、秘密鍵は管理者が **nsupdate** コマンドの管理モードで使用するために残されます。

1. **nsupdate** コマンドを使用してキーのペアを生成するには、次のように入力します。

   ```
   nsupdate -g -h ZoneName -p ServerName -k AdminKeyFile
   ```

   これによってゾーン用のキーが生成されます。 この例では、**nsupdate** は **nsupdate4** にリンクされています。これは次のように入力して行います。

   ```
   ln -fs /usr/sbin/nsupdate4 /usr/sbin/nsupdate
   ```

2. ペアの最後のキーを、次のようにゾーンの先頭セクションに置いてください。

   ```
   IN     KEY     0x0100  0  1  Key
   ```

   named.abc.data ファイルのエントリーは次のとおりです。

   ```
   $TTL 3h    ;3 hour

   @ IN    SOA     venus.abc.aus.century.com. gail.zeus.abc.aus.century.com. (
                   1       ;serial
                   3600    ;refresh
                   600     ;retry
                   3600000 ;expire
                   86400   ;negative caching TTL
   )
           IN      NS      venus.abc.aus.century.com.
           IN      KEY     0x0100  0 1 AQPlwHmIQeZzRk6Q/nQYhs3xwnhfTgF/
                           8Y1BVzKSoKxVKPNLINnYW0mB7attTcfhHaZZcZr4u/
                           vDNikKnhnZwgn/
   venus   IN      A       192.9.201.1
   ```

```
earth   IN      A       192.9.201.5
mars    IN      A       192.9.201.3
```

3. これで、ゾーンはネーム・サーバーのリフレッシュでロードできるようになりました。 ゾーンを更新するクライアントまたは DHCP サーバーに、AdminKeyFile を置きます。 AdminKeyFile が入っているゾーン・キーは、ネーム・サーバーに対する更新および保守操作に使用できます。

BIND 9 セキュリティー

BIND 9 は、**named** のセキュリティー方法としてトランザクション署名 (TSIG) と署名 (SIG) を提供します。

BIND 9 で機能するネーム・サーバーは、デフォルトでは、BIND 8 の場合と同様に、権限のあるゾーンの動的更新はできません。

BIND 9 は、主にサーバー間の通信のためのトランザクション署名 (TSIG) をサポートします。 これには、ゾーン転送、通知、および再帰的照会メッセージが含まれます。 また、TSIG は動的更新にも有用です。 動的ゾーンの 1 次サーバーは更新を制御するためのアクセス制御に使用しますが、IP ベースのアクセス制御では十分ではありません。

アクセス制御リストの現行方式ではなくキーをベースにした暗号化を使用することで、TSIG を、動的ゾーンを更新できる人を制限するために使用できます。 動的更新の ACL (アクセス制御リスト) 方式と異なり、TSIG キーは、ネーム・サーバー上の構成ファイルを変更せずに、他の更新者に配布できます。これは、ネーム・サーバーが構成ファイルを再度読み取る必要はないということです。

BIND 9 は、BIND 8 でインプリメントされたすべてのキーワードを備えているわけではないことに注意してください。 今回の例では、BIND 8 からの簡単なマスター構成を使用しています。

注 : named 9 を使用するには、次のコマンドを入力して、**named** デーモンから **named9**、および **nsupdate** から **nsupdate9** のシンボリック・リンクを、再リンクする必要があります。

1. `ln -fs /usr/sbin/named9 /usr/sbin/named`
2. `ln -fs /usr/sbin/nsupdate9 /usr/sbin/nsupdate`

1. **dnssec-keygen** コマンドを使用して、キーを生成します。

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST keyname
```

   - HMAC-MD5 は、暗号化に使用するアルゴリズム
   - 128 は使用するキーの長さ (ビット数)
   - HOST: HOST は、共有鍵暗号化に関するホスト・キーの生成に使用する TSIG キーワードです。

   次のコマンドは、

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST venus-batman.abc.aus.century.com
```

   次の 2 つのキー・ファイルを生成します。

```
Kvenus-batman.abc.aus.century.com.+157+35215.key
Kvenus-batman.abc.aus.century.com.+157+35215.private
```

   - 157 は使用するアルゴリズム (HMAC-MD5)
   - 35215 は指紋。これは 1 つのゾーンに複数のキーが許されるため、DNNSEC 内で有用

2. エントリーを、マスター・ネーム・サーバー上の named.conf に追加します。

```
// TSIG Key
key venus-batman.abc.aus.century.com. {
        algorithm hmac-md5;
        secret "+UWSvbpxHWFdNwEAdy1Ktw==";
};
```

   HMAC-MD5 が使用されていると想定すると、両方のキー・ファイルには共有鍵が含まれ、これらのキー・ファイルは、ファイル内の最後のエントリーとして保管されます。 共有秘密鍵をクライアントにコピーするセキュアな方法を見付けます。 共有秘密鍵のコピーだけで、キー・ファイルはコピーする必要はありません。

次は、ファイル Kvenus-batman.abc.aus.century.com.+157+35215.private のエントリーです。

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: +UWSvbpxHWFdNwEAdy1Ktw==
```

次は、マスター・ネーム・サーバーの named.conf ファイルの例です。 ゾーン abc.aus.century.com によって、キー venus-batman.abc.aus.century.com を持つサーバーに対してのみ、ゾーン転送と動的更新ができます。 同じことを逆ゾーンに行います。こちらは、更新者は共有鍵を持っている必要があります。

```
// TSIG Key
key venus-batman.abc.aus.century.com. {
        algorithm hmac-md5;
        secret "+UWSvbpxHWFdNwEAdy1Ktw==";
};

options {
        directory "/usr/local/domain";
};

zone "abc.aus.century.com" in {
        type master;
        file "named.abc.data";
        allow-transfer { key venus-batman.abc.aus.century.com.;};
        allow-update{ key venus-batman.abc.aus.century.com.; };
};
```

ゾーン転送がキーを持つものに制限されているため、スレーブ・ネーム・サーバーの named.conf ファイルも編集する必要があります。 192.9.201.1(venus.abc.aus.century.com) に対するすべての要求は、キーによって署名されます。 キーの名前 (venus-batman.abc.aus.century.com.) は、それを使用するサーバー上のキーの名前と一致している必要があることに注意してください。

次は、スレーブ・ネーム・サーバー上の named.conf ファイルの例です。

```
// TSIG Key
key venus-batman.abc.aus.century.com. {
        algorithm hmac-md5;
        secret "+UWSvbpxHWFdNwEAdy1Ktw==";
};

server 192.9.201.1{
        keys { venus-batman.abc.aus.century.com.;};
};

options {
        directory "/usr/local/domain";
};

zone "abc.aus.century.com" IN {
    type slave;
    file "named.abc.data.bak";
    masters { 192.9.201.1; };
};
```

BIND 9 トランザクション署名

BIND 9 は、主にサーバー間の通信のためのトランザクション署名 (TSIG) をサポートします。

これには、ゾーン転送、通知、および再帰的照会メッセージが含まれます。 また、TSIG は動的更新にも有用です。 動的ゾーンの 1 次サーバーは更新を制御するためのアクセス制御に使用しますが、IP ベースのアクセス制御では十分ではありません。

アクセス制御リストの現行方式ではなくキーをベースにした暗号化を使用することで、TSIG を、動的ゾーンを更新できる人を制限するために使用できます。 動的更新の ACL (アクセス制御リスト) 方式と異なり、TSIG キーは、ネーム・サーバー上の構成ファイルを変更せずに、他の更新者に配布できます。 これは、ネーム・サーバーが構成ファイルを再度読み取る必要はないということです。

BIND 9 は、BIND 8 でインプリメントされたすべてのキーワードを備えているわけではないことに注意してください。 今回の例では、BIND 8 からの簡単なマスター構成を使用しています。

注 : named 9 を使用するには、次のコマンドを入力して、**named** デーモンから **named9**、および **nsupdate** から **nsupdate9** のシンボリック・リンクを、再リンクする必要があります。

1. `ln -fs /usr/sbin/named9 /usr/sbin/named`
2. `ln -fs /usr/sbin/nsupdate9 /usr/sbin/nsupdate`

1. **dnssec-keygen** コマンドを使用して、キーを生成します。

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST keyname
```

   - HMAC-MD5 は、暗号化に使用するアルゴリズム
   - 128 は使用するキーの長さ (ビット数)
   - HOST: HOST は、共有鍵暗号化に関するホスト・キーの生成に使用する TSIG キーワードです。

   次のコマンドは、

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST venus-batman.abc.aus.century.com
```

   次の 2 つのキー・ファイルを生成します。

```
Kvenus-batman.abc.aus.century.com.+157+35215.key
Kvenus-batman.abc.aus.century.com.+157+35215.private
```

   - 157 は使用するアルゴリズム (HMAC-MD5)
   - 35215 は指紋。これは 1 つのゾーンに複数のキーが許されるため、DNNSEC 内で有用

2. エントリーを、マスター・ネーム・サーバー上の named.conf に追加します。

```
// TSIG Key
key venus-batman.abc.aus.century.com. {
         algorithm hmac-md5;
         secret "+UWSvbpxHWFdNwEAdy1Ktw==";
};
```

   HMAC-MD5 が使用されていると想定すると、両方のキー・ファイルには共有鍵が含まれ、これらのキー・ファイルは、ファイル内の最後のエントリーとして保管されます。 共有秘密鍵をクライアントにコピーするセキュアな方法を見付けます。 共有秘密鍵のコピーだけで、キー・ファイルはコピーする必要はありません。

   次は、ファイル Kvenus-batman.abc.aus.century.com.+157+35215.private のエントリーです。

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: +UWSvbpxHWFdNwEAdy1Ktw==
```

   次は、マスター・ネーム・サーバーの named.conf ファイルの例です。 ゾーン abc.aus.century.com によって、キー venus-batman.abc.aus.century.com を持つサーバーに対してのみ、ゾーン転送と動的更新ができます。 同じことを逆ゾーンに行います。こちらは、更新者は共有鍵を持っている必要があります。

```
// TSIG Key
key venus-batman.abc.aus.century.com. {
         algorithm hmac-md5;
         secret "+UWSvbpxHWFdNwEAdy1Ktw==";
};

options {
         directory "/usr/local/domain";
};
zone "abc.aus.century.com" in {
         type master;
         file "named.abc.data";
```

```
        allow-transfer { key venus-batman.abc.aus.century.com.;};
        allow-update{ key venus-batman.abc.aus.century.com.; };
};
```

ゾーン転送がキーを持つものに制限されているため、スレーブ・ネーム・サーバーの named.conf ファイルも編集する必要があります。 192.9.201.1(venus.abc.aus.century.com) に対するすべての要求は、キーによって署名されます。 キーの名前 (venus-batman.abc.aus.century.com.) は、それを使用するサーバー上のキーの名前と一致している必要があることに注意してください。

次は、スレーブ・ネーム・サーバー上の named.conf ファイルの例です。

```
// TSIG Key
key venus-batman.abc.aus.century.com. {
        algorithm hmac-md5;
        secret "+UWSvbpxHWFdNwEAdy1Ktw==";
};

server 192.9.201.1{
        keys { venus-batman.abc.aus.century.com.;};
};

options {
        directory "/usr/local/domain";
};

zone "abc.aus.century.com" IN {
     type slave;
     file "named.abc.data.bak";
     masters { 192.9.201.1; };
};
```

BIND 9 署名

BIND 9 は、RFC 2535 に指定されているとおりに、DNSSEC SIG トランザクション署名の一部をサポートします。

SIG は公開鍵と秘密鍵を使用してメッセージを認証します。

SIG レコードによって、管理者はゾーン・データに署名でき、その結果、それが本物であると表明します。

ルート・ゾーンの保護

ルート・ゾーンを保護するための手順を使用する場合は、インターネット上の他のネーム・サーバーが BIND 9 を使用しないことを前提とし、ゾーン・データを保護して、他のサーバーがそのゾーン・データを検証できるようにします。

ゾーン (この例では aus.century.com) がセキュア・ルートであることを表明し、その下のセキュア・ゾーン・データをすべて検証します。

1. **dnssec-keygen** コマンドを使用して、キーを生成します。

   ```
   dnssec-keygen -a RSA -b 512 -r /usr/sbin/named -n ZONE aus.century.com.
   ```

   注 : まず次のコマンドを実行して DNS ライブラリーをセキュア DNS ライブラリーに再リンクする必要がありますが、OpenSSL がインストールされていれば、キー生成のアルゴリズムとして RSA 暗号化を使用できます。

   ```
   ln -fs /usr/lib/libdns_secure.a /usr/lib/libdns.a
   ```

   - ZONE: ZONE は、公開鍵/秘密鍵暗号化のためのゾーン・キーの生成に使用する DNSSEC キーワード
   - r フラグはランダム・デバイスを指定

2. named.conf ファイルと同じように、公開鍵のエントリーを追加します。

   この例で使用するエントリーは次のとおりです。 次は、キー・ファイル Kaus.century.com.+001+03254.key の内容です。

   ```
   abc.aus.century.com. IN KEY 256 3 1
   AQOnfGEAg0xpzSdNRe7KePq3Dl4NqQiq7HkwKl6TygUfaw6vz6ldmauB4UQFcGKOyL68/
   Zv5ZnEvyB1fMTAaDLYz
   ```

公開鍵は、ファイル Kzonename.+algor.+fingerprint.key に入っています。すなわち、ここでの例では Kaus.century.com.+001+03254.key に入っています。 クラス IN を除去し、KEY を入力し、キーを引用符で囲む必要があります。 このエントリーを /etc/named.conf ファイルに追加したあと、ネーム・サーバーをリフレッシュすると、ゾーン aus.century.com がセキュア・ルートになります。

```
trusted-keys {
        aus.century.com. 256 3 1 "AQOnfGEAg0xpzSdNRe7KePq3Dl4NqQiq7HkwKl6Tyg
Ufaw6vz6ldmauB 4UQFcGKOyL68/Zv5ZnEvyB1fMTAaDLYz";
};
options {
        directory "/usr/local/domain";
};

zone "abc.aus.century.com" in {
        type master;
        file "named.abc.data.signed";
        allow-update{192.9.201.1;};
};
```

トラストのチェーンの設定
セキュア・ルートが得られたため、残りの子ゾーンを保護することができます。 ここでは、ゾーン abc.aus.century.com を保護します。

次のステップに従って、残りの子ゾーンを保護します。

1. **dnssec-keygen** コマンドを使用して、キーのペアを生成します。

   ```
   dnssec-keygen -a RSA -b 512 -r /usr/sbin/named -n ZONE abc.aus.century.com.
   ```

   r フラグはランダム入力ファイルを指定。
2. **dnssec-makekeyset** コマンドを実行して、キー・セットを作成します。

   ```
   dnssec-makekeyset -t 172800 Kabc.aus.century.com.+001+11515.key
   ```

   ここで、*Kabc.aus.century.com.+001+11515.key* はユーザー独自の公開鍵です。

   これによって、keyset-abc.aus.century.com というキー・セット・ファイルが作成されます。
3. このキー・セット・ファイルを親ゾーンに送信して、署名を取得します。 ここでは、親ゾーンはセキュア・ルート・ゾーンの aus.century.com です。
4. 親は、親の秘密鍵を使用してキーに署名する必要があります。

   ```
   dnssec-signkey keyset-abc.aus.century.com. Kaus.century.com.+001+03254.private
   ```

   これによって、signedkey-abc.aus.century.com というファイルが生成され、親は、このファイルを子ゾーンに送り返す必要があります。
5. ゾーン abc.aus.century.com の子ネーム・サーバー上で、$INCLUDE Kabc.aus.century.com.+001+11515.key をプレーン・ゾーン・ファイル named.abc.data に追加します。 signedkey-abc.aus.century.com ファイルを、ゾーン・ファイル named.abc.data と同じロケーションに置きます。 次のステップでゾーンが署名されると、プログラムは、signedkey-abc.aus.century.com をインクルードすることを理解します。これは、親から受信したものです。

   ```
   $TTL 3h    ;3 hour

   @ IN    SOA     venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (
                   1       ;serial
                   3600    ;refresh
                   600     ;retry
                   3600000 ;expire
                   86400   ;negative caching TTL
   )
   $INCLUDE Kabc.aus.century.com.+001+03254.key
   ```
6. **dnssec-signzone** コマンドを使用して、ゾーンを署名します。

```
dnssec-signzone -o abc.aus.century.com. named.abc.data
```

7. 子ゾーン abc.aus.century.com にある **named.conf** ファイルを変更して、新規署名ゾーン・ファイル (named.abc.data.signed) を使用するようにします。 例えば、次のとおりです。

```
options {
        directory "/usr/local/domain";
};

zone "abc.aus.century.com" in {
        type master;
        file "named.abc.data.signed";
        allow-update{192.9.201.1;};
};
```

8. ネーム・サーバーをリフレッシュします。

詳しくは、474 ページの『ネーム・レゾリューションに関する問題』を参照してください。

LDAP ネーム・レゾリューションの計画と構成 (IBM SecureWay ディレクトリー・スキーマ)

Lightweight Directory Access Protocol (LDAP) は、ディレクトリー内の情報へのアクセスおよび更新の方法を定義する、業界標準です。

LDAP スキーマは、データの順序付けの規則を定義します。 IBM SecureWay Directory スキーマの一部である **ibm-HostTable** オブジェクト・クラスを使用すれば、ネットワーク上のすべてのホストのための、「name-to-Internet-address mapping information (名前からインターネット・アドレスへのマッピング情報)」を保管できます。

ibm-HostTable オブジェクト・クラスは、次のように定義します。

```
Object Class name:     ibm-HostTable
Description:           Host Table entry which has a collection of hostname to
                       IP address mappings.
OID:                   TBD
RDN:                   ipAddress
Superior object class: top
Required Attributes:   host, ipAddress
Optional Attributes:   ibm-hostAlias, ipAddressType, description
```

属性の定義は、次のとおりです。

```
Attribute Name: ipAddress
Description:    IP Address of the hostname in the Host Table
OID:            TBD
Syntax:         caseIgnoreString
Length:         256
Single Valued:  Yes
Attribute Name: ibm-hostAlias
Description:    Alias of the hostname in the Host Table
OID:            TBD
Syntax:         caseIgnoreString
Length:         256
Single Valued:  Multi-valued
Attribute Name: ipAddressType
Description:    Address Family of the IP Address (1=IPv4, 2=IPv6)
OID:            TBD
Syntax:         Integer
Length:         11
Single Valued:  Yes
Attribute Name: host
Description:    The hostname of a computer system.
OID:            1.13.18.0.2.4.486
Syntax:         caseIgnoreString
Length:         256
Single Valued:  Multi-valued
Attribute Name: description
Description:    Comments that provide a description of a directory object entry.
OID:            2.5.4.13
Syntax:         caseIgnoreString
Length:         1024
Single Valued:  Multi-valued
```

名前からインターネット・アドレスへのマッピング・ホスト情報を保管できるように、IBM SecureWay ディレクトリー・スキーマに準拠する **LDAP** サーバーを構成するには、次の手順に従ってください。

1. **LDAP** サーバーにサフィックスを追加します。

   サフィックスは、ホスト・データベースの開始点です。 例えば、「cn=hosts」を追加します。 サフィックスは、Web ベースの IBM SecureWay Directory サーバー管理ツールを使用して追加することができます。
2. LDAP データ交換フォーマット (LDIF) ファイルを作成します。

   これは、手動で、または /etc/hosts ファイルから LDIF を作成する **hosts2ldif** コマンドで、行うことができます。 詳しくは、hosts2ldif コマンドを参照してください。 LDIF ファイルのサンプルを、次に示します。

   ```
   dn: cn=hosts
   objectclass: top
   objectclass: container
   cn: hosts
   dn: ipAddress=1.1.1.1, cn=hosts
   host: test
   ipAddress: 1.1.1.1
   objectclass: ibm-HostTable
   ipAddressType: 1
   ibm-hostAlias: e-test
   ibm-hostAlias: test.austin.ibm.com
   description: first ethernet interface
   dn: ipAddress=fe80::dead, cn=hosts
   host: test
   ipAddress: fe80::dead
   objectclass: ibm-HostTable
   ipAddressType: 2
   ibm-hostAlias: test-ll
   ibm-hostAlias: test-ll.austin.ibm.com
   description: v6 link level interface
   ```
3. **LDAP** サーバー上の LDIF ファイルからホスト・ディレクトリー・データをインポートします。

   これは、**ldif2db** コマンドを使用するか、Web ベースの IBM SecureWay Directory サーバー管理ツールを使用して行うことができます。

LDAP メカニズムを使用して、LDAP サーバー上のホスト・データベースにアクセスできるようにクライアントを構成するには、次の手順に従ってください。

1. /etc/resolv.ldap ファイルを作成します。 詳細および resolv.ldap ファイルの詳細な例については、ファイル参照 の『resolv.ldap File Format for TCP/IP』を参照してください。
2. NSORDER 環境変数、/etc/netsvc.conf ファイル、または /etc/irs.conf ファイルを使用して、デフォルトのネーム・レゾリューションを変更します。 詳しくは、ファイル参照 の『netsvc.conf File Format for TCP/IP』または『irs.conf File Format for TCP/IP』を参照してください。

依然としてサポートされてはいますが、ldap メカニズムは使用すべきではありません。 この既存の ldap メカニズムは、IBM SecureWay Directory スキーマで機能します。これに対し、nis_ldap (NIS_LDAP) は RFC 2307 スキーマで機能します。ldap メカニズムではなく、nis_ldap メカニズムの使用をお勧めします。 nis_ldap ネーム・レゾリューションについては、211 ページの『NIS_LDAP ネーム・レゾリューションの計画と構成 (RFC 2307 スキーマ)』を参照してください。

NIS_LDAP ネーム・レゾリューションの計画と構成 (RFC 2307 スキーマ)

AIX 5.2 では、NIS_LDAP と呼ばれる新しい命名メカニズムを提供します。

既存の LDAP メカニズムと新しい NIS_LDAP メカニズムとの違いは、LDAP スキーマ (エンティティーを記述するための属性のグループ分けを決定する属性およびオブジェクト・クラスのセット) にあります。 既存の LDAP メカニズムは、IBM SecureWay Directory スキーマに準拠した LDAP サーバーで機能し、ホスト命名サービスのみをサポートします。 NIS_LDAP メカニズムは、RFC 2307 スキーマに準拠する LDAP サーバーで機能し、すべての NIS サービス (ユーザーとグループ、ホスト、サービス、プロトコル、ネットワーク、およびネットグループ) をサポートします。 RFC 2307 は、ユーザーとグループをはじめとするネットワーク情報サービスの記述に使用する、一連の属性とオブジェクト・クラスを定義します。

- LDAP サーバーを構成するには、LDAP サーバーをセットアップし、そのサーバーに必要なデータをマイグレーションする必要があります。

  a) **mksecldap** コマンドを使用して、サーバーをセットアップします。

  nis_ldap メカニズムは、RFC 2307 スキーマでしか機能しません。 LDAP サーバーのセットアップの間に、**mksecldap** コマンドが -S rfc2307 または -S rfc2307aix オプションを付けて呼び出されます (-S aix オプションではありません。このオプションは IBM SecureWay Directory スキーマを指定するものです)。 デフォルトでは、**mksecldap** コマンドは、ローカル・システムに定義されたユーザーとグループを LDAP サーバーに移行します。 この移行を行わないようにするには、-u NONE オプションを使用します。

```
mksecldap -s -a cn=admin -p adminpwd -S rfc2307aix
```

  これは、LDAP サーバーを、管理者 DN を「cn=admin」、パスワードを「adminpwd」でセットアップします。 デフォルトのサフィックス cn=aixdata も、LDAP サーバー構成ファイル /etc/slapd32.conf に追加されます。

  デフォルトでは、**mksecldap** コマンドは、ローカル・システムに定義されたユーザーとグループを LDAP サーバーに移行します。 この移行をしない場合は、-u NONE オプションを使用します。これは、ローカルのユーザーとグループを LDAP サーバーに移行しないため、あとから NIS ユーザーとグループのみ追加できます。

```
mksecldap -s -a cn=admin -p adminpwd -u NONE
```

  b) NIS データを移行します。 NIS サーバーから **nistoldif** コマンドを使用して、NIS マップを LDAP サーバーに移行します。 **nistoldif** コマンドを使用して、フラット・ファイルからのデータを移行することもできます。

  LDAP サーバーに移行する必要のある NIS データが入っているシステムで、**nistoldif** コマンドを実行します。

```
nistoldif -h server1.ibm.com -a cn=admin -p adminpwd -d cn=aixdata
```

  これは、NIS マップを、ローカル・システムから LDAP サーバー server1.ibm.com に移行します。NIS データは、cn=aixdata DN の下に置かれます。 **nistoldif** コマンドを実行して、任意のシステム上にあるフラット・ファイルからのデータを、LDAP サーバーに移行することもできます。 フラット・ファイルは、NIS サーバーには存在していないマップのために使用されます。

  注 : 名前は、LDAP サーバーの cn 属性によって表されます。 RFC 2307 で定義された cn 属性は、大文字小文字の区別はしません。 大文字小文字が異なるだけの名前は、サーバー上で一緒にされます。突き合わせでも大文字小文字の区別はされません。TCP、tcp、または Tcp を検索すると、すべて TCP のプロトコルのエントリーが戻されます。

- LDAP クライアントを、LDAP サーバーからの名前をアクセスするように構成するには、クライアントのセットアップのオプションを 付けて **mksecldap** コマンドを実行します。

  a) **mksecldap** コマンドは、LDAP サーバー名、ポート、管理者 DN、パスワード、およびベース DN を /etc/security/ldap/ldap.cfg ファイルに保管します。このファイルは、**secldapclntd** デーモンが起動時に読み取ります。 セットアップが正常であれば、**mksecldap** コマンドは **secldapclntd** デーモンを自動的に始動します。

  詳しくは、コマンド・リファレンス 第 *5* 巻の /etc/security/ldap/ldap.cfg ファイル ファイル参照 の および **secldapclntd** デーモンを参照してください。

  b) **mksecldap** コマンドは、nis_ldap メカニズムを /etc/netsvc.conf ファイルと /etc/irs.conf ファイルに追加して、ネーム・レゾリューションに LDAP を使用するようにします。 NSORDER 環境変数を手動で nis_ldap に設定して、NIS_LDAP ネーム・レゾリューションを使用するようにすることもできます。

```
mksecldap -c -a cn=admin -p adminpwd -h server1.ibm.com
```

これは、ローカル・システムが server1.ibm.com LDAP サーバーを使用するようにセットアップします。 LDAP サーバーの管理者 DN とパスワードを提供して、このクライアントがサーバーを 認証できるようにしなければなりません。 /etc/netsvc.conf と /etc/irs.conf ファイルが更新され、ネーム・レゾリューションが NIS_LDAP を介して解決できるようになります。

詳しくは、ファイル参照の TCP/IP の /etc/netsvc.conf ファイル・フォーマットまたは TCP/IP の /etc/irs.conf ファイル・フォーマットを参照してください。

c) ユーザーとグループのネーム・レゾリューションは、/etc/netsvc.conf または /etc/irs.conf ファイルでは制御されません。 これは /etc/security/user ファイルを通して制御されます。 LDAP ユーザーが AIX システムにログインできるようにするには、そのクライアント・システムの /etc/security/user ファイル内で、ユーザーの SYSTEM と registry 変数を「LDAP」に設定します。
これを行うには、**chuser** コマンドを実行します。

```
chuser -R LDAP SYSTEM=LDAP registry=LDAP foo
```

ユーザーのシステムを、すべての LDAP ユーザーがシステムにログインできるように 構成できます。 それには、/etc/security/user ファイルを編集します。 root スタンザに registry = files を追加します。 次に、SYSTEM = LDAP と registry = LDAP を default スタンザに追加します。

ユーザー認証の詳細については、セキュリティーの LDAP を参照してください。

関連情報
NIS から RFC 2307 準拠の LDAP サービスへの移行

TCP/IP アドレスとパラメーターの割り当て - 動的ホスト構成プロトコル

伝送制御プロトコル/インターネット・プロトコル (TCP/IP) を使用すると、構成済みアドレスを持つマシン間で通信が可能になります。 ネットワーク管理者が直面する難題の 1 つとして、ネットワーク上のすべてのマシンに対してアドレスを割り当て、パラメーターを分配しなければならないという問題があります。 通常は、この作業は、各ユーザーが各自でマシンを構成できるように、管理者が各ユーザーに対して構成を記述するプロセスです。 しかし、構成間違いや誤解によって、サービス・コールがかかり、管理者が個々に処理しなければならなくなることがあります。 **動的ホスト構成プロトコル (DHCP)** を使用すると、ネットワーク管理者はそのような構成問題からエンド・ユーザーを解放するとともに、ネットワーク構成を中央で維持することができます。

DHCP は、ネットワーク上のクライアント・マシンが IP アドレスやその他の構成パラメーターをサーバーから入手できるようにするアプリケーション層のプロトコルです。 これは、クライアント上のあるデーモンとサーバー上の別のデーモンとの間でパケットを交換することによって、情報を入手します。 現在では、ほとんどのオペレーティング・システムが、その基本パッケージに **DHCP** クライアントを提供しています。

アドレスを入手するために、**DHCP** クライアント・デーモン (**dhcpcd**) は **DHCP** 検出メッセージをブロードキャストします。そのメッセージはサーバーによって受信されて処理されます。 (ネットワーク上に複数のサーバーを余分に構成することができます。) そのクライアントに使用可能なアドレスがあれば、**DHCP** オファー・メッセージが作成されます。このメッセージには、そのクライアント用の IP アドレスとその他のオプションが入っています。 クライアントは、サーバー **DHCP** オファーを受信し、一方ではその他のオファーを待ちながらそれを格納します。 クライアントは最適なオファーを選択し、そのオファーが必要とするサーバーを指定して **DHCP** 要求をブロードキャストします。

要求は、すべての構成済み **DHCP** サーバーにより受信されます。 そして、各サーバーは、要求されているサーバーが自分であるかどうかを検査します。 もし違えば、サーバーはそのクライアントに割り当てられているアドレスを解放します。 要求されたサーバーはこのアドレスを割り当て済みとマークし、**DHCP** 肯定応答を戻します。これでトランザクションは完了します。 クライアントはサーバーによって 指定された時間 (貸出時間) だけアドレスを保持します。

この貸出時間の半分が使用されると、クライアントは貸出時間を延長するために、サーバーに更新 パケットを送信します。 サーバーは、更新に応じる場合には **DHCP** 肯定応答を送信します。 クライアントがそのクライアントの現行アドレスを所有するサーバーから応答を取得できなかった場合、例えば、そのサーバーがあるネットワークから別のネットワークに移動していたとすると、そのサーバーに届くように **DHCP** 再バインド・パケットがブロードキャストされます。 貸出時間を使い切っても、クライアントがアドレス

を更新できなかった場合、このインターフェースは終了し、このプロセスが最初から繰り返されます。 このサイクルは、ネットワーク上の複数のクライアントが同じアドレスに割り当てられるのを防ぎます。

DHCP サーバーは、キーに基づいてアドレスを割り当てます。 4 つの共通キーがあります。ネットワーク、クラス、ベンダー、およびクライアント ID です。 サーバーは、これらのキーを使用して、クライアントへ戻すアドレスと 1 組の構成オプションを取得します。

network
どのネットワーク・セグメントからのパケットであるかを識別する。 ネットワーク・キーを使用すると、サーバーは、そのアドレス・データベースを検査し、ネットワーク・セグメントによってアドレスを割り当てることができます。

class
クライアントが完全に構成できるキーである。 これでアドレスとオプションを指定できます。 このキーは、ネットワーク内のマシン機能を示したり、管理目的でマシンがどのようにグループ化されるのかを示したりするために使用できます。 例えば、ネットワーク管理者は、NetBIOS クライアントのためのオプションを含む `netbios` クラスを作成したり、特定のプリンターにアクセスする必要のあるアカウンティング部門マシンを表す `accounting` クラスを 作成することができます。

vendor
クライアントをそのハードウェア/ソフトウェアのプラットフォーム (Microsoft Windows 95 クライアントや OS/2 Warp クライアントなど) によって識別できるようにする。

client ID
クライアントを、マシンのホスト名またはそのメディア・アクセス制御 (MAC) 層アドレスによって識別する。 クライアント ID は、**dhcpcd** デーモンの構成ファイル内で指定されます。 また、サーバーはクライアント ID を使用して、特定のクライアントへオプションを 渡したり、特定のクライアントにパラメーターの受信を禁止したりすることができます。

これらのキーは、単独に、または組み合わせて、構成で使用することができます。 複数のキーがクライアントから提供され、複数のアドレスを割り当てることができる場合は、キーは 1 つだけ選択され、その選択されたキーからまず最初にオプション・セットが生成されます。 キーとアドレスの選択の詳細については、217 ページの『DHCP の構成』 を参照してください。

リレー・エージェントは、クライアントからの初期のブロードキャストがローカル・ネットワークから出られるようにするために必要です。 このエージェントは、BOOTP リレー・エージェントと呼ばれます。 リレー・エージェントは、**DHCP** パケットと **BOOTP** パケット用の転送エージェントとして機能します。

DHCP サーバー
AIX オペレーティング・システムでは、**DHCP** サーバーは 3 つの主要部分に区分されています。

DHCP サーバーの主要コンポーネントは、データベース、プロトコル・エンジン、1 組のサービス・スレッドから成り、それぞれに構成情報があります。

DHCP データベース
`db_file.dhcpo` データベースは、クライアントとアドレスを追跡するため、また、アクセス制御 (例えば、あるネットワーク上の特定のクライアントにはアクセスを 許可するがその他には許可しない場合や、特定のネットワーク上の **BOOTP** クライアントを使用不可にする場合) のために使用されます。

オプションも、検索用およびクライアントへの配布用に、このデータベースに格納されます。 このデータベースは、動的にロード可能なオブジェクトとしてインストールされ、これによって、サーバーのアップグレードと保守が容易になります。

このデータベースは、構成ファイル内の情報を使用して用意され、整合性を検査されます。 1 組のチェックポイント・ファイルがデータベースに対する更新を処理し、主記憶ファイルへの書き込みのオーバーヘッドを削減します。 また、このデータベースには、アドレスとオプションのプールが 入っていますが、これらは静的なプールです。これについては、217 ページの『DHCP の構成』 に説明があります。

主記憶ファイルとそのバックアップは、フラット ASCII ファイルで、編集することができます。 データベース主記憶ファイルのフォーマットは、以下のとおりです。

```
DF01
"CLIENT ID" "0.0.0.0" State LeaseTimeStart LeaseTimeDuration LeaseTimeEnd
  "Server IP Address" "Class ID" "Vendor ID" "Hostname" "Domain Name"
"CLIENT ID" "0.0.0.0" State LeaseTimeStart LeaseTimeDuration LeaseTimeEnd
```

```
  "Server IP Address" "Class ID" "Vendor ID" "Host Name" "Domain Name"
...
```

最初の行はこのファイルのバージョン ID である DF01c です。 その後の行は、クライアント・レコード定義行です。 サーバーは、この 2 行目からファイルの終わりまでを読み取ります。 (引用符で囲まれたパラメーターは、引用符で囲まれている必要があります。)

"*CLIENT ID*"
クライアントがサーバーにそれ自体を表すために使用する ID。

"0.0.0.0"
現在、**DHCP** サーバーに割り当てられている IP アドレス。 アドレスが割り当てられていない場合は、"0.0.0.0" となります。

State
クライアントの現在の状態。 **DHCP** プロトコル・エンジンには許容セットが含まれ、その状態は **DHCP** データベースで維持されます。 *State* の次の数字はその値を表します。 状態としては、次のものがあります。

(1) FREE (空き)
使用できるようにアドレスが空いていることを表す。 一般に、クライアントにアドレスが割り当てられている場合は、クライアントはこの状態にはなりません。 **dadmin** および **lssrc** からの出力は、この状態を「Free」と報告します。

(2) BOUND (バインド済み)
クライアントとアドレスが結合され、クライアントはこのアドレスにしばらく割り当てられていたことを示す。 **dadmin** および **lssrc** からの出力は、この状態を「Leased」と報告します。

(3) EXPIRED (有効期限切れ)
クライアントとアドレスが結合されているが、解放済みアドレスの場合と同様、情報目的だけであることを示す。 ただし、この有効期限切れ状態は、その貸出期間を期限切れにしたクライアントを表します。 有効期限切れアドレスは、新たに使用に回すことができ、すべての空いているアドレスが使用不能になった後で、解放済みアドレスが再度割り当てられる前に、再割り当てされます。 **dadmin** および **lssrc** からの出力は、この状態を「Expired」と報告します。

(4) RELEASED (解放済み)
クライアントとアドレスが情報目的でのみ結合されていることを示す。 **DHCP** プロトコルは、**DHCP** サーバーがサービスしたクライアントについての 情報を将来の参照に備えて保持するように提案します (主な目的は、参照したときに以前そのアドレスに割り当てられていたクライアントに同じアドレスを与えるためです)。 この状態は、クライアントがそのアドレスを解放したことを示します。 このアドレスは、他に使用できるアドレスがない場合に、別のクライアントで使用することができます。 **dadmin** および **lssrc** からの出力は、この状態を「Released」と報告します。

(5) RESERVED (予約済み)
クライアントとアドレスがゆるやかに結合されていることを示す。 このクライアントは **DHCP** 検出メッセージを出しており、**DHCP** サーバーが応答したが、クライアントはそのアドレスに 対する **DHCP** 要求でまだ応答していません。 **dadmin** および **lssrc** からの出力は、この状態を「Reserved」と報告します。

(6) BAD (正しくない)
ネットワークで使用されているが、**DHCP** サーバーによって分配されていないアドレスを表す。 この状態は、クライアントがリジェクトしたアドレスも 表します。 この状態はクライアントには適用されません。 **dadmin** および **lssrc** からの出力は、この状態を「Used」および「Bad」と報告します。

LeaseTimeStart
現在の貸出時間の開始 (1970 年 1 月 1 日からの秒数)

LeaseTimeDuration
貸出時間の期間 (秒数)

LeaseTimeEnd
貸出時間の終了。フォーマットは *LeaseTimeStart* と同じ。 構成オプションで貸出時間の開始と終了にこれとは異なる値を指定した場合は、値は構成ファイル・オプションで上書きされます。 236 ページの『db_file データベースに対する DHCP サーバー・ファイルの構文』を参照してください。

"*Server IP Address*"
このレコードを所有している DHCP サーバーの IP アドレス

"*Class ID*" "*Vendor ID*" "*Host Name*" "*Domain Name*"
サーバーに送信されるオプション (引用符付き文字列として格納されている) を判別するためにサーバーが使用する値。 これらのクライアントのために **DHCP** サーバーの始動時にオプション・リストが事前生成されるため、これらのパラメーターによりパフォーマンスが向上します。

DHCP チェックポイント・ファイル
チェックポイント・ファイルのための構文は指定しません。

サーバーがクラッシュしたり、あるいは、ユーザーがサーバーをシャットダウンしたためにデータベースを正常にクローズできない場合には、サーバーがチェックポイントとバックアップ・ファイルを処理して、有効なデータベースを構築し直すことができます。 サーバーがクラッシュした時点でチェックポイント・ファイルに書き込み中であったクライアントは失われます。 デフォルト・ファイルは、次のとおりです。

/etc/db_file.cr
正常なデータベース操作

/etc/db_file.crbk
データベースのバックアップ

/etc/db_file.chkpt および /etc/db_file.chkpt2
回転用チェックポイント・ファイル

DHCP サーバーはスレッド化されます。スループットを高く維持するためには、データベース操作 (保存操作を含む) でスレッドが有効である必要があります。 保存が要求されると、既存のチェックポイント・ファイルは次のチェックポイント・ファイルへと回転し、既存のデータベース・ファイルはバックアップ・ファイルにコピーされ、新しい保存ファイルが作成されます。 次に、各クライアント・レコードがログに記録されるとともに、トグルのビットによって、クライアントはログの記録のために新しいチェックポイント・ファイルを使用するように指示されます。 すべてのクライアント・レコードが記録されると、保存ファイルがクローズされて、バックアップ・ファイルと古いチェックポイント・ファイルは削除されます。 このようにして、クライアントは処理を続けることができます。データベースの変更は、クライアントのレコードが保存されていたかどうかによって、新しい保存ファイルまたは新しいチェックポイント・ファイルに入ります。

DHCP プロトコル・エンジン
DHCP プロトコル・エンジンは RFC 2131 をサポートし、RFC 1541 との互換性も依然として備えています。(サーバーは RFC 2132 に定義されているようにオプションを処理することもできます。) プロトコル・エンジンは、データベースを使用して、どの情報をクライアントに戻すかを判別します。

アドレス・プールの構成には、それぞれのマシンの状態に影響を与える構成オプションがいくつかあります。 例えば、**DHCP** サーバーは、アドレスを分配する前に ping で接続をテストします。 現在では、サーバーが応答を待機する時間は、アドレス・プールごとに構成することができます。

DHCP のスレッド化操作
DHCP サーバーの最後の部分は、実行を続行するために使用される一連の操作です。 **DHCP** サーバーはスレッド化されるため、これらの操作は、すべてが一緒に行われるようにするために時々実行されるスレッドとしてセットアップされます。

最初のスレッドである `main` スレッドは、SRC 要求 (`startsrc`、`stopsrc`、`lssrc`、`traceson`、`refresh` など) を処理します。 また、このスレッドは、すべてのスレッドに影響するすべての操作を調整し、シグナルを処理します。 例えば、次のとおりです。

- SIGHUP (-1) は構成ファイル内のすべてのデータベースを再表示する。
- SIGTERM (-15) はサーバーを正常中止にする。

次のスレッドである `dadmin` スレッドは、`dadmin` クライアント・プログラムと **DHCP** サーバーとのインターフェースです。 `dadmin` ツールを使用すると、手動でデータベース・ファイルを編集するのを避けるために、データベースを変更するとともに、その状況を入手することができます。 以前の **DHCP** サーバーのバージョンでは、状況要求の実行中は、どのクライアントもアドレスを入手することができませんでした。 `dadmin` と `src` のスレッドが追加されたおかげで、サーバーはサービス要求を処理する一方でクライアントの要求も処理することができるようになりました。

その次のスレッドは garbage スレッドです。これは、定期的にデータベースをクリーンアップするとともにデータベースを保存し、アドレスを持っていないクライアントを除去し、余りにも長く予約済み状態になっている予約済みアドレスを除去するためのタイマーを動かします。 これらのタイマーはすべて構成することができます (217 ページの『DHCP の構成』を参照)。 その他のスレッドはパケット・プロセッサーです。 パケット・プロセッサーの数は構成できます。デフォルトは 10 です。 いずれも、**DHCP** クライアントからの要求を処理することができます。 必要となるパケット・プロセッサー数は、ロード状況とマシンによっていくらか異なります。 マシンが **DHCP** 以外のサービスに使用される場合は、500 スレッドを始動しないようにしてください。

DHCP の計画

このプロトコルを使用するには、ネットワーク管理者は **DHCP** サーバーをセットアップし、**DHCP** サーバーがないリンク上に BOOTP リレー・エージェントを構成する必要があります。 適切に計画を立てると、ネットワークで **DHCP** のロードを減らすことができます。

例えば、あるサーバーにすべてのクライアントを 処理させるように構成できますが、それではすべてのパケットがそのサーバーを通過しなければなりません。 2 つの大きなネットワークの間にルーターが 1 つだけある場合、ネットワーク内に 2 つのサーバー (1 つのリンクに 1 つのサーバー) を配置する方が賢明です。

また、**DHCP** がトラフィックのパターンを暗黙指定することも検討してください。 例えば、デフォルトの貸出時間が 2 日未満に設定されており、マシンの電源が週末にオフにされる場合は、月曜日の朝に **DHCP** トラフィックが増大します。 **DHCP** トラフィックによりネットワークのオーバーヘッドが大量になることはありませんが、**DHCP** サーバーをネットワークのどこに配置するか、いくつ使用するかを決めるときは、そのことも考慮する必要があります。

DHCP がネットワーク上でクライアントを取得できるようになったら、クライアントは何も入力する必要はありません。 **DHCP** クライアントの dhcpcd は、dhcpcd.ini ファイルを読み取ります。このファイルには、ログ・パラメーター、および実行の開始に必要なその他のパラメーターが入っています。 インストールの後、ユーザーはどのメソッドで **TCP/IP** を構成するか、すなわち、最小構成にするか **DHCP** にするかを決めます。 **DHCP** を選択した場合は、インターフェースを 選択し、さらにオプション・パラメーターをいくつか指定します。 インターフェースを 選択するには、キーワードの any を選択します。これは、使用できる最初のインターフェースを 見つけ、それを使用するように dhcpcd に指示します。 これにより、クライアント側での入力量が最小になります。

DHCP の構成

DHCP サーバーは、デフォルトでは /etc/dhcpsd.cnf ファイルを読み取ることで構成されます。このファイルにより、オプションとアドレスの 入った初期データベースが指定されます。

サーバーは、/etc/rc.tcpip ファイル内で始動します。 SMIT からの始動や SRC コマンドでの始動も可能です。 **DHCP** クライアントは、システム 管理インターフェース・ツール (SMIT) を実行するか、フラット ASCII ファイルを編集することで構成できます。

DHCP サーバーの構成は、通常、ネットワークで **DHCP** を使用する際の最も難しい作業です。 まず、どのネットワークに **DHCP** クライアントが必要かを決めてください。 ネットワーク内の各サブネットは、**DHCP** サーバーがそのデータベースに加えなければならないアドレスのプールを表しています。 例えば、次のとおりです。

```
database db_file
{

    subnet 9.3.149.0 255.255.255.0
      {  option 3 9.3.149.1 # The default gateway clients on this network should use
         option 6 9.3.149.2 # The nameserver clients on this network should use
      }
    ... options or other containers added later
}
```

上記の例は、サブネット・マスク 255.255.255.0 を付けたサブネット 9.3.149.0 を示しています。 このサブネット内のすべてのアドレス、9.3.149.1 から 9.3.149.254 がプールに入ります。 オプションとして、行の終わりに範囲を指定することができ、また、範囲または除外ステートメントをサブネット・コンテナーに含めることができます。 共通の構成方法および定義については、225 ページの『DHCP サーバー・ファイルの既知のオプション』を参照してください。

db_file の付いた database 文節は、この構成ファイル部分を処理するために使用するデータベースのメソッドを指示します。 コメントは # (ポンド記号) で始まります。 **DHCP** サーバーは、始めの # からその行の終わりまでのテキストを無視します。 サーバーは、各 option 行を使用して、クライアントに行うべきことを指示します。 現在サポートされ、認識されるオプションについては、225 ページの『DHCP サーバー・ファイルの既知のオプション』に説明があります。 サーバーに既知ではないオプションを指定する方法については、231 ページの『汎用サーバー操作に対する DHCP サーバー・ファイルの構文』を参照してください。

サーバーは、オプションの構文解析方法が分からない場合は、デフォルトのメソッドを使用して、クライアントにそのオプションを送信します。 これにより、**DHCP** サーバーは、RFC 定義ではないがある種のクライアントまたはクライアント 構成で使用できる、サイト固有のオプションを送信することもできます。

DHCP 構成ファイル

構成ファイルには、アドレス・セクションとオプション定義セクションがあります。 これらのセクションは、コンテナーを使用して、オプション、修飾子、場合によっては他のコンテナーを保持します。

コンテナー (基本的には、オプションをグループ化するための方法) は、クライアントをグループに 分類するのに ID を使用します。 コンテナーには 4 つのタイプあります。サブネット、クラス、ベンダー、およびクライアントです。 現在、一般的なユーザー定義可能コンテナーはありません。 ID は、例えばクライアントがサブネット間を移動するときにクライアントを追跡できるように、クライアントを固有に定義するものです。 クライアント・アクセスを定義するために、複数のコンテナー・タイプを使用することができます。

オプション は、デフォルトのゲートウェイや DNS アドレスなど、クライアントに戻される ID です。

修飾子 は、デフォルトの貸出時間など、コンテナーのある特性を変更する単一ステートメントです。

DHCP コンテナー

DHCP サーバーが要求を受信すると、パケットが構文解析され、抽出するコンテナー、オプション、およびアドレスが ID キーによって判別されます。

217 ページの『DHCP の構成』の例は、サブネット・コンテナーを示しています。 この ID キーは、ネットワーク内のクライアントの位置です。 クライアントがそのネットワークからのものである場合は、そのクライアントはそのコンテナーに入ります。

コンテナーは、タイプごとに、クライアントを識別するための異なるオプションを使用します。

- サブネット・コンテナーは、**giaddr** フィールドまたは受信インターフェースのインターフェース ・アドレスを使用して、クライアントがどのサブネットから発生したものであるかを判別する。
- クラス・コンテナーは、オプション 77 の値 (ユーザー・サイト・クラス ID) を使用する。
- ベンダー・コンテナーは、オプション 60 の値 (ベンダー・クラス ID) を使用する。
- クライアント・コンテナーは、**DHCP** クライアントに対してオプション 61 (クライアント ID)、**BOOTP** クライアントに対して **BOOTP** パケットの **chaddr** フィールドを使用する。

サブネット・コンテナーを除いて、どのコンテナーについても、(正規表現マッチングを 含む) マッチングする値を指定することができます。

暗黙のコンテナーもあります。 グローバル・コンテナーがそれです。 オプションと修飾子は、オーバーライドされたり拒否されたりしない限りグローバル・コンテナーに入れられます。 ほとんどのコンテナーは、可視範囲を暗黙指定して他のコンテナーの中に入れることができます。 コンテナーは、それに関連するアドレス範囲を持つこともできますし、持たなくてもかまいません。 サブネットは、その性質上それに関連した範囲があります。

コンテナーとサブコンテナーの基本規則は次のとおりです。

- グローバル・レベルではどのコンテナーも有効である。
- サブネットは、他のコンテナーの中に入れることはできない。
- 制限付きコンテナーの中には、同じタイプのレギュラー・コンテナーを入れることはできない。 (例えば、Accounting クラスだけが許されたオプションのコンテナーに、文字「a」で始まるすべてのクラスが許されたオプションのコンテナーを入れることはできません。 これは違反です。)
- 制限付きクライアント・コンテナーはサブコンテナーを持てない。

上記の規則に従えば、コンテナーの階層を作成することができます。この階層によって、オプションは、特定のクライアントまたは 1 組のクライアントのグループに 分けられます。

クライアントが複数のコンテナーに合致する場合は、オプションとアドレスはどのように分配されるのでしょうか。 **DHCP** サーバーは、メッセージを受信し、要求をデータベース (この場合は db_file) に渡します。そこでコンテナー・リストが作成されます。 このリストは、奥行きと優先順位の順に並べられます。優先順位は、コンテナー内の暗黙の階層として定義されます。 ストリクト・コンテナーは、レギュラー・コンテナーよりも高い優先順位を持ちます。 クライアント、クラス、ベンダー、最後にサブネットがこの順に、また、それぞれのコンテナー・タイプ内では奥行きの順に記載されます。 すなわち、特定度の最も高いものから特定度の低いものの順に並べられたリストができあがります。 例えば、次のとおりです。

```
Subnet 1
--Class 1
--Client 1
Subnet 2
--Class 1
----Vendor 1
----Client 1
--Client 1
```

この例では、2 つのサブネット Subnet 1 と Subnet 2 が示されています。 クラス名は 1 つで Class 1、ベンダー名も 1 つで Vendor 1、そしてクライアント名が Client 1 の 1 つです。Class 1 と Client 1 は複数の場所で定義されています。これらは異なるコンテナーに入っているので、名前は同じですがその中の値は異なることがあります。 Client 1 がオプション・リストに Class 1 を指定している Subnet 1 からメッセージを **DHCP** サーバーに送ると、**DHCP** サーバーは次のようなコンテナー・パスを生成します。

```
Subnet 1, Class 1, Client 1
```

最も特定度の高いコンテナーが最後にリストされています。 アドレスを取得するには、このリストを階層の逆向きに調べ、最初に使用可能なアドレスを探します。 次に、このリストを階層の前方向に調べ、オプションを入手します。 オプションは、**deny** がコンテナーに指定されていなければ、以前の値を指定変更します。 また、Subnet 1 には Class 1 と Client 1 が入っているため、これらはコンテナーの優先順位に従って並べられます。 同じクライアントが Subnet 2 に入っていて、同じメッセージを送信するのであれば、コンテナー・リストは次のように生成されます。

Subnet 2、Class 1、Client 1 (Subnet 2 レベル)、Client 1 (Class 1 レベル)

Subnet 2 が最初にリストされ、次に Class 1、最後に Subnet 2 レベルの Client 1 の順にリストされます (このクライアント・ステートメントが階層では 1 レベル下にあるためです)。 この階層は、最初のクライアント・ステートメントに合致するクライアントが、Subnet 2 内の Class 1 の Client 1 に合致するクライアントよりも特定度が低いことを示唆しています。

階層内の奥行きによって選択された優先順位が、コンテナーそれ自体の優先順位で取り替えられることはありません。 例えば、同一のクライアントが同じメッセージを出し、ベンダー ID を指定している場合、そのコンテナー・リストは次のようになります。

Subnet 2、Class 1、Vendor 1、Client 1 (Subnet 2 レベル)、Client 1 (Class 1 レベル)

コンテナー優先順位は、クライアント・コンテナーが 1 つ以上のクライアントを定義するための最も特定度の高いコンテナー優先順位であるとする一般的な概念に従っているため、検索パフォーマンスを向上させます。 クラス・コンテナーは、クライアント・コンテナーよりも特定度が低いアドレスを保持しています。ベンダーはさらに特定度が低く、サブネットは特定度が最も低くなっています。

DHCP アドレスとアドレス範囲

どのコンテナー・タイプも関連したアドレス範囲を持つことができます。サブネットにはアドレス範囲が必須です。 コンテナー内の各範囲は、その範囲のサブセットにする必要があり、また、他のコンテナーの範囲とオーバーラップすることはできません。

例えば、あるクラスがサブネット内で定義されており、そのクラスが範囲を持っている場合、その範囲は当該サブネットの範囲のサブセットである必要があります。 また、そのクラス・コンテナー内の範囲は、そのレベルにあるその他の範囲とオーバーラップすることはできません。

範囲は、コンテナーの行で表現し、範囲および除外ステートメントで変更して、あるコンテナーに関連付けられたばらばらのアドレス・セットを使用可能にすることができます。 したがって、サブネットの最初の 10 アドレスと次の 10 アドレスを使用可能にする場合、そのサブネットがサブネット文節でこれらのアドレス範囲を指定することによって、使用するメモリーを削減できると同時に、指定された範囲にないその他のクライアントとのアドレス競合が起きるのを避けることができます。

アドレスを選択すると、アドレス範囲を含んでいるリスト内の後続のコンテナーは、その子とともにリストから除去されます。 アドレスがそのコンテナー内から使用されなかった場合、除去されたコンテナー内のネットワーク固有のオプションは無効です。

DHCP 構成ファイル・オプション
アドレスを判別するためにリストが選別された後、1 組のオプションがクライアントのために 生成されます。

この選択プロセスでは、*deny* が検出されなければ、以前選択されたオプションが上書きされます。検出された場合は、否定されたオプションがクライアントに 送られるリストから除去されます。 このメソッドによって、親コンテナーからの継承で、指定しなければならないデータ量が削減されます。

DHCP 修飾子
修飾子は、アクセス時間または貸出時間など、特定のコンテナーのある種の特性を変更するものです。

コンテナーを変更する前に、アドレスとオプション・プールを定義してください。 最も一般的な修飾子は、`leasetimedefault`、`supportBootp`、および `supportUnlistedclients` です。

leasetimedefault
アドレスをクライアントに貸し出す時間を定義する。

supportBootp
サーバーが **BOOTP** クライアントに応答するかどうかを定義する。

supportUnlistedclients
アドレスを受信するために、クライアント・ステートメントで明示的にクライアントを定義するかどうかを指示する。 `supportUnlistedClients` の値は、`none`、`dhcp`、`bootp`、または `both` のいずれかです。 これを使用すると、アクセスを bootp クライアントに限定したり、すべての DHCP クライアントにアドレス取得を許可したりすることができます。

その他の修飾子は、236 ページの『db_file データベースに対する DHCP サーバー・ファイルの構文』にリストされています。

DHCP のロギング
修飾子を選択した後、次にセットアップするのはロギングのための 項目です。

ロギング・パラメーターは、データベースのようなコンテナーに指定しますが、コンテナーの場合のキーワードは `logging_info` です。 **DHCP** の構成方法を学んでいる間は、ロギングのレベルを最高レベルに設定しておくことをお勧めします。 また、ロギング・サブシステムの 初期化後に構成エラーが確実に記録されるようにするために、他のどの構成ファイル・データよりも先にロギング構成を指定しておくのが最善です。 `logitem` キーワードは、あるロギング・レベルをオンにするために使用します。ロギング・レベルを使用不可にするには `logitem` キーワードを除去してください。 ロギングのためのその他のキーワードを使用すると、ログのためのファイル名、ファイルのサイズ、回転ログ・ファイル数を指定することができます。

DHCP サーバー固有オプション
指定する最後のパラメーターのセットは、サーバー固有オプションです。これは、パケット・プロセッサー数を制御したり、不要情報コレクション・スレッドを実行する頻度を決めたりするためのものです。

例えば、次の 2 つのサーバー固有オプションがあります。

reservedTime
OFFER を **DHCP** クライアントに送信した後、アドレスを予約済み状態に留めておく期間を指示する。

reservedTimeInterval
reservedTime よりも長く予約済み状態に置かれているアドレスがないかどうか、**DHCP** サーバーがアドレスをスキャンする頻度を指示する。

これらのオプションは、DISCOVER メッセージをブロードキャストするクライアントがいくつかあり、それらがその REQUEST メッセージをブロードキャストしないか、またはその REQUEST メッセージがネットワーク内で失われた場合に、有益です。 これらのパラメーターを使用すれば、非準拠クライアントのためにアドレスが無期限に予約されることはありません。

もう 1 つの特に有益なオプションとして、**SaveInterval** があります。これは、保存を行う頻度を指示します。 231 ページの『汎用サーバー操作に対する DHCP サーバー・ファイルの構文』 には、すべてのサーバー固有オプションが、ロギングのためのキーワードとともに、リストされています。

DHCP パフォーマンスの考慮事項

特定の構成キーワードと構成ファイルの構造が **DHCP** サーバーのメモリー使用量とパフォーマンスに影響を与えるということを理解するのは重要です。

まずはじめに、親コンテナーから子コンテナーへと継承されるオプションの継承モデルを理解することにより、過度のメモリー使用を回避することができます。 リストされていないクライアントは一切サポートされない環境では、管理者は当該のファイルに各クライアントを明示的にリストする必要があります。 どのクライアントについても 個々にオプションがリストされている場合、サーバーは、その構成ツリーを保管するために、オプションが親コンテナー (例えば、サブネット・コンテナー、ネットワーク・コンテナー、またはグローバル・コンテナーなど) から継承される場合よりも多くのメモリーを使用します。 したがって、管理者は、構成ファイル内においてクライアント・レベルで繰り返し現れるオプションがあるかどうかを検査し、それらのオプションをひとまとめにして親コンテナーの中に指定して一群のクライアントで共用できるかどうかを判断する必要があります。

また、**logItem** エントリー INFO と TRACE を使用していると、**DHCP** クライアントのメッセージを 1 つ処理するたびに大量のメッセージがログに記録されます。 ログ・ファイルに 1 行を追加する操作は消費が多くなる可能性があります。したがって、ロギングの量を制限すると、**DHCP** サーバーのパフォーマンスが向上します。 **DHCP** サーバーにエラーが発生している疑いがある場合は、SRC の **traceson** コマンドまたは **dadmin** コマンドを使用して、ロギングを動的に再度使用可能にすることができます。

最後に、**numprocessors** 値の選択は、**DHCP** がサポートするネットワークのサイズ、**pingTime db_file** 構成パラメーター、およびネットワーク上の通常の伝搬遅延に応じて行います。 各パケット・プロセッサー・スレッドは、サーバー所有のアドレスに関するオファリングをクライアントに 対して行う前に、そのアドレスの状況を検査するために ICMP エコー要求を発行します。そのため、エコー応答が戻されるまでの待ち時間が DISCOVER メッセージの処理時間に直接影響します。 つまり、パケット・プロセッサー・スレッドは、いずれかのサーバーから応答が戻されるまで、あるいは **pingTime** がタイムアウトになるまで、待つ以外には何もできないということです。 したがって、**numprocessors** 値を小さくすれば、クライアントによる再送の回数が減少することでサーバーの応答時間が向上し、しかも ping を使用することによって得られるサーバー設計上の利点はそのまま生かされます。

最高のパフォーマンスを実現するために、**pingTime** の選択は、当該 **DHCP** サーバーがサポートしているすべてのリモート・ネットワークの伝搬遅延に基づいて行ってください。 また、**numprocessors** 値の選択は、この **pingTime** 値およびネットワークのサイズに基づいて行ってください。 選択した値が あまりにも小さいと、すべてのパケット処理スレッドが停止してしまうことがあります。 その後、サーバーはエコー応答を待ち、着信の **DHCP** クライアント・メッセージがサーバーのポート上でキューイングされることになります。 これによって、サーバーは、クライアント・メッセージを絶え間ないストリームとしてではなく、バッチで処理するようになります。

選択した値があまりにも小さいと、エコー応答を待っている状態ですべてのパケット処理スレッドが停止してしまいます。

このような状態にならないようにするために、**numprocessors** の値は、**DHCP** クライアントのアクティビティーが活発なときに 1 **pingTime** インターバル内で受信される可能性のある DISCOVER メッセージの予想数よりも大きい数に設定します。 ただし、スレッド管理のためにカーネルに高い負荷がかかるほど大きな値を **numprocessors** に設定してはなりません。

例えば、1 秒当たり 10 個の DISCOVER メッセージが発生する可能性のある環境で値 **numprocessors 5** および **pingTime 300** を設定すると、要求のピーク時にはメッセージが 3 秒ごとに 5 つずつしか処理されないために、パフォーマンスが低下してしまいます。 このような環境は、**numprocessors 20** および **pingTime 80** のような値を使用して構成します。

DHCP 構成ファイルのカスタマイズ

DHCP 構成ファイルをカスタマイズする場合は、さまざまな要因が関係します。

多くのネットワークには、複数のクライアント・タイプが含まれます。例えば、 単一ネットワークに、Windows、OS/2、Java™ OS、 および UNIX など のさまざまなオペレーティング・システムが稼働しているコンピューターが含まれていることがあります。 これらは、それぞれが固有のベンダー ID (マシンのタイプを DHCP サーバーに明らかにするために使用されるフィールド) を必要とします。 Java OS クライアントと IBM シン・クライアント・マシンは、bootfile などの固有のパラメーターを必要とすることがあり、また、構成オプションをそれらのために特別に調整する必要がある場合があります。 Windows 95 コンピューターは、Java 固有のオプションは処理しません。

ある種のマシンの基本的な使用がそれらのマシンのユーザー・タイプに基づいている場合、マシン固有のオプションをベンダー・コンテナー内にカプセル化することができます。 例えば、開発担当部員はプログラミングのためにこのオペレーティング・システムのクライアントを 使用し、販売企画員は OS/2 クライアントを使用し、営業担当部員は Java OS クライアントと IBM シン・クライアント・マシンを、さらに、会計担当者は Windows 95 マシンを使用していることがあります。 これらのユーザーは、それぞれ異なる構成オプション (種々のプリンター、ネーム・サーバー、デフォルトの Web サーバーなど) を必要とする場合があります。 この場合、それぞれのグループが異なるマシン・タイプを使用するため、このようなオプションはベンダー・コンテナーに入れることができます。

しかし、複数のグループが同じマシン・タイプを使用する場合は、従属クラス ID にオプションを 入れることによって、例えば、他の部門の人がアクセスできない特定のプリンターのセットを販売企画マネージャーが使用できるようにすることができます。

注 : 次の例は、構成ファイルの一部を示すためのものです。 ポンド記号 (#) を前に付けたコメントが、その行の設定理由を説明しています。

```
vendor "AIX_CLIENT"
{
# No specific options, handles things based on class
}

vendor "OS/2 Client"
{
# No specific options, handles things based on class
}

vendor "Windows 95"
{ option 44 9.3.150.3          # Default NetBIOS Nameserver
}

vendor "Java OS"
{ bootstrapserver 9.3.150.4    # Default TFTP server for the Java OS boxes
  option 67 "javaos.bin"       # The bootfile of the Java OS box
}

vendor "IBM Thin Client"
{ bootstrapserver 9.3.150.5    # Default TFTP server for Thin Client boxes
  option 67 "thinos.bin"       # Default bootfile for the Thin Client boxes
}

subnet 9.3.149.0 255.255.255.0
{ option 3 9.3.149.1           # The default gateway for the subnet
  option 6 9.3.150.2           # This is the nameserver for the subnet
  class accounting 9.3.149.5-9.3.149.20
  {          # The accounting class is limited to address range 9.3.149.5-9.3.149.20
          # The printer for this group is also in this range, so it is excluded.
     exclude 9.3.149.15
     option 9 9.3.149.15       # The LPR server (print server)
     vendor "Windows 95"
     {
     option 9 deny              # This installation of Windows 95 does not support
                                # this printer, so the option is denied.
     }
  }
  . . .
}
```

DHCP と動的ドメイン・ネーム・システム

DHCP サーバーは、動的ドメイン・ネーム・システム (DDNS) 環境で操作を行うためのオプションを提供しています。

DDNS 環境で **DHCP** を使用するには、DNS サーバーで動的ゾーンを設定して使用する必要があります。

DDNS サーバーが構成できたら、**DHCP** サーバーに A レコード更新、PTR レコード更新、または両レコード・タイプの更新、または更新なし、のいずれかを決定してください。 これは、クライアント・マシンがこの作業の一部または全部を行えるどうかによって決まります。

- クライアントが更新を一緒に担当できる場合は、PTR レコード更新を行うようにサーバーを構成し、A レコード更新を行うようにクライアントを構成する。
- クライアントが両方の更新を行える場合は、サーバーには何もさせないように構成する。
- クライアントが更新を行えない場合は、サーバーに両方を行わせるように構成する。

DHCP サーバーには、更新が必要なときに実行するコマンドをユーザーが指定できる 1 組の構成キーワードがあります。 これらのキーワードは、次のものです。

updatedns
(使用すべきではありません。) コマンドがどのタイプの更新も行えることを示す。 これは、PTR レコードと A レコードの両方の更新に対して呼び出されます。

updatednsA
コマンドが A レコードを更新することを指定する。

updatednsP
コマンドが PTR レコードを更新することを指定する。

これらのキーワードは、更新が必要なときに **DHCP** サーバーが実行する実行可能文字列を指定します。 キーワード文字列には、4 つの %s (パーセント記号と文字 s) が入っている必要があります。 最初の %s はホスト名、2 番目はドメイン名、3 番目は IP アドレス、4 番目は貸出時間です。 これらは、**dhcpaction** コマンドの最初の 4 つのパラメーターとして使用されます。 **dhcpaction** コマンドのための残り 2 つのパラメーターは、更新するレコード (A、PTR、NONE または BOTH) と、NIM を更新すべきかどうか (NIM または NONIM) を指定します。 NIM と **DHCP** の相互作用の詳細については、306 ページの『DHCP とネットワーク・インストール管理における推奨事項』を参照してください。 例えば、次のとおりです。

```
updatednsA "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' '%s' A NONIM"
                    # This does the dhcpaction command only on the A record
updatednsP "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' '%s' PTR NONIM"
                    # This does the command only on the PTR record
updatedns "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' '%s' BOTH NIM"
                    # This does the command on both records and updates NIM
```

DHCP サーバーには、貸出時間が解除されたか、または貸出時間が経過したときに DNS エントリーを除去するための一連のキーワードもあります。 これらのキーワードは次のものです。

releasednsA
A レコードを除去する。

releasednsP
PTR レコードを除去する。

removedns
両方のレコード・タイプを除去する。

これらのキーワードは、アドレスが解放されたか、または有効期限切れのときに **DHCP** サーバーが実行する実行可能文字列を指定します。 **dhcpremove** コマンドの動作は、**dhcpaction** コマンドの動作と似ていますが、指定するパラメーターは 3 つだけです。

1. IP アドレス。これはコマンド文字列の中で %s として指定されます。
2. 除去すべきレコード (A、PTR、NONE または BOTH)
3. NIM を更新すべきかどうかの別 (NIM または NONIM)

例:

```
releasednsA "/usr/sbin/dhcpremove '%s' A NONIM"
                    # This does the dhcpremove command only the A record
releasednsP "/usr/sbin/dhcpremove '%s' PTR NONIM"
                    # This does the command only on the PTR record
removedns "/usr/sbin/dhcpremove '%s' BOTH NIM"
                    # This does the command on both records and updates NIM
```

dhcpaction と **dhcpremove** のスクリプトは、パラメーター検査を行ってから、**nsupdate** へのコールをセットアップします。このコマンドは、このオペレーティング・システムのサーバーと OS/2 DDNS サーバーで機能するように更新されています。 詳しくは、**nsupdate** コマンドの説明を参照してください。

名前の更新に NIM 対話が必要でない場合は、DHCP デーモンと **nsupdate** コマンドとの間でソケット転送を使用するように DHCP サーバーを構成して、パフォーマンスを向上させ、障害発生時における DNS 更新の再試行を可能にすることができます。 このオプションを構成するには、キーワード updateDNSA、updateDNSP、releaseDNSA、または releaseDNSP で「nsupdate_daemon」を最初の引用ワードとして指定する必要があります。 この更新用のパラメーターおよびフラグは、**nsupdate** コマンドが受け入れるパラメーターおよびフラグと同一です。 さらに、置換のために以下の変数名を使用することもできます。

項目	説明
$hostname	DNS 更新時にはそのクライアントのホスト名によって置き換えられ、DNS からの除去の場合は、以前にそのクライアントに関連付けられていたホスト名によって置き換えられます。
$domain	更新の場合は、その DNS ドメインによって置き換えられ、DNS からの除去の場合は、そのクライアントのホスト名が以前に使用されていたドメインによって置き換えられます。
$ipadress	その **DHCP** クライアント名と関連付けられるか、あるいはそのクライアント名との関連付けを解除される IP アドレスによって置き換えられます。
$leasetime	貸出時間 (秒数) によって置き換えられます。
$clientid	その **DHCP** クライアント ID の文字列表示によって置き換えられます。また、**BOOTP** クライアントの場合は、組み合わせハードウェア・タイプとハードウェア・アドレスによって置き換えられます。

例えば、次のとおりです。

```
updateDNSA "nsupdate_daemon -p 9.3.149.2 -h $hostname -d $domain
 -s"d;a;*;a;a;$ipaddress;s;$leasetime;3110400""

updateDNSP "nsupdate_daemon -p 9.3.149.2 -r $ipaddress
 -s"d;ptr;*;a;ptr;$hostname.$domain.;s;$leasetime;3110400""

releaseDNSA "nsupdate_daemon -p 9.3.149.2 -h $hostname -d $domain -s"d;a;*;s;1;3110400""

releaseDNSP "nsupdate_daemon -p 9.3.149.2 -r $ipaddress -s"d;ptr;*;s;1;3110400""
```

詳しくは、**nsupdate** コマンドの説明を参照してください。

また、管理者定義によるポリシーが、サーバーとクライアントの間でのホスト名交換のために追加されています。 デフォルトでは、クライアントに戻されて DDNS 更新に使用されるホスト名は、オプション 12 (サーバー構成ファイルに定義されている) です。 あるいは、オプション 81 (DHCPDDNS オプション) またはオプション 12 (HOSTNAME オプション) によって、デフォルトのホスト名をクライアント提案のホスト名とすることもできます。 ただし、管理者は、hostnamepolicy、proxyarec および appenddomain 構成キーワードを使用することによって、これらのデフォルト・ホスト名をオーバーライドすることができます。 これらのオプションおよびそのパラメーターは、236 ページの『db_file データベースに対する DHCP サーバー・ファイルの構文』で定義されています。

DHCP の旧バージョンとの互換性

DHCP サーバーは、旧バージョンの構成ファイルとデータベース・ファイル (dhcps.ar と dhcps.cr) を認識します。

以前の構成ファイルは構文解析され、新しいデータベース・ファイルがもとのロケーションに生成されます。 以前のデータベースは、自動的に新しいファイルに変換されます。 構成ファイルそのものは変換されません。

DHCP サーバー・データベース・モジュール db_file は、以前のフォーマットを読み取ることができます。 **DHCP** サーバーは、データベース・コンテナーが構成ファイル内に存在しないときにはそれを認識し、サーバー・パラメーター、ロギング・パラメーター、および db_file データベース・パラメーターを構成するときにファイル全体を処理することができます。

注 :

1. 以前の構成ファイルの構文の中には、使用すべきではないものがいくつかありますが、それもサポートされます。 その他には、次のものは使用すべきではありません。
2. ネットワーク・コンテナーはいずれも使用すべきでない。 正しく指定するには、範囲を持つ network 文節は、サブネット・アドレス、サブネット・ネットマスク、および範囲を持つ有効なサブネット・コンテナーに変換します。 ネットワーク・コンテナーにサブネット・コンテナーがある場合は、network コンテナー・キーワードとその中括弧を除去し、サブネット・マスクをその行の適切な場所に入れます。 データベース・コンテナーの使用を開始するには、ネットワークとクライアント・アクセスに関係するすべてをタイプ db_file の 1 つのデータベース・コンテナーに入れます。
3. updatedns と removedns のキーワードは使用すべきでなく、代わりに、それぞれ A レコードと PTR レコードのためのアクションを指定する。
4. clientrecorddb と addressrecorddb キーワードは、使用すべきでなく、clientrecorddb と backupfile にそれぞれ置換される。
5. option sa と option ga キーワードは、bootstrapserver と giaddrfield キーワードにそれぞれ置換されている。 詳しくは、231 ページの『汎用サーバー操作に対する DHCP サーバー・ファイルの構文』 および 236 ページの『db_file データベースに対する DHCP サーバー・ファイルの構文』 を参照してください。

DHCP サーバー・ファイルの既知のオプション

ここでは、**DHCP** サーバー・ファイルの既知のオプションを示します。

注 : 以下の表において指定できないもの (「指定可能?」列の値が「いいえ」) として示されているオプションは、構成ファイルに指定できますが、適切な値で上書きされます。 各オプションをうまく定義するためには、RFC 2132 を参照してください。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明/使用
0	なし	いいえ	サーバーは、必要に応じてオプション・フィールドを埋め込む。
1	ドット付きクワッド	いいえ	アドレスを取り出した元のサブネットのネットマスク。
2	32 ビットの整数	はい	クライアントのサブネットのオフセットを、協定世界時間 (UTC) からの秒数で指定する。
3	1 つ以上のドット付きクワッド	はい	デフォルト・ゲートウェイの IP アドレスのリスト。
4	1 つ以上のドット付きクワッド	はい	タイム・サーバーの IP アドレスのリスト。
5	1 つ以上のドット付きクワッド	はい	ネーム・サーバーの IP アドレスのリスト。
6	1 つ以上のドット付きクワッド	はい	DNS IP アドレスのリスト。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明/使用
7	1 つ以上のドット付きクワッド	はい	ログ・サーバーの IP アドレスのリスト。
8	1 つ以上のドット付きクワッド	はい	Cookie サーバーの IP アドレスのリスト。
9	1 つ以上のドット付きクワッド	はい	LPR サーバーの IP アドレスのリスト。
10	1 つ以上のドット付きクワッド	はい	Impress サーバーの IP アドレスのリスト。
11	1 つ以上のドット付きクワッド	はい	リソース・ロケーション・サーバーの IP アドレスのリスト。
12	ASCII 文字列	はい	使用するクライアントのホスト名。
13	16 ビットの無符号整数	はい	ブート・ファイルのサイズ。
14	ASCII 文字列	はい	メリット・ダンプ・ファイルのためのパス。
15	ASCII 文字列	はい	デフォルトの DNS ドメイン名。
16	IP アドレス	はい	スワップ・サーバーのアドレス。
17	ASCII 文字列	はい	デフォルトのルート・パス。
18	ASCII 文字列	はい	クライアントのための拡張機能へのパス。
19	Yes、No、True、False、1、0	はい	IP 転送をオンにするかどうかを指定する。
20	Yes、No、True、False、1、0	はい	非ローカルのソースの経路指定を使用するかどうかを指定する。
21	1 つ以上のペアから成るドット付きクワッド。フォーマットは *DottedQuad* :*DottedQuad*	はい	IP アドレスのフィルター・ポリシー。
22	16 ビットの無符号整数	はい	データグラム・フラグメントを可能にする最大サイズ。
23	8 ビットの無符号整数	はい	IP 存続時間 (TTL)。
24	32 ビットの無符号整数	はい	パス MTU エージング・タイムアウトで使用する秒数。
25	1 つ以上の 16 ビット無符号整数のリスト	はい	パス MTU Plateau テーブル。 パス MTU ディスカバリーの使用時に使われる MTU サイズを表す 1 組の値を指定する。
26	16 ビットの無符号整数	はい	受信インターフェースのための MTU サイズを指定する。
27	Yes、No、True、False、1、0	はい	すべてのサブネットがローカルかどうかを指定する。
28	IP アドレス (ドット付きクワッド)	はい	インターフェースのためのブロードキャスト・アドレスを指定する。
29	Yes、No、True、False、1、0	はい	ICMP ネットマスク・ディスカバリーを使用するかどうかを指定する。
30	Yes、No、True、False、1、0	はい	クライアントが ICMP ネットマスクを提供するべきかどうかを指定する。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明/使用
31	Yes、No、True、False、1、0	はい	ICMP Router Discovery メッセージを使用するかどうかを指定する。
32	IP アドレス (ドット付きクワッド)	はい	ルーター送信請求に使用するアドレスを指定する。
33	1 つ以上の IP アドレスのペア。フォーマットは *DottedQuad :DottedQuad*	はい	各アドレス・ペアが静的経路指定を表す。
34	Yes/No、True/False、1/0	はい	トレーラーのカプセル化を行うかどうかを指定する。
35	32 ビットの無符号整数	はい	ARP キャッシュ・タイムアウト値。
36	Yes/No、True/False、1/0	はい	イーサネット・カプセル化を使用するかどうかを指定する。
37	8 ビットの無符号整数	はい	TCP 存続時間 (TTL)。
38	32 ビットの無符号整数	はい	**TCP** キープアライブ・インターバル。
39	Yes/No、True/False、1/0	はい	TCP キープアライブを使用するかどうかを指定する。
40	ASCII 文字列	はい	デフォルトの NIS ドメイン。
41	1 つ以上のドット付きクワッド	はい	NIS サーバーの IP アドレスを指定する。
42	1 つ以上のドット付きクワッド	はい	NTP サーバーの IP アドレスを指定する。
43	16 進の文字列 (hex "*digits*", hex "*digits*"、または 0x*digits* のフォーマットで)	はい。ただし、ベンダー・コンテナーにだけ指定可能	ベンダー・コンテナーのためのカプセル化されたオプション・コンテナー。
44	1 つ以上のドット付きクワッド	はい	NetBIOS ネーム・サーバーの IP アドレスを指定する。
45	1 つ以上のドット付きクワッド	はい	NetBIOS データグラム配布サーバーの IP アドレスを指定する。
46	8 ビットの無符号整数	はい	NetBIOS ノード・タイプを指定する。
47	16 進数列 (hex "*digits*", hex "*digits*"、または 0x*digits* のフォーマットで)	はい	NetBIOS 有効範囲。
48	1 つ以上のドット付きクワッド	はい	X Windows フォント・サーバーの IP アドレスを指定する。
49	1 つ以上のドット付きクワッド	はい	Windows Display Manager を指定する。
50	なし	いいえ	要求された IP アドレス (必要なアドレスを指示するためにクライアントが使用)。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明/使用
51	32 ビットの無符号整数	はい	リターン・アドレスのための貸出時間。 デフォルトでは、**DHCP** サーバーは `leasetimedefault` キーワードを使用するが、直接指示オプション 51 はそれを指定変更する。
52	なし	いいえ	オプション多重定義。 クライアントは、**BOOTP** パケットの **sname** と **file** フィールドにオプションを指定できることを指示するためにこれを使用する。
53	なし	いいえ	**DHCP** サーバーまたはクライアントはこのオプションを使用して、**DHCP** メッセージのタイプを指示する。
54	なし	いいえ	**DHCP** サーバーまたはクライアントはこのオプションを使用して、メッセージの宛先とするサーバーのアドレスまたはサーバーを指示する。
55	なし	いいえ	**DHCP** クライアントはこれを 使用して、必要なオプションを指示する。
56	ASCII 文字列	はい	**DHCP** サーバーがクライアントに送信する文字列。 一般的に、これは問題を指示するために **DHCP** サーバーとクライアントが 使用できる。
57	いいえ	いいえ	**DHCP** クライアントはこのオプションを 使用して、クライアントが受信できる最大 **DHCP** パケット・サイズを **DHCP** サーバーに伝える。
58	32 ビットの無符号整数	はい	更新パケット送信までにクライアントに許される時間 (秒数) を指定する。
59	32 ビットの無符号整数	はい	再バインド・パケット送信までにクライアントに許される時間 (秒数) を指定する。
60	なし	いいえ	**DHCP** クライアントはこのオプションを 使用して、そのベンダー・タイプを指示する。 **DHCP** サーバーはこのフィールドを使用して、ベンダー・コンテナーとのマッチングを行う。
61	なし	いいえ	**DHCP** クライアントはこれを使用して、クライアントを固有のものにする。 **DHCP** サーバーはこのフィールドを使用して、クライアント・コンテナーとのマッチングを行う。
66	ASCII 文字列	はい	**TFTP** サーバー名を指定する。 これはホスト名であり、クライアントがこのオプションを 識別する場合は、**siaddr** フィールドの代わりとして使用される。
67	ASCII 文字列	はい	ブート・ファイル名を指定する。 これは `bootfile` キーワードの代わりに使用できる。これによって、ファイルはパケットの `filename` フィールドに置かれる。
68	1 つ以上のドット付きクワッド、またはなし	はい	ホーム・エージェントのアドレスを指定する。
69	1 つ以上のドット付きクワッド	はい	使用するデフォルトの SMTP サーバーを指定する。
70	1 つ以上のドット付きクワッド	はい	使用するデフォルトの POP3 サーバーを指定する。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明/使用
71	1 つ以上のドット付きクワッド	はい	使用するデフォルトの NNTP サーバーを指定する。
72	1 つ以上のドット付きクワッド	はい	使用するデフォルトの WWW サーバーを指定する。
73	1 つ以上のドット付きクワッド	はい	使用するデフォルトの Finger サーバーを指定する。
74	1 つ以上のドット付きクワッド	はい	使用するデフォルトの IRC サーバーを指定する。
75	1 つ以上のドット付きクワッド	はい	使用するデフォルトの Street Talk サーバーを指定する。
76	1 つ以上のドット付きクワッド	はい	使用するデフォルトの Street Talk ディレクトリー援助サーバーを指定する。
77	ASCII 文字列	はい	ユーザー・サイト・クラス ID。 **DHCP** サーバーはこのフィールドを使用して、クラス・コンテナーとのマッチングを行う。
78	必須バイト、1 つ以上のドット付きクワッド	はい	SLP ディレクトリー・エージェント・オプションは、ディレクトリー・エージェントの IP アドレスのリストを指定する。
79	必須バイト、ASCII 文字列	はい	ASCII 文字列は、コンマで区切られた有効範囲リストで、それを使用して SLP エージェントが構成される有効範囲を示す。
81	ASCII 文字列にその他の項目を加えたもの	いいえ	DHCP クライアントはこのオプションを 使用して、**DHCP** サーバーが DDNS に対して使用すべきポリシーを定義する。
85	1 つ以上のドット付きクワッド	はい	NDS サーバー・オプションは、NDS データベースにアクセスするために接続する、クライアントの 1 つ以上のサーバーを指定する。 サーバーは優先順位に従ってリストしなければなりません。
86	ASCII 文字列	はい	NDS ツリー名オプションは、クライアントが接続する NDS ツリーの名前を指定する。
87	ASCII 文字列	はい	NDS コンテキスト・オプションは、クライアントが使用する初期 NDS コンテキストを指定する。
93	なし	いいえ	DHCP クライアントは、このオプションを使用してクライアント・システム体系を定義する。
94	なし	いいえ	DHCP クライアントは、このオプションを使用してクライアント・ネットワーク・インターフェース ID を定義する。

117	1 つ以上の符号なし整数のリスト	はい	ネーム・サービス検索オプションは、ネーム・サービスの整数オプション・コードの優先順位を指定する。 例:
			`Name Services value`
			`Domain Name Server Option 6`
			`NIS Option 41`

118	1 つのドット付きクワッド	いいえ	サブネット選択オプションは、クライアントによって送信され、指定したサブネットからの IP アドレスの割り当てを DHCP サーバーに依頼する。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明/使用
255	なし	いいえ	DHCP サーバーとクライアントはこのオプションでオプション・リストの終わりを指示する。

プリブート実行環境のベンダー・コンテナーのサブオプション
プリブート実行環境 (PXE) クライアントをサポートしている場合、**DHCP** サーバーは BINLD が自己構成するときに使用する次のオプションを BINLD サーバーに渡します。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明
7	1 つのドット付きクワッド	はい	マルチキャスト IP アドレス。 ブート・サーバー・ディスカバリー・マルチキャスト IP アドレス。

このオプションの使用例を、次に示します。

```
pxeservertype    proxy_on_dhcp_server

Vendor pxeserver
{
     option   7    9.3.4.68
}
```

上記の例では、プロキシー・サーバーが同じマシン上で稼働しているがポート 4011 でクライアント要求を listen していることを **DHCP** サーバーがクライアントに通知します。 この場合、BINLD サーバーは、オプション 60 を「PXEServer」に設定した状態で、INFORM/REQUEST メッセージをポート 67 でブロードキャストするため、ベンダー・コンテナーが必要です。 応答として、**DHCP** サーバーは BINLD が PXEClient の要求を listen する必要があるマルチキャスト IP アドレスを送ります。

次の例では、**dhcpsd** サーバーは、PXEClient クライアントにブート・ファイル名を渡すか、あるいはサブオプションを送ることにより PXEClient を BINLD サーバーへと誘導します。 pxeboofile キーワードは、指定されたクライアント・アーキテクチャーの作成と、メジャー・バージョンおよびマイナー・バージョンのクライアント・システム用の ブート・ファイル・リストの作成に使用されます。

```
pxeservertype     dhcp_pxe_binld

subnet default
{
     vendor pxe
     {
         option 6 2    # Disable Multicast
         option 8 5 4 10.10.10.1 12.1.1.15 12.5.5.5 12.6.6.6¥
                  2 2 10.1.1.10 9.3.4.5 1 1 10.5.5.9¥
                  1 1 9.3.149.15¥
                  4 0
         option 9 5 "WorkSpace On Demand" 2 "Intel"¥
                  1 "Microsoft Windows NT" 4 "NEC ESMPRO"
         option 10 2 "Press F8 to View Menu"
     }
     vendor pxeserver
     {
         option 7 239.0.0.239
     }

}

subnet  9.3.149.0  255.255.255.0
{
     option 3   9.3.149.1
     option 6   9.3.149.15

     vendor pxe
     {
       option    6    4    # bootfile is present in the offer packet
```

```
        pxebootfile   1   2   1   os2.one
        pxebootfile   2   2   1   aix.one
      }
}
```

サーバーは、PXE コンテナー内の各オプション行を使用して、クライアントに行うべきことを指示します。現在サポートされている既知の PXE サブオプションについては、335 ページの『PXE ベンダー・コンテナー・サブオプション』を参照してください。

汎用サーバー操作に対する DHCP サーバー・ファイルの構文

ここでは、汎用サーバー操作に対する **DHCP** ファイルの構文と各フィールドの有効値について説明します。

注 : 次の表の中の時間の単位 (*time_units*) は、オプションであり、実際の時間に対する修飾子を表しています。 デフォルトの時間単位は分です。 有効な値は、秒 (1)、分 (60)、時間 (3600)、日 (86400)、週 (604800)、月 (2392000) および年 (31536000) です。 ここで、括弧内の数字は、指定値 *n* を秒数で表現したい場合に、乗数として使用する値です。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
database	database *db_type*	はい	なし	アドレス・プール、オプション、およびクライアント・アクセス・ステートメントのための定義を保持する 1 次コンテナー。 *db_type* は、ファイルのこの部分を処理するためにロードされるモジュールの名前。 現在使用できる値は、`db_file` だけである。
logging_info	logging_info	はい	なし	ロギング・パラメーターを定義する 1 次ロギング・コンテナー。
logitem	logitem NONE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem SYSERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem OBJERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem PROTOCOL	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem PROTERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem WARN	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
logitem	logitem WARNING	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem CONFIG	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem EVENT	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem PARSEERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem ACTION	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem ACNTING	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem STAT	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem TRACE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem RTRACE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem START	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
numLogFiles	numLogFiles *n*	いいえ	0	作成するログ・ファイル数を指定する。 ログは、最初のファイルがいっぱいになると、次のファイルに回転する。 *n* は作成するファイル数。
logFileSize	logFileSize *n*	いいえ	0	各ログ・ファイルのサイズを 1024 バイト単位で指定する。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
logFileName	logFileName *path*	いいえ	なし	最初のログ・ファイルへのパスを指定する。 最初のログ・ファイルの名前は、*filename* または *filename.extension* である。 ファイルが次に回転するとき、その名前は、ベースの *filename* で始まり、その後ろに番号を追加するか、または拡張子を番号で置き換えたものになる。 例えば、もとのファイル名が `file` である場合、回転後のファイル名は、`file01` となる。 また、もとのファイル名が `file.log` である場合は、`file.01` となる。
CharFlag	charflag yes	いいえ	true	このオペレーティング・システムの **DHCP** サーバーには適用されない。ただし、OS/2 **DHCP** サーバーはこれを使用してデバッグ・ウィンドウを作成する。
CharFlag	charflag true	いいえ	true	このオペレーティング・システムの **DHCP** サーバーには適用されない。ただし、OS/2 **DHCP** サーバーはこれを使用してデバッグ・ウィンドウを作成する。
CharFlag	charflag false	いいえ	true	このオペレーティング・システムの **DHCP** サーバーには適用されない。ただし、OS/2 **DHCP** サーバーはこれを使用してデバッグ・ウィンドウを作成する。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
CharFlag	charflag no	いいえ	true	このオペレーティング・システムの **DHCP** サーバーには適用されない。ただし、OS/2 **DHCP** サーバーはこれを使用してデバッグ・ウィンドウを作成する。
StatisticSnapShot	StatisticSnapShot *n*	いいえ	-1、移らない	統計情報をログ・ファイルに書き込む頻度を秒数で指定する。
UsedIpAddressExpireInterval	UsedIpAddressExpireInterval *n time_units*	いいえ	-1、移らない	BAD 状態に置かれたアドレスを別の状態に移し、その妥当性を再テストする頻度を指定する。
leaseExpireInterval	leaseExpireInterval *n time_units*	いいえ	900 秒	BOUND 状態のアドレスが有効期限切れになっていないか検査する頻度を指定する。 アドレスの有効期限が切れている場合は、状態は EXPIRED に移る。
reservedTime	reservedTime *n time_units*	いいえ	-1、移らない	RESERVED 状態のロング・アドレスが FREE 状態に移るまでのインターバルを指定する。
reservedTimeInterval	reservedTimeInterval *n time_units*	いいえ	900 秒	RESERVE 状態のアドレスを FREE 状態に移してよいか検査する頻度を指定する。
saveInterval	saveInterval *n time_units*	いいえ	3600 秒	**DHCP** サーバーがオープン・データベースの保存を行う頻度を指定する。 多数がロードされているサーバーでは、これは 60 または 120 秒がよい。
clientpruneintv	clientpruneintv *n time_units*	いいえ	3600 秒	クライアントがアドレスに関連付けられていない (UNKNOWN 状態) データベースの除去を **DHCP** サーバーが行う頻度を指定する。 これにより、**DHCP** サーバーのメモリー使用が削減される。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
numprocessors	numprocessors *n*	いいえ	10	作成するパケット・プロセッサー数を指定する。 最小値は 1。
userObject	userObject *obj_name*	はい	なし	サーバーはユーザー定義の共用オブジェクトをロードし、**DHCP** クライアントとの個々の対話のたびにこのオブジェクト内のルーチンをコールすることを指示する。 ロードすべきオブジェクトは、`obj_name.dhcpo` という名前で `/usr/sbin` ディレクトリーに配置されている。 詳しくは、DHCP サーバーのユーザー定義拡張機能 API を参照。
pxeservertype	pxeservertype *server_type*	いいえ	dhcp_only	**dhcpd** サーバーのタイプを示す。 *server_type* には次のいずれかを指定できる。 **dhcp_pxe_binld** **DHCP** は、**dhcpsd**、**pxed**、および **bindl** 機能を実行する。 **proxy_on_dhcp_server** **DHCP** は、PXE クライアントに同じマシンのプロキシー・サーバーを参照させる。 デフォルトは `dhcp_only`。この値は、**dhcpsd** がデフォルト・モードの PXE クライアントをサポートしないことを意味する。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportsubnetselection	supportsubnetselection global supportsubnetselection subnetlevel supportsubnetselection no	いいえ	なし	DHCP サーバーが、クライアント DISCOVER または REQUEST パケットで オプション 118 (サブネット選択オプション) をサポートするかどうかを指示する。 global: 構成ファイル内のすべてのサブネットは、オプション 118 をサポートする。 subnetlevel: キーワード supportoption118 によりこのオプションをサポートすると 構成されたサブネットは、このオプションをサポートする。 no: オプション 118 をサポートしない。

db_file データベースに対する DHCP サーバー・ファイルの構文
db_file データベースのファイル構文には、以下の属性があります。

注 :

1. 次の表の中の時間の単位 (*time_units*) は、オプションであり、実際の時間に対する修飾子を表しています。 デフォルトの時間単位は分です。 有効な値は、秒 (1)、分 (60)、時間 (3600)、日 (86400)、週 (604800)、月 (2392000) および年 (31536000) です。 ここで、括弧内の数字は、指定値 *n* を秒数で表現したい場合に、乗数として使用する値です。
2. 1 つのコンテナー内に指定されている項目は、サブコンテナー内で指定変更されることがあります。 例えば、グローバルに **BOOTP** クライアントを定義する一方で、supportBootp キーワードを両方のコンテナーに指定することで、ある一定のサブネット内で **BOOTP** クライアントをサポートできます。
3. クライアント、クラス、およびベンダーの各コンテナーは、正規表現のサポートを受けることができます。 クラスとベンダーでは、引用符の後ろの最初の文字を感嘆符 (!) にした引用符付き文字列は、文字列の残りを正規表現として処理されるように指示します。 クライアント・コンテナーでは、「hwtype」フィールドと「hwaddr」フィールドの両方で正規表現を使用することができます。 両方のフィールドを表すために、単一の文字列が次のフォーマットで使用されます。

   ```
   decimal_number-data
   ```

   decimal_number がゼロの場合は、data は ASCII 文字列です。 ゼロ以外の場合は、data は 16 進数です。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
subnet	subnet default	はい	なし	関連した範囲なしで、サブネットを指定する。 このサブネットは、クライアントからのクライアント INFORM/REQUEST パケットに対応する場合にのみ、サーバーによって使用され、クライアントのアドレスには、別の一致するサブネット・コンテナーは含まれない。
subnet	subnet *subnet id netmask*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
subnet	subnet *subnet id netmask range*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
subnet	subnet *subnet id netmask label*:*priority*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
subnet	subnet *subnet id netmask range label*:*priority*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。
subnet	subnet *subnet id range*	はい	なし	ネットワーク・コンテナー内で使用されるサブネットを指定する。 これは、オプションの範囲が指定されない限り、サブネット全体のアドレス範囲を定義する。 このサブネットに関連するネットマスクは、周辺のネットワーク・コンテナーから取得される。 **注:** このメソッドは、使用すべきではありません。他のサブネット・フォーマットを使用してください。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
option	option *number data ...*	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 option * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
option	option *number*deny	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 option * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
option	option * deny	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 option * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。
exclude	exclude *IP address*	いいえ	なし	除外ステートメントが入っているコンテナーの範囲を変更する。 除外ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 除外ステートメントは、指定されたアドレスまたは範囲をコンテナーの現在の範囲から除去する。 除外ステートメントを使用すると、サブネットまたはその他のコンテナーに不連続な範囲を作成することができる。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
exclude	exclude *dotted_quad-dotted_quad*	いいえ	なし	除外ステートメントが入っているコンテナーの範囲を変更する。 除外ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 除外ステートメントは、指定されたアドレスまたは範囲をコンテナーの現在の範囲から除去する。 除外ステートメントを使用すると、サブネットまたはその他のコンテナーに不連続な範囲を作成することができる。
range	range *IP_address*	いいえ	なし	範囲ステートメントが入っているコンテナーの範囲を変更する。 範囲ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 range が、コンテナー定義行で範囲を指定していないコンテナーの最初のものである場合は、そのコンテナーのための範囲は、この範囲ステートメントの指定した範囲になる。 最初の range の後ろにある範囲ステートメント、または、その定義に範囲を指定しているコンテナーのための範囲ステートメントは、現在の範囲に追加される。 範囲ステートメントを使用すると、範囲にアドレスを 1 つ、または 1 組追加することができる。 範囲は、サブネット・コンテナー定義の中に収まる必要がある。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
range	range *dotted_quad-dotted_quad*	いいえ	なし	範囲ステートメントが入っているコンテナーの範囲を変更する。 範囲ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 range が、コンテナー定義行で範囲を指定していないコンテナーの最初のものである場合は、そのコンテナーのための範囲は、この範囲ステートメントの指定した範囲になる。 最初の range の後ろにある範囲ステートメント、または、その定義に範囲を指定しているコンテナーのための範囲ステートメントは、現在の範囲に追加される。 範囲ステートメントを使用すると、範囲にアドレスを 1 つ、または 1 組追加することができる。 範囲は、サブネット・コンテナー定義の中に収まる必要がある。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
client	client *hwtype hwaddr* NONE	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
client	client *hwtype hwaddr* ANY	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
client	client *hwtype hwaddr dotted_quad*	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
client	client *hwtype hwaddr range*	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。
class	class *string*	はい	なし	名前が *string* であるクラス・コンテナーを指定する。 文字列には引用符を付けても付けなくてもよい。 引用符を付ける場合は、比較の前に引用符は除かれる。 スペースまたはタブを含む文字列には、引用符が必要である。 このコンテナーは、どのレベルでも有効である。 このクラスのクライアント分配する 1 組のアドレスを指示するために、範囲を指定することができる。 範囲は、ドット付きクワッドの IP アドレスを 1 つで、またはドット付きクワッドの IP アドレスを 2 つをダッシュで区切って示す。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
class	class *string range*	はい	なし	名前が *string* であるクラス・コンテナーを指定する。 文字列には引用符を付けても付けなくてもよい。 引用符を付ける場合は、比較の前に引用符は除かれる。 スペースまたはタブを含む文字列には、引用符が必要である。 このコンテナーは、どのレベルでも有効である。 このクラスのクライアント分配する 1 組のアドレスを指示するために、範囲を指定することができる。 範囲は、ドット付きクワッドの IP アドレスを 1 つで、またはドット付きクワッドの IP アドレスを 2 つをダッシュで区切って示す。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
network	network *network id netmask*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
network	network *network id*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
network	network *network id range*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id* hex""	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id* hex ""	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"! (感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id* 0xdata	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id* ""	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id range*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id range* hex""	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id range* hex ""	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id range* 0xdata	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
vendor	vendor *vendor_id range* ""	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
inoption	inoption *number option_data*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
inoption	inoption *number option_data range*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"!(感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
virtual	virtual fill *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
virtual	virtual sfill *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
virtual	virtual rotate *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
virtual	virtual srotate *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
inorder:	inorder: *id id* ...	いいえ	なし	fill ポリシーを持つ仮想サブネットを指定する。 このポリシーは、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
balance:	balance: *id id* ...	いいえ	なし	rotate ポリシーを持つ仮想サブネットを指定する。このポリシーは、次のコンテナーの次のアドレスを使用することを意味する。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
supportBootp	supportBootp true	いいえ	はい	現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。
supportBootp	supportBootp 1	いいえ	はい	現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。
supportBootp	supportBootp yes	いいえ	はい	現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。
supportBootp	supportBootp false	いいえ	はい	現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。
supportBootp	supportBootp 0	いいえ	はい	現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportBootp	supportBootp no	いいえ	はい	現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。
supportBootp				現行のコンテナーとその下にあるすべてが (指定変更されるまで) **BOOTP** クライアントをサポートするかどうかを指定する。
supportUnlistedclients	supportUnlistedclients BOTH	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportUnlistedclients	supportUnlistedclients DHCP	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。
supportUnlistedclients	supportUnlistedclients BOOTP	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportUnlistedclients	supportUnlistedclients NONE	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。
supportUnlistedclients	supportUnlistedclients true	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportUnlistedclients	supportUnlistedclients yes	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。
supportUnlistedclients	supportUnlistedclients 1	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportUnlistedclients	supportUnlistedclients false	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。
supportUnlistedclients	supportUnlistedclients no	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportUnlistedclients	supportUnlistedclients 0	いいえ	両方	現行のコンテナーとその下にあるすべてが (指定変更されるまで)、リストされていないクライアントをサポートするかどうかを指定する。 この値によって、特定のクライアント・ステートメントを使用しないでアクセスを許可するクライアントを指定する。つまり、すべてのクライアント、**DHCP** クライアントのみ、**BOOTP** クライアントのみ、または許可しない、のいずれかを指定する。 **注 :** 以前のバージョンとの互換性のために、true と false の値がサポートされていますが、使用すべきではありません。 true 値は両方、false 値はなしに対応します。
addressrecorddb	addressrecrddb *path*	いいえ	なし	これを指定すると、`backupfile` キーワードのように働く。 グローバル/データベース・コンテナー・レベルでのみ有効。 **注:** このメソッドは使用すべきではありません。
backupfile	backupfile *path*	いいえ	/etc/db_file.crbk	データベース・バックアップのために使用するファイルを指定する。 グローバル/データベース・コンテナー・レベルでのみ有効。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
checkpointfile	checkpointfile *path*	いいえ	/etc/db_file.chkpt	データベース・チェックポイント・ファイルを指定する。 最初のチェックポイント・ファイルは *path* となる。 2 番目のチェックポイント・ファイルは、末尾の文字が 2 に置き換えられた *path* となる。 そのため、そのチェックポイント・ファイルは 2 で終わることはできない。グローバルまたはデータベースのコンテナー・レベルでのみ有効。
clientrecorddb	clientrecorddb *path*	いいえ	/etc/db_file.cr	データベース保存ファイルを指定する。 このファイルには、**DHCP** サーバーが処理したすべてのクライアント・レコードが含まれる。 グローバル/データベース・コンテナー・レベルでのみ有効。
bootstrapserver	bootstrapserver *IP address*	いいえ	なし	**BOOTP** または **DHCP** パケットの受信後、**TFTP** ファイルに対してクライアントが使用すべきサーバーを指定する。 この値は、パケットの **siaddr** フィールドに入る。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
giaddrfield	giaddrfield *IP address*	いいえ	なし	応答パケットのための giaddrfield を指定する。 **注:** この指定は **BOOTP** と **DHCP** のプロトコルには正しくありませんが、ネットワークのデフォルト・ゲートウェイとして **giaddr** フィールドを必要とするクライアントもあります。 giaddrfield は、どのレベルでも有効ですが、このように競合の可能性があるため、クライアント・コンテナー内でのみ使用します。
pingTime	pingTime *n time_unit*	いいえ	3 秒	ping 応答の待機からアドレスを分配するまでの時間を指定する。 デフォルトの時間単位は 1 秒の 100 分の 1 である。 時間単位の値については、この表の先頭の注に定義してある。 これは、どのコンテナー・レベルにも有効である。 *time_unit* パラメーターはオプションである。
bootptime	bootptime *n time_unit*	いいえ	-1、無限	**BOOTP** クライアントにアドレスを貸し出す時間を指定する。 デフォルトは -1、すなわち無限である。 通常の時間単位値が使用できる。 *time_unit* パラメーターはオプションである。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
AllRoutesBroadcast	allroutesbroadcast no	いいえ	0	ブロードキャスト応答が必要である場合に、応答をすべての経路指定にブロードキャストするかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。 クライアントの実際の MAC アドレスは RIF も含めてリターン・パケットのために格納されるので、この指定はオペレーティング・システムの **DHCP** サーバーでは無視される。 これは、どのコンテナー・レベルにも有効である。
AllRoutesBroadcast	allroutesbroadcast false	いいえ	0	ブロードキャスト応答が必要である場合に、応答をすべての経路指定にブロードキャストするかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。 クライアントの実際の MAC アドレスは RIF も含めてリターン・パケットのために格納されるので、この指定はオペレーティング・システムの **DHCP** サーバーでは無視される。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
AllRoutesBroadcast	allroutesbroadcast 0	いいえ	0	ブロードキャスト応答が必要である場合に、応答をすべての経路指定にブロードキャストするかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。 クライアントの実際の MAC アドレスは RIF も含めてリターン・パケットのために格納されるので、この指定はオペレーティング・システムの **DHCP** サーバーでは無視される。 これは、どのコンテナー・レベルにも有効である。
AllRoutesBroadcast	allroutesbroadcast yes	いいえ	0	ブロードキャスト応答が必要である場合に、応答をすべての経路指定にブロードキャストするかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。 クライアントの実際の MAC アドレスは RIF も含めてリターン・パケットのために格納されるので、この指定はオペレーティング・システムの **DHCP** サーバーでは無視される。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
AllRoutesBroadcast	allroutesbroadcast true	いいえ	0	ブロードキャスト応答が必要である場合に、応答をすべての経路指定にブロードキャストするかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。 クライアントの実際の MAC アドレスは RIF も含めてリターン・パケットのために格納されるので、この指定はオペレーティング・システムの **DHCP** サーバーでは無視される。 これは、どのコンテナー・レベルにも有効である。
AllRoutesBroadcast	allroutesbroadcast 1	いいえ	0	ブロードキャスト応答が必要である場合に、応答をすべての経路指定にブロードキャストするかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。 クライアントの実際の MAC アドレスは RIF も含めてリターン・パケットのために格納されるので、この指定はオペレーティング・システムの **DHCP** サーバーでは無視される。 これは、どのコンテナー・レベルにも有効である。
addressassigned	addressassigned "*string*"	いいえ	なし	アドレスがクライアントに割り当てられたときに実行する引用符付き文字列を指定する。 この文字列には 2 つの %s が必要である。最初の %s は *type-string* のフォーマットのクライアント ID である。 2 番目の %s はドット付きクワッド・フォーマットの IP アドレスである。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
addressreleased	addressreleased "*string*"	いいえ	なし	アドレスがクライアントによって解放されたときに実行する引用符付き文字列を指定する。 この文字列には 1 つだけ %s を入れる。 %s はドット付きクワッドのフォーマットの IP アドレスである。 これは、どのコンテナー・レベルにも有効である。
appenddomain	appenddomain 0	いいえ	いいえ	クライアントがドメイン名を提案していない場合に、クライアント提案のホスト名に、定義済みオプション 15 ドメイン名を付加するかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。
appenddomain	appenddomain no	いいえ	いいえ	クライアントがドメイン名を提案していない場合に、クライアント提案のホスト名に、定義済みオプション 15 ドメイン名を付加するかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。
appenddomain	appenddomain false	いいえ	いいえ	クライアントがドメイン名を提案していない場合に、クライアント提案のホスト名に、定義済みオプション 15 ドメイン名を付加するかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。
appenddomain	appenddomain 1	いいえ	いいえ	クライアントがドメイン名を提案していない場合に、クライアント提案のホスト名に、定義済みオプション 15 ドメイン名を付加するかどうかを指定する。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
appenddomain	appenddomain yes	いいえ	いいえ	クライアントがドメイン名を提案していない場合に、クライアント提案のホスト名に、定義済みオプション 15 ドメイン名を付加するかどうかを指定する。これは、どのコンテナー・レベルにも有効である。
appenddomain	appenddomain true	いいえ	いいえ	クライアントがドメイン名を提案していない場合に、クライアント提案のホスト名に、定義済みオプション 15 ドメイン名を付加するかどうかを指定する。これは、どのコンテナー・レベルにも有効である。
canonical	canonical 0	いいえ	0	クライアント ID が正規フォーマットであることを指定する。 これはクライアント・コンテナーでだけ有効である。
canonical	canonical no	いいえ	0	クライアント ID が正規フォーマットであることを指定する。 これはクライアント・コンテナーでだけ有効である。
canonical	canonical false	いいえ	0	クライアント ID が正規フォーマットであることを指定する。 これはクライアント・コンテナーでだけ有効である。
canonical	canonical 1	いいえ	0	クライアント ID が正規フォーマットであることを指定する。 これはクライアント・コンテナーでだけ有効である。
canonical	canonical yes	いいえ	0	クライアント ID が正規フォーマットであることを指定する。 これはクライアント・コンテナーでだけ有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
canonical	canonical true	いいえ	0	クライアント ID が正規フォーマットであることを指定する。 これはクライアント・コンテナーでだけ有効である。
leaseTimeDefault	leaseTimeDefault *n time_unit*	いいえ	86400 秒	クライアントのためのデフォルトの貸出時間を指定する。 これは、どのコンテナー・レベルにも有効である。 *time_unit* パラメーターはオプションである。
proxyarec	proxyarec never	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。 usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。 usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。 always は、すべてのクライアントに A レコード更新を 行うことを示す。 XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。 standard は always の同義語である。 protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec usedhcpddns	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec usedhcpddnsplus	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec always	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec usedhcpddnsprotected	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec usedhcpddnsplusprotected	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。 usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。 usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。 always は、すべてのクライアントに A レコード更新を 行うことを示す。 XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。 standard は always の同義語である。 protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec alwaysprotected	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
proxyarec	proxyarec standard	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
	proxyarec protected	いいえ	usedhcpddnsplus	DNS の A レコード更新に使用するオプションとメソッドを指定する。never は、A レコード更新を行わないこと を示す。usedhcpddns は、クライアントが指定している 場合はオプション 81 を使用することを示す。usedhcpddnsplus は、指定があれば、オプション 81、またはオプション 12 と 15 を使用することを 示す。always は、すべてのクライアントに A レコード更新を 行うことを示す。XXXXprotected は、**nsupdate** コマンドを変更して、そのクライアントが確実に許可されるようにする。standard は always の同義語である。protected は alwaysprotected の同義語である。 これは、どのコンテナー・レベルにも有効である。
releasednsA	releasednsA "*string*"	いいえ	なし	アドレスが解放されたときに使用する実行文字列を指定する。 この文字列は、解放されたアドレスに関連する A レコードを除去するために使用される。 これは、どのコンテナー・レベルにも有効である。
releasednsP	releasednsP "*string*"	いいえ	なし	アドレスが解放されたときに使用する実行文字列を指定する。 この文字列は、解放されたアドレスに関連する PTR レコードを除去するために使用される。これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
removedns	removedns "*string*"	いいえ	なし	アドレスが解放されたときに使用する実行文字列を指定する。 この文字列は、解放されたアドレスに関連する PTR レコードと A レコードを除去するために使用される。 これは、どのコンテナー・レベルにも有効である。 **注:** これは使用すべきではありません。 `releasednsA` と `releasednsP` キーワードを使用してください。
updatedns	updatedns "*string*"	いいえ	なし	アドレスがバインドされたときに使用する実行文字列を指定する。 この文字列は、そのアドレスに関連する A レコードと PTR レコードの両方を更新するために使用される。 これは、どのコンテナー・レベルにも有効である。 **注:** これは使用すべきではありません。 `updatednsA` と `updatednsP` キーワードを使用してください。
updatednsA	updatednsA "*string*"	いいえ	なし	アドレスがバインドされたときに使用する実行文字列を指定する。 この文字列は、そのアドレスに関連する A レコードを更新するために使用される。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
updatednsP	updatednsP "*string*"	いいえ	なし	アドレスがバインドされたときに使用する実行文字列を指定する。 この文字列は、そのアドレスに関連する PTR レコードを更新するために使用される。 これは、どのコンテナー・レベルにも有効である。
hostnamepolicy	hostnamepolicy suggested	いいえ	デフォルト	クライアントに戻すホスト名を指定する。 デフォルト・ポリシーは、提案名よりも定義済みのホスト名とドメイン名を優先する指示である。 その他のポリシーは、それぞれ指定のキーワードに厳しく従う (例えば、`defined` は定義済み名を戻す、または構成に名前が定義されていない場合は何も戻さない)。 また、ポリシーに `always` 修飾子を使用すると、クライアントがパラメーター・リスト・オプションによって要求したかどうかに関係なく、サーバーに対してホスト名オプションを戻す指示になる。 ホスト名を提案することは、それを要求することにもなる点に注意。ホスト名は、オプション 81 またはオプション 12 と 15 によって提案することができる。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
hostnamepolicy	hostnamepolicy resolved	いいえ	デフォルト	クライアントに戻すホスト名を指定する。 デフォルト・ポリシーは、提案名よりも定義済みのホスト名とドメイン名を優先する指示である。 その他のポリシーは、それぞれ指定のキーワードに厳しく従う (例えば、`defined` は定義済み名を戻す、または構成に名前が定義されていない場合は何も戻さない)。 また、ポリシーに `always` 修飾子を使用すると、クライアントがパラメーター・リスト・オプションによって要求したかどうかに関係なく、サーバーに対してホスト名オプションを戻す指示になる。 ホスト名を提案することは、それを要求することにもなる点に注意。ホスト名は、オプション 81 またはオプション 12 と 15 によって提案することができる。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
hostnamepolicy	hostnamepolicy always_resolved	いいえ	デフォルト	クライアントに戻すホスト名を指定する。 デフォルト・ポリシーは、提案名よりも定義済みのホスト名とドメイン名を優先する指示である。 その他のポリシーは、それぞれ指定のキーワードに厳しく従う (例えば、`defined` は定義済み名を戻す、または構成に名前が定義されていない場合は何も戻さない)。 また、ポリシーに `always` 修飾子を使用すると、クライアントがパラメーター・リスト・オプションによって要求したかどうかに関係なく、サーバーに対してホスト名オプションを戻す指示になる。 ホスト名を提案することは、それを要求することにもなる点に注意。ホスト名は、オプション 81 またはオプション 12 と 15 によって提案することができる。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
hostnamepolicy	hostnamepolicy defined	いいえ	デフォルト	クライアントに戻すホスト名を指定する。 デフォルト・ポリシーは、提案名よりも定義済みのホスト名とドメイン名を優先する指示である。 その他のポリシーは、それぞれ指定のキーワードに厳しく従う (例えば、`defined` は定義済み名を戻す、または構成に名前が定義されていない場合は何も戻さない)。 また、ポリシーに `always` 修飾子を使用すると、クライアントがパラメーター・リスト・オプションによって要求したかどうかに関係なく、サーバーに対してホスト名オプションを戻す指示になる。 ホスト名を提案することは、それを要求することにもなる点に注意。ホスト名は、オプション 81 またはオプション 12 と 15 によって提案することができる。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
hostnamepolicy	hostnamepolicy always_defined	いいえ	デフォルト	クライアントに戻すホスト名を指定する。 デフォルト・ポリシーは、提案名よりも定義済みのホスト名とドメイン名を優先する指示である。 その他のポリシーは、それぞれ指定のキーワードに厳しく従う (例えば、`defined` は定義済み名を戻す、または構成に名前が定義されていない場合は何も戻さない)。 また、ポリシーに `always` 修飾子を使用すると、クライアントがパラメーター・リスト・オプションによって要求したかどうかに関係なく、サーバーに対してホスト名オプションを戻す指示になる。 ホスト名を提案することは、それを要求することにもなる点に注意。ホスト名は、オプション 81 またはオプション 12 と 15 によって提案することができる。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
hostnamepolicy	hostnamepolicy default	いいえ	デフォルト	クライアントに戻すホスト名を指定する。 デフォルト・ポリシーは、提案名よりも定義済みのホスト名とドメイン名を優先する指示である。 その他のポリシーは、それぞれ指定のキーワードに厳しく従う (例えば、`defined` は定義済み名を戻す、または構成に名前が定義されていない場合は何も戻さない)。 また、ポリシーに `always` 修飾子を使用すると、クライアントがパラメーター・リスト・オプションによって要求したかどうかに関係なく、サーバーに対してホスト名オプションを戻す指示になる。 ホスト名を提案することは、それを要求することにもなる点に注意。ホスト名は、オプション 81 またはオプション 12 と 15 によって提案することができる。 これは、どのコンテナー・レベルにも有効である。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
bootfilepolicy	bootfilepolicy suggested	いいえ	suggested	ブート・ファイル名をクライアントに戻す際の選択を指定する。 suggested は、どのサーバー構成名よりもクライアント提案のブート・ファイル名を優先する。 merge は、サーバー構成ホーム・ディレクトリーにクライアント提案名を付加する。 defined はどの提案ブート・ファイル名よりも定義済み名を選択する。 always は、クライアントがパラメーター・リスト・オプションによってブート・ファイル・オプションを要求するかどうかにはかかわらず、定義済み名を戻す。
bootfilepolicy	bootfilepolicy merge	いいえ	suggested	ブート・ファイル名をクライアントに戻す際の選択を指定する。 suggested は、どのサーバー構成名よりもクライアント提案のブート・ファイル名を優先する。 merge は、サーバー構成ホーム・ディレクトリーにクライアント提案名を付加する。 defined はどの提案ブート・ファイル名よりも定義済み名を選択する。 always は、クライアントがパラメーター・リスト・オプションによってブート・ファイル・オプションを要求するかどうかにはかかわらず、定義済み名を戻す。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
bootfilepolicy	bootfilepolicy defined	いいえ	suggested	ブート・ファイル名をクライアントに戻す際の選択を指定する。 suggested は、どのサーバー構成名よりもクライアント提案のブート・ファイル名を優先する。 merge は、サーバー構成ホーム・ディレクトリーにクライアント提案名を付加する。 defined はどの提案ブート・ファイル名よりも定義済み名を選択する。 always は、クライアントがパラメーター・リスト・オプションによってブート・ファイル・オプションを要求するかどうかにはかかわらず、定義済み名を戻す。
bootfilepolicy	bootfilepolicy always	いいえ	suggested	ブート・ファイル名をクライアントに戻す際の選択を指定する。 suggested は、どのサーバー構成名よりもクライアント提案のブート・ファイル名を優先する。 merge は、サーバー構成ホーム・ディレクトリーにクライアント提案名を付加する。 defined はどの提案ブート・ファイル名よりも定義済み名を選択する。 always は、クライアントがパラメーター・リスト・オプションによってブート・ファイル・オプションを要求するかどうかにはかかわらず、定義済み名を戻す。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
stealfromchildren	stealfromchildren true	いいえ	いいえ	親コンテナーにアドレスがなくなったとき、親コンテナーが子コンテナーからアドレスを「スチール」すべきかどうかを指定する。 すなわち、クラスを範囲アドレスによって定義したサブネットがある場合、それらのアドレスは、そのクラスを指定しているクライアントのために予約される。 `stealfromchildren` true の場合は、アドレスを子から取って、要求を試行し、満足させようとする。 デフォルトはアドレスをスチールしないことである。
stealfromchildren	stealfromchildren 1	いいえ	いいえ	親コンテナーにアドレスがなくなったとき、親コンテナーが子コンテナーからアドレスを「スチール」すべきかどうかを指定する。 すなわち、クラスを範囲アドレスによって定義したサブネットがある場合、それらのアドレスは、そのクラスを指定しているクライアントのために予約される。 `stealfromchildren` true の場合は、アドレスを子から取って、要求を試行し、満足させようとする。 デフォルトはアドレスをスチールしないことである。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
stealfromchildren	stealfromchildren yes	いいえ	いいえ	親コンテナーにアドレスがなくなったとき、親コンテナーが子コンテナーからアドレスを「スチール」すべきかどうかを指定する。 すなわち、クラスを範囲アドレスによって定義したサブネットがある場合、それらのアドレスは、そのクラスを指定しているクライアントのために予約される。 `stealfromchildren` true の場合は、アドレスを子から取って、要求を試行し、満足させようとする。 デフォルトはアドレスをスチールしないことである。
stealfromchildren	stealfromchildren false	いいえ	いいえ	親コンテナーにアドレスがなくなったとき、親コンテナーが子コンテナーからアドレスを「スチール」すべきかどうかを指定する。 すなわち、クラスを範囲アドレスによって定義したサブネットがある場合、それらのアドレスは、そのクラスを指定しているクライアントのために予約される。 `stealfromchildren` true の場合は、アドレスを子から取って、要求を試行し、満足させようとする。 デフォルトはアドレスをスチールしないことである。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
stealfromchildren	stealfromchildren 0	いいえ	いいえ	親コンテナーにアドレスがなくなったとき、親コンテナーが子コンテナーからアドレスを「スチール」すべきかどうかを指定する。 すなわち、クラスを範囲アドレスによって定義したサブネットがある場合、それらのアドレスは、そのクラスを指定しているクライアントのために予約される。 `stealfromchildren` true の場合は、アドレスを子から取って、要求を試行し、満足させようとする。 デフォルトはアドレスをスチールしないことである。
stealfromchildren	stealfromchildren no	いいえ	いいえ	親コンテナーにアドレスがなくなったとき、親コンテナーが子コンテナーからアドレスを「スチール」すべきかどうかを指定する。 すなわち、クラスを範囲アドレスによって定義したサブネットがある場合、それらのアドレスは、そのクラスを指定しているクライアントのために予約される。 `stealfromchildren` true の場合は、アドレスを子から取って、要求を試行し、満足させようとする。 デフォルトはアドレスをスチールしないことである。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
homedirectory	homedirectory *path*	いいえ	なし	応答パケットのファイル・セクションで使用するホーム・ディレクトリーを指定する。 これは、どのコンテナー・レベルにも指定できる。 着信パケットのファイル・セクションに指定されている項目が、ブート・ファイルおよびホーム・ディレクトリー・ステートメントとどのように対話するかは、bootfile ポリシーで定義する。
bootfile	bootfile *path*	いいえ	なし	応答パケットのファイル・セクションで使用するブート・ファイルを指定する。 これは、どのコンテナー・レベルにも指定できる。 着信パケットのファイル・セクションに指定されている項目が、ブート・ファイルおよびホーム・ディレクトリー・ステートメントとどのように対話するかは、bootfile ポリシーで定義する。
pxebootfile	pxebootfile *system_architecture major_version minor_version boofilename*	いいえ	なし	クライアントに渡すブート・ファイル名を指定する。 **dhcpsd** が PXE クライアントをサポートしている (`pxeservertype` に `dhcp_pxe_binld` が指定されている) 場合にのみ使用する。 `pxebootfile` の後に続くパラメーターの数が 4 つ未満の場合、構成ファイル・パーサーはエラーを生成し、すべての追加パラメーターを無視する。 `pxebootfile` は、コンテナー内でのみ使用できる。

キーワード	フォーマット	サブコンテナー	デフォルト値	意味
supportoption118	supportoption118 *no*/*yes*	いいえ。サブネット・コンテナーにおいてのみ定義可能。	なし	このキーワードは、このコンテナーがオプション 118 をサポートするかどうかを指定する。「はい」はサポートすることを意味し、「いいえ」はサポートしないことを意味する。このオプションを有効にするには、キーワード supportsubnetselection も使用する必要がある。

DHCP とネットワーク・インストール管理における推奨事項
IP アドレスの動的割り当ての概念は、かなり新しいものです。 以下の推奨事項は、**DHCP** とネットワーク・インストール管理 (NIM) の対話に役立ちます。

1. NIM 環境でオブジェクトを構成する場合は、可能な限りホスト名を使用してください。 それによって動的ネーム・サーバーを使用できるようになります。動的ネーム・サーバーは、NIM 環境でホスト名が IP アドレスに変換されたときに IP アドレスを更新します。
2. NIM マスターと **DHCP** サーバーは、同じシステム上に配置してください。 **DHCP** サーバーは更新 DNS 文字列の中にオプションを持っており、「NIM」に設定された場合、静的 IP アドレスが変更されても NIM オブジェクトが静的 IP アドレスを必要としない状態を保持しようとします。
3. NIM クライアントの場合は、デフォルトの貸出時間を、クライアントのインストールに要する時間の 2 倍に設定してください。 これによって、貸し出された IP アドレスを、インストールの間、有効にしておくことができます。 インストール後、クライアントを再始動します。 インストールのタイプに応じて **DHCP** が開始するか、DHCP を構成する必要が生じます。
4. dhcpsd サーバーは、PTR と A の両方の DNS レコードに責任を負う必要があります。 NIM がマシンを再インストールされた場合、RSA が入っているファイルが削除され、クライアントはレコードを更新できません。 サーバーは、システム・レコードを更新します。 そのためには、/etc/dhcpcd.ini 内の updatedns の行を次のように変更します。

   ```
   updatedns "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' '%s' NONE NONIM"
   ```

   /etc/dhcpsd.cnf 内で updatedns の行を次のように変更します。

   ```
   updatedns "/usr/sbin/dhcpaction '%s' '%s' '%s' '%s' '%s' BOTH NIM"
   ```

   注 : NIM オブジェクトが BOS インストール保留状態に入った場合、 dhcpsd サーバーは、最初に意図したのと異なる引数を渡すことがあります。 これを避けるには、クライアントがこの保留状態にある時間を最小限にとどめるようにしてください。

これらのヒントに従えば、NIM 環境で動的クライアントを処理できます。

動的ホスト構成プロトコル、バージョン 6

動的ホスト構成プロトコル (DHCP) は、中央でネットワーク構成を保守するためのメソッドを提供します。このトピックは **DHCPv6** に特定のものです。特に明記していない限り、「IP アドレス」への言及はすべて IPv6 アドレスのことであり、「**DHCP**」への言及はすべて **DHCPv6** のことです。

DHCPv4 サーバーは、**DHCPv6** サーバーと同じリンク上に共存できます。 プロトコルの詳細な説明については、RFC 3315 を参照してください。

DHCP は、ネットワーク上のクライアント・マシンが IP アドレスやその他の構成パラメーターをサーバーから入手できるようにするアプリケーション層のプロトコルです。 これらのパラメーターは、オプションで定義されています。 オプションは、クライアント上のあるデーモンとサーバー上の別のデーモンとの間でパケットを交換することによって、入手されます。 これらのメッセージ交換は、**UDP** パケットの形式です。 クライアントは、**autoconf6** コマンドまたは他のメソッドを介してリンク・ローカル・アドレスを使用し、そのサーバーに対する送信元アドレスを識別します。 サーバーは、予約済みリンク範囲のマルチキャスト・アドレスを listen します。 リレー・エージェントにより、クライアントおよびサーバーは、同じリンク上になくても通信することができます。

このトピックでは、1 つの IA_NA およびこの IA_NA の 1 つのアドレスを持つ単一のインターフェースを 対象とする、4 つのメッセージの交換ハンドシェークを説明しています。 IP アドレスを入手するために、**DHCP** クライアント・デーモン (**dhcpcd6**) は SOLICIT メッセージを **All_DHCP_Relay_Agents_and_Servers** アドレスに送信します。そのメッセージはサーバーによって受信されて処理されます。 (ネットワーク上に複数のサーバーを余分に構成することができます。) そのクライアントに使用可能なアドレスがあれば、ADVERTISE メッセージが作成され、クライアントに再度送信されます。 このメッセージには、そのクライアント用の IP アドレスとその他のオプションが入っています。 クライアントは、サーバー DHCP ADVERTISE メッセージを受信し、一方ではその他の通知を待ちながらそれを格納します。 クライアントが最適な通知を選択すると、DHCP REQUEST が **All_DHCP_Relay_Agents_and_Servers** アドレスに送信され、必要なサーバー通知が指定されます。

REQUEST メッセージは、すべての構成済み **DHCP** サーバーにより受信されます。 そして、各サーバーは、要求されているサーバーが自分であるかどうかを検査します。 サーバーは、独自のものと一致しないサーバー DUID を持つパケットは処理しません。 要求されたサーバーはこのアドレスを割り当て済みとマークし、DHCP REPLY を戻します。これでトランザクションは完了します。 クライアントはサーバーによって指定された時間 (貸出時間: valid-lifetime) だけアドレスを持ちます。

アドレスの推奨存続時間の期限が切れると、クライアントは貸出時間を延長するために、サーバーに RENEW パケットを送信します。 サーバーは、アドレスの更新に応じる場合には DHCP REPLY を送信します。 クライアントがそのクライアントの 現行アドレスを所有するサーバーから応答を取得できなかった場合、例えば、そのサーバーがあるネットワークから別のネットワークに移動していたとすると、DHCP REBIND パケットがマルチキャストされます。 有効存続時間を使い切っても、クライアントがアドレスを 更新できなかった場合、アドレスはインターフェースから 除去され、このプロセスが最初から繰り返されます。 このサイクルは、ネットワーク上の複数のクライアントが同じアドレスに割り当てられるのを防ぎます。

クライアントは複数の IA_NA オプションを持つことができ、各 IA_NA は複数のアドレスを持つことができます。 また、クライアントは複数の IA_TA オプションを持つことができ、それぞれが複数のアドレスを持つことができます。

- **非一時アドレス用 ID 関連付け (Identity association for non-temporary addresses (IA_NA))**: 一時アドレスでない割り当て済みアドレスを実行する IA
- **一時アドレス用 ID 関連付け (Identity association for temporary addresses (IA_TA))**: 一時アドレスを実行する IA (RFC 3041 を参照)
- **DUID**: **DHCP** 参加者用の **DHCP** 固有の ID。各 **DHCP** クライアントおよびサーバーには、リブートにおいて変化しない固有の DUID があります。

DHCP サーバーは、キーに基づいてアドレスを割り当てます。 4 つの共通キーとして、クラス、ベンダー、クライアント ID、および 着信オプションがあります。 サーバーは、これらのキーを使用して、クライアントへ戻すアドレスと一連の構成オプションを割り振ります。

class
クラス・キーは完全にクライアント構成可能です。 これでアドレスとオプションを指定できます。 このキーは、ネットワーク内のマシン機能を示したり、管理目的でマシンがどのようにグループ化されるのかを示したりするために使用できます。 例えば、ネットワーク管理者は、NetBIOS クライアントのためのオプションを含む NetBIOS クラスを作成したり、特定のプリンターにアクセスする必要のあるアカウンティング部門マシンを表す accounting クラスを作成することができます。

vendor
ベンダー・キーは、そのハードウェアおよびソフトウェア・プラットフォームによってクライアントを識別するのに役立ちます。

client ID
client ID キーは、DUID を介してクライアントを識別します。 クライアント ID は、**dhcpcd** デーモンの duid ファイル内で指定されます。 また、サーバーはクライアント ID を使用して、特定のクライアントへオプションを渡したり、特定のクライアントにパラメーターの受信を禁止したりすることができます。

Inoption
着信オプション・キーは、クライアントが要求するオプションにより、クライアントを識別します。

これらのキーは、単独で、または組み合わせて使用することができます。 複数のキーがクライアントから提供され、複数のアドレスを割り当てることができる場合は、キーは 1 つだけ選択され、その選択されたキーからまず最初にオプション・セットが生成されます。

リレー・エージェントは、クライアントからの初期のマルチキャストがローカル・ネットワークから出られるようにするために必要です。 リレー・エージェントは、**DHCP** パケット用の転送エージェントとして機能します。

DHCPv6 サーバー
DHCPv6 サーバーには 3 つの主要コンポーネントがあります。

DHCP サーバーは 3 つの主要コンポーネント (データベース、プロトコル・エンジン、一連のサービス・スレッド) に区分されています。 それぞれのコンポーネントに独自の構成情報があります。

DHCPv6 データベース
db_filev6.dhcpo データベースは、クライアントおよびアドレスをトラックするため、およびアクセス制御のために使用されます。

オプションも、検索用およびクライアントへの配布用に、このデータベースに格納されます。 データベースは、動的にロード可能なオブジェクトとしてインプリメントされます。

このデータベースは、構成ファイル内の情報を使用して用意され、整合性を検査されます。 また、このデータベースには、アドレスとオプションのプールが入っています。

主記憶ファイルとそのバックアップは、ASCII ファイルです。 データベース主記憶ファイルのフォーマットは、以下のとおりです。

注 : これらのファイルを手動で編集しないでください。

```
DB6-1.0
Client-Info {
duid 1-0006085b68e20004ace491d3
state 7
authinfo {
    protocol 2
    algorithm 1
    rdm 0
    replay 1206567640
}
  Interface 0 {
    Inoptions {
    interface-id "en1"
    policies 2
    maxopcode 16
    numiana 1
      Ianalist {
      option 3 40
00000001000000320000005000050018deaddeadaaaaaaaa0000000000000006000000640000000c8
      }
    numiata 0
      Optiontable {
      option 6 10 00030004001700180237
      option 8 2 e659
      option 15 14 000369626d000373756e00026870
      option 16 18 000004d200073078313131313100369626d
      }
    }
    Ianarec {
    IAID 1
    t1 50
    t2 80
      Addrec {
      Address dead:dead:aaaa:aaaa::6
```

```
        state 3
        starttime  1087592918
        preferred-lifetime 100
        valid-lifetime 200
        }
      }
    }
  }
```

最初の行はこのファイルのバージョン ID である DB6-1.0 です。 その後の行は、クライアント・レコード定義行です。 サーバーは、この 2 行目からファイルの終わりまでを読み取ります。 (引用符で囲まれたパラメーターは、引用符で囲まれている必要があります。)

duid
クライアントがサーバーにそれ自体を表すために使用する ID。

Interface
クライアントは、複数のインターフェースを 持つことができます。 クライアントが単一のインターフェースを持ってあり、各 IA_NA または IA_TA ごとに個々の **SOLICIT** メッセージを作成する場合、ファイルにはこのクライアントについて 複数のインターフェースが 含まれることになります。

Inoptions
クライアントからの着信オプション。

policies
ユニキャスト、再構成オプション、および高速コミットを識別するためのフラグ。

maxopcode
最大のオプション・コード。

numiana
このインターフェースの IA_NA の数。

Ianalist
クライアントから着信する IA_NA オプションのリスト。

numiata
このインターフェースの IA_TA の数。

Optiontable
IA_NA および IA_TA オプション以外にクライアントが要求するオプションのリスト。

Ianarec
サーバー・データベースからの保管された IA_NA レコード・コンテナー。

IAID
IA_NA の ID。

t1
この IA_NA に対する推奨存続期間のパーセント。

t2
この IA_NA に対する有効存続時間のパーセント。

Addrec
サーバー・データベースからのアドレス・レコード・コンテナー。

Address
このアドレス・レコードに応じてクライアントに指定されるアドレス。

state
クライアントの現在の状態。 **DHCP** プロトコル・エンジンには許容セットが含まれ、その状態は **DHCP** データベースで維持されます。 **state** の次の数字はその値を表します。 状態としては、次のものがあります。

(1) FREE (空き)
使用できるようにアドレスが空いていることを表す。 一般に、クライアントにアドレスが割り当てられている場合は、クライアントはこの状態にはなりません。 **dadmin** および **lssrc** コマンドは、この状態を Free として報告します。

(2) BOUND (バインド済み)
クライアントとアドレスが結合され、クライアントはこのアドレスにしばらく割り当てられていたことを示す。 **dadmin** コマンドと **lssrc** コマンドは、この状態を「Leased」と報告します。

(3) EXPIRED (有効期限切れ)
クライアントとアドレスが結合されているが、解放済みアドレスの場合と同様、情報目的だけであることを示す。 ただし、この有効期限切れ状態は、その貸出期間を期限切れにしたクライアントを表します。 有効期限切れアドレスは、新たに使用に回すことができ、すべての空いているアドレスが使用不能になった後で、解放済みアドレスが再度割り当てられる前に、再割り当てされます。 **dadmin** コマンドと **lssrc** コマンドは、この状態を「Expired」と報告します。

(4) RELEASED (解放済み)
クライアントとアドレスが情報目的でのみ結合されていることを示す。 **DHCP** プロトコルは、**DHCP** サーバーがサービスしたクライアントについての 情報を将来の参照に備えて保持するように提案します (主な目的は、参照したときに以前そのアドレスに割り当てられていたクライアントに同じアドレスを与えるためです)。 この状態は、クライアントがそのアドレスを解放したことを示します。 このアドレスは、他に使用できるアドレスがない場合に、別のクライアントで使用することができます。 **dadmin** コマンドと **lssrc** コマンドは、この状態を「Released」と報告します。

(5) RESERVED (予約済み)
クライアントとアドレスがゆるやかに結合されていることを示す。 このクライアントは **DHCP** 検出メッセージを出しており、**DHCP** サーバーが応答したが、クライアントはそのアドレスに対する **DHCP** 要求でまだ応答していません。 **dadmin** コマンドと **lssrc** コマンドは、この状態を「Reserved」と報告します。

(6) BAD (正しくない)
ネットワークで使用されているが、**DHCP** サーバーによって分配されていないアドレスを表す。 この状態は、クライアントがリジェクトしたアドレスも 表します。 この状態はクライアントには適用されません。 **dadmin** コマンドはこの状態を「Used」として報告し、**lssrc** コマンドはこの状態を「Bad」と報告します。

Starttime
このアドレスが分配された時刻。2000 年 1 月 1 日以降の秒数で表されます。

preferred-lifetime
このアドレスに更新の必要が生じるまでの秒数。

valid-lifetime
このアドレスが無効になり、使用できなくなるまでの秒数。

protocol
クライアントが使用している認証プロトコル:

(1) DELAYED
クライアントは遅延認証を使用しています。

(2) RECONFIGURE KEY
クライアントは再構成鍵認証を使用しています。

algorithm
クライアントが使用している認証アルゴリズム:

(1) HMAC-MD5
クライアントは、メッセージ・ダイジェストを作成するのに鍵付き MD5 アルゴリズムを使用しています。

rdm
クライアントが使用しているリプレイ 検出メソッド:

(0) Monotonically increasing counter
クライアントは、リプレイ 値を変更するのに、単調に増えるカウンターを使用しています。

replay
リプレイ・フィールドの現在値。

チェックポイント・ファイルのための構文は指定しません。 サーバーがクラッシュしたり、あるいは、ユーザーがサーバーをシャットダウンしたためにデータベースを正常にクローズできない場合には、サーバ

ーがチェックポイントとバックアップ・ファイルを処理して、有効なデータベースを構築し直すことができます。 サーバーがクラッシュした時点でチェックポイント・ファイルに書き込み中であったクライアントがあれば、それらは失われます。 現在、クライアントの処理時に断続的な保管はありません。 デフォルト・ファイルは、次のとおりです。

/etc/dhcpv6/db_file6.cr
正常なデータベース操作

/etc/dhcpv6/db_file6.crbk
データベースのバックアップ

DHCP のスレッド化操作

DHCP サーバーの最後の部分は、実行を続行するために使用される一連の操作です。

DHCP サーバーはスレッド化されるため、これらの操作は、すべてが一緒に行われるようにするために時々実行されるスレッドとしてセットアップされます。

main スレッド
このスレッドはシグナルを処理します。 例えば、次のとおりです。

- SIGHUP (-1) は構成ファイル内のすべてのデータベースを再表示する。
- SIGTERM (-15) はサーバーを正常中止にする。
- SIGUSR1 (-30) はサーバーに構成データベースをダンプさせる。

src スレッド
このスレッドは、SRC 要求 (**startsrc**、**stopsrc**、**lssrc**、**traceson**、および **refresh** など) を処理します。

dadmin スレッド
このスレッドは、**dadmin** クライアント・プログラムと **DHCP** サーバーとのインターフェースです。 **dadmin** ツールを使用すると、手動でデータベース・ファイルを編集するのを避けるために、データベースを変更するとともに、その状況を入手することができます。 **dadmin** と **src** のスレッドが追加されたおかげで、サーバーはサービス要求を処理する一方でクライアントの要求も処理することができるようになりました。

garbage スレッド
このスレッドは、定期的にデータベースをクリーンアップするとともにデータベースを 保存し、アドレスを持っていないクライアントを除去し、かなりの長期間予約済み状態になっている予約済みアドレスを除去するためのタイマーを動かします。 これらのタイマーはすべて構成することができます。

パケット・プロセッサー
いずれも、**DHCPv6** クライアントからの要求を処理することができます。 必要となるパケット・プロセッサー数は、ロード状況とマシンによっていくらか異なります。 この数は構成可能です。デフォルトは 1 です。パケット・スレッドの最大数は 50 です。

logging スレッド
大量のデータがログ・ファイルにログ記録されているシステムでは、logging スレッドの数をデフォルト (1) より多く、最大 (50) まで増やすことができます。

table manager スレッド
このスレッドは、**dhcpsdv6** デーモンが重複パケットを処理しないようにします。

process スレッド
これらのスレッドは、**DHCPv6** クライアント・パケットを処理します。

reconfigure スレッド
このスレッドは、サーバーが (`dadmin -x 6 -i` コマンドなどを使用して) リフレッシュされる場合に、クライアントの再構成を管理します。

DHCPv6 の構成

DHCP サーバーは、デフォルトでは `/etc/dhcpv6/dhcpsdv6.cnf` ファイルを読み取ることで構成されます。このファイルにより、オプションとアドレスの 入った初期データベースが指定されます。

サーバーは SRC コマンドから開始されます。 **dhcpsdv6** がリブートで開始する場合、`/etc/rc.tcpip` ファイルにエントリーを追加します。

DHCP サーバーの構成は、通常、ネットワークで **DHCP** を使用する際の最も難しい作業です。 まず、どのネットワークに **DHCP** クライアントが必要かを決めてください。 ネットワーク内の各サブネットは、**DHCP** サーバーがそのデータベースに加えなければならないアドレスのプールを表しています。 以下に例を示します。

```
subnet dead:dead:aaaa:: 48 {
        option 23 dead::beef beef:aaaa::bbbb:c aaaa:bbbb::cccc  #nameserver list
        option 24 austin.ibm.com ibm.com                        # domain list
 }
```

上記の例は、48 ビットのプレフィックスを付けたサブネット dead:dead:aaaa:: を示しています。 このサブネット中のすべてのアドレス、dead:dead:aaaa::1 から dead:dead:aaaa:ffff:ffff:ffff:ffff:ff7f はプールに入っています。 オプションとして、行の終わりの '{' の前に範囲を指定することができ、また、範囲または除外ステートメントをサブネット・コンテナーに含めることができます。

コメントは # (ポンド記号) で始まります。 **DHCP** サーバーは、始めの # からその行の終わりまでのテキストを無視します。 サーバーは、各 option 行を使用して、クライアントに行うべきことを指示します。

サーバーは、オプションの構文解析方法が分からない場合は、デフォルトのメソッドを使用して、クライアントにそのオプションを送信します。 これにより、**DHCP** サーバーは、RFC 定義ではないがある種のクライアントまたはクライアント 構成で使用できる、サイト固有のオプションを送信することもできます。

DHCPv6 構成ファイル

構成ファイルには、アドレス・セクションとオプション定義セクションがあります。 これらのセクションは、コンテナーを使用して、オプション、修飾子、場合によっては他のコンテナーを保持します。

コンテナー (オプションをグループ化するための方法) は、クライアントをグループに 分類するのに ID を使用します。 コンテナーには 5 つのタイプがあります。サブネット、クラス、ベンダー、着信オプション、およびクライアント です。 現在、一般的なユーザー定義可能コンテナーはありません。 ID は、例えばクライアントがサブネット 間を移動するときにクライアントを追跡できるように、クライアントを固有に定義するものです。 クライアント・アクセスを定義するために、複数のコンテナー・タイプを使用することができます。

オプションは、DNS アドレスやドメイン名など、クライアントに戻される ID です。

修飾子を選択した後、次にセットアップするのはロギングのための 項目です。 ロギング・パラメーターは、データベースのようなコンテナーに指定しますが、コンテナーの場合のキーワードは logging_info です。 **DHCP** の構成方法を学んでいる間は、ロギングのレベルを最高レベルに設定しておくことをお勧めします。 また、ロギング・サブシステムの 初期化後に構成エラーが確実に記録されるようにするために、他のどの構成ファイル・データよりも先にロギング構成を指定しておくのが最善です。 logitem キーワードは、あるロギング・レベルをオンにするために使用します。ロギング・レベルを使用不可にするには logitem キーワードを除去してください。 ロギングのためのその他のキーワードを使用すると、ログのためのファイル名、ファイルのサイズ、回転ログ・ファイル数を指定することができます。

DHCPv6 コンテナー

DHCP サーバーが要求を受信すると、パケットが構文解析され、抽出するコンテナー、オプション、およびアドレスが ID キーによって判別されます。

コンテナーは、タイプごとに、クライアントを識別するための異なるオプションを使用します。

- サブネット・コンテナーは、**hintlist** フィールドまたは受信インターフェースのインターフェース ・アドレスを使用して、クライアントがどのサブネットに所属するかを判別する。
- クラス・コンテナーは、オプション 15 の値 (OPTION_USER_CLASS ID) を使用する。
- ベンダー・コンテナーはオプション 16 の値 (OPTION_VENDOR_CLASS) を使用する。
- クライアント・コンテナーは、DHCP クライアントの DUID からオプション 1 (OPTION_CLIENTID) を使用する。
- 着信オプション・コンテナーは、クライアントの要求オプションに一致する。

サブネット・コンテナーを除いて、どのコンテナーについても、(正規表現マッチングを 含む) マッチングする値を指定することができます。

暗黙のコンテナーもあります。グローバル・コンテナーがそれです。 オプションと修飾子は、オーバーライドされたり拒否されたりしない限りグローバル・コンテナーに入れられます。 ほとんどのコンテナーは、可視範囲を暗黙指定して他のコンテナーの中に入れることができます。 コンテナーは、それに関連するアドレス範囲を持つこともできますし、持たなくてもかまいません。 サブネットは、その性質上それに関連した範囲があります。

コンテナーとサブコンテナーの基本規則は次のとおりです。

- グローバル・レベルではサブネット・コンテナーのみが有効である。
- サブネットは、それ自体も含め、他のコンテナーの中に入れることはできない。
- 制限付きコンテナーの中には、同じタイプのレギュラー・コンテナーを入れることはできない。 (例えば、Accounting クラスだけが許されたオプションのコンテナーに、文字 a で始まるすべてのクラスが許されたオプションのコンテナーを 入れることはできません。)
- 制限付きクライアント・コンテナーはサブコンテナーを持てない。
- 着信オプションはサブコンテナーを 持てない。

上記の規則に従えば、コンテナーの階層を作成することができます。この階層によって、オプションは、特定のクライアントまたは 1 組のクライアントのグループに 分けられます。

クライアントが複数のコンテナーと一致する場合、**DHCP** サーバーは要求をデータベースに渡します。そこでコンテナー・リストが作成されます。 このリストは、奥行きと優先順位の順に並べられます。 優先順位は、コンテナー内の暗黙の階層として定義されます。 ストリクト・コンテナーは、レギュラー・コンテナーよりも高い優先順位を持ちます。 クライアント、クラス、ベンダー、サブネットがこの順に、また、それぞれのコンテナー・タイプ内では奥行きの順に記載されます。 すなわち、特定度の最も高いものから特定度の低いものの順に並べられたリストができあがります。 以下に例を示します。

```
Subnet 1
    --Class 1
    --Client 1
Subnet 2
    --Class 1
    ----Vendor 1
    ----Client 1
    --Client 1
```

この例では、2 つのサブネット Subnet 1 と Subnet 2 が示されています。 クラス名は 1 つで Class 1、ベンダー名も 1 つで Vendor 1、そしてクライアント名が Client 1 の 1 つです。Class 1 と Client 1 は複数の場所で定義されています。これらは異なるコンテナーに入っているので、名前は同じですがその中の値は異なることがあります。 Client 1 がオプション・リストに Class 1 を指定している Subnet 1 からメッセージを **DHCP** サーバーに送ると、**DHCP** サーバーは次のようなコンテナー・パスを生成します。

```
Subnet 1, Class 1, Client 1
```

最も特定度の高いコンテナーが最後にリストされています。 アドレスを取得するには、このリストを階層の逆向きに調べ、最初に使用可能なアドレスを探します。 次に、このリストを階層の前方向に調べ、オプションを入手します。 オプションは、deny がコンテナーに指定されていなければ、以前の値を指定変更します。 また、Subnet 1 には Class 1 と Client 1 が入っているため、これらはコンテナーの優先順位に従って並べられます。 同じクライアントが Subnet 2 に入っていて、同じメッセージを送信するのであれば、コンテナー・リストは次のように生成されます。

```
Subnet 2, Class 1, Client 1 (at the Subnet 2 level), Client 1 (at the Class 1 level)
```

Subnet 2 が最初にリストされ、次に Class 1、最後に Subnet 2 レベルの Client 1 の順にリストされます (このクライアント・ステートメントが階層では 1 レベル下にあるためです)。 この階層は、最初のクライアント・ステートメントに合致するクライアントが、Subnet 2 内の Class 1 の Client 1 に合致するクライアントよりも特定度が低いことを示唆しています。

階層内の奥行きによって選択された優先順位が、コンテナーそれ自体の優先順位で取り替えられることはありません。 例えば、同一のクライアントが同じメッセージを出し、ベンダー ID を指定している場合、そのコンテナー・リストは次のようになります。

```
Subnet 2, Class 1, Vendor 1, Client 1 (at Subnet 2 level), Client 1 (at Class 1 level)
```

コンテナー優先順位は、クライアント・コンテナーが 1 つ以上のクライアントを定義するための最も特定度の高いコンテナー優先順位であるとする一般的な概念に従っているため、検索パフォーマンスを向上させます。 クラス・コンテナーは、クライアント・コンテナーよりも特定度が低いアドレスを保持しています。ベンダーはさらに特定度が低く、サブネットは特定度が最も低くなっています。

DHCPv6 アドレスとアドレス範囲
どのコンテナー・タイプも関連したアドレス範囲を持つことができます。サブネットにはアドレス範囲が必須です。

コンテナー内の各範囲は、その範囲のサブセットにする必要があり、また、他のコンテナーの範囲とオーバーラップすることはできません。 例えば、あるクラスがサブネット内で定義されており、そのクラスが範囲を持っている場合、その範囲は当該サブネットの範囲のサブセットである必要があります。 また、そのクラス・コンテナー内の範囲は、そのレベルにあるその他の範囲とオーバーラップすることはできません。

範囲は、コンテナーの行で表現し、範囲および除外ステートメントで変更して、あるコンテナーに関連付けられたばらばらのアドレス・セットを使用可能にすることができます。 したがって、サブネットの最初の 10 アドレスと次の 10 アドレスを使用可能にする場合、そのサブネットがサブネット文節でこれらのアドレス範囲を指定することによって、使用するメモリーを削減できると同時に、指定された範囲にないその他のクライアントとのアドレス競合が起きるのを避けることができます。

アドレスを選択すると、アドレス範囲を含んでいるリスト内の後続のコンテナーは、その子とともにリストから除去されます。 アドレスがそのコンテナー内から使用されなかった場合、除去されたコンテナー内のネットワーク固有のオプションは無効です。

DHCPv6 構成ファイル・オプション
アドレスを判別するためにリストが選別された後、1 組のオプションがクライアントのために 生成されます。

この選択プロセスでは、deny が検出されなければ、以前選択されたオプションが上書きされます。検出された場合は、否定されたオプションがクライアントに 送られるリストから除去されます。 このメソッドによって、親コンテナーからの継承で、指定しなければならないデータ量が削減されます。

DHCPv6 サーバー固有オプション
指定する最後のパラメーターのセットは、サーバー固有オプションです。これは、パケット・プロセッサー数を制御したり、不要情報コレクション・スレッドを実行する頻度を決めたりするためのものです。

例えば、次の 2 つのサーバー固有オプションがあります。

reservedTime
ADVERTISE を **DHCP** クライアントに送信した後、アドレスを予約済み状態に留めておく期間を指示する。

reservedTimeInterval
reservedTime よりも長く予約済み状態に置かれているアドレスがないかどうか、**DHCP** サーバーがアドレスをスキャンする頻度を指示する。

これらのオプションは、SOLICIT メッセージをマルチキャストするクライアントがいくつかあり 、それらがその REQUEST メッセージをマルチキャストしないか、またはその REQUEST メッセージがネットワーク内で失われた場合に有益です。 これらのパラメーターを使用すれば、非準拠クライアントのためにアドレスが無期限に予約されることはありません。

もう 1 つの特に有益なオプションとして、*SaveInterval* があります。これは、保存を行う頻度を指示します。

/etc/dhcpv6/dhcpsdv6.cnf ファイル
DHCPv6 サーバーを構成するには、`/etc/dhcpv6/dhcpsdv6.cnf` ファイルを編集します。

キーワードは大/小文字を区別します。 '{' がリストされる場合は、キーワードと同じ行になければなりません。 構成ファイルのサンプルは、 `/usr/samples/tcpip/dhcpv6` にあります。

以下は、/etc/dhcpv6/dhcpsdv6.cnf ファイルの説明です。 次のスタンザがこのファイルで許可されています。

- ロギング
- グローバル・キーワード
- ネストなしコンテナー・ステートメント
- ネストありコンテナー・ステートメント
- オプション
- 共通オプション

DHCPv6 のロギング
ここでは、ロギング・スタンザで入力する **DHCPv6** サーバーのキーワードについて説明します。

このスタンザの存在は必須ではありませんが、存在する場合は構成ファイルの先頭になければなりません。フォーマットは次のとおりです。

```
logging_info { log_options }
```

log_options の値は次のいずれかを指定できます。

表 *64.* ロギング・スタンザで入力するキーワード、値、および説明

キーワード	**値**	**説明**
logFileSize	*num*	ログ・ファイルのサイズを指定する。 *num* 値はキロバイト単位でのログ・ファイルの最大サイズ。 ログ・ファイルは、このサイズに達すると循環する。 logFileSize が指定されないと、無限サイズが前提になる。
logFileName	"*filename*"	ログ・ファイルの名前を指定する。 *filename* 値がログ・ファイルの名前になる。 デフォルトのファイル名とロケーションは、/var/tmp/dhcpsdv6.log。
numLogFiles	*num*	ファイル循環するログ・ファイルの数を指定する。 デフォルトは 0。

表 64. ロギング・スタンザで入力するキーワード、値、および説明 (続き)

キーワード	値	説明
logItem	*type*	望ましいロギングのタイプを指定する。 有効なタイプは次のとおり。 **SYSERR** プラットフォームに対するインターフェースにおけるシステム・エラー。 **OBJERR** プロセス中のオブジェクト間でのオブジェクト・エラー。 **PROTERR** クライアントとサーバーとの間のプロトコル・エラー。 **WARNING** ユーザーが注意を払うべき警告。 **EVENT** プロセスに対して発生するイベント。 **ACTION** プロセスが実行するアクション。 **INFO** 役立つ可能性のある情報。 **ACNTING** サービスを受ける対象とその時期。 **TRACE** デバッグ用のコード・フロー。

DHCPv6 グローバル・キーワード
ここでは、グローバル・キーワード・スタンザで入力するキーワード値について説明します。

グローバル・キーワードは、コンテナーの外部でのみ有効です。 許可される値は次のとおりです。

表 65. グローバル・キーワード・スタンザで入力するキーワード、値、および説明

キーワード	値	説明
UsedIpAddressExpiredInterval	num [*units*]	BAD 状態に置かれたアドレスを別の状態に移し、その妥当性を再テストする頻度を指定する。 単位が設定されないと、システム・デフォルトは秒に設定される。 デフォルト値は -1。
leaseExpiredInterval	num [*units*]	BOUND 状態のアドレスが有効期限切れになっていないか検査する頻度を指定する。 アドレスの有効期限が切れている場合は、状態は EXPIRED に移る。 単位が設定されないと、システム・デフォルトは秒に設定される。 デフォルト値は 900 秒。
reservedTime	num [*units*]	RESERVED 状態のロング・アドレスが FREE 状態に移るまでのインターバルを指定する。 単位が設定されないと、システム・デフォルトは秒に設定される。 デフォルト値は -1。
reservedTimeInterval	num [*units*]	RESERVE 状態のアドレスを FREE 状態に移してよいか検査する頻度を指定する。 単位が設定されないと、システム・デフォルトは秒に設定される。 デフォルト値は 900 秒。

表 65. グローバル・キーワード・スタンザで入力するキーワード、値、および説明 (続き)

キーワード	値	説明
saveInterval	num [*units*]	**DHCP** サーバーがオープン・データベースの保存を行う頻度を指定する。 多数がロードされているサーバーでは、これは 60 または 120 秒がよい。 単位が設定されないと、システム・デフォルトは秒に設定される。 デフォルト値は 3600 秒。
clientpruneintv	num [*units*]	クライアントがアドレスに関連付けられていない (UNKNOWN 状態) データベースの除去を **DHCP** サーバーが行う頻度を指定する。 これにより、**DHCP** サーバーのメモリー使用が削減される。 単位が設定されないと、システム・デフォルトは秒に設定される。 デフォルト値は 3600 秒。
numprocessthreads	num	作成するパケット・プロセッサー・スレッド数を指定する。 最小値は 1。 それぞれの processthread が 1 つのクライアントを処理する。 デフォルトでは 30。
numpacketthreads	num	作成するパケット・スレッド数を指定する。 最小は 1、デフォルトでは 5 に設定される。
numloggingthreads	num	ロギング・スレッドの数を指定する。 デフォルトは 1。
numduidbuckets	num	テーブル・マネージャーにより使用され、numprocessthreads に対して直接相関する。 デフォルトでは、53 に設定される。
numclientbuckets	num	クライアント・レコードを保管するのに使用されるバケット数。 デフォルトでは 1021。
ignoreinterfacelist	*interface* [*interface*]	無視するインターフェースのリスト 。 インターフェースの数は、1 つ以上。
backupfile	"*filename*"	データベース・バックアップのために使用するファイル。 デフォルトのファイルは /etc/dhcpv6/db_file6.crbk。
checkpointfile	"*filename*"	データベース・チェックポイント・ファイルを指定する。 最初のチェックポイント・ファイルは path となる。 2 番目のチェックポイント・ファイルは、末尾の文字が 2 に置き換えられた path となる。そのため、そのチェックポイント・ファイルは 2 で終わることはできない。
clientrecorddb	"*filename*"	データベース保存ファイルを指定する。 このファイルには、**DHCP** サーバーが処理したすべてのクライアント・レコードが含まれる。 デフォルトのファイルは **/etc/dhcpv6/db_file6.cr**。
duid	idtype *value* [*value*]	サーバーの識別に使用される。 許可される値は次のとおり。 • duid 1 *interface* • duid 2 *interface* • duid 3 enterprise number *identifier* • duid number 0x*hexdigit*

表 65. グローバル・キーワード・スタンザで入力するキーワード、値、および説明 (続き)		
キーワード	**値**	**説明**
preference-number	num	クライアントが、情報を入手しようとするサーバーを識別できるようにする。 値が高いほど、クライアントがそのサービスのためにこのサーバーを使用する可能性が高くなる。 デフォルトは、最大値 255。
unicast-enable	*policy*	サーバーでのユニキャスト・ポリシー。これにより、サーバーはユニキャストを使って通信できる。 デフォルトではオン。
tablemgr-policy	*policy*	サーバーによって、テーブル・マネージャーが着信クライアントをより良い方法で管理できるようにする。 デフォルトではオン。
auth	*policy*	サーバーが遅延認証をサポートするのを許可する。 デフォルトではオフ。
auth-keyfile	"*filename*"	クライアント用の遅延認証キーを含むファイル。 デフォルトのファイルは、**/etc/dhcpv6/dhcpsdv6.keys** です。

DHCPv6 ネストなしコンテナー・ステートメント
DHCPv6 サーバー・キーワード subnet は、ネストなしコンテナー・ステートメントで入力します。

ネストなしコンテナー・ステートメントは、グローバル・キーワードの一部としてのみ存在可能です。

表 66. ネストなしコンテナー・ステートメントで入力するキーワード、値、および説明		
項目	**説明**	
subnet	*subnetid prefix-length* [*range*] {OPTIONS}	使用するサブネットの名前。 *subnetid* は IPv6 アドレスでなければならない。 *prefix-length* は 128 未満の正の整数でなければならない。

DHCPv6 ネストありコンテナー・ステートメント
ネストありコンテナー・ステートメントは、サブネットの内部でオプションとしてのみ存在可能です。

すべてのコンテナーは、特に注記がない限り、その内部に他のコンテナーをネストできます。 ネストの最大の深さは、サブネットおよびグローバル・コンテナーを含めて 7 です (サブネット・コンテナーの下には 5 つまでのネストありコンテナーを置くことができます)。

ベンダーおよび着信オプション・コンテナーは、他のコンテナーをネストすることはできません。

表 67. ネストありコンテナー・ステートメントで入力するキーワード、値、および説明		
キーワード	**値**	**説明**
class	*name* [*range*] {OPTIONS| COMMON OPTIONS }	クラス・コンテナー。 *name* 値は 1 つのストリングで、スペースで区切られる。正規表現は hex 0x*hexdigit*、0x*hexdigit*。
vendor	*name* [*range*] {OPTIONS| COMMON OPTIONS }	ベンダー・コンテナー。 *name* 値は 1 つのストリングで、スペースで区切られる。正規表現は hex 0x*hexdigit*、0x*hexdigit*。

表 67. ネストありコンテナー・ステートメントで入力するキーワード、値、および説明 (続き)		
キーワード	**値**	**説明**
client	<*id* | 0 0xhexdigit | regular expression> <ip | *range* | none | any> {OPTIONA|COMMON OPTIONS }	クライアント・コンテナー。 *id*- 1-hexdigit、2-hexdigit、3-hexdigit <ip|range| none|any> - ID と一致するクライアントに指定する IP アドレス
inoption	*icode keytomatch* [*range*] { OPTIONS|COMMON OPTIONS }	着信オプション・コンテナー。 *icode* - クライアントが指定する着信オプション・コードまたは番号 *keytomatch* - 対応して一致するオプション・データ。

DHCPv6 cnf ファイル・オプション
ここで説明する **DHCPv6** に対する cnf ファイル・オプションは、コンテナー内でのみ存在できます。

表 68. オプション・スタンザで入力するキーワード、値、および説明		
キーワード	**値**	**説明**
exclude	*range*	現行の範囲から除外する IP 範囲。通常､範囲がコンテナー・ステートメントの一部として指定されていない場合に使用される。
exclude	ip	現行の範囲から除外する IP アドレス
range	*range*	現行の範囲を拡張するのに使用する IP 範囲。通常、範囲がコンテナー・ステートメントの一部として指定されていない場合に使用される。
range	ip	追加する IP アドレス。範囲を拡張するのに使用される
stealfromchildren	*policy*	すべてのアドレスを使い果たした場合に、子コンテナーからアドレスを流用する。 デフォルトではオフ。
stealfrompeer	*policy*	すべてのアドレスを使い果たした場合に、ピア・コンテナーからアドレスを流用する。 デフォルトではオフ。
stealfromparent	*policy*	すべてのアドレスを使い果たした場合に、親コンテナーからアドレスを流用する。 デフォルトではオフ。
balance-option	{ balance-policy | <option | option option ...> }	バランス・オプション・コンテナー。このコンテナー内で指定されるオプションは、ポリシー上のクライアント・ベースに指定される。 このキーワードは、サブネット・コンテナーの下にのみ存在できる。
balance-policy	b_policy	b_policy 値は fill または rotate。 デフォルトは rotate。
fill-count	num	オプションが、その同じオプションの次のインスタンスを配布する前に分配される回数。

表 68. オプション・スタンザで入力するキーワード、値、および説明 (続き)		
キーワード	**値**	**説明**
interface-id	"*interface*"	これはサブネットの下にのみリストされる。 このインターフェース 上で受け取られるクライアント要求は、アドレスを入手するために許可される。

DHCPv6 共通オプション
以下のキーワードは、**DHCPv6** 共通オプションです。

これらは、コンテナーの内部、またはグローバル・セクションに存在できます。

表 69. 共通オプションのキーワード、値、および説明		
キーワード	**値**	**説明**
reconfig-policy	*policy*	サーバーが、再構成メッセージをクライアントに送信するのを許可する。 デフォルトでは未設定で、オフと見なされる。
rapid-commit	*policy*	サーバーが、コンテナーに対して高速コミットを実行するか、またはグローバルに設定されるのを許可する。 デフォルトでは未設定で、オフと見なされる。
prefered-lifetime	num [*units*]	IANA または IATA の推奨存続時間。 デフォルトは 43200 秒。
valid-lifetime	num [*units*]	IANA または IATA の有効な存続時間。 デフォルトは 86400 秒。
rebind	num	アドレスのための再バインド時間のパーセント 0-100。 デフォルト値は 80 パーセント。
renew	num	アドレスのための更新時間のパーセント 0-100。 デフォルト値は 50 パーセント。
unicast-option	policy	コンテナーが、ユニキャストによってメッセージ交換を提供するのを許可する。これは、サーバー・ポリシーが異なる場合でも個々のコンテナーとサブネットをオンにしたりオフにしたりするのに使用できる。 デフォルトでは未設定で、オフと見なされる。
option	num <string| stings| hex>	オプションのリストについては、320 ページの『DHCPv6 サーバー・ファイルの既知のオプション』を参照。
change-optiontable	optiontable	ベンダー・コンテナー内でのみ許可される。

DHCPv6 サーバー・ファイルの既知のオプション
ここでは、**DHCPv6** サーバー・ファイルの既知のオプションについて説明します。

次のオプションは、**DHCPv6** サーバー・ファイルの既知のオプションです。 「**指定可能?**」列が「いいえ」となっているオプションは、構成ファイルで指定することができません。指定されても無視されます。

オプション番号	デフォルト・ データ・タイプ	指定可能?	説明
1	なし	いいえ	請求
2	なし	いいえ	通知
3	なし	いいえ	要求

オプション番号	デフォルト・ データ・タイプ	指定可能?	説明
4	なし	いいえ	確認
5	なし	いいえ	アドレス
6	なし	いいえ	オプション要求
7	数値	いいえ	サーバーの優先回数
8	なし	いいえ	経過時間
9	なし	いいえ	リレー・メッセージ
11	なし	いいえ	Auth
12	ASCII ストリング yes、no、true、false	はい	ユニキャスト
13	なし	いいえ	状況
14	ASCII ストリング yes、no、true、false	はい	高速コミット
15	なし	いいえ	ユーザー・クラス
16	なし	いいえ	ベンダー・クラス
17	なし	いいえ	ベンダー・オプション
18	なし	いいえ	インターフェース ID
19	なし	いいえ	再構成メッセージ
20	ASCII ストリング yes、no、true、false	はい	再構成受信
23	スペースで区切った IPv6 アドレス	はい	DNS サーバー
24	ASCII ストリング	はい	ドメイン・リスト

DHCPv6 パラメーター値
以下の値は **DHCPv6** パラメーターに使用できます。

units: second、 seconds、 minute、 minutes、 hour、 hours、 day、 days、 week、 weeks、 month、 months、 year、 years

interface: en0、en1、tr0

identifier: 数値または文字

policy: yes、no、true、false

range: ipv6addresss-ipv6addresss

regular expression: "!epression to match$"、"!epression to match^"

/etc/dhcpv6/dhcpsdv6.cnf ファイルの例
以下の `/etc/dhcpv6/dhcpsdv6.cnf` ファイルの例は、ファイル内容の一部を示します。

```
logging_info{
        logFileSize 4000
        logItem     SYSERR
        logItem     PROTERR
        logItem     WARNING
        logItem     EVENT
        logItem     ACTION
        logItem     INFO
```

```
        logItem     ACNTING
        logItem     TRACE
        numLogFiles     3
        logFileName "/var/tmp/dhcpsdv6.log"
}
duid 1 en0
numprocessthreads 10
numpacketthreads  5
preference-number 255
reconfig-policy no
rapid-commit  no
unicast-option    yes
leaseExpiredInterval  3000 seconds
unicast-enable          yes
saveInterval            60 seconds
reservedTimeInterval    8000 seconds
reservedTime            10000 seconds
clientpruneintv         20  seconds

subnet bbbb:aaaa:: 40 bbbb:aaaa::0004-bbbb:aaaa::000f {
        balance-option {
               option 23 dead::beef
               option 23 beef::aaaa
               option 24 yahoo.com
        }

subnet dead:dead:aaaa:: 48 dead:dead:aaaa:aaaa::0006-dead:dead:aaaa:aaaa::000a {
        interface-id "en1"
        preferred-lifetime      100 seconds
        valid-lifetime          200 seconds
        rapid-commit  yes
        option 23 dead::beef beef:aaaa::bbbb:c aaaa:bbbb::cccc
        option 24 ibm.com austin.ibm.com
}
```

DHCPv6 クライアント構成

/etc/dhcpv6/dhcpc6.cnf ファイルは、**DHCPv6** クライアントを構成するのに使用されます。

このファイルで指定可能な指示が、ここに含まれています。 **dhcpcd6** がリブートで開始する場合、/etc/rc.tcpip ファイルにエントリーを追加します。

ロギング・キーワード

ここでは、**DHCPv6** サーバーの有効なロギング・キーワードについて説明します。

有効なキーワードは次のとおりです。

表 *70.* ロギング・キーワード用のキーワードと説明

キーワード	説明
log-file-name	最新のログ・ファイルのパスおよびファイル名。 最新のものより前のファイル名には、1 が追加される (n-1)。この数値が大きいほど古いファイルになる。
log-file-size	ログ・ファイルの最大サイズを KB で指定する。 最新のログ・ファイルのサイズがこの値に達すると、名前が変更され、新しいファイルが作成される。
log-file-num	最新のログ・ファイルのサイズが log-file-size 値に達し、ファイルの名前が変更されて、新しいファイルが生成されるときに保守されるログ・ファイルの最大数を指定する。

表 70. ロギング・キーワード用のキーワードと説明 (続き)	
キーワード	**説明**
log-item	ログ記録する必要のあるログ項目を指定する。 **SYSERR** システム・エラー **OBJERR** オブジェクト・エラー **PROTERR** プロトコル・エラー **WARNING** 警告 **EVENT** 発生したイベント **ACTION** プロセスが実行するアクション **INFO** 追加情報 **ACNTING** サービスを受ける対象とその時期 **TRACE** コード・フロー、デバッグ

DUID キーワード

以下のキーワード値は、DUID エントリーの値です。

DUID エントリーのフォーマットは次のとおりです。

```
duid  <duid_type> <value>  <value> ...
```

DUID タイプはキーワードまたは数値で、今後定義される可能性のある DUID タイプのための余地を残します。 現在、RFC 3315 によって定義されている DUID タイプは次の 3 つです。

表 71. *DUID* エントリー用のキーワードおよび値	
キーワード	**説明**
LLT	DUID-LLT タイプ (値 1)
LL	DUID-LL (値 2)
EN	DUID-EN タイプ (値 3)

DUID エントリーの特定のフォーマットは、使用されているキーワードによって異なります。

```
duid  LLT          <interface name>
duid  LL           <interface name>
duid  EN           <enterprise number>  <enterprise identifier>
duid  <number>     <hex data (prefixed with '0x')
```

情報専用キーワード

情報専用キーワードの形式は、`info-only` *`interface name`* です。

情報専用キーワードを次に示します。

表 72. 情報専用キーワードのキーワードと説明	
キーワード	**説明**
info-only *interface name*	このキーワードは、クライアントがサーバーからのアドレスではなく、構成情報のみを入手する必要のあるインターフェース 名を指定します。

貸出更新および再バインド・キーワード

ここでは、**DHCPv6** サーバーの貸出更新および再バインド・キーワードについて説明します。

表 73. 貸出更新および再バインド・キーワードのキーワードと説明	
キーワード	**説明**
rebind-time *value*	(サーバーが応答しなかったため) クライアントが貸出の更新に失敗した場合に、rebind-time はクライアントが他のサーバーと連絡をとり、貸出を再バインドする時間を指定する。
renew-time *value*	renew-time は、クライアントが貸出を更新するために貸出情報を入手した先のサーバーと、クライアントが接触する時間を指定する。

請求再送キーワード

請求再送キーワードには、solicit-maxcount と solicit-timeout があります。

表 74. 請求再送キーワードのキーワードと説明	
キーワード	**説明**
solicit-maxcount	solicit-maxcount キーワードは、クライアントが、サーバーからの応答を受信する前に、サーバーに対して送信する請求メッセージの数を指定します。
solicit-timeout	solicit-timeout キーワードは、クライアントが、サーバーからの応答を受信する前に、サーバーに対する請求メッセージの送信を試みるまでの時間を指定します。

オプション・キーワード

オプション・キーワードが「インターフェース」スタンザの外部に表示される場合、それらのキーワードはグローバルと見なされます。 このようなオプションは、すべてのインターフェースに 適用されます。 オプション・キーワードが「インターフェース」スタンザ内に表示される場合、これらのオプションはそのインターフェースにのみ 適用されます。

オプション・スタンザのフォーマットは次のとおりです。

```
option <keyword | option code>
option <keyword | option code> exec "exec string"
option <keyword | option code> { option specific parameters }
option <keyword | option code> { option specific parameters } exec "exec string"
```

オプション・コードは、IANA 登録オプション・コードを使用して指定できます。 しかし、オプションの中には、次のキーワードを使って指定できるものもあります。

キーワード	オプション・コード
ia-na	3
ia-ta	4
request-option	6

キーワード	オプション・コード
rapid-commit	14
user-class	15
vendor-class	16
vendor-opts	17
reconf-accept	20
dns-servers	23
domain-list	24

各キーワードの付加的な説明は次のとおりです。

キーワード	目的、フォーマット、およびパラメーター
ia-na	**目的** オプション 3 を指定する。指定されると、クライアントはサーバーから 非一時アドレスを要求する。 **フォーマット** option ia-na [{ *parameters* }] [exec "exec string"] **パラメーター** オプション ia-na は次のパラメーターを使用する。 ia-id *value* renew-time *value* rebind-time *value* これらのパラメーターは、ユーザーにとって望ましい値を指定し、オプションである。指定される *value* (値) は 10 進数、または '0x' が最初に付けられる 16 進数。
ia-ta	**目的** オプション 4 を指定する。指定されると、クライアントはサーバーから 一時アドレスを要求する。 **フォーマット** option ia-ta [{ *parameters* }] [exec "exec string"] **パラメーター** オプション ia-ta は次のパラメーターを使用する。 ia-id *value* このパラメーターは、ユーザーにとって望ましい値を指定し、オプションである。 指定される *value* (値) は 10 進数、または '0x' が最初に付けられる 16 進数。
request-option	**目的** オプション 6 を指定する。指定されると、クライアントはサーバーからオプションのリストを要求する。 **フォーマット** option request-option { *parameters* } [exec "exec string"] **パラメーター** オプション request-option は、オプション・コード (10 進数) のスペースで区切ったリストを引数として使用する。

キーワード	目的、フォーマット、およびパラメーター
rapid-commit	**目的** オプション 14 を指定する。 指定されると、クライアントは、クライアントが Solicit-Reply メッセージ交換を実行する準備ができたことを示す。 **フォーマット** option rapid-commit [exec "exec string"] **パラメーター** オプションの exec ステートメント以外のパラメーターは使用しない。
user-class	**目的** オプション 15 を指定する。 指定されると、クライアントは、それが表現するユーザーまたはアプリケーションのタイプまたはカテゴリーを 示す。 **フォーマット** option user-class { *parameters* } [exec "exec string"] **パラメーター** オプション user-class は、ユーザー・クラス・データの 1 つ以上のインスタンスを使用する。 ユーザー・クラス・データの各インスタンスは、任意の長さのストリングで、引用符で囲まれていてもいなくてもよい。 ストリングにブランク・スペースが入っている場合は、引用符で囲む必要がある。 パラメーターは必須。 パラメーターのフォーマットは次のとおり。 class *value* class *value* *value* (値) は引用符で囲まれているか、または囲まれていないストリング。
vendor-class	**目的** オプション 16 を指定する。 指定されると、クライアントはクライアントが 実行しているハードウェアを製造したベンダーを示す。 **フォーマット** option vendor-class { *parameters* } [exec "exec string"] **パラメーター** オプション vendor-class は、ベンダーの登録済み企業番号およびベンダー・クラス・データの 1 つ以上のインスタンスを使用する。 ベンダー・クラス・データの各インスタンスは、任意の長さのストリングで、引用符で囲まれていてもいなくてもよく、それぞれがクライアントのハードウェア 構成のいくつかの特性を説明する。 パラメーターはオプションではない。 フォーマットは次のとおりです。 vendor-id *value* class *value* class *value* *value* (値) は引用符で囲まれているか、または囲まれていないストリング。

キーワード	目的、フォーマット、およびパラメーター
vendor-opts	**目的** オプション 17 を指定する。 指定されると、クライアントは、サーバーに対してベンダー固有の情報を示す。 **フォーマット** option vendor-opts <*enterprise-number*> { *parameters* } [exec "exec string"]] **パラメーター** オプション vendor-opts は、ベンダーの登録済み企業番号およびベンダー・オプション・データの 1 つ以上のインスタンスを使用する。 ベンダー・オプション・データの各インスタンスは、ベンダー・オプション・コードにストリングまたは 16 進フォーマットでオプション・データが続くものである。 パラメーターはオプションではない。 フォーマットは次のとおりです。 vendor-id *value* option *opcode option-data* option *opcode option-data* *option-data* は、引用符で囲まれるか、または囲まれないストリング、あるいは 16 進ストリング (前に '0x' を付ける)
reconf-accept	**目的** オプション 20 を指定する。 指定されると、クライアントは、サーバーに対して、クライアントがサーバーからの 再構成メッセージを受け入れるかどうかを示す。 **フォーマット** option reconf-accept [{ exec "exec string" }] **パラメーター** オプション reconf-accept は、exec ステートメント以外はオプションに特定のパラメーターを使用しない。
dns-servers	**目的** オプション 23 を指定する。 指定されると、クライアントはサーバーに 対して DNS サーバーの推奨セットを示す。 **フォーマット** option dns-servers [{ *parameters* }] [exec "exec string"] **パラメーター** オプション dns-servers は、IPv6 アドレスのスペースまたは行で区切られたリストを引数として使用する。
domain-list	**目的** オプション 24 を指定する。 指定されると、クライアントは推奨されるドメイン・リストを示す。 **フォーマット** option domain-list [{ *parameters* }] [exec "exec string"] **パラメーター** オプション domain-list は、ドメイン名ストリングのスペースまたは行で区切られたリストを使用する。

インターフェース・キーワード

インターフェース・キーワードの形式は、interface <*interface name*> [{ option declaration/s }] です。

表 75. インターフェース・キーワードのキーワードと説明

キーワード	説明
interface <*interface name*> [{ option declaration/s }]	インターフェース・ステートメントは、1 つ以上の宣言を引数として使用する。 これらのオプションは、すべてのインターフェースに 適用する、インターフェース・スタンザの外部で宣言されるオプションとは異なり、このインターフェースに 特定のものである。

```
interface  en1  {
                option ia-na {
                  ia-id        01
                  renew-time    0x40
                  rebind-time    0x60
    }

    option  request-option  { 3 23 24 }

       option  user-class {
                class ibm
               class "userclassA and B"
               class "userclassB"
        }

      option  vendor-class {
               vendor-id    1234
               class "vendorclassA"
               class "vendorclassB"
      }

      option  vendor-opts {
                vendor-id    2343
                  option 89       vendoroption89
                  option 90       vendoroption90
      }

    option    reconf-accept
```

DHCP リレー・エージェント

/etc/dhcprd.cnf ファイルは、**DHCP** と **BOOTP** リレー・エージェント用の構成ファイルです。 ファイルのフォーマットおよび許可される指示とキーワードをここで説明します。

指示は、次のフォーマットで指定されます。

```
<keyword> <value1> ... <valueN>
```

これらのパラメーターの存在および値は、始動または再始動されるリレー・エージェントにより使用されます。

パラメーターのこのセットは、このサーバーが保守するログ・ファイルを指定します。 各パラメーターは、キーワードにより識別され、その値が続きます。

キーワード	値	定義
numLogFiles	0 から *n*	ログ・ファイルの数。 0 が指定される場合、保守されるログ・ファイルはなく、ログ・メッセージはどこにも表示されない。 *n* は最新のログ・ファイルのサイズが最大サイズに達し、新しいログ・ファイルが作成されるときに保守されるログ・ファイルの最大数。
logFileSize	KB 単位	ログ・ファイルの最大サイズ。 最新のログ・ファイルのサイズがこの値に達すると、名前が変更され、新しいログ・ファイルが作成される。
logFileName	ファイル・パス	最新のログ・ファイルの名前。 最新のものより前のログ・ファイルは、その名前 1 を追加したもの (*n* - 1) になる。この数値が大きいほど古いファイルになる。
logItem	ログ記録される 1 つの項目。	**SYSERR** プラットフォームに対するインターフェースにおけるシステム・エラー。 **OBJERR** プロセス中のオブジェクト間でのオブジェクト・エラー。 **PROTERR** クライアントとサーバーとの間のプロトコル・エラー。 **WARNING** ユーザーが注意を払うべき警告。 **EVENT** プロセスに対して発生するイベント。 **ACTION** プロセスが実行するアクション。 **INFO** 役立つ可能性のある情報。 **ACNTING** サービスを受ける対象とその時期。 **TRACE** デバッグ用のコード・フロー。

例えば、/etc/dhcprd.cnf ファイルのエントリーは次のようなものであることが考えられます。

```
numLogFiles     4
logFileSize     1000
logFileName       /usr/tmp/dhcprd.log
logItem           SYSERR
logItem           OBJERR
logItem           PROTERR
```

```
logItem           WARNING
logItem           EVENT
logItem           ACTION
logItem           INFO
logItem           ACNTING
logItem           TRACE
```

キーワード	値	定義
relay	IPv4、IPv6、または ALL	パケット・リレーのモードを指定する。 IPv4 が指定される場合、リレー・エージェントは **DHCPv4** リレー・エージェントとしてのみ動作する。 これはリレー・エージェントのデフォルト・モードである。 IPv6 が指定される場合、リレー・エージェントは **DHCPv6** リレー・エージェントとしてのみ動作する。 ALL が指定される場合、リレー・エージェントは **DHCPv4** と **DHCPv6** の両方のリレー・エージェントとして動作する。
server	IP アドレス	**BOOTP** または **DHCP** サーバーの IP アドレスを指定する。 このファイルにリストされるサーバーにパケットが転送される。
server6	IPv6 アドレス	**DHCPv6** サーバーの IPv6 アドレスを指定する。 ここにリストされるサーバーにパケットが転送される。
option6	<*option code*> <*option data*>	**DHCPv6** リレー・エージェント・オプションを指定する。 キーワードは、リレー・モードが IPv6 に設定されている場合のみ有効である。 *option code* 値は、10 進数として指定される。 *option data* 値は引用符で囲まれた、または囲まれないストリングか、16 進形式 (0x が前に付く) として指定される。
single-site		リレー・エージェントが実行しているデバイスが、1 つのサイトにしか所属しないことを指定する。

プリブート実行環境プロキシー DHCP デーモン

PXE プロキシー **DHCP** サーバーは、通常の **DHCP** クライアント・トラフィックを listen し、特定のクライアント要求に応答することで、**DHCP** サーバーと同様に動作します。 ただし、**DHCP** サーバーとは異なり、PXE プロキシー **DHCP** サーバーはネットワーク・アドレスを管理せず、PXE クライアントの ID を持つクライアントにのみ応答します。

PXE プロキシー **DHCP** サーバーが渡す応答には、クライアントがブート・サーバーを探し出すときに使用するメカニズム、またはサポートされている互換性のあるブート・サーバーのネットワーク・アドレスと記述が含まれています。

DHCP サーバーのほかに PXE プロキシー **DHCP** サーバーも使用した場合、3 つの主要機能が使用可能になります。 1 つ目として、ネットワーク・アドレスの管理とブート・イメージの管理を分離することができます。 同じシステム上で 2 つの異なるプロセスを使用すると、**DHCP** サーバー構成へのアクセスを妨害したり、要求したりすることなく、PXE プロキシー **DHCP** サーバーによって管理されるブート情報を構成することができます。 2 つ目として、複数のブート・サーバーを定義して、ブート時に PXE クライアントが特定のサーバーを選択できるようにすることができます。 例えば、各ブート・サーバーは、異なるタイプのオペレーティング・システムまたはシステム 構成を提供することができます。 3 つ目として、プロキシー・サーバーを使用することにより、互換性のあるブート・サーバーの場所を探し出すためにマルチキャスト IP アドレッシングを使用できるように PXE クライアントを構成することが可能となります。

PXE プロキシー **DHCP** サーバーは、**DHCP** サーバーが稼働中の同じシステムで稼働するように構成することも、異なるシステムで稼働するように構成することもできます。 また、ブート・サーバー・デーモンを稼働中のシステムで稼働するようにそれを構成することも、異なるシステムで稼働するように構成することもできます。

PXE プロキシー DHCP サーバー・コンポーネント

PXED サーバーには 3 つのコンポーネントがあります。

PXED サーバーは 3 つの主要部分 (データベース、プロトコル・エンジン、一連のサービス・スレッド) に区分されており、それぞれの部分に構成情報があります。

PXED データベース

`db_file.dhcpo` データベースは、クライアントが REQUEST パケットを送信したときにクライアントに送られるオプションの生成に使用されます。

データベースから戻されるオプションは、選択したサーバーのタイプによって異なります。 オプションは、`pxed.cnf` ファイル内の `pxeservertype` キーワードを使用して設定します。

このデータベースは、構成ファイル内の情報を使用して用意され、整合性を検査されます。

PXED プロトコル・エンジン

プロトコル・エンジンは、データベースを使用して、どの情報をクライアントに戻すべきかを判別します。

PXED プロトコル・エンジンは Intel Preboot Execution Environment (PXE) 仕様バージョン 2.1 に基づいており、依然として Intel PXE 仕様バージョン 1.1 との互換性もあります。

PXED のスレッド化操作

PXED サーバーの最後の部分は、実行を続行させるために使われる、一連の操作です。 PXED サーバーはスレッド化されるため、これらの操作は、実際は、すべてが一緒に行われるようにするために時々実行される、スレッドとしてセットアップされます。

最初のスレッドである *main* スレッドは、SRC 要求 (**startsrc**、**stopsrc**、**lssrc**、**traceson**、**refresh** など) を処理します。 また、このスレッドは、すべてのスレッドに影響するすべての操作を調整し、シグナルを処理します。 例えば、次のとおりです。

- SIGHUP (-1) は構成ファイル内のすべてのデータベースを再表示する。
- SIGTERM (-15) はサーバーを正常中止にする。

もう 1 つのスレッドは、パケットを処理します。 サーバーのタイプによって、1 つまたは 2 つのスレッドがあります。 1 つのスレッドがポート 67 を listen し、もう 1 つのスレッドがポート 4011 を listen します。 各スレッドが、クライアントからの要求を処理することができます。

PXED サーバーの構成

PXED サーバーは、デフォルトでは `/etc/pxed.cnf` ファイルを読み取ることによって構成されます。このファイルは、オプションとアドレスの入ったサーバーの初期データベースを指定しています。

サーバーは、SMIT または SRC コマンドのいずれかを使用して始動します。

PXED サーバーの構成は、通常は、ネットワークで PXED を使用する際の最も難しい仕事です。 まず最初に、どのネットワークが PXE クライアントを持つ必要があるかを見極めてください。 次の例では、**pxed** デーモンを、DHCP サーバーと同じマシンで稼働するように構成しています。

```
pxeservertype       proxy_on_dhcp_server

subnet default
{
    vendor pxe
    {
        option    6      2       # Disable Multicast boot server discovery
        option    8      1   2     9.3.4.5    9.3.4.6   2    1    9.3.149.29
                                   # The above option gives the list of bootservers
        option    9      0       "PXE bootstrap server" ¥
                                   1       "Microsoft Windows NT Boot Server"  ¥
                                   2       "DOS/UNDI Boot Server"
        option    10    20    "seconds left before the first item in the boot menu is auto-
selected"
     }
}
```

PXE クライアントの IP アドレスがサブネットの IP アドレス範囲内にある (例えば、9.3.149.0 から 9.3.149.255) 場合のみ、ベンダー・コンテナー内のサブオプションが PXE クライアントに送られます。

次の例では、**pxed** デーモンを、**DHCP** サーバーとは異なるマシンで稼働するように構成しています。

```
subnet default
{
    vendor pxe
    {
        option    6      10      # The bootfile name is present in the client's initial pxed
                                   # offer packet.
        option    8      1   2     9.3.4.5    9.3.4.6   2    1    9.3.149.29
                                   # The above option gives the list of bootservers
        option    9      0       "PXE bootstrap server" ¥
                                   1       "Microsoft Windows NT Boot Server"  ¥
                                   2       "DOS/UNDI Boot Server"
        option    10    20    "seconds left before the first item in the boot menu is auto-
selected"
        bootstrapserver           9.3.148.65
        pxebootfile    1    2    1   window.one
        pxebootfile    2    2    1   linux.one
        pxebootfile    1    2    1   hello.one
        client 6 10005a8ad14d any
        {
           pxebootfile    1    2    1   aix.one
           pxebootfile    2    2    1   window.one
        }
    }

Vendor pxeserver
{
    option     7     224.234.202.202
}
```

pxeservertype キーワードが構成ファイル内に設定されていないため、デフォルト値 **pdhcp_only** が採用されます。この値は、PXED サーバーを **DHCP** サーバーとは異なるマシンで稼働させることを意味します。 この構成により、PXED サーバーは、2 つのポート (67 と 4011) でクライアントの BINLD REQUEST/INFORM パケットを listen します。 PXED サーバーが、BINLD サーバーから REQUEST/INFORM パケットをポート 67 で受信すると、オプション 7 が BINLD サーバーに送られ、オプション 60 が PXED サーバーに設定されます。

db_file database 文節は、この構成ファイル部分を処理するために使用するデータベースのメソッドを指示します。 コメントはポンド記号 (#) で始まります。 PXED サーバーは、# からその行の終わりまでを無視します。 サーバーは、各 option 行を使用して、クライアントに行うべきことを指示します。 現在サポートされ、認識されるオプションについては、335 ページの『PXE ベンダー・コンテナー・サブオプション』に説明があります。 サーバーに既知ではないオプションを指定する方法については、338 ページの『汎用サーバー操作に対する PXED サーバー・ファイルの構文』を参照してください。

PXED 構成ファイル

構成ファイルには、アドレス・セクションとオプション定義セクションとがあります。これらのセクションは、オプション、修飾子、それにおそらくはその他のコンテナーを保持しているコンテナーの概念に基づいています。

コンテナー (基本的には、オプションをグループ化するための方法) は、クライアントをグループに 分類するのに ID を使用します。 コンテナーには 4 つのタイプあります。サブネット、クラス、ベンダー、およびクライアントです。 現在、一般的なユーザー定義可能コンテナーはありません。 ID は、例えばクライアントがサブネット間を移動するときにクライアントを追跡できるように、クライアントを固有に定義するものです。 クライアント・アクセスを定義するために、複数のコンテナー・タイプを使用することができます。

オプション は、デフォルトのゲートウェイや DNS アドレスなど、クライアントに戻される ID です。

PXED コンテナー

DHCP サーバーが要求を受信すると、パケットが構文解析され、抽出するコンテナー、オプション、およびアドレスが ID キーによって判別されます。

PXED サーバー構成の例は、サブネット・コンテナーを示しています。 この ID キーは、ネットワーク内のクライアントの位置です。 クライアントがそのネットワークからのものである場合は、そのクライアントはそのコンテナーに入ります。

コンテナーは、タイプごとに、クライアントを識別するための異なるオプションを使用します。

- サブネット・コンテナーは、「`giaddr`」フィールドまたは受信インターフェースのインターフェース ・アドレスを使用して、クライアントがどのサブネットから 発生したものであるかを判別する。
- クラス・コンテナーは、オプション 77 の値 (ユーザー・サイト・クラス ID) を使用する。
- ベンダー・コンテナーは、オプション 60 の値 (ベンダー・クラス ID) を使用する。
- クライアント・コンテナーは、PXE クライアントに対してオプション 61 (クライアント ID)、**BOOTP** クライアントに対して **BOOTP** パケットの `chaddr` フィールドを使用する。

サブネット・コンテナーを除いて、どのコンテナーも、正規表現マッチングを 含む、マッチングによる 値の指定を行うことができます。

暗黙のコンテナーもあります。 グローバル・コンテナーがそれです。 グローバル・コンテナー内のオプションと修飾子は、指定変更されたり、拒否されたりしない限りは、すべてのコンテナーに適用されます。 ほとんどのコンテナーは、可視範囲を暗黙指定して他のコンテナーの中に入れることができます。 コンテナーは、関連するアドレス範囲を持つことも、持たないこともあります。 サブネットは、その性質上それに関連した範囲があります。

コンテナーとサブコンテナーの基本規則は、次のとおりです。

- グローバル・レベルではどのコンテナーも有効である。
- サブネットは、他のコンテナーの中に入れることはできない。
- 制限付きコンテナーの中には、同じタイプのレギュラー・コンテナーを入れることはできない。 (例えば、`Accounting` クラスだけが許されたオプションのコンテナーに、文字「`a`」で始まるすべてのクラスが許されたオプションのコンテナーを 入れることはできません。 これは違反です。)
- 制限付きクライアント・コンテナーはサブコンテナーを持てない。

上記の規則に従えば、コンテナーの階層を作成することができます。この階層によって、オプションは、特定のクライアントまたは 1 組のクライアントのグループに 分けられます。

クライアントが複数のコンテナーに合致する場合は、オプションとアドレスはどのように分配されるのでしょうか。 **DHCP** サーバーは、メッセージを受信し、要求をデータベース (この場合は `db_file`) に渡します。そこでコンテナー・リストが作成されます。 このリストは、奥行きと優先順位の順に並べられます。 優先順位は、コンテナー内の暗黙の階層として定義されます。 ストリクト・コンテナーは、レギュラー・コンテナーよりも高い優先順位を持ちます。 クライアント、クラス、ベンダー、最後にサブネットがこの順に、また、それぞれのコンテナー・タイプ内では奥行きの順に記載されます。 すなわち、特定度の最も高いものから特定度の低いものの順に並べられたリストができあがります。 例えば、次のとおりです。

```
Subnet 1
--Class 1
--Client 1
Subnet 2
--Class 1
----Vendor 1
----Client 1
--Client 1
```

この例では、2 つのサブネット Subnet 1 と Subnet 2 が示されています。 クラス名は 1 つで Class 1、ベンダー名も 1 つで Vendor 1、そしてクライアント名が Client 1 の 1 つです。Class 1 と Client 1 は複数の場所で定義されています。これらは異なるコンテナーに入っているので、名前は同じですがその中の値は異なることがあります。 Client 1 がオプション・リストに Class 1 を指定している Subnet 1 からメッセージを **DHCP** サーバーに送ると、**DHCP** サーバーは次のようなコンテナー・パスを生成します。

```
Subnet 1, Class 1, Client 1
```

最も特定度の高いコンテナーが最後にリストされています。 アドレスを取得するには、このリストを階層の逆向きに調べ、最初に使用可能なアドレスを探します。 次に、このリストを階層の前方向に調べ、オプションを入手します。 オプションは、deny がコンテナーに指定されていなければ、以前の値を指定変更します。 また、Subnet 1 には Class 1 と Client 1 が入っていますが、これらは、コンテナーの優先順位に従って並べられます。 同じクライアントが Subnet 2 に入っていて、同じメッセージを送信するのであれば、コンテナー・リストは次のように生成されます。

Subnet 2、Class 1、Client 1 (Subnet 2 レベル)、Client 1 (Class 1 レベル)

Subnet 2 が最初にリストされ、次に Class 1、最後に Subnet 2 レベルの Client 1 の順にリストされます (このクライアント・ステートメントが階層では 1 レベル下にあるためです)。 この階層は、最初のクライアント・ステートメントに合致するクライアントが、Subnet 2 内の Class 1 の Client 1 に合致するクライアントよりも特定度が低いことを示唆しています。

階層内の奥行きによって選択された優先順位が、コンテナーそれ自体の優先順位で取り替えられることはありません。 例えば、同一のクライアントが同じメッセージを出し、ベンダー ID を指定している場合、そのコンテナー・リストは次のようになります。

Subnet 2、Class 1、Vendor 1、Client 1 (Subnet 2 レベル)、Client 1 (Class 1 レベル)

コンテナー優先順位は、クライアント・コンテナーが 1 つ以上のクライアントを定義するための最も特定度の高いコンテナー優先順位であるとする一般的な概念に従っているため、検索パフォーマンスを向上させます。 クラス・コンテナーは、クライアント・コンテナーよりも特定度が低いアドレスを保持しています。ベンダーはさらに特定度が低く、サブネットは特定度が最も低くなっています。

PXED アドレスとアドレス範囲

どのコンテナー・タイプも関連したアドレス範囲を持つことができます。サブネットにはアドレス範囲が必須です。 コンテナー内の各範囲は、親コンテナーの範囲のサブセットでなければならず、他のコンテナーの範囲とオーバーラップしてはなりません。

例えば、あるクラスがサブネット内で定義されており、そのクラスが範囲を持っている場合、その範囲は当該サブネットの範囲のサブセットである必要があります。 また、そのクラス・コンテナー内の範囲は、そのレベルにあるその他の範囲とオーバーラップすることはできません。

範囲は、コンテナーの行で表現し、範囲および除外ステートメントで変更して、あるコンテナーに関連付けられたばらばらのアドレス・セットを使用可能にすることができます。 したがって、サブネットの最初の 10 アドレスと次の 10 アドレスを使用可能にする場合、そのサブネットがサブネット文節でこれらのアドレス範囲を指定することによって、使用するメモリーを削減すると同時に、指定された範囲にないその他のクライアントとアドレス競合が起きるのを避けることができます。

アドレスを選択すると、アドレス範囲を含んでいるリスト内の後続のコンテナーは、その子とともにリストから除去されます。 これは、除去されたコンテナー内のネットワーク固有のオプションは、アドレスがそのコンテナーから使用されたものでなければ有効ではないからです。

PXED 構成ファイル・オプション
アドレスを判別するためにリストが選別された後、1 組のオプションがクライアントのために 生成されます。

この選択プロセスでは、*deny* が検出されなければ、以前選択されたオプションが上書きされます。検出された場合は、否定されたオプションがクライアントに 送られるリストから除去されます。 このメソッドによって、親コンテナーからの継承で、指定しなければならないデータ量が削減されます。

PXED のロギング
ロギング・パラメーターは、データベースのようなコンテナーに指定しますが、コンテナーの場合のキーワードは `logging_info` です。

PXED の構成方法を学んでいる間は、ロギングのレベルを最高レベルに設定しておくことをお勧めします。また、ロギング・サブシステムの 初期化後に構成エラーが確実に記録されるようにするために、他の構成ファイル・データのどれよりも先にロギング構成を指定することが最善です。 `logitem` キーワードは、あるロギング・レベルをオンにするために使用します。ロギング・レベルを使用不可にするには `logitem` キーワードを除去してください。 ロギングのためのその他のキーワードを使用すると、ログのためのファイル名、ファイルのサイズ、回転ログ・ファイル数を指定することができます。

PXED パフォーマンスの考慮事項
特定の構成キーワードおよび構成ファイルの構造が PXED サーバーのメモリー使用量とパフォーマンスに影響を与えるということを理解することが重要です。

まずはじめに、親コンテナーから子コンテナーへと継承されるオプションの継承モデルを理解することにより、過度のメモリー使用を回避することができます。 リストされていないクライアントは一切サポートされない環境では、管理者は当該のファイルに各クライアントを明示的にリストする必要があります。 どのクライアントについても 個々にオプションがリストされている場合、サーバーは、その構成ツリーを保管するために、オプションが親コンテナー (例えば、サブネット・コンテナー、ネットワーク・コンテナー、またはグローバル・コンテナーなど) から継承される場合よりも多くのメモリーを使用します。 したがって、管理者は、構成ファイル内においてクライアント・レベルで繰り返し現れるオプションがあるかどうかを検査し、そのようなオプションがある場合には、それらのオプションをひとまとめにして親コンテナーの中に指定して一群のクライアントで共有することができるかどうかを判別する必要があります。

また、`logItem` エントリー INFO および TRACE を使用していると、PXE クライアントのメッセージを 1 つ処理するたびに大量のメッセージがログに記録されます。 ログ・ファイルに 1 行を追加する操作が高くつく可能性があります。したがって、ロギングの量を制限すれば PXED サーバーのパフォーマンスが向上します。 PXED サーバーにエラーが発生している疑いがある場合は、SRC の **traceson** コマンドを使用して、ロギングを動的に再度使用可能にすることができます。

PXE ベンダー・コンテナー・サブオプション
PXE クライアントをサポートする 場合、**DHCP** サーバーは BINLD が自己構成に使用する次のオプションを BINLD サーバーに渡します。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明
6	10 進数	はい	PXE_DISCOVERY_CONTROL。 0 から 16 に制限。 これはビット・フィールド。 ビット 0 は最下位ビット。 **ビット 0** 設定されていると、ブロードキャスト・ディスカバリーが使用不可になる。 **ビット 1** 設定されていると、マルチキャスト・ディスカバリーが使用不可になる。 **ビット 2** 設定されていると、PXE_BOOT_SERVERS に設定されているサーバーのみを使用する/受け入れる。 **ビット 3** 設定され、初期 PXED オファー・パケット内にブート・ファイル名が存在していると、そのブート・ファイルをダウンロードする (プロンプト表示/メニュー表示/ブート・サーバーの検出をしない)。 **ビット 4-7** 0 に設定しなくてはならない。このオプションが設定されていないと、クライアントは、すべてのビットを 0 と見なす。
7	1 つのドット付きクワッド	はい	マルチキャスト IP アドレス。 ブート・サーバー・ディスカバリー・マルチキャスト IP アドレス。 マルチキャスト・ディスカバリー機能のあるブート・サーバーは、このマルチキャスト・アドレスを listen しなければならない。 PXE_DISCOVERY_ CONTROL オプションのマルチキャスト・ディスカバリー使用不可化ビット (ビット 1) が設定されていない場合、このオプションが必要。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明
8	ブート・サーバー・タイプ *(0* から *65535)*	はい	PXE_BOOT_SERVERS *IP* アドレス・カウント *(0* から *256)* **タイプ 0** Miscrosoft Windows *IP* アドレス...*IP* アドレス NT Boot Server ブート・サーバー・タイプ *IP* アドレス **タイプ 1** Intel LCM Boot Server カウント *IP* アドレス ... **タイプ 3** DOS/UNDI Boot Server *IP* アドレス **タイプ 4** NEC ESMPRO Boot Server **タイプ 5** IBM WSoD Boot Server **タイプ 6** IBM LCCM Boot Server **タイプ 7** CA Unicenter TNG Boot Server **タイプ 8** HP OpenView Boot Server **タイプ 9 から 32767** 予約済み **タイプ 32768 から 65534** ベンダー使用 **タイプ 65535** PXE API Test Server サーバー・タイプの *IP* アドレス・カウントがゼロの場合、クライアントは、そのタイプのあらゆるブート・サーバーからオファーを受けることができる。 ブート・サーバーは、サポートしないタイプのディスカバリー要求には応じない。
9	ブート・サーバー・タイプ *(0* から *65535)*	はい	PXE_BOOT_MENU「記述」ブート・サーバーのブート「順序」は、暗黙タイプ。 「記述」 ...メニュー順序。

オプション番号	デフォルト・データ・タイプ	指定可能?	説明
10	秒単位のタイムアウト値 *(0 から 255)*	はい	PXE_MENU_PROMPT "プロンプト" ・タイムアウトは、最初のブート・メニュー項目を自動選択するまでの待ち時間 (秒数)。 クライアント・システムでは、プロンプトに続いて、ブート・メニューの最初の項目が自動選択されるまでの残り時間 (秒数) が表示される。 クライアント・システムで F8 キーを押すと、メニューが表示される。 このオプションがクライアントに渡されると、プロンプトおよびタイムアウトなしでメニューが表示される。 タイムアウトが 0 の場合、メニューの最初の項目が自動選択される。 タイムアウトが 255 の場合、自動選択またはタイムアウトなしでメニューとプロンプトが表示される。

汎用サーバー操作に対する PXED サーバー・ファイルの構文
ここでは、汎用サーバー操作に対する **DHCPv6** サーバーの PXED サーバー・ファイル・キーワードについて説明します。 構文の形式、サブコンテナー、デフォルト値、および意味を示します。

注 : 次の表の中の時間の単位 (*time_units*) は、オプションであり、実際の時間に対する修飾子を表しています。 デフォルトの時間単位は分です。 有効な値は、秒 (1)、分 (60)、時間 (3600)、日 (86400)、週 (604800)、月 (2392000) および年 (31536000) です。 ここで、括弧内の数字は、指定値 *n* を秒数で表現したい場合に、乗数として使用する値です。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
database	database *db type*	はい	なし	アドレス・プール、オプション、およびクライアント・アクセス・ステートメントのための定義を保持する 1 次コンテナー。 *db type* は、ファイルのこの部分を処理するためにロードされるモジュールの名前。 現在使用できる値は、`db_file` だけである。
logging_info	logging_info	はい	なし	ロギング・パラメーターを定義する 1 次ロギング・コンテナー。
logitem	logitem NONE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem SYSERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem OBJERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem PROTOCOL	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
logitem	logitem PROTERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem WARN	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem WARNING	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem CONFIG	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem EVENT	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem PARSEERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem ACTION	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem ACNTING	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem STAT	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem TRACE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem RTRACE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
logitem	logitem START	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される
numLogFiles	numLogFiles *n*	いいえ	0	作成するログ・ファイル数を指定する。 ログは、最初のファイルがいっぱいになると、次のファイルに回転する。 *n* は作成するファイル数。
logFileSize	logFileSize *n*	いいえ	0	各ログ・ファイルのサイズを 1024 バイト単位で指定する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
logFileName	logFileName *path*	いいえ	なし	最初のログ・ファイルへのパスを指定する。 最初のログ・ファイルの名前は、*filename* または *filename.extension* である。 *filename* は 8 文字以内でなければならない。 ファイルが次に回転するとき、その名前は、ベースの *filename* で始まり、その後ろに番号を追加するか、または拡張子を番号で置き換えたものになる。 例えば、もとのファイル名が `file` である場合、回転後のファイル名は、`file01` となる。 また、もとのファイル名が `file.log` である場合は、`file.01` となる。
pxeservertype	pxeservertype *servertype*	いいえ	dhcp_only	**dhcpsd** サーバーのタイプを示す。 *servertype* に設定できる値は、PXED が **DHCP** サーバーと同じマシンで稼働している場合は、ポート 4011 でのみ PXE クライアント要求を listen していることを意味する `proxy_on_dhcp_server`、PXED が別のマシンで稼働している場合は、ポート 67 と 4011 でクライアント・パケットを listen しなければならないことを意味するデフォルト値 `pdhcp_only` のいずれか。

db_file データベースに対する PXED サーバー・ファイルの構文

ここでは、`db_file` データベースに対する PXED サーバー・ファイルの構文について説明します。 構文の形式、サブコンテナー、デフォルト値、および意味を示します。

注 :

1. 次の表の中の時間の単位 (*time_units*) は、オプションであり、実際の時間に対する修飾子を表しています。 デフォルトの時間単位は分です。 有効な値は、秒 (1)、分 (60)、時間 (3600)、日 (86400)、週 (604800)、月 (2392000) および年 (31536000) です。 ここで、括弧内の数字は、指定値 *n* を秒数で表現したい場合に、乗数として使用する値です。
2. 1 つのコンテナー内に指定されている項目は、サブコンテナー内で指定変更されることがあります。 例えば、グローバルに **BOOTP** クライアントを定義する一方で、supportBootp キーワードを両方のコンテナーに指定してある一定のサブネット内で **BOOTP** クライアントをサポートできます。
3. クライアント、クラス、およびベンダーの各コンテナーは、正規表現のサポートを受けることができます。 クラスとベンダーでは、引用符の後ろの最初の文字を感嘆符 (!) にした引用符付き文字列は、文字列の残りを正規表現として処理されるように指示します。 クライアント・コンテナーでは、「**hwtype**」

フィールドと「**hwaddr**」フィールドの両方で正規表現を使用することができます。 両方のフィールドを表すために、単一の文字列が次のフォーマットで使用されます。

```
decimal_number-data
```

decimal_number がゼロの場合は、data は ASCII 文字列です。 ゼロ以外の場合は、data は 16 進数です。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
subnet	subnet default	はい	なし	範囲なしのサブネットを指定する。 サブネットは、サーバーがクライアントからの INFORM パケットに応答するときのみ使用される。
subnet	subnet *subnet id netmask*			サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
subnet	subnet *subnet id netmask range*			サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。
subnet	subnet *subnet id netmask label*:*priority*			サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
subnet	subnet *subnet id netmask range label*:*priority*			サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。
subnet	subnet *subnet id range*	はい	なし	ネットワーク・コンテナー内で使用されるサブネットを指定する。 これは、オプションの範囲が指定されない限り、サブネット全体のアドレス範囲を定義する。 このサブネットに関連するネットマスクは、周辺のネットワーク・コンテナーから取得される。 **注 :** このメソッドは、使用すべきではありません。 他のサブネット・フォーマットを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
option	option *number data ...*	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 オプションの * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
option	option *number*deny	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 オプションの * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
option	option * deny	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 オプションの * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。
exclude	exclude *IP address*	いいえ	なし	除外ステートメントが入っているコンテナーの範囲を変更する。 除外ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 除外ステートメントは、指定されたアドレスまたは範囲をコンテナーの現在の範囲から除去する。 除外ステートメントを使用すると、サブネットまたはその他のコンテナーに不連続な範囲を作成することができる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
exclude	exclude *dotted_quad-dotted_quad*	いいえ	なし	除外ステートメントが入っているコンテナーの範囲を変更する。 除外ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 除外ステートメントは、指定されたアドレスまたは範囲をコンテナーの現在の範囲から除去する。 除外ステートメントを使用すると、サブネットまたはその他のコンテナーに不連続な範囲を作成することができる。
range	range *IP_address*	いいえ	なし	範囲ステートメントが入っているコンテナーの範囲を変更する。 範囲ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 range が、コンテナー定義行で範囲を指定していないコンテナーの最初のものである場合は、そのコンテナーのための範囲は、この範囲ステートメントの指定した範囲になる。 最初の range の後ろにある範囲ステートメント、または、その定義に範囲を指定しているコンテナーのための範囲ステートメントは、現在の範囲に追加される。 範囲ステートメントを使用すると、範囲にアドレスを 1 つ、または 1 組追加することができる。 範囲は、サブネット・コンテナー定義の中に収まる必要がある。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
range	range *dotted_quad-dotted_quad*	いいえ	なし	範囲ステートメントが入っているコンテナーの範囲を変更する。 範囲ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 range が、コンテナー定義行で範囲を指定していないコンテナーの最初のものである場合は、そのコンテナーのための範囲は、この範囲ステートメントの指定した範囲になる。 最初の range の後ろにある範囲ステートメント、または、その定義に範囲を指定しているコンテナーのための範囲ステートメントは、現在の範囲に追加される。 範囲ステートメントを使用すると、範囲にアドレスを 1 つ、または 1 組追加することができる。 範囲は、サブネット・コンテナー定義の中に収まる必要がある。
client	client *hwtype hwaddr* NONE	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
client	client *hwtype hwaddr* ANY	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。
client	client *hwtype hwaddr dotted_quad*	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
client	client *hwtype hwaddr range*	はい	なし	*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。
class	class *string*	はい	なし	名前が *string* であるクラス・コンテナーを指定する。 文字列には引用符を付けても付けなくてもよい。 引用符を付ける場合は、比較の前に引用符は除かれる。 スペースまたはタブを含む文字列には、引用符が必要である。 このコンテナーは、どのレベルでも有効である。 このクラスのクライアント分配する 1 組のアドレスを指示するために、範囲を指定することができる。 範囲は、ドット付きクワッドの IP アドレスを 1 つで、またはドット付きクワッドの IP アドレスを 2 つをダッシュで区切って示す。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
class	class *string range*	はい	なし	名前が *string* であるクラス・コンテナーを指定する。 文字列には引用符を付けても付けなくてもよい。 引用符を付ける場合は、比較の前に引用符は除かれる。 スペースまたはタブを含む文字列には、引用符が必要である。 このコンテナーは、どのレベルでも有効である。 このクラスのクライアント分配する 1 組のアドレスを指示するために、範囲を指定することができる。 範囲は、ドット付きクワッドの IP アドレスを 1 つで、またはドット付きクワッドの IP アドレスを 2 つをダッシュで区切って示す。
network	network *network id netmask*	はい	なし	クラス情報を使用したネットワーク ID を指定する (例えば、9.3.149.0 にネットマスク 255.255.255.0 を付けたものは、ネットワーク 9.0.0.0 255.255.255.0 となる)。 このバージョンのネットワーク・コンテナーは、同じネットワーク ID とネットマスクを持つサブネットを保持するために使用される。 範囲を指定した場合は、その範囲内のすべてのアドレスがそのプールに入る。 範囲は、そのネットワーク ID のネットワーク内にある必要がある。 これは、クラス・フル・アドレッシングを使用する。 これは、グローバルまたはデータベースのコンテナー・レベルでだけ有効である。 **注 :** network キーワードは、使用すべきではありません。代わりにサブネット・コンテナーを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
network	network *network id*	はい	なし	クラス情報を使用したネットワーク ID を指定する (例えば、9.3.149.0 にネットマスク 255.255.255.0 を付けたものは、ネットワーク 9.0.0.0 255.255.255.0 となる)。このバージョンのネットワーク・コンテナーは、同じネットワーク ID とネットマスクを持つサブネットを保持するために使用される。 範囲を指定した場合は、その範囲内のすべてのアドレスがそのプールに入る。 範囲は、そのネットワーク ID のネットワーク内にある必要がある。 これは、クラス・フル・アドレッシングを使用する。 これは、グローバルまたはデータベースのコンテナー・レベルでだけ有効である。 **注 :** network キーワードは、使用すべきではありません。代わりにサブネット・コンテナーを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
network	network *network id range*	はい	なし	クラス情報を使用したネットワーク ID を指定する (例えば、9.3.149.0 にネットマスク 255.255.255.0 を付けたものは、ネットワーク 9.0.0.0 255.255.255.0 となる)。 このバージョンのネットワーク・コンテナーは、同じネットワーク ID とネットマスクを持つサブネットを保持するために使用される。 範囲を指定した場合は、その範囲内のすべてのアドレスがそのプールに入る。 範囲は、そのネットワーク ID のネットワーク内にある必要がある。 これは、クラス・フル・アドレッシングを使用する。 これは、グローバルまたはデータベースのコンテナー・レベルでだけ有効である。 **注 :** network キーワードは、使用すべきではありません。代わりにサブネット・コンテナーを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id*	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* hex""			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* hex ""			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* 0xdata			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* ""			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range*			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* hex""			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* hex ""			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* 0xdata			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* ""			ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
inoption	inoption *number option_data*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって 既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"! (感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
inoption	inoption *number option_data range*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって 既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"! (感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual	virtual fill *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 fill は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 rotate は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 sfill と srotate は、fill と rotate と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual	virtual sfill *id id ...*	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 fill は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 rotate は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 sfill と srotate は、fill と rotate と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual	virtual rotate *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 fill は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 rotate は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 sfill と srotate は、fill と rotate と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual	virtual srotate *id* *id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 fill は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 rotate は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 sfill と srotate は、fill と rotate と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
inorder:	inorder: *id id* ...	いいえ	なし	fill ポリシーを持つ仮想サブネットを指定する。このポリシーは、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
balance:	balance: *id id ...*	いいえ	なし	rotate ポリシーを持つ仮想サブネットを指定する。 このポリシーは、次のコンテナーの次のアドレスを使用することを意味する。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
bootstrapserver	bootstrapserver *IP address*	いいえ	なし	**BOOTP** または **DHCP** パケットの受信後、**TFTP** ファイルに対してクライアントが使用すべきサーバーを指定する。 この値は、パケットの **siaddr** フィールドに入る。 これは、どのコンテナー・レベルにも有効である。
giaddrfield	giaddrfield *IP address*	いいえ	なし	応答パケットのための giaddrfield を指定する。 **注 :** この指定は **BOOTP** と **DHCP** のプロトコルには正しくありませんが、ネットワークのデフォルト・ゲートウェイとして **giaddr** フィールドを必要とするクライアントもあります。 giaddrfield は、どのレベルでも有効ですが、このように競合の可能性があるため、クライアント・コンテナー内でのみ使用します。
bootfile	bootfile *path*	いいえ	なし	応答パケットのファイル・セクションで使用するブート・ファイルを指定する。 これは、どのコンテナー・レベルにも指定できる。 着信パケットのファイル・セクションに指定されている項目が、ブート・ファイルおよびホーム・ディレクトリー・ステートメントとどのように対話するかは、bootfile ポリシーで定義する。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
pxebootfile	pxebootfile *System Arch MajorVer MinorVer Bootfilename*	いいえ	なし	クライアントに渡すブート・ファイル名を指定する。 構成ファイル・パーサーは、キーワードの後に続くパラメーターの数が 4 つ未満の場合はエラーを生成し、4 つを超える場合は無視する。このキーワードは、コンテナーだけでしか使用できない。

その他のオプションの詳細については、225 ページの『DHCP サーバー・ファイルの既知のオプション』 および 230 ページの『プリブート実行環境のベンダー・コンテナーのサブオプション』 を参照してください。

ブート・イメージ・ネゴシエーション・レイヤー・デーモン

ブート・イメージ・ネゴシエーション・レイヤー・デーモン (BINLD) サーバーは、PXE (Preboot Execution Environment) クライアントのコンタクトの第 3 ステージです。

DHCP サーバーとの通信により IP アドレスが獲得され、PXE プロキシー DHCP サーバーとの通信によりブート・サーバーのロケーションが獲得されると、そのブート・サーバーにコンタクトし、ブート・イメージをダウンロードするファイル名とロケーションが取得されます。 PXE クライアントは、そのブート・プロセスにおいて複数のファイルを必要とする場合、ブート中に戻ってブート・サーバーと何度か通信をすることができます。

PXE ネットワーク・ブートにおける最終ステージは、ブート・サーバーが提供するブート・イメージのダウンロードです。 TFTP のロケーションとダウンロードするファイル名が、ブート・サーバーにより PXE クライアントに提供されます。

BINLD サーバー・コンポーネント

ここでは、BINLD サーバーの 3 つの主要なコンポーネントについて説明します。

BINLD サーバーは 3 つの主要部分 (データベース、プロトコル・エンジン、一連のサービス・スレッド) に区分されており、それぞれの部分に構成情報があります。

BINLD データベース

クライアントの REQUEST パケットに応答するオプションを生成するために、`db_file.dhcpo` データベースが使用されます。

データベースから戻されるオプションは、選択したサーバーのタイプによって異なります。 オプションは、`binld.cnf` ファイル内の `pxeservertype` キーワードを使用して設定します。

このデータベースは、構成ファイル内の情報を使用して用意され、整合性を検査されます。

BINLD プロトコル・エンジン

プロトコル・エンジンは、データベースを使用して、どの情報をクライアントに戻すべきかを判別します。

PXED プロトコル・エンジンは Intel のプリブート実行環境 (PXE) 仕様バージョン 2.1 に基づいていますが、Intel PXE 仕様バージョン 1.1 との互換性もあります。

BINLD のスレッド化操作

BINLD サーバーの最後の部分は、実行を続行するために使用される一連の操作です。

BINLD サーバーはスレッド化されるため、これらの操作は、すべてが一緒に行われるようにするために時々実行されるスレッドとしてセットアップされます。

最初のスレッドである *main* スレッドは、SRC 要求 (**startsrc**、**stopsrc**、**lssrc**、**traceson**、**refresh** など) を処理します。 また、このスレッドは、すべてのスレッドに影響するすべての操作を調整し、シグナルを処理します。 例えば、次のとおりです。

- SIGHUP (-1) は構成ファイル内のすべてのデータベースを再表示する。
- SIGTERM (-15) はサーバーを正常中止にする。

もう 1 つのスレッドは、パケットを処理します。 サーバーのタイプによって、1 つまたは 2 つのスレッドがあります。 1 つはポート 67、もう 1 つはポート 4011 を listen します。 それぞれ、クライアントからの要求を処理します。

BINLD の構成

BINLD サーバーは、デフォルトでは /etc/binld.cnf ファイルを読み取ることによって構成されます。 このファイルは、オプションとアドレスの入ったサーバーの初期データベースを指定します。

サーバーは、SMIT または SRC コマンドのいずれかを使用して始動します。

BINLD サーバーの構成は、通常は、ネットワークで BINLD を使用するときの最も難しい仕事です。 まず最初に、どのネットワークが PXE クライアントを持つ必要があるかを見極めてください。 DHCP サーバーと同じマシンで BINLD サーバーを実行するように構成する例を次に示します。

```
pxeservertype       binld_on_dhcp_server

subnet default
{
    vendor pxe
    {
                bootstrapserver  9.3.149.6      #TFTP server IP address
                pxebootfile   1   2   1   window.one   1   0
                pxebootfile   2   2   1   linux.one    2   3
                pxebootfile   1   2   1   hello.one    3   4
                client 6 10005a8ad14d any
                {
                   pxebootfile   1   2   1   aix.one     5   6
                   pxebootfile   2   2   1   window.one  6   7
        }
     }
}
```

上記の構成では、BINLD サーバーはポート 4011 でクライアントのユニキャスト・パケットを listen し、BINLD が dhcpsd/pxed からマルチキャスト・アドレスを取得した場合、ポート 4011 でマルチキャスト・パケットを listen します。 BINLD サーバーは、ブート・ファイル名と TFTP サーバーの IP アドレスにより、クライアントの REQUEST/INFORM パケットに応答します。 BINLD が、クライアントの指定するマッチング・レイヤーでブート・ファイルを検出できない場合は、次のレイヤーのブート・ファイルの検出を試みます。 クライアントの要求 (*Type*、*SystemArch*、*MajorVers*、*MinorVers*、および *Layer*) に一致するブート・ファイルがない場合、BINLD は応答しません。

BINLD が別個のマシンで実行するよう構成されている (DHCP と PXED を同じマシンで実行しない) 例を次に示します。

```
subnet 9.3.149.0 255.255.255.0
{

     vendor pxe
     {

     bootstrapserver        9.3.149.6      # TFTP server ip address.
     pxebootfile    1    2    1   window.one    1   0
     pxebootfile    2    2    1   linux.one     2   3
     pxebootfile    1    2    1   hello.one     3   4
     client 6 10005a8ad14d any
       {
       pxebootfile    1    2    1   aix.one      5   6
       pxebootfile    2    2    1   window.one   6   7
       }
     }
}
```

上記の例では、*pxeservertype* が設定されていないため、デフォルトのサーバー・タイプは **binld_only** です。 BINLD サーバーは、ポート 4011 でクライアントのユニキャスト・パケットを listen し、BINLD が dhcpsd/pxed からマルチキャスト・アドレスを取得した場合、ポート 67 でブロードキャスト & ユニキャスト・パケットを listen し、ポート 4011 でマルチキャスト・パケットを listen します。 ブート・ファイ

ル名および TFTP サーバーの IP アドレスは、クライアントの IP アドレスがサブネットの IP アドレスの範囲内 (9.3.149.0 から 9.3.149.255) にある場合にのみ、PXE クライアントに送られます。

PXED サーバーと同じマシンで BINLD が実行されるように構成する例を次に示します。

```
pxeservertype       binld_on_proxy_server
subnet default
{
     vendor
     {
     bootstrapserver        9.3.149.6          # TFTP server ip address.
     pxebootfile    1    2    1   window.one    1    0
     pxebootfile    2    2    1   linux.one     2    3
     pxebootfile    1    2    1   hello.one     3    4
     client 6 10005a8ad14d any
       {
       pxebootfile    1    2    1   aix.one     5   6
       pxebootfile    2    2    1   window.one  6   7
       }
     }
}
```

上記の構成では、BINLD が dhcpsd/pxed からマルチキャスト・アドレスを取得した場合のみ、BINLD サーバーはポート 4011 でマルチキャスト・パケットを listen します。 マルチキャスト・アドレスを受信しない場合、BINLD は終了し、エラー・メッセージがログ・ファイルに記録されます。

データベースの db_file 文節は、この構成ファイル部分を処理するために使用するデータベースのメソッドを指示します。 コメントはポンド記号 (#) で始まります。 PXED サーバーは、# からその行の終わりまでを無視します。 サーバーは、各 option 行を使用して、クライアントに行うべきことを指示します。 現在サポートされ、認識されるサブオプションについては、335 ページの『PXE ベンダー・コンテナー・サブオプション』に説明があります。 サーバーに既知ではないオプションを指定する方法については、376 ページの『汎用サーバー操作に対する BINLD サーバー・ファイルの構文』を参照してください。

BINLD 構成ファイル

構成ファイルには、アドレス・セクションとオプション定義セクションとがあります。これらのセクションは、オプション、修飾子、それにおそらくはその他のコンテナーを保持しているコンテナーの概念に基づいています。

コンテナー (基本的には、オプションをグループ化するための方法) は、クライアントをグループに 分類するのに ID を使用します。 コンテナーには 4 つのタイプあります。サブネット、クラス、ベンダー、およびクライアントです。 現在、一般的なユーザー定義可能コンテナーはありません。 ID は、例えばクライアントがサブネット間を移動するときにクライアントを追跡できるように、クライアントを固有に定義するものです。 クライアント・アクセスを定義するために、複数のコンテナー・タイプを使用することができます。

オプション は、デフォルトのゲートウェイや DNS アドレスなど、クライアントに戻される ID です。

BINLD コンテナー

DHCP サーバーが要求を受信すると、パケットが構文解析され、抽出するコンテナー、オプション、およびアドレスが ID キーによって判別されます。

BINLD の構成の最後の例は、サブネット・コンテナーを示しています。 この ID キーは、ネットワーク内のクライアントの位置です。 クライアントがそのネットワークからのものである場合は、そのクライアントはそのコンテナーに入ります。

コンテナーは、タイプごとに、クライアントを識別するための異なるオプションを使用します。

- サブネット・コンテナーは、「giaddr」フィールドまたは受信インターフェースのインターフェース ・アドレスを使用して、クライアントがどのサブネットから 発生したものであるかを判別する。
- クラス・コンテナーは、オプション 77 の値 (ユーザー・サイト・クラス ID) を使用する。
- ベンダー・コンテナーは、オプション 60 の値 (ベンダー・クラス ID) を使用する。
- クライアント・コンテナーは、PXED クライアントについてはオプション 61 (クライアント ID)、BOOTP クライアントに ついては BOOTP パケットの「chaddr」フィールドを使用する。

サブネット・コンテナーを除いて、どのコンテナーも、正規表現マッチングを含む、マッチングによる値の指定を行うことができます。

暗黙のコンテナーもあります。 グローバル・コンテナーがそれです。 グローバル・コンテナーに入れられたオプションと修飾子は、指定変更されたり、拒否されたりしない限りは、すべてのコンテナーに適用されます。 ほとんどのコンテナーは、可視範囲を暗黙指定して他のコンテナーの中に入れることができます。コンテナーは、それに関連するアドレス範囲を持つこともできますし、持たなくてもかまいません。 サブネットは、その性質上それに関連した範囲があります。

コンテナーとサブコンテナーの基本規則は、次のとおりです。

- グローバル・レベルではどのコンテナーも有効である。
- サブネットは、他のコンテナーの中に入れることはできない。
- 制限付きコンテナーの中には、同じタイプのレギュラー・コンテナーを入れることはできない。 (例えば、Accounting クラスだけが許されたオプションのコンテナーに、文字「a」で始まるすべてのクラスが許されたオプションのコンテナーを入れることはできません。 これは違反です。)
- 制限付きクライアント・コンテナーはサブコンテナーを持てない。

上記の規則に従えば、コンテナーの階層を作成することができます。この階層によって、オプションは、特定のクライアントまたは 1 組のクライアントのグループに分けられます。

クライアントが複数のコンテナーに合致する場合は、オプションとアドレスはどのように分配されるのでしょうか。 DHCP サーバーは、メッセージを受信し、要求をデータベース (この場合は db_file) に渡します。そこでコンテナー・リストが作成されます。 このリストは、奥行きと優先順位の順に並べられます。優先順位は、コンテナー内の暗黙の階層として定義されます。 ストリクト・コンテナーは、レギュラー・コンテナーよりも高い優先順位を持ちます。 クライアント、クラス、ベンダー、最後にサブネットがこの順に、また、それぞれのコンテナー・タイプ内では奥行きの順に記載されます。 すなわち、特定度の最も高いものから特定度の低いものの順に並べられたリストができあがります。 以下に例を示します。

```
Subnet 1
--Class 1
--Client 1
Subnet 2
--Class 1
----Vendor 1
----Client 1
--Client 1
```

この例では、2 つのサブネット Subnet 1 と Subnet 2 が示されています。 クラス名は 1 つで Class 1、ベンダー名も 1 つで Vendor 1、そしてクライアント名が Client 1 の 1 つです。Class 1 と Client 1 は複数の場所で定義されています。これらは異なるコンテナーに入っているので、その名前は同じですがその中の値は異なっていてかまいません。 Client 1 がオプション・リストに Class 1 を指定している Subnet 1 からメッセージを DHCP サーバーに送ると、DHCP サーバーは次のようなコンテナー・パスを生成します。

```
Subnet 1, Class 1, Client 1
```

最も特定度の高いコンテナーが最後にリストされています。 アドレスを取得するには、このリストを階層の逆向きに調べ、最初に使用可能なアドレスを探します。 次に、このリストを階層の前方向に調べ、オプションを入手します。 オプションは、*deny* がコンテナーに指定されていなければ、以前の値を指定変更します。 また、Subnet 1 には Class 1 と Client 1 が入っていますが、これらは、コンテナーの優先順位に従って並べられます。 同じクライアントが Subnet 2 に入っていて、同じメッセージを送信するのであれば、コンテナー・リストは次のように生成されます。

Subnet 2、Class 1、Client 1 (Subnet 2 レベル)、Client 1 (Class 1 レベル)

Subnet 2 が最初にリストされ、次に Class 1、最後に Subnet 2 レベルの Client 1 の順にリストされます (このクライアント・ステートメントが階層では 1 レベル下にあるためです)。 この階層は、最初のクライアント・ステートメントに合致するクライアントが、Subnet 2 内の Class 1 の Client 1 に合致するクライアントよりも特定度が低いことを示唆しています。

階層内の奥行きによって選択された優先順位が、コンテナーそれ自体の優先順位で取り替えられることはありません。 例えば、同一のクライアントが同じメッセージを出し、ベンダー ID を指定している場合、そのコンテナー・リストは次のようになります。

Subnet 2、Class 1、Vendor 1、Client 1 (Subnet 2 レベル)、Client 1 (Class 1 レベル)

コンテナー優先順位は、クライアント・コンテナーが 1 つ以上のクライアントを定義するための最も特定度の高いコンテナー優先順位であるとする一般的な概念に従っているため、検索パフォーマンスを向上させます。 クラス・コンテナーは、クライアント・コンテナーよりも特定度が低いアドレスを保持しています。ベンダーはさらに特定度が低く、サブネットは特定度が最も低くなっています。

BINLD アドレスとアドレス範囲
いずれのコンテナー・タイプも関連したアドレス範囲を持つことがあります。サブネットには関連したアドレス範囲が必須です。

コンテナー内の各範囲は、親コンテナーの範囲のサブセットでなければならず、他のコンテナーの範囲とオーバーラップしてはなりません。 例えば、あるクラスがサブネット内で定義されており、そのクラスが範囲を持っている場合、その範囲は当該サブネットの範囲のサブセットである必要があります。 また、そのクラス・コンテナー内の範囲は、そのレベルにあるその他の範囲とオーバーラップすることはできません。

範囲は、コンテナーの行で表現し、範囲および除外ステートメントで変更して、あるコンテナーに関連付けられたばらばらのアドレス・セットを使用可能にすることができます。 したがって、サブネットの最初の 10 アドレスと次の 10 アドレスを使用可能にする場合、そのサブネットがサブネット文節でこれらのアドレス範囲を指定することによって、使用するメモリーを削減すると同時に、指定された範囲にないその他のクライアントとアドレス競合が起きるのを避けることができます。

一度アドレスを選択すると、アドレス範囲を含んでいるリスト内の後続のコンテナーは、その子とともにリストから除去されます。 これは、除去されたコンテナー内のネットワーク固有のオプションは、アドレスがそのコンテナーから使用されたものでなければ有効ではないからです。

BINLD 構成ファイル・オプション
アドレスを判別するためにリストが選別された後、1 組のオプションがクライアントのために 生成されます。

この選択プロセスでは、*deny* が検出されなければ、以前選択されたオプションが上書きされます。検出された場合は、否定されたオプションがクライアントに 送られるリストから除去されます。 このメソッドによって、親コンテナーからの継承で、指定しなければならないデータ量が削減されます。

BINLD ロギング
ロギング・パラメーターは、データベースのようなコンテナーに指定しますが、コンテナーの場合のキーワードは `logging_info` です。

PXED の構成方法を学んでいる間は、ロギングのレベルを最高レベルに設定しておくことをお勧めします。また、ロギング・サブシステムの 初期化後に構成エラーが確実に記録されるようにするために、他の構成ファイル・データのどれよりも先にロギング構成を指定することが最善です。 `logitem` キーワードは、あるロギング・レベルをオンにするために使用します。ロギング・レベルを使用不可にするには `logitem` キーワードを除去してください。 ロギングのためのその他のキーワードを使用すると、ログのためのファイル名、ファイルのサイズ、回転ログ・ファイル数を指定することができます。

BINLD パフォーマンスの考慮事項
特定の構成キーワードおよび構成ファイルの構造が PXED サーバーのメモリー使用量とパフォーマンスに影響を与えるということを理解することが重要です。

まずはじめに、親コンテナーから子コンテナーへと継承されるオプションの継承モデルを理解することにより、過度のメモリー使用を回避することができます。 リストされていないクライアントは一切サポートされない環境では、管理者は当該のファイルに各クライアントを明示的にリストする必要があります。 どのクライアントについても 個々にオプションがリストされている場合、サーバーは、その構成ツリーを保管するために、オプションが親コンテナー (例えば、サブネット・コンテナー、ネットワーク・コンテナー、またはグローバル・コンテナーなど) から継承される場合よりも多くのメモリーを使用します。 したがって、管理者は、構成ファイル内においてクライアント・レベルで繰り返し現れるオプションがあるかど

うかを検査し、そのようなオプションがある場合には、それらのオプションをひとまとめにして親コンテナーの中に指定して一群のクライアントで共有することができるかどうかを判別する必要があります。

また、logItem エントリー INFO および TRACE を使用していると、PXE クライアントのメッセージを 1 つ処理するたびに大量のメッセージがログに記録されます。 ログ・ファイルに 1 行を追加する操作が高くつく可能性があります。したがって、ロギングの量を制限すれば PXED サーバーのパフォーマンスが向上します。 PXED サーバーにエラーが発生している疑いがある場合は、SRC の traceson コマンドを使用して、ロギングを動的に再度使用可能にすることができます。

汎用サーバー操作に対する BINLD サーバー・ファイルの構文

ここでは、汎用サーバー操作に対する BINLD サーバー・ファイルの構文について説明します。 構文の形式、サブコンテナー、デフォルト値、および意味を示します。

注 : 次の表の中の時間の単位 (*time_units*) は、オプションであり、実際の時間に対する修飾子を表しています。 デフォルトの時間単位は分です。 有効な値は、秒 (1)、分 (60)、時間 (3600)、日 (86400)、週 (604800)、月 (2392000) および年 (31536000) です。 ここで、括弧内の数字は、指定値 *n* を秒数で表現したい場合に、乗数として使用する値です。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
database	database *db type*	はい	なし	アドレス・プール、オプション、およびクライアント・アクセス・ステートメントのための定義を保持する 1 次コンテナー。 *db type* は、ファイルのこの部分を処理するためにロードされるモジュールの名前。 現在使用できる値は、db_file だけである。
logging_info	logging_info	はい	なし	ロギング・パラメーターを定義する 1 次ロギング・コンテナー。
logitem	logitem NONE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem SYSERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem OBJERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem PROTOCOL	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem PROTERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem WARN	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem WARNING	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem CONFIG	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem EVENT	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
logitem	logitem PARSEERR	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem ACTION	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem ACNTING	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem STAT	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem TRACE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem RTRACE	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
logitem	logitem START	いいえ	すべて使用不可	ロギング・レベルを使用可能にする。 複数行が許される。
numLogFiles	numLogFiles *n*	いいえ	0	作成するログ・ファイル数を指定する。 ログは、最初のファイルがいっぱいになると、次のファイルに回転する。 *n* は作成するファイル数。
logFileSize	logFileSize *n*	いいえ	0	各ログ・ファイルのサイズを 1024 バイト単位で指定する。
logFileName	logFileName *path*	いいえ	なし	最初のログ・ファイルへのパスを指定する。 最初のログ・ファイルの名前は、*filename* または *filename.extension* である。 ファイルが次に回転するとき、その名前は、ベースの *filename* で始まり、その後ろに番号を追加するか、または拡張子を番号で置き換えたものになる。 例えば、もとのファイル名が `file` である場合、回転後のファイル名は、`file01` となる。 また、もとのファイル名が `file.log` である場合は、`file.01` となる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
pxeservertype	pxeservertype *servertype*	いいえ	dhcp_only	dhcpsd サーバーのタイプを指示する。 *servertype* は、次のいずれかにすることができる。 `binld_on_dhcp_server` (BINLD が DHCP サーバーと同じマシンで実行され、ポート 4011 および DHCP/PXED から受信したとき、マルチキャスト・アドレスで PXE クライアント要求を listen する)。 `binld_on_proxy_server` (BINLD が PXED サーバーと同じマシンで実行され、DHCP/PXED から受信したとき、マルチキャスト・アドレスで PXE クライアント要求を listen する)。 デフォルト値は `binld_only` で、BINLD は別個のマシンで実行され、ポート 67、4011、および DHCP/PXED から受信したとき、マルチキャスト・アドレスでクライアントのパケットを listen する。
dhcp_or_proxy_address	dhcp_or_proxy_address *IP address*	いいえ	なし	BINLD サーバーが、マルチキャスト・アドレスを受信するためのタイプ REQUEST/INFORM のユニキャスト・パケットを送信する先の dhcp または pxed サーバーの IP アドレスを指定する。 このキーワードは、dhcp または pxed が BINLD と異なるサブネットにある場合にのみ、定義する。

db_file データベースに対する BINLD サーバー・ファイルの構文

ここでは、`db_file` データベースに対する BINLD サーバー・ファイルの構文について説明します。 構文の形式、サブコンテナー、デフォルト値、および意味を示します。

注 :

1. 次の表の中の時間の単位 (*time_units*) は、オプションであり、実際の時間に対する修飾子を表しています。 デフォルトの時間単位は分です。 有効な値は、秒 (1)、分 (60)、時間 (3600)、日 (86400)、週 (604800)、月 (2392000) および年 (31536000) です。 ここで、括弧内の数字は、指定値 *n* を秒数で表現したい場合に、乗数として使用する値です。
2. 1 つのコンテナー内に指定されている項目は、サブコンテナー内で指定変更されることがあります。 例えば、グローバルに BOOTP クライアントを定義する一方で、supportBootp キーワードを両方のコンテナーに指定することで、ある一定のサブネット内で BOOTP クライアントをサポートできます。
3. クライアント、クラス、およびベンダーの各コンテナーは、正規表現のサポートを受けることができます。 クラスとベンダーでは、引用符の後ろの最初の文字を感嘆符 (!) にした引用符付き文字列は、文字列の残りを正規表現として処理されるように指示します。 クライアント・コンテナーでは、「hwtype」フィールドと「hwaddr」フィールドの両方で正規表現を使用することができます。 両方のフィールドを表すために、単一の文字列が次のフォーマットで使用されます。

```
decimal_number-data
```

decimal_number がゼロの場合は、data は ASCII 文字列です。 ゼロ以外の場合は、data は 16 進数です。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
subnet	subnet default	はい	なし	範囲なしのサブネットを指定する。 サーバーでは、クライアントからの INFORM パケットに応答するときに、クライアントのアドレスが他に一致するサブネット・コンテナーを持たない場合にのみ、このサブネットを使用します。
subnet	subnet *subnet id netmask*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
subnet	subnet *subnet id netmask range*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。
subnet	subnet *subnet id netmask label*:*priority*	はい	なし	サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
subnet	subnet *subnet id netmask range label*:*priority*			サブネットとアドレスのプールを指定する。 すべてのアドレスは、範囲がその行に指定されているか、またはアドレスがコンテナー内で後に範囲または除外ステートメントによって変更されない限り、そのプールに入っているものと見なされる。 オプションの範囲は、ダッシュで分離されたドット付きクワッドの 1 対の IP アドレスである。 オプションのラベルと優先順位を指定することができる。 これらは、サブネットを識別し、仮想サブネット内で順序を付けるために、仮想サブネットが使用する。 ラベルと優先順位は、コロンで区切られる。 これらのコンテナーは、グローバルまたはデータベースのコンテナー・レベルでだけ使用できる。
subnet	subnet *subnet id range*	はい	なし	ネットワーク・コンテナー内で使用されるサブネットを指定する。 これは、オプションの範囲が指定されない限り、サブネット全体のアドレス範囲を定義する。 このサブネットに関連するネットマスクは、周辺のネットワーク・コンテナーから取得される。 **注 :** このメソッドは、使用すべきではありません。他のサブネット・フォーマットを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
option	option *number data* ...	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 option * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。
option	option *number*deny	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 option * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
option	option * deny	いいえ	なし	クライアントに送信すべきオプション、または、否定の場合は、クライアントに送信してはならないオプションを指定する。 option * deny 文節は、現在のコンテナーに指定されていないオプションのすべてをクライアントに戻すのではないことを意味する。 option *number*deny は、指定のオプションだけを否定する。 *number* は、無符号の 8 ビット整数である。 *data* は、該当オプションに固有 (上記参照) か、または引用符付き文字列 (ASCII テキストであることを指示) であるか、あるいは、0x*hexdigits* または hex"*hexdigits*" か hex "*hexdigits*" とすることができる。 オプションがベンダー・コンテナー内にある場合は、他のオプションとオプション 43 にカプセル化される。
exclude	exclude *IP address*	いいえ	なし	除外ステートメントが入っているコンテナーの範囲を変更する。 除外ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 除外ステートメントは、指定されたアドレスまたは範囲をコンテナーの現在の範囲から除去する。 除外ステートメントを使用すると、サブネットまたはその他のコンテナーに不連続な範囲を作成することができる。
exclude	exclude *dotted_quad-dotted_quad*	いいえ	なし	除外ステートメントが入っているコンテナーの範囲を変更する。 除外ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 除外ステートメントは、指定されたアドレスまたは範囲をコンテナーの現在の範囲から除去する。 除外ステートメントを使用すると、サブネットまたはその他のコンテナーに不連続な範囲を作成することができる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
range	range *IP_address*	いいえ	なし	範囲ステートメントが入っているコンテナーの範囲を変更する。 範囲ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 range が、コンテナー定義行で範囲を指定していないコンテナーの最初のものである場合は、そのコンテナーのための範囲は、この範囲ステートメントの指定した範囲になる。 最初の range の後ろにある範囲ステートメント、または、その定義に範囲を指定しているコンテナーのための範囲ステートメントは、現在の範囲に追加される。 範囲ステートメントを使用すると、範囲にアドレスを 1 つ、または 1 組追加することができる。 範囲は、サブネット・コンテナー定義の中に収まる必要がある。
range	range *dotted_quad-dotted_quad*	いいえ	なし	範囲ステートメントが入っているコンテナーの範囲を変更する。 範囲ステートメントは、グローバルまたはデータベースのコンテナー・レベルでは無効である。 range が、コンテナー定義行で範囲を指定していないコンテナーの最初のものである場合は、そのコンテナーのための範囲は、この範囲ステートメントの指定した範囲になる。 最初の range の後ろにある範囲ステートメント、または、その定義に範囲を指定しているコンテナーのための範囲ステートメントは、現在の範囲に追加される。 範囲ステートメントを使用すると、範囲にアドレスを 1 つ、または 1 組追加することができる。 範囲は、サブネット・コンテナー定義の中に収まる必要がある。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
client	client *hwtype hwaddr* NONE			*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。
client	client *hwtype hwaddr* ANY			*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
client	client *hwtype hwaddr dotted_quad*			*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。
client	client *hwtype hwaddr range*			*hwaddr* と *hwtype* で指定されるクライアントがアドレスを取得することを否定するクライアント・コンテナーを指定する。 *hwtype* が 0 の場合は、*hwaddr* は ASCII 文字列である。 0 以外の場合は、*hwtype* はそのクライアントのハードウェア・タイプ、*hwaddr* はハードウェア・アドレスである。 *hwaddr* が文字列の場合は、文字列の前後に引用符を使用できる。 *hwaddr* が 16 進数列の場合は、アドレスは 0x*hexdigits* または hex *digits* と指定できる。 *range* を使用すると、*hwaddr* と *hwtype* で指定されるクライアントは、*range* 内のアドレスを取得できる。 複数のクライアントとマッチングを行うには、正規表現でなければならない。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
class	class *string*	はい	なし	名前が *string* であるクラス・コンテナーを指定する。 文字列には引用符を付けても付けなくてもよい。 引用符を付ける場合は、比較の前に引用符は除かれる。 スペースまたはタブを含む文字列には、引用符が必要である。 このコンテナーは、どのレベルでも有効である。 このクラスのクライアント分配する 1 組のアドレスを指示するために、範囲を指定することができる。 範囲は、ドット付きクワッドの IP アドレスを 1 つで、またはドット付きクワッドの IP アドレスを 2 つをダッシュで区切って示す。
class	class *string range*	はい	なし	名前が *string* であるクラス・コンテナーを指定する。 文字列には引用符を付けても付けなくてもよい。 引用符を付ける場合は、比較の前に引用符は除かれる。 スペースまたはタブを含む文字列には、引用符が必要である。 このコンテナーは、どのレベルでも有効である。 このクラスのクライアント分配する 1 組のアドレスを指示するために、範囲を指定することができる。 範囲は、ドット付きクワッドの IP アドレスを 1 つで、またはドット付きクワッドの IP アドレスを 2 つをダッシュで区切って示す。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
network	network *network id netmask*	はい	なし	クラス情報を使用したネットワーク ID を指定する (例えば、9.3.149.0 にネットマスク 255.255.255.0 を付けたものは、ネットワーク 9.0.0.0 255.255.255.0 となる)。 このバージョンのネットワーク・コンテナーは、同じネットワーク ID とネットマスクを持つサブネットを保持するために使用される。 範囲を指定した場合は、その範囲内のすべてのアドレスがそのプールに入る。 範囲は、そのネットワーク ID のネットワーク内にある必要がある。 これは、クラス・フル・アドレッシングを使用する。 これは、グローバルまたはデータベースのコンテナー・レベルでだけ有効である。 **注 :** network キーワードは、使用すべきではありません。代わりにサブネット・コンテナーを使用してください。
network	network *network id*	はい	なし	クラス情報を使用したネットワーク ID を指定する (例えば、9.3.149.0 にネットマスク 255.255.255.0 を付けたものは、ネットワーク 9.0.0.0 255.255.255.0 となる)。 このバージョンのネットワーク・コンテナーは、同じネットワーク ID とネットマスクを持つサブネットを保持するために使用される。 範囲を指定した場合は、その範囲内のすべてのアドレスがそのプールに入る。 範囲は、そのネットワーク ID のネットワーク内にある必要がある。 これは、クラス・フル・アドレッシングを使用する。 これは、グローバルまたはデータベースのコンテナー・レベルでだけ有効である。 **注 :** network キーワードは、使用すべきではありません。代わりにサブネット・コンテナーを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
network	network *network id range*			クラス情報を使用したネットワーク ID を指定する (例えば、9.3.149.0 にネットマスク 255.255.255.0 を付けたものは、ネットワーク 9.0.0.0 255.255.255.0 となる)。 このバージョンのネットワーク・コンテナーは、同じネットワーク ID とネットマスクを持つサブネットを保持するために使用される。 範囲を指定した場合は、その範囲内のすべてのアドレスがそのプールに入る。 範囲は、そのネットワーク ID のネットワーク内にある必要がある。 これは、クラス・フル・アドレッシングを使用する。 これは、グローバルまたはデータベースのコンテナー・レベルでだけ有効である。 **注 :** network キーワードは、使用すべきではありません。代わりにサブネット・コンテナーを使用してください。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id*	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* hex""	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* hex ""	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* 0xdata	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id* ""	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range*	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* hex""	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* hex ""	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* 0xdata	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor *vendor_id range* ""	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor pxe	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
vendor	vendor pxeserver	はい	なし	ベンダー・コンテナーを指定する。 ベンダー・コンテナーはクライアントにオプション 43 を戻すために使用される。 ベンダー ID は、引用符付き文字列、あるいは 0x*hexdigits* または hex"*digits*" のフォーマットのバイナリー文字列として指定できる。 ベンダー ID の後にはオプションで範囲を指定できる。 範囲は、2 つのドット付きクワッドをダッシュで区切って指定する。 オプションの範囲の後に、任意で 16 進数列または ASCII 文字列をオプション 43 の最初の部分として指定することができる。 コンテナー内にオプションがあると、オプション 43 データに付加される。 すべてのオプションが処理されると、オプション・リスト・オプションの終わり (End Of Option List Option) がデータに付加される。 オプション 43 以外のオプションを戻すためには、通常のオプションをベンダー ID に基づいて戻すことを指定するために、すべてのクライアントに合致する正規表現クライアントを使用する。 キーワード vendor の後ろの pxe は、PXEClient のベンダー・コンテナーを作成します。 キーワード vendor の後ろの pxeserver は、PXEServer のベンダー・コンテナーを作成します。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
inoption	inoption *number option_data*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって 既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"! (感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの 文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
inoption	inoption *number option_data range*	はい	なし	クライアント指定のあらゆる任意の着信オプションと一致するコンテナーを指定する。 *number* は、オプション番号を指定する。 *option_data* は、当該のクライアント用のアドレスおよびオプションの選択時に選択されるこのコンテナーと一致するキーを指定する。 *option_data* は、既知のオプションの場合は予期される形式 — 引用符付き文字列、IP アドレス、整数値 — で指定される。あるいは、オプションで、文字 0x を先頭に付ければ 16 進数のバイト文字列として指定することができる。 サーバーにとって 既知ではないオプションの場合は、同じ方法で 16 進数のバイト文字列を指定することができる。 さらに、 *option_data* は、クライアントのオプション・データの文字列表示と比較される正規表現を指示することができる。 正規表現は、"! (感嘆符の前に二重引用符を付ける) で始まる引用符付き文字列の中に指定する。 サーバーにとって既知ではないオプションの 文字列形式は、文字 0x が先頭に付いていない 16 進数のバイト文字列となる。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual	virtual fill *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
virtual	virtual sfill *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual	virtual rotate *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
virtual	virtual srotate *id id* ...	いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
virtual		いいえ	なし	ポリシー付きの仮想サブネットを指定する。 `fill` は、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 `rotate` は、要求ごとにリスト内の次のプールからアドレスを選択することを意味する。 `sfill` と `srotate` は、`fill` と `rotate` と同じだが、クライアントがサブネット内のコンテナー、ベンダー、またはクラスと合致するかどうかを調べるために検索が行われる。 アドレスを提供できる合致が検出された場合は、ポリシーには従わずに、そのアドレスがそのコンテナーから取得される。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
inorder:	inorder: *id id* ...	いいえ	なし	fill ポリシーを持つ仮想サブネットを指定する。このポリシーは、コンテナー内のすべてのアドレスを使用してから次のコンテナーに進むことを意味する。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。
balance:	balance: *id id* ...	いいえ	なし	rotate ポリシーを持つ仮想サブネットを指定する。このポリシーは、次のコンテナーの次のアドレスを使用することを意味する。 必要なだけ多くの ID を置くことができる。 *id* は、サブネット定義からのサブネット ID またはサブネット定義からのラベルである。 このラベルは、同じサブネット ID を持つサブネットが複数ある場合に必要である。

キーワード	フォーマット	サブコンテナー?	デフォルト値	意味
bootstrapserver	bootstrapserver *IP address*	いいえ	なし	BOOTP または DHCP パケットの受信後、TFTP ファイルに対してクライアントが使用すべきサーバーを指定する。 この値は、パケットの **siaddr** フィールドに入る。 これは、どのコンテナー・レベルにも有効である。
giaddrfield	giaddrfield *IP address*	いいえ	なし	応答パケットのための giaddrfield を指定する。 **注 :** この指定は BOOTP と DHCP のプロトコルには正しくありませんが、ネットワークのデフォルト・ゲートウェイとして **giaddr** フィールドを必要とするクライアントもあります。 giaddrfield は、どのレベルでも有効ですが、このように競合の可能性があるため、クライアント・コンテナー内でのみ使用します。
bootfile	bootfile *path*	いいえ	なし	応答パケットのファイル・セクションで使用するブート・ファイルを指定する。 これは、どのコンテナー・レベルにも指定できる。 着信パケットのファイル・セクションに指定されている項目が、ブート・ファイルおよびホーム・ディレクトリー・ステートメントとどのように対話するかは、bootfile ポリシーで定義する。
pxebootfile	pxebootfile *SystemArch MajorVer MinorVer Bootfilename Type Layer*	いいえ	なし	PXEClient に提供するブート・ファイルを指定する。 キーワードの後ろのパラメーターの数が 4 に満たない場合、config ファイル・パーサーはエラーを生成する。 7 を超えると無視し、4 つの場合は、Type = 0 と Layer = 0 の値が想定される。このキーワードは、コンテナーだけでしか使用できない。

その他のオプションの詳細については、225 ページの『DHCP サーバー・ファイルの既知のオプション』および 335 ページの『PXE ベンダー・コンテナー・サブオプション』 を参照してください。

TCP/IP デーモン

デーモン (サーバー とも呼ばれる) とは、常にバックグラウンドで実行され、他のプロセスが必要とする機能を実行するプロセスのことです。 **TCP/IP (伝送制御プロトコル/インターネット・プロトコル)** には、特定の機能をオペレーティング・システム内に実装するためのデーモンがあります。

これらデーモンはバックグラウンド・プロセスであり、他のプロセス (そのプロセスがデーモンの機能の一部でない場合) に割り込まずに実行されます。

デーモンは、システム管理レベルのコマンド、他のデーモン、シェル・スクリプトのいずれかによって起動されます。 また、**inetd** デーモン、rc.tcpip シェル・スクリプト、およびシステム・リソース・コントローラー (SRC) を使用してデーモンを制御することもできます。

サブシステムとサブサーバー

サブシステム とは、SRC によって制御されるデーモン、すなわち、サーバーのことです。 サブシステムによって制御されるデーモンのことをサブサーバー といいます (デーモン・コマンドとデーモン名は通常、名前の末尾に **d** が付きます)。

サブシステムと サブサーバーのカテゴリーは、互いに排他的です。 つまり、デーモンはサブシステムとサブサーバーの両方としてリストされることはありません。 他のデーモンを制御する **TCP/IP** サブシステムは、**inetd** デーモンだけです。 したがって、すべての **TCP/IP** サブサーバーは、**inetd** サブサーバーでもあります。

TCP/IP デーモンのリストについては、487 ページの『TCP/IP デーモン』を参照してください。

システム・リソース・コントロール

数ある機能の中の、特に SRC を使用すると、デーモンの始動と停止、およびデーモンのアクティビティーのトレースができます。 また、SRC は複数のデーモンをサブシステムとサブサーバーにグループ 化する機能も備えています。

システム・リソース・コントロールは、デーモンを制御する際に役立つように設計されたツールです。 SRC を使用すると、各デーモン・コマンドに使用できるフラグやパラメーター以上の制御ができます。

システム・リソース・コントローラーの詳細については、オペレーティング・システムおよびデバイスの管理のシステム・リソース・コントローラーを参照してください。

SRC コマンドのリストについては、485 ページの『SRC コマンド』を参照してください。

inetd デーモンの構成

TCP/IP inetd デーモンを構成するには、以下の手順を使用します。

inetd デーモンを構成するには、次のようにします。

1. **inetd** デーモンを追加することで、呼び出されるサブサーバーを指定します。
2. **inetd** デーモンの再始動特性を変更することによって、再始動特性を指定します。

表 *76. inetd* デーモンの構成タスク

タスク	SMIT 高速パス	コマンドまたはファイル
inetd デーモンの始動	smit mkinetd	**startsrc -s inetd**
inetd デーモンの再始動特性の変更	smit chinetd または smit lsinetd	
inetd デーモンの停止	smit rminetd	**stopsrc -s inetd**
すべての **inetd** サブサーバーのリスト表示	smit inetdconf	
inetd サブサーバーの追加 [1]	smit mkinetdconf	edit /etc/inetd.conf then run **refresh -s inetd** or **kill -1** *inetdPID*[2]
inetd サブサーバーの特性の変更/表示	smit inetdconf	edit /etc/inetd.conf then run **refresh -s inetd** or **kill -1** *inetdPID*[2]

表 76. *inetd* デーモンの構成タスク (続き)		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
inetd サブサーバーの除去	`smit rminetd`	edit `/etc/inetd.conf` then run **refresh -s inetd** or **kill -1** *inetdPID*[2]

注:

1. 「**inetd** サブサーバーの追加」のアクションでは、必要に応じてサブサーバーを起動するように **inetd** デーモンが構成される。
2. **refresh** コマンドと **kill** コマンドはどちらも、**inetd** デーモンに対して、構成ファイルに変更があったことを通知する。

クライアント・ネットワーク・サービス

クライアント・ネットワーク・サービス (SMIT 高速パス `smit clientnet` を使用してアクセスできる) は、このオペレーティング・システムで使用可能な **TCP/IP** プロトコルを参照します。

各プロトコル (またはサービス) は、そのプロトコルがネットワーク上で使用するポート番号によって認識されるので、予約済みポート とも呼ばれています。 プログラマーにとって便利なように、ポート番号は番号だけでなく名前でも参照できます。 例えば、**TCP/IP** メール・プロトコルはポート 25 を使用し、**smtp** という名前で認識されます。 プロトコルが `/etc/services` ファイル内にリストされている (コメント化されていない) 場合、ホストはそのプロトコルを使用できます。

デフォルトでは、すべての **TCP/IP** プロトコルが `/etc/services` ファイル内に定義されています。 このファイルを構成する必要はありません。 独自のクライアント/サーバー・プログラムを作成する際、独自のサービスを `/etc/services` ファイルに追加し、そのサービスに対して特定のポート番号と名前を予約することができます。 独自のサービスを `/etc/services` に追加する場合は、ポート番号 0 から 1024 がシステム用に予約済みである点に注意してください。

表 77. クライアント・ネットワーク・サービス・タスク		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
すべてのサービスのリスト	`smit lsservices`	`/etc/services` を表示する
サービスの追加	`smit mkservices`	`/etc/services` を編集する
サービスの特性の変更/表示	`smit chservices`	`/etc/services` を編集する
サービスの除去	`smit rmservices`	`/etc/services` を編集する

サーバー・ネットワーク・サービス

サーバー・ネットワーク・サービスには、次の表に示すようにリモート・アクセスの制御、**TCP/IP** の始動と停止、pty デバイス・ドライバーの管理などの機能があります。

pty デバイス・ドライバーは、システムとともに自動的にインストールされます。 デフォルトでは 16 個の BSD 型のシンボリック・リンクをサポートするように構成され、ブート時にシステムによる使用が可能です。

表 78. サーバー・ネットワーク・サービス・タスク		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
リモート・アクセスの制御		セキュリティーのリモート・コマンド実行アクセスおよび制限付きファイル転送プログラム・ユーザーを参照。

表 78. サーバー・ネットワーク・サービス・タスク (続き)		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
TCP/IP サブシステムの始動、再始動、または停止	`smit otherserv`	408 ページの『システム・リソース・コントロール』を参照してください。
pty デバイス・ドライバーの特性の変更/表示	`smit chgpty`	**chdev -l pty0 -P -a num=***X* (*X* の範囲は 0 から 64)
pty デバイス・ドライバーを使用不可にする	`smit pty` の後、「**Remove the PTY; Keep Definition (PTY の除去; 定義は保持)**」を選択する	関連コマンドまたはファイルはなし。
pty デバイス・ドライバーを使用可能にする	`smit pty` の後、「**Configure the Defined PTY (定義済み PTY の構成)**」を選択する	関連コマンドまたはファイルはなし。
エラー・レポートの生成	`smit errpt`	関連コマンドまたはファイルはなし。
pty のトレース	`smit trace`	関連コマンドまたはファイルはなし。

TCP/IP 経路指定

経路 は、インターネット・ネットワークを介して別のネットワーク上のアドレスへパケットを送信するためのパスを定義します。

経路は完全なパスを定義するのではなく、1 つのホストから、宛先へパケットを転送できるゲートウェイまで (または、あるゲートウェイから別のゲートウェイまで) のパス・セグメントだけを定義します。 経路には次の 5 つのタイプがあります。

項目	説明
ホスト経路	パケットを別のネットワーク上の特定のホストに転送できるゲートウェイを定義します。
ネットワーク経路	パケットを特定のネットワーク上の任意のホストへ転送できるゲートウェイを定義します。
デフォルト経路	宛先へのホスト経路またはネットワーク経路が他の方法で定義されていない場合に使用されるゲートウェイを定義します。
ループバック経路	ローカル・ネットワーク・アドレスに送信されるすべてのパケットのデフォルト経路。 ループバック経路 IP は常に 127.0.0.1 です。
ブロードキャスト経路	すべてのブロードキャスト・パケットのデフォルト経路。 2 つのブロードキャスト経路は、ネットワークが IP を持つ各サブネットに自動的に割り当てられます (1 つはサブネット・アドレス、もう 1 つはサブネットのブロードキャスト・アドレス)。

経路はカーネルのルーティング・テーブル の中で定義されます。 経路定義には、ローカル・ホストから到達可能なネットワークに関する情報や、リモート・ネットワークに到達するために使用するゲートウェイに関する情報が含まれています。 ゲートウェイは、データグラムを受信すると、その宛先までのパスに沿った次の送信先を知るために、ルーティング・テーブルを調べます。

カーネル・ルーティング・テーブルに、同じ宛先への複数の経路を追加できます。 経路指定の検索では、要求に一致するすべての経路が評価され、最低距離メトリックの経路が選択されます。 一致する複数の経路の距離が等しい場合、検索では最も固有の経路が選択されます。 両方の基準が、複数の経路について等しい場合、一致する経路の条件を替えて経路指定の検索を行います。

静的経路指定と動的経路指定

TCP/IP では、経路指定を静的 または動的 のいずれかのタイプにすることができます。

静的経路指定の場合は、**route** コマンドを使用してルーティング・テーブルを手動で保守します。 静的経路指定は、1 つのネットワークが他の 1 つか 2 つのネットワークと通信する場合に適します。 しかし、もっと多くのネットワークと通信するようになると、ゲートウェイの数が増え、したがってルーティング・テーブルを手動で保持するために要する時間と労力も増えます。

動的経路指定では、デーモンが自動的にルーティング・テーブルを更新します。 経路指定デーモンは、他の経路指定デーモンからの情報のブロードキャストを絶えず受信し、したがってルーティング・テーブルを絶えず更新します。

TCP/IP では、動的経路指定に **routed** と **gated** という 2 つのデーモンを使用できます。 **gated** デーモンは、**Routing Information Protocol (RIP)**、**Routing Information Protocol Next Generation (RIPng)**、**外部ゲートウェイ・プロトコル (EGP)**、**ボーダー・ゲートウェイ・プロトコル (BGP)** と **BGP4+**、**DCN ローカル・ネットワーク・プロトコル (HELLO)**、**Open Shortest Path First (OSPF)**、**Intermediate System to Intermediate System (IS-IS)**、および **インターネット制御メッセージ・プロトコル (ICMP と ICMPv6)/ Router Discovery** の経路指定プロトコルを同時にサポートします。 さらに、**gated** デーモンは**シンプル・ネットワーク管理プロトコル (SNMP)** もサポートします。 **routed** デーモンは、**Routing Information Protocol** のみをサポートします。

経路指定デーモンは、始動時に使用されたオプションに応じて、パッシブ とアクティブ という 2 つのモードのどちらかで動作します。 アクティブ・モードでは、経路指定デーモンはローカル・ネットワークに関する経路指定情報をゲートウェイとホストへ定期的にブロードキャストし、また、ホストとゲートウェイからの経路指定情報を受信します。 パッシブ・モードでは、経路指定デーモンはホストとゲートウェイから経路指定情報を受信しますが、リモート・ゲートウェイを更新しようとしません (経路指定情報を公示しません)。

この 2 つのタイプの経路指定は、ゲートウェイだけでなくネットワーク上にある別のホストに対しても同様に使用できます。 静的経路指定は、他のホストの場合もゲートウェイの場合と同様に機能します。 しかし、動的経路指定デーモンは、ゲートウェイでないホスト上で実行する場合にはパッシブ (抑制) モードで実行しなければなりません。

TCP/IP 経路指定ゲートウェイ

ゲートウェイは、一種のルーターです。 ルーター は 2 つ以上のネットワークを接続し、経路指定機能を提供します。 例えば、一部のルーターは、ネットワーク・インターフェース ・レベルまたは物理レベルで経路を指定します。 しかし、ゲートウェイ は、ネットワーク・レベルで経路を指定します。

ゲートウェイは、他のゲートウェイまたはホストからローカル・ネットワーク上のホストへ送達する IP データグラムを受信し、1 つのネットワークから別のネットワークへ IP データグラムを経路指定します。 例えば、2 つのトークンリング・ネットワークへ接続しているゲートウェイには 2 つのトークンリング・アダプター・カードがあり、それぞれのカードが独自のトークンリング・ネットワーク・インターフェースを備えています。 情報を渡す場合、ゲートウェイは 1 つのネットワーク・インターフェースを 介してデータグラムを受信し、それらのデータグラムをもう 1 つのネットワーク・インターフェースを 介して送信します。 ゲートウェイは、インターフェース 状況メッセージを使用して、ネットワーク接続を定期的に検査します。

ゲートウェイは、宛先ホストでなく宛先ネットワークに応じてパケットを経路指定します。 つまり、ゲートウェイ・マシンは、あるパケットについて考えられるすべてのホスト宛先を追跡するためには、なくても構いません。 ホスト宛先を追跡する代わりに、ゲートウェイは宛先ホストのネットワークに応じてパケットを経路指定します。 その後、宛先ネットワークが責任を持ってパケットを宛先ホストへ送信します。 したがって、通常のゲートウェイ・マシンは、限られたディスク記憶容量 (ディスク・デバイスがある場合) と限られたメイン・メモリー容量だけを必要とします。

メッセージが発信元ホストから宛先ホストに届くまでに移動しなければならない距離は、必要なゲートウェイ・ホップ の数によって異なります。 ゲートウェイは、それが直接接続しているネットワークからはゼロ・ホップの距離になり、1 つのゲートウェイを介して到達可能なネットワークからは 1 ホップの距離になります (以下同様です)。 通常、メッセージの距離は必要なゲートウェイ・ホップの数、つまりホップ・カウント (メトリック とも呼ばれる) を単位として表されます。

内部と外部の経路指定ゲートウェイ

内部ゲートウェイとは、同じ自律システムに属するゲートウェイのことです。 内部ゲートウェイは、**Routing Information Protocol (RIP)**、**Routing Information Protocol Next Generation (RIPng)**、**Intermediate System to Intermediate System** プロトコル、**Open Shortest Path First (OSPF)** プロトコル、または **HELLO プロトコル (HELLO)** を使用して互いに通信します。 外部ゲートウェイは、異なる自律システムに属します。 外部ゲートウェイは、**外部ゲートウェイ・プロトコル (EGP)**、**ボーダー・ゲートウェイ・プロトコル (BGP)**、または **BGP4+** を使用します。

例えば、2 つの自律システムがあるとします。 第 1 のシステムは、Widget Company によって管理されるすべてのネットワークです。 第 2 のシステムは、Gadget Company によって管理されるすべてのネットワークです。 Widget Company には apple という名前の 1 台のマシンがあり、このマシンはインターネットへの Widget 社のゲートウェイになっています。 Gadget Company には orange という名前の 1 台のマシンがあり、このマシンはインターネットへの Gadget 社のゲートウェイになっています。 どちらの会社も、社内に複数の異なる内部ネットワークがあります。 それらの内部ネットワークを接続しているゲートウェイは、内部ゲートウェイです。 しかし、apple と orange は外部ゲートウェイです。

それぞれの外部ゲートウェイは、他のすべての外部ゲートウェイと通信するわけではありません。 外部ゲートウェイは、通信の相手となる一連の隣接 (他の外部ゲートウェイ) を獲得します。 それらの隣接は、地理的な近さによって定義されるのではなく、相互に確立された通信によって定義されます。 それらの隣接ゲートウェイは、今度は別の外部ゲートウェイ隣接を持ちます。 このようにして、外部ゲートウェイのルーティング・テーブルが更新され、経路指定情報が外部ゲートウェイ間に伝搬されて行きます。

経路指定情報は (N、D) というペアで送信されます。ここで、N はネットワーク、D は指定されたネットワークに到達するコストを反映した距離です。 個々のゲートウェイは、自分が到達できるネットワークと、それらのネットワークへ到達するコストを公示します。 受信側のゲートウェイは、他のネットワークへの最短パスを計算し、その情報を隣接たちに渡します。 このようにして、個々の外部ゲートウェイは頻繁に経路指定情報を受信し、ルーティング・テーブルを更新したあと、その情報を外部隣接に渡します。

ゲートウェイ・プロトコル

すべてのゲートウェイは、内部ゲートウェイでも外部ゲートウェイでも、プロトコルを使用して互いに通信します。 ここでは、一般的に使用される **TCP/IP** のゲートウェイ・プロトコルについて簡潔に説明します。

HELLO プロトコル (HELLO)
HELLO は、内部ゲートウェイ同士の通信に使用されるプロトコルのうちの 1 つです。 **HELLO** は、遅延時間が最小のパスを判別することで、他のネットワークへの最短パスを計算します。

Routing Information Protocol (RIP)
Routing Information Protocol は、内部ゲートウェイ同士の通信に使用されるプロトコルのうちの 1 つです。 **HELLO プロトコル**と同様に、**RIP** も他のネットワークへの最短パスを計算します。 しかし、**HELLO** とは異なり、**RIP** は遅延時間でなくホップ・カウントによって距離を見積もります。 **gated** デーモンは、すべてのメトリックを時間遅延として内部に保管するため、**RIP** のホップ・カウントを時間遅延に変換します。

Routing Information Protocol Next Generation
RIPng は、**IPv6** をサポートするために拡張された **RIP** プロトコルです。

Open Shortest Path First (OSPF)
OPSF は、内部ゲートウェイ同士の通信に使用されるプロトコルのうちの 1 つです。 このプロトコルはリンク状態プロトコルであり、多くのルーターが存在する複雑なネットワークの場合に **RIP** より適しています。 このプロトコルは、等しい費用のマルチパス経路指定を提供します。

外部ゲートウェイ・プロトコル (EGP)
外部ゲートウェイは、**外部ゲートウェイ・プロトコル**を使用して互いに通信することができます。 **EGP** は、他のネットワークへの最短パスを計算しません。 その代わりに、単に特定のネットワークが到達可能であるかどうかを示します。

ボーダー・ゲートウェイ・プロトコル (BGP)
ボーダー・ゲートウェイは、このプロトコルを使用して互いに通信することができます。 このゲートウェイは、到達可能性情報を自律システム間で交換しますが、**EGP** より多くの機能を提供します。 **BGP** はパス属性を使用して、最適な経路を選択する際に役立つ、経路ごとの情報をより多く提供します。

ボーダー・ゲートウェイ・プロトコル 4+
BGP4+ は **BGP** プロトコルのバージョン 4 であり、**IPv6** をサポートし、このプロトコルの旧バージョンの機能を拡張しています。

Intermediate System to Intermediate System (IS-IS)
内部ゲートウェイは、**IS-IS** プロトコルを使用して互いに通信します。 このプロトコルはリンク状態プロトコルであり、IP パケットおよび ISO/CLNP パケットを経路指定することができ、**OSPF** と同様に「最短パス優先」アルゴリズムを使用して経路を決定します。

ゲートウェイの考慮事項
ゲートウェイを構成する前に、以下の処理を実行してください。

ネットワーク用にゲートウェイを構成する前に、まず次のことを実行してください。

1. 使用するゲートウェイの数を検討します。
   構成する必要があるゲートウェイの数は、次の要因によって異なります。
   - 接続したいネットワークの数
   - ネットワークへ接続する方法
   - 接続したネットワーク上でのアクティビティー・レベル

   例えば、ネットワーク 1、ネットワーク 2、ネットワーク 3 上のすべてのユーザーが、互いに通信する必要があるとします。

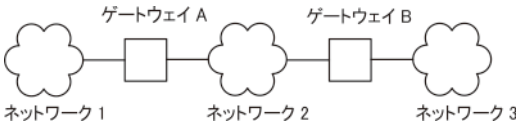


   図 *24.* 簡易ゲートウェイの構成

   この図には、1、2、3 と番号が振られた 3 つのネットワークを表す雲が含まれています。 ネットワーク 1 と 2 はゲートウェイ A で接続され、ネットワーク 2 と 3 はゲートウェイ B で接続されています。

   ネットワーク 1 をネットワーク 2 に直接接続するには、ゲートウェイを 1 つだけ (ゲートウェイ A) 使用します。 ネットワーク 2 をネットワーク 3 に直接接続するには、別のゲートウェイ (ゲートウェイ B) を使用します。 これで、正しい経路が定義されたとすると、3 つのネットワークの上のすべてのユーザーが通信できます。

   しかし、ネットワーク 2 が非常に混雑している場合は、ネットワーク 1 とネットワーク 3 間の通信に受け入れ難いほどの遅延が発生する可能性があります。 また、ほとんどのインターネットワーク通信がネットワーク 1 とネットワーク 3 の間で行われる場合は、ネットワーク 1 とネットワーク 3 を直接接続したくなります。そのためには、別の 1 対のゲートウェイ、つまり (ネットワーク 1 上の) ゲートウェイ C と (ネットワーク 3 上の) ゲートウェイ D を直接接続して使用することができます。 ただし、これは非効率な解決法かもしれません。なぜなら、1 つのゲートウェイに 3 つ以上のネットワークを接続できるからです。

   より効率的な解決策は、ゲートウェイ A をネットワーク 2 だけでなくゲートウェイ B にも直接接続することです。このようにする場合は、ゲートウェイ A とゲートウェイ B の両方にネットワーク・アダプターがもう 1 つずつ必要になります。通常、1 つのゲートウェイを介して接続するネットワーク数は、ゲートウェイ・マシンがサポートできるネットワーク・アダプター・カードの数によって制限されます。
2. 使用する経路指定のタイプを決定します。

   ネットワークが小規模で、その構成がほとんど変化しなければ、多くの場合、静的経路指定を使用することになります。 しかし、構成が頻繁に変更される大規模なネットワークでは、多くの場合、動的経路指定を使用することになります。 静的と動的の経路指定を組み合わせて使用することもできます。 つまり、少数の特定の経路に静的定義を与え、その他の経路はデーモンが更新できるようにします。 作成した静的経路は他のゲートウェイへ公示されず、経路指定デーモンによって更新されることもありません。
3. 動的な経路指定を使用している場合は、必要なゲートウェイのタイプと、ゲートウェイがサポートしなければならないプロトコルに応じて、経路指定デーモンを選択します。

ゲートウェイが内部ゲートウェイであり、**RIP** だけをサポートする必要がある場合は、**routed** デーモンを選択します。 ゲートウェイがその他のプロトコルもサポートしなければならないか、外部ゲートウェイである場合は、**gated** デーモンを選択します。

注 : gated デーモンと **routed** デーモンを同じホスト上で同時に実行した場合、予期せぬ結果が生じる恐れがあります。

ゲートウェイの構成

ゲートウェイとして機能するようにマシンを構成するには、以下の手順を使用します。

分かりやすくするため、この手順では、ゲートウェイ・マシンが 2 つのネットワークへ接続する予定であり、既にそのうちの 1 つのネットワーク上で最小限の構成が終わっているものとします。

1. まだ終わっていなければ、第 2 のネットワーク・アダプターのインストールと 構成を行います。 (166 ページの『ネットワーク・アダプターのインストール』 と 167 ページの『アダプターの 管理および構成』 を参照してください。)
2. 第 2 のネットワーク・インターフェースの IP アドレスを選択した後、173 ページの『ネットワーク・インターフェースの 管理』 の説明に従って ネットワーク・インターフェースを 構成します。
3. 第 2 のネットワークへの経路を追加します。
4. マシンを **TCP/IP** ネットワーク上のインターネットワーク・ルーターとして使用するには、次のコマンドを入力します。

```
no -o ipforwarding=1
```

この時点で、このゲートウェイ・マシンは直接接続した両方のネットワークへアクセスできます。

1. 静的経路指定を使用して、この 2 つのネットワーク以外のネットワークまたはホストと通信したい場合は、必要な経路を追加してください。
2. 動的経路指定を使用したい場合は、416 ページの『routed デーモンの構成』 または 417 ページの『gated デーモンの構成』 の指示に従ってください。 使用するインターネットワークがインターネットに 参加する場合は、420 ページの『自律システム 番号』 の指示にも従ってください。

表 79. ゲートウェイの構成タスク

タスク	SMIT 高速パス	コマンド・ファイル
ルーティング・テーブルの表示	`smit lsroute`	**netstat**[1]
静的経路の追加	`smit mkroute`	**route add** *destination gateway*[2]
静的経路の除去	`smit rmroute`	**route delete** *destination gateway*[2]
ルーティング・テーブルのフラッシュ	`smit fshrttbl`	**route flush**

注 :

1. テーブルは宛先アドレス、ゲートウェイ・アドレス、フラグ、参照カウント (ホップ・カウント)、ネットワーク・インターフェースの 各列に分かれています。 フレームが宛先に到達せず、かつルーティング・テーブルが正しい経路を指示している場合は、次の条件が 1 つ以上存在する可能性があります。
   - ネットワークに障害がある。
   - リモート・ホストまたはゲートウェイに障害がある。
   - リモート・ホストまたはゲートウェイがダウンしているか、フレームを受信する準備が整っていない。
   - リモート・ホストに送信元ネットワークに戻るための経路がない。
2. *destination* 値は宛先のホストまたはネットワークの小数点付き 10 進数アドレスかシンボル名で、*gateway* 値はこのゲートウェイの小数点付き 10 進数アドレスかシンボル名です。 (デフォルトの経路は、宛先として 0 を指定します。)

経路の使用制限

経路に制限を付けて、一部のユーザーだけがその経路を使用できるようにすることができます。 この制限は、ユーザーの 1 次グループ ID が基礎となります。

route コマンドを使用すると、ある経路の使用を許可された (または許可されない) 最大 32 個までのグループ ID を、1 つのリストにして指定できます。 そのリストが、許可されたグループのリストである場合、リストにあるいずれかのグループに属するすべてのユーザーは、その経路を使用できます。 そのリストが、許可されないグループのリストである場合、リストにあるいずれかのグループに属さないユーザーだけがその経路を使用できます。 root ユーザーは、すべての経路を使用できます。

ifconfig コマンドを使用して、あるインターフェースにグループを 関連付けることもできます。 その場合、転送可能パケットは、その着信インターフェースへ 関連付けられているグループに許可されたすべての経路を使用できます。

同じ宛先まで到達するのに 2 つ以上の経路が存在する場合、その宛先用に受信されたすべての ICMP リダイレクトは無視され、それらの経路に対してパス MTU ディスカバリーは行われません。

デッド・ゲートウェイの検出

使用しているゲートウェイがダウンしているかどうかを検出し、それに応じてルーティング・テーブルを調整するように、ホストを構成することができます。

ネットワーク・オプション **-passive_dgd** が 1 の場合、パッシブ・デッド・ゲートウェイ検出がシステム全体に対して使用可能になります。 ゲートウェイへの **dgd_packets_lost** 連続 **ARP** 要求に対する応答が受信されない場合、そのゲートウェイはダウンしていると見なされ、そのゲートウェイを使用するすべての経路の距離メトリック (ホップ・カウント またはコスト とも呼ばれる) が最大可能値に引き上げられます。 **dgd_retry_time** に指定されている時間 (分) が経過すると、経路のコストはユーザー構成値に復元されます。 また、ホストは障害が起こった **TCP** 接続に基づいて、アクションを実行します。 **dgd_packets_lost** 連続 **TCP** パケットが失われた場合、使用中のゲートウェイに関する **ARP** エントリーが削除され、**TCP** 接続は次の最善経路を試行します。 そのゲートウェイが、次回の使用時に実際にダウンしていると、上記のアクションが実行されます。 **passive_dgd**、**dgd_packets_lost**、および **dgd_retry_time** パラメーターは、すべて **no** コマンドを使用して構成できます。

route コマンドの **-active_dgd** フラグを使用して、アクティブ・デッド・ゲートウェイ検出を経路ごとに実行するようにホストを構成することもできます。 アクティブ・デッド・ゲートウェイ検出は、**dgd_ping_time** で指定されているインターバル (秒) で使用可能になり、対象となる経路で使用されているすべてのゲートウェイに PING します。 ゲートウェイから応答がない場合、アクティブ・デッド・ゲートウェイ検出は、**dgd_packets_lost** に指定されている最大回数までそのゲートウェイに PING を繰り返します。 それでも無応答の場合、そのゲートウェイを使用するすべての経路のコストが引き上げられます。そのゲートウェイへの PING が続けられ、最終的に応答が受信されると、経路のコストがユーザー定義値に復元されます。 **dgd_ping_time** パラメーターは、**no** コマンドを使用して構成できます。

デッド・ゲートウェイ検出は、動的経路指定ではなく静的経路指定を使用するホストに対して最も有効です。 パッシブ・デッド・ゲートウェイ検出は、パフォーマンスの問題がより少ないので、冗長ゲートウェイがあるすべてのネットワークでの使用をお勧めします。ただし、パッシブ・デッド・ゲートウェイ検出は、ベストエフォート・ベースでのみ行われます。 **UDP** などのプロトコルでは、データ送信が失敗している場合、ホストにフィードバックを戻しません。この場合、パッシブ・デッド・ゲートウェイ検出ではアクションは実行されません。

ホストがゲートウェイがダウンしたことをすぐに発見しなければならない場合は、アクティブ・デッド・ゲートウェイ検出が最も有効です。 アクティブ・デッド・ゲートウェイ検出は、数秒おきに使用可能になり、対象の各ゲートウェイに照会するので、その使用に応じてネットワーク使用量が多少増えます。アクティブ・デッド・ゲートウェイ検出は、クリティカルなサービスを提供するホスト、およびホスト数が限定されているネットワークに対してのみ使用することをお勧めします。

注 : デッド・ゲートウェイ検出と **gated** デーモンおよび **routed** デーモンで使用される経路指定プロトコルは、ネットワーク構成で変更を検出し、それに応じてルーティング・テーブルを調整することで同様の機能を実行します。 ただし、双方は異なるメカニズムを使用して上記の機能を実行し、さらに双方が同時に実行される場合は、互いに矛盾が生じる可能性があります。 このため、デッド・ゲートウェイ検出は、**gated** や **routed** デーモンを実行しているシステム上で使用してはなりません。

デッド・ゲートウェイ検出で 1 次経路がオンラインに戻ったことが検出されたときに、**dgd_flush_cached_route** パラメーターが使用可能に設定されている場合は、アクティブなすべての 接続の

キャッシュされた現行経路がフラッシュされます。 データの送信に最良の経路を検出するために、アクティブなすべての現行接続の経路が再度検証されます。**dgd_flush_cached_route** パラメーターは、**no** コマンドを使用して構成できます。デフォルトでは、**dgd_flush_cached_route** パラメーターは使用不可になっています。

注 : dgd_flush_cached_route パラメーターは、安定したネットワーク環境でのみ使用可能に設定する必要があります。そうでない場合、不良または不安定なハードウェア・ルーターによるパフォーマンスの問題がより大きくなり、デッド・ゲートウェイ検出で頻繁にルーティング・テーブルが更新される可能性があります。キャッシュされた経路の頻繁なフラッシュは、コストも大きくなります。

経路クローン作成

経路クローン作成により、システムが通信するそれぞれのホストに対するホスト経路を作成できます。

ネットワーク・トラフィックが送信される直前にルーティング・テーブルの中で検索が行われ、そのホストへの経路が検出されます。 特定のホスト経路が検出された場合は、その経路が使用されます。 特定のホスト経路が検出されなければ、ネットワーク経路またはデフォルト経路が検出されます。 検出された経路にクローン作成フラグ 'c' が設定されている場合、宛先のホスト経路は、クローン作成された経路からのゲートウェイを使用して作成されます。 その宛先のその後のルーティング・テーブルの検索で、クローン作成されたホスト経路が検出されます。 クローン作成された経路には 'W' フラグが設定されます。 これらの経路は、*route_expire* 分間使用されなければ、タイムアウトになり、ルーティング・テーブルから削除されます。 *route_expire* は、**no** コマンドを使用して変更できます。

経路クローン作成機能は、AIX オペレーティング・システム内で、主にパス MTU ディスカバリー・プロトコルが各通信先のパス MTU 情報を追跡するために使用します。 ネットワーク・オプション **tcp_pmtu_discover** または **udp_pmtu_discover** (**no** コマンドで設定できる) が 1 であれば、システム上のすべてのネットワーク経路のクローン作成フラグがオンにされます。 パス MTU ディスカバリー・プロトコルは、デフォルトでオンになります。

注 : クローン作成経路項目を手動で追加する場合は、**route** コマンドを使用して、ルーティング・テーブルを操作することができます。

関連情報

route コマンド

動的経路除去

routed デーモンを使用している場合は、着信 RIP 情報が手動で除去された経路の代わりになることはありません (ioctl が使用されているため)。

gated デーモンを使用しており、**-n** フラグを使用していない場合は、着信 RIP 情報の中で発見された経路が手動で除去された経路の代わりになります。

routed デーモンの構成

routed デーモンを構成するには、以下の手順を使用します。

routed デーモンを構成するには、次のようにします。

1. `/etc/rc.tcpip` シェル・スクリプトのコメント記号 (*#*) を外し、routed 文節を修正します。
   これによって、システムが始動するたびに **routed** デーモンが自動的に始動します。
   - ゲートウェイをパッシブ・モード (**-s** フラグ) とアクティブ・モード (**-q** フラグ) のどちらで実行するかを指定します。
   - パケット・トレースをオンとオフのどちらにするか (**-t** フラグ) を指定します。 パケット・トレースは、**routed** デーモンを始動した後でも、**kill** コマンドを使用してそのデーモンに **SIGUSR1** シグナルを送信することでオンにできます。 また、このシグナルを使用してトレースのレベルを 4 レベルまで上げることもできます。 さらに、**kill** コマンドを使用して **routed** デーモンに **SIGUSR2** シグナルを送信することで、そのデーモンの実行中にパケット・トレースをオフにできます。 詳しくは、**routed** デーモンと **kill** コマンドを参照してください。
   - デバッグをオンとオフのどちらにするか (**-d** フラグ) を指定します。 このフラグを使用する場合は、デバッグ情報を保管するログ・ファイルを指定するか、デバッグ情報をコンソール・ディスプレイへ送信する方法を指定するかを選択します。

- ゲートウェイ上で **routed** デーモンを実行するかどうか (**-g** フラグ) を指定します。

  注 : ゲートウェイでないホストでも **routed** デーモンを実行できますが、パッシブ・モードで実行しなければなりません。

2. 既知のネットワークがあれば、/etc/networks ファイルの中でリストにして識別します。

   詳しくは、ファイル参照 の Networks File Format for TCP/IP を参照してください。 サンプルの networks ファイルが、/usr/samples/tcpip ディレクトリーにあります。

3. /etc/gateways ファイルの中で、ネットワークに直接接続していない既知のゲートウェイへの経路をセットアップします。

   /etc/gateways ファイル内のエントリーの詳細例については、ファイル参照 の Gateways File Format for TCP/IP を参照してください。 サンプルの gateways ファイルが /usr/samples/tcpip ディレクトリーにあります。

重要 : routed デーモンと **gated** デーモンを同じマシン上で実行しないでください。 予測不能な結果となる可能性があります。

gated デーモンの構成

gated デーモンを構成する場合は、システムに最適なゲートウェイ・プロトコルを決定する必要があります。

gated デーモンを構成するには、次のようにします。

1. どのゲートウェイ・プロトコルがシステムに最適であるかを判断します。

   選択できる経路指定プロトコルは、**EGP**、**BGP**、**RIP**、**RIPng**、**HELLO**、**OSPF**、**ICMP/Router Discovery**、および **IS-IS** です。 **SNMP** も使用でき、これを使用するとリモート・ホストからネットワーク・エレメントの管理情報を変更したり表示したりできます。

   注 : 自律システム内のネットワークのアドレスを他の自律システム内のゲートウェイに公示するには、**EGP**、**BGP**、または **BGP4+** を使用します。 インターネット上では、必ず **EGP**、**BGP**、または **BGP4+** を使用して、ネットワークの到達可能性をコア・ゲートウェイ・システムに公示しなければなりません。 ある 1 つの自律システム内で到達可能性を公示するには、内部経路指定プロトコルを使用します。

2. 既知のネットワークがあれば、/etc/networks ファイルの中でリストにして識別します。

   詳しくは、ファイル参照 の Networks File Format for TCP/IP を参照してください。 サンプルの networks ファイルが、/usr/samples/tcpip ディレクトリーにあります。

3. /etc/gated.conf ファイルを編集し、求める **gated** デーモン構成を反映させます。

   注 : AIX 4.3.2 以降の gated デーモン のバージョンは 3.5.9 です。 /etc/gated.conf ファイルの構文が変更されました。 以下の例は、3.5.9 バージョンの **gated** 用のものです。 AIX 4.3.2 より前のバージョンの /etc/gated.conf ファイルを構成するには、/etc/gated.conf ファイル内に指定されている構文を使用します。

   a) 求めるトレース出力のレベルを指定します。 gated.conf ファイルを構文解析する前にトレースが必要な場合は、**-t** フラグを使用してデーモンの始動時にトレースをオンにします。

   b) 使用したい経路指定プロトコルを指定します。

   個々のプロトコルには、独自のプロトコル・ステートメントがあります。 コメント記号 (#) を外し、使用したいプロトコルに対応するステートメントを修正してください。

   - **EGP** を使用する場合:
     - **EGP** の autonomoussystem 文節をセットアップします。 インターネット上にいない場合は、ネットワーク上にある他のシステムの自律システム番号を考慮しながら、自律システム番号を割り当ててください。
     - **EGP** ステートメントを「yes (はい)」に設定します。
     - 個々の自律システムに 1 つずつ group 文節をセットアップします。
     - その自律システム内にある隣接ごとに 1 つずつ neighbor 文節をセットアップします。 以下に例を示します。

```
autonomoussystem 283 ;

egp yes {
        group maxup 1 {
             neighbor nogendefault 192.9.201.1 ;
             neighbor nogendefault 192.9.201.2 ;
       } ;
       group {
             neighbor 192.10.201.1 ;
             neighbor 192.10.201.2 ;
       } ;
} ;
```

- **RIP** または **HELLO** を使用する場合:
  - **RIP** または **HELLO** ステートメントを「yes（はい)」に設定します。
  - ゲートウェイが経路指定情報を受け入れるだけで、情報をブロードキャストしないようにしたい場合は、**RIP** または **HELLO** ステートメント内に nobroadcast を指定します。 あるいは、ゲートウェイが経路指定情報を受け入れるだけでなく、経路指定情報をブロードキャストするようにしたい場合は、**RIP** または **HELLO** ステートメント内に broadcast を指定します。
  - ゲートウェイから送信元ゲートウェイに直接送信するには、sourcegateways ステートメントを使用します。 sourcegateways 文節内に、ゲートウェイ名またはインターネット・アドレス (ドット 10 進数) で指定します。 以下に例を示します。

    ```
    # Send directly to specific gateways

    rip/hello yes {
              sourcegateways
                    101.25.32.1
                    101.25.32.2 ;
    } ;
    ```

    次の例は、**RIP** パケットを送信せず、tr0 インターフェースで **RIP** パケットを受信しないマシンの gated.conf ファイル内に記述されている **RIP**/**HELLO** スタンザを示します。

    ```
    rip/hello nobroadcast {
            interface tr0 noripin ;
    } ;
    ```

- **BGP** を使用する場合:
  - **BGP** の autonomoussystem 文節をセットアップします。 インターネット上にいない場合は、ネットワーク上にある他のシステムの 自律システム 番号を考慮しながら、自律システム 番号を割り当ててください。
  - **BGP** ステートメントを「yes（はい)」に設定します。
  - その自律システム 内にある隣接ごとに 1 つずつ peer 文節をセットアップします。 以下に例を示します。

    ```
    # Perform all BGP operations

    bgp yes {
            peer 192.9.201.1 ;
    } ;
    ```

- **SNMP** を使用する場合:
  - **SNMP** ステートメントを「yes（はい)」に設定します。

    ```
    snmp yes ;
    ```

IPv6 を実行するための gated デーモンの構成

インターネット・プロトコル・バージョン 6 (IPv6) を実行するために **gated** デーモンを構成するには、以下の手順を使用します。

インターネット・プロトコル・バージョン 6 (IPv6) のもとで実行するために **gated** デーモンを構成するには、最初に、ユーザーのシステムが **IPv6** と **IPv6** 経路指定用に構成されていることを確認します。

1. **autoconf6** を実行して、インターフェースを **IPv6** に合わせて自動的に構成します。
2. **IPv6** 経路指定を使用したい **IPv6** インターフェースごとにサイト 内ローカル・アドレスを構成するために、次のコマンドを使用します。

   ```
   ifconfig interface inet6 fec0:n::address/64 alias
   ```

   この場合、

   interface
   インターフェースの 名前。例えば、tr0 または en0。

   n
   任意の 10 進数。例えば、11。

   address
   2 つのコロンに続く、**IPv6** インターフェース・アドレスの部分。例えば、**IPv6** アドレスが fe80::204:acff:fe86:298d の場合、*address* エントリーは 204:acff:fe86:298d になります。

   注 : コマンド **netstat -i** を使用すると、構成済みのインターフェースごとの **IPv6** アドレスを表示できます。

   トークンリング tr0 の **IPv6** アドレスが fe80::204:acff:fe86:298d である場合は、次のコマンドを使用します。

   ```
   ifconfig tr0 inet6 fec0:13::204:acff:fe86:298d/64 alias
   ```

3. 次のコマンドを使用して **IPv6** 転送をオンにします。

   ```
   no -o ip6forwarding=1
   ```

4. 次のコマンドを使用して **ndpd-router** を始動します。

   ```
   ndpd-router -g
   ```

   ndpd-router を始動すると、ユーザーのシステムは**隣接ディスカバリー・プロトコル**のルーターとして機能します。 **隣接ディスカバリー・プロトコル**のルーターは隣接ディスカバリー・ホストに経路指定情報を通知するため、ホストは **IPv6** パケットを経路指定できるようになります。

   IPv6 ネットワークの一部にしたいネットワーク上のすべてのホストが **ndpd-host** を実行する必要があります。 **ndpd-host** を実行するネットワーク上の各ホストは、それ自体を **IPv6** ネットワークの一部として認識し、**隣接ディスカバリー・プロトコル**を使用します。このプロトコルにより、隣接経路指定を可能にし、パケット転送用の隣接ルーターを検索するためのリンク層アドレスの判別とモニターを行うことができます。
5. 次に、**gated** デーモンを構成します。
   a) システムに最適な **IPv6** ゲートウェイ・プロトコルを決定します。

   IPv6 経路指定プロトコルに選択できるのは、**IPv6** 用に拡張された**ボーダー・ゲートウェイ・プロトコル** (**BGP4+**) と **Routing Information Protocol Next Generation** (**RIPng**) です。

   b) /etc/gated.conf ファイルを編集し、求める **gated** デーモン構成を反映させます。

   注 : AIX 4.3.2 以降は **gated** バージョン 3.5.9 を実行します。 gated.conf ファイルの構文は、旧バージョンと多少異なります。 正しい構文については、 ファイル参照 の gated.conf の資料を参照してください。/usr/sample/tcpip ディレクトリーに入っているサンプル・ファイルを使用することもできます。

BGP4+ または **RIPng** を構成する場合は、**IPv6** アドレスを使用してください。この構文により、IP アドレスが指定されます。

注 : デフォルトにより、**RIPng** はそのパケットをマルチキャストします。

/etc/gated.conf ファイルを変更した後、**gated** デーモンを始動することができます。

自律システム番号
EGP または **BGP** を使用する場合は、使用するゲートウェイ用に公認の自律システム番号 を入手する必要があります。

公認の自律システム番号を入手するには、次のインターネット・アドレスから NIC に連絡してください。

```
INFO@INTERNIC.NET
```

モバイル IPv6

モバイル **IPv6** は、**IPv6** のモバイル機能をサポートします。 これによって、同じインターネット・アドレスを世界中のどこに居ても保持することができ、ロケーションを変えたときに、そのアドレスを使用しているアプリケーションはトランスポート層と上位層との接続を維持できます。 これによって、同種、および異種のメディアに渡るモバイル機能が可能になります。

例えば、モバイル **IPv6** によって、モバイル・ノードの IP アドレスを変更しないでイーサネット・セグメントから無線 LAN セルにノードを移動できます。

モバイル **IPv6** では、各モバイル・ノードはホーム・アドレスとケアオブ・アドレスの 2 つの IP アドレスによって識別されます。 ホーム・アドレスは、モバイル・ノードがどのロケーションにあってもそのノードを識別する永続 IP アドレスです。 ケアオブ・アドレスは、新たに接続される各ポイントで変わり、モバイル・ノードの現行状態の情報を提供します。 モバイル・ノードは、次のネットワークに到達したときに、到達先のケアオブ・アドレスを獲得する必要があります。そのケアオブ・アドレスは、モバイル・ノードが到達先ネットワークのこのロケーションの下にある間、使用されます。 **IPv6** 隣接ディスカバリーのメソッドを使用して、ケアオブ・アドレスを獲得できます (130 ページの『IPv6 拡張経路指定とアドレッシング』を参照)。 ステートレスとステートフルの 両方の自動構成が可能です。 ケアオブ・アドレスは手動で構成することもできます。 ケアオブ・アドレスがどのように獲得されるかは、モバイル **IPv6** とは関係ありません。

ホーム・ネットワーク上には少なくとも 1 つのホーム・エージェントが構成されている必要があります。また、モバイル・ノードも構成されて、既知のホーム・エージェントの IP アドレスが分かるようになっている必要があります。 モバイル・ノードは、バインディング更新が入ったパケットをホーム・エージェントに送信します。 ホーム・エージェントは、そのパケットを受け取り、モバイル・ノードのホーム・アドレスと受け取ったケアオブ・アドレスとの間を関連付けます。 ホーム・エージェントはバインディング確認が入ったパケットを応答します。

ホーム・エージェントは、サービスを提供するモバイル・ノードのホーム・アドレスとケアオブ・アドレスとの関連付けが入ったバインディング・キャッシュを保持します。 ホーム・エージェントは、ホーム・アドレスが宛先となっているパケットを代行受信して、それをモバイル・ノードに転送します。 そして、モバイル・ノードはバインディング更新をコレスポンデント・ノードに送信してモバイル・ノードのケアオブ・アドレスを通知し、コレスポンデント・ノードはバインディング・キャッシュ・エントリーを作成して、このあとのトラフィックをモバイル・ノードのケアオブ・アドレスに直接送信できるようにします。

AIX のモバイル・サポートでは、次の機能が提供されます。

ホーム・エージェント・ノードでは:

- サービスを提供している各モバイル・ノードのバインディング・キャッシュ内にあるエントリーを保持します。
- 現在ホーム・エージェントとしてサービスを提供しているモバイル・ノードが宛先となっているパケットについて、モバイル・ノードがホームから離れている間は、そのモバイル・ノードのホーム・リンク上でそのパケットを代行受信します。
- そのように代行受信したパケットを、ホーム・エージェントのバインディング・キャッシュ内のバインディングに示されたモバイル・ノードの 1 次ケアオブ・アドレスにトンネルするために、カプセル化します。

- 確認のビットが設定されている受信したバインディング更新オプションの応答として、バインディング確認オプションを戻します。
- モバイル・ノードのケアオブ・アドレスに関する重複アドレスの検出を処理し、**IPv6** アドレスが固有であることを確認します。
- 動的ホーム・エージェント・アドレス・ディスカバリーをサポートして、モバイル・ノードによるホーム・エージェントのアドレス検出を支援します。
- モバイル・プレフィックス 請求の受信と、モバイル・プレフィックス 公示の送信をサポートします。

ステーショナリー・コレスポンデント・ノードでは:

- **IPv6** パケットで受信したホーム・アドレス・オプションを処理します。
- パケットで受信したバインディング更新オプションを処理し、受信したバインディング更新で確認 (A) ビットが設定されていれば、バインディング確認オプションを戻します。
- 受け入れたバインディング更新で、受信したバインディングのバインディング・キャッシュを保持します。
- モバイル・ノードのバインディング・キャッシュ・エントリーにモバイル・ノードの現行ケアオブ・アドレスが含まれていれば、経路指定ヘッダーを使用してパケットを送信します。

モバイル・ノードの到達先のネットワーク内の**ルーター**・ノードでは:

- ルーター公示内の公示インターバル・オプションを送信して、モバイル・ノードによる移動検出を支援します。 これは、**ndpd-router** デーモン内の **-m** パラメーターによって構成できます。
- RFC 2461 に説明されている高速での非送信請求マルチキャスト・ルーター公示の送信をサポートします。 これは、**ndpd-router** デーモン内の **-m** パラメーターと **-D** パラメーターによって構成できます。
- ルーター公示内のホーム・エージェント情報オプション (ホーム・エージェントの設定とライフタイム) を送信して、モバイル・ノードによるホーム・エージェント選択を支援します。 これは、**ndpd-router** デーモン内の **-H** パラメーターによって構成できます。

モバイル IPv6 のセキュリティー

モバイル・ノードとホーム・エージェントの間で交換されるバインディング更新メッセージとバインディング確認メッセージは、ヌル以外のペイロード認証アルゴリズムのカプセル化セキュリティー・ペイロード (ESP) 保護を使用した、IP セキュリティーによって保護しなければなりません。

IP セキュリティーについて詳しくは、セキュリティーを参照してください。

モバイル・ノードとコレスポンデント・ノードの間のバインディング確立は、Return Routability プロシージャーを使用して保護できます。 このプロシージャーでは、ホーム・エージェント・ノードとモバイル・ノードの間で交換されるメッセージも、ESP を使用した IP セキュリティーにより保護する必要があります。 コレスポンデント・ノードとモバイル・ノードの間で交換されるバインディング更新メッセージとバインディング確認メッセージは Return Routability プロシージャーによって保護されるので、コレスポンデント・ノードには IP セキュリティーは必要ありません。 しかし、コレスポンデント・ノードで IP セキュリティーを使用してアクセスを制限する場合は、プロトコル MH (135) を使用したメッセージを許可しなければなりません。

手動で、または応答側の働きをする IKE を使用して、トンネルを定義できます (アグレッシブ・モードのみサポートされます)。 少なくとも以下の IP セキュリティー・トンネルを、ESP ヘッダーを使用してホーム・エージェント上で定義します。

- ホーム・エージェントの IP アドレスと、このホーム・エージェント上で登録できる個々のモバイル・ノードのホーム・アドレスの間のトンネル。プロトコル MH (135) を使用したトランスポート・モードです。
- 任意の IP アドレスと、このホーム・エージェント上で登録できる個々のモバイル・ノードのホーム・アドレスの間のトンネル。プロトコル MH (135) を使用したトンネル・モードです。

対応するトンネルをモバイル・ノード上で定義しなければなりません。

注 : バインディング更新メッセージとバインディング確認メッセージはモバイル機能ヘッダーを使用して送信されるので、ESP を使用した IP セキュリティーによって保護しなければなりません。

AIX でのモバイル IPv6 の旧インプリメンテーションでは、宛先オプション・パケットを使用してバインディング更新メッセージを送信するモバイル・ノードに関するサポートが提供されていました。 これらのメッセージは、認証ヘッダーを使用した IP セキュリティーにより保護できました。

ホーム・エージェントまたはコレスポンデント・ノードが、宛先オプションを使用したこの種のバインディング更新メッセージを受け入れるようにするには、/etc/rc.mobip6 ファイルを編集し、**Enable_Draft13_Mobile** 変数を使用可能にしてから、モバイル **IPv6** を開始してください。 この場合、IP セキュリティーを使用してバインディング更新メッセージを保護するには、プロトコル 60 のトランスポート・モードの手動トンネルまたは IKE トンネルを定義しなければなりません。 こうすると、バインディング更新メッセージとバインディング確認メッセージが保護されます。

ホーム・エージェントまたはコレスポンデント・ノードが、IP セキュリティーによって保護されないバインディング更新メッセージを受け入れるようにするには、/etc/rc.mobip6 ファイルを編集し、**Check_IPsec** 変数を使用不可にしてください。 モバイル・ノードにアドレス指定されたパケットの経路指定に影響する機能があるので、セキュリティーが非常にぜい弱になるため、この方式はお勧めしません。

モバイル IPv6 の構成
ここでは、モバイル **IPv6** の構成について説明します。 モバイル **IPv6** を使用するには、まず bos.net.mobip6.rte ファイルセットをインストールしておく必要があります。

ファイルセットのインストールについては、インストールおよび移行のオプショナル・ソフトウェア製品および保守更新を参照してください。

ホーム・エージェントのモバイル *IPv6* の始動
ホーム・エージェントとしてモバイル **IPv6** を始動するには、以下の手順を使用します。

1. コレスポンデントが通信できる、ホーム・エージェント IP アドレスと各モバイル・ホーム・アドレスとの間の IKE トンネル (フェーズ 1 および 2、**ESP** プロトコルを使用する応答側として定義する) または手動 ESP IP セキュリティー関連を定義します。
2. システムをモバイル **IPv6** ホーム・エージェントおよびコレスポンデント・ノードとして使用可能にします。 コマンド・ラインで、smit enable_mobip6_home_agent と入力します。
3. 使用可能にしたい時点を選択します。

コレスポンデントのモバイル *IPv6* の始動
コレスポンデントとしてモバイル IPv6 を始動するには、以下の手順を使用します。

1. コレスポンデントが通信できる、ホーム・エージェント IP アドレスと各モバイル・ホーム・アドレスとの間の IKE トンネル (フェーズ 1 および 2、**ESP** プロトコルを使用する応答側として定義する) または手動 ESP IP セキュリティー関連を定義します。
2. システムをモバイル **IPv6** コレスポンデント・ノードとして使用可能にします。 コマンド・ラインで、smit enable_mobip6_correspondent と入力します。
3. 使用可能にしたい時点を選択します。

ルーターのモバイル *IPv6* の始動
ルーターとしてモバイル **IPv6** を始動するには、以下の手順を使用します。

移動検出のために、次のコマンドを実行します。

```
ndpd-router -m
```

モバイル *IPv6* の停止
モバイル **IPv6** を停止するには、以下の手順を使用します。

1. コマンド・ラインに smit disable_mobip6 と入力します。
2. モバイル **IPv6** を停止したい時点を選択します。
3. **ndpd-router** デーモンを停止するかどうかを選択します。
4. **IPv6** 転送を使用不可にするかどうかを選択します。

モバイル IPv6 のトラブルシューティング

モバイル **IPv6** のトラブルシューティングを 行う場合は、**mobip6ctrl -b** コマンドを使用します。

1. 次を実行して、バインディング状態を取得します。

```
mobip6ctrl -b
```

2. **TCP/IP** トラブルシューティング ・ユーティリティーの 使用については、474 ページの『TCP/IP のトラブルシューティング』を参照してください。

仮想 IP アドレス

仮想 IP アドレスによって、ホストは個々のネットワーク・インターフェースに 左右されなくなります。

着信パケットはシステムの VIPA アドレスに送信されますが、すべてのパケットは実際のネットワーク・インターフェースを 通って移動します。

これまでは、インターフェースに 障害が発生すると、そのインターフェースへの 接続はいずれも切れてしまいました。 システムの VIPA と自動転送を提供するネットワーク内の経路指定プロトコルにより、パケットが別の物理インターフェースを 通して到達できる限り、仮想インターフェースを 使用する既存のユーザー接続を切断せずに障害からリカバリーします。 VIPA が稼働中のシステムでは、アダプター 障害がアクティブな接続に影響を与えるということがなくなるため、使用可能度が一層高くなります。 複数の物理アダプターによってシステム IP トラフィックが送信されるため、全体の負荷が 1 つのアダプターとそれに関連したサブネットに集中することがありません。

AIX VIPA 機能は、ネットワーク装置に対して透過的です。 特殊なネットワーク装置または他のハードウェアは必要ありません。 VIPA を実装するには、次のものが必要です。

- 会社のネットワークに接続される異なるサブネット上に、既存の 2 つ以上の物理タイプの IP インターフェース
- 会社のネットワーク内部で稼働する IP 経路指定プロトコル

VIPA の構成

VIPA は、IP ネットワーク・インターフェースと 同じように SMIT 内に定義できます。 さらに、VIPA を構成する間に、インターフェースのグループを 指定することもできます。

このように構成すると、これらのインターフェースを 介して VIPA ホストが開始する すべての発信接続は、VIPA の使用を指定され、仮想アドレスは発信パケットの **TCP/IP** パケット・ヘッダーに置かれた発信元アドレスになります。

1. IPv4 VIPA の場合、コマンド・ラインに `smit mkinetvi` と入力します。 IPv6 VIPA の場合、コマンド・ラインに `smit mkinetvi6` と入力します。
2. 必要なフィールドに値を入れます。 追加情報については、424 ページの『VIPA 環境の例』を参照してください。 Enter キーを押します。

VIPA へのアダプターの追加

仮想 IP アドレスにアダプターを 追加するには、以下の手順を使用します。

アダプターをユーザーの VIPA インターフェースに 追加するには、次のステップを実行してください。

1. コマンド・ラインに `smit chvi` と入力します。
2. アダプターを追加する VIPA を選択して、Enter キーを押します。
3. 追加するアダプターを 「**Interface Name(s) (インターフェース 名)**」フィールドに入力します。
4. 「**ADD/REMOVE interface(s) (インターフェースの 追加/除去)**」フィールドに ADD と入力して、Enter キーを押します。

VIPA からのアダプターの除去

仮想 IP アドレスからアダプターを 除去するには、以下の手順を使用します。

VIPA からアダプターを除去するには、次のステップを実行します。

1. コマンド・ラインに `smit chvi` と入力します。
2. そこからアダプターを除去する VIPA を選択して、Enter キーを押します。

3. 削除するアダプターを「**Interface Name(s) (インターフェース 名)**」フィールドに入力します。
4. 「**ADD/REMOVE interface(s) (インターフェースの 追加/除去)**」フィールドに REMOVE と入力して、Enter キーを押します。

VIPA 環境の例
イーサネット接続での VIPA 環境を示す以下の例では、仮想 IP アドレスと 2 つの物理接続を持つシステムを使用します。

システムは、仮想 IP アドレス vi0 (10.68.6.1)、および 2 つの物理接続 en1 (IP アドレス 10.68.1.1) と en5 (IP アドレス 10.68.5.1) を持っています。 この例では、物理接続はどちらもイーサネットですが、サブネットが最終的により大きな会社のネットワークに接続されて会社のルーターとして認識される限り、トークンリングまたは FDDI などの混合された IP インターフェースもサポートされます。

「**lsattr -El vi0**」コマンドを実行すると、次の結果が出力されます。

```
netaddr          10.68.6.1      N/A                                      True
state            up             Standard Ethernet Network Interface      True
netmask          255.255.255.0  Maximum IP Packet Size for This Device   True
netaddr6                        Maximum IP Packet Size for REMOTE Networks  True
alias6                          Internet Address                         True
prefixlen                       Current Interface Status                 True
alias4                          TRAILER Link-Level Encapsulation         True
interface_names  en1,en5        Interfaces using the Virtual Address     True
```

「**ifconfig vi0**」コマンドを実行すると、次の結果が出力されます。

```
vi0: flags=84000041<UP,RUNNING,64BIT>
       inet 10.68.6.1 netmask 0xffffff00
       iflist : en1 en5
```

「**netstat -rn**」コマンドを実行すると、次の結果が出力されます。

```
Routing tables
Destination       Gateway       Flags     Refs    Use    If     PMTU Exp Groups

Route Tree for Protocol Family 2 (Internet):
default           10.68.1.2      UG        3      1055   en1     -   -
10.68.1/24        10.68.1.1      U         0      665    en1     -   -
10.68.5/24        10.68.5.1      U         0      1216   en5     -   -
127/8             127.0.0.1      U         4      236    lo0     -   -
10.68.6.1         127.0.0.1      UH        0      0      lo0     -   -
```

発信元アドレス・セットを持っていなくて、インターフェース en1 と en5 を通して経路指定された発信パケットは、仮想アドレスの発信元アドレス (10.68.6.1) を持ちます。 着信パケットは、ネットワーク上に公示された VIPA アドレス (10.68.6.1) に経路指定されます。 vi0 が仮想 (すなわち、デバイスに関連付けられていない) であるため、**netstat -rn** コマンドを使用して表示されるシステム全体のルーティング・テーブルにはそのエントリーはありません。 つまり、インターフェースが SMIT 内に構成されるときに、インターフェース 経路は追加されないということです。

物理インターフェース、ネットワーク接続、あるいはネットワーク・パスの 1 つに障害があれば、ネットワーク・プロトコルは、同じシステム内の別の物理インターフェースに 経路指定されます。 リモート・システムが telnet で vi0 アドレスに接続すると、パケット vi0 は、en1 または en5 のどちらかを使用して到着します。 例えば en1 がダウンしたとしても、パケットは en5 に到着できます。 経路指定プロトコルは、経路の伝搬に時間がかかることもあることに注意してください。

VIPA を使用するとき、終端システムと 介在するルーターは、VIPA (vi0) 宛てのパケットを 物理インターフェース (en1 または en5) の 1 つに経路指定する必要があります。

VIPA と別名の比較
VIPA の概念は、アドレスがハードウェア・インターフェースと 関連付けられていないことを除けば、IP 別名と類似しています。

VIPA には IP 別名にはない幾つかの利点があります。

- VIPA は、物理インターフェースに 影響を及ぼさずに、独立してアップしたりダウンしたりできる、仮想デバイスを提供します。

- VIPA アドレスは変更できます。別名の場合は追加または削除ができるのみです。

実アダプターの IP アドレスを使用したアクセス
VIPA がインプリメントされたあとも、個々のインターフェースは 他のシステムにアクセス可能です。 しかし、ping および telnet セッションに実 IP アドレスを使用すると、物理アダプターに独立して通信できるという VIPA の利点を損なうことになります。 VIPA は、物理アダプター障害を外にあるクライアントには見せません。 実アドレスを使用することで、再び物理アダプターに依存することになります。

リモート・システムが VIPA アドレスを使用して VIPA システムにコンタクトするか、VIPA システム上のアプリケーションが 他のシステムとの通信を開始するとき、VIPA アドレスは、パケット内の送信元 IP アドレスとして使用されます。 しかし、リモート・システムが実インターフェースの IP アドレスを使用してセッションを開始するときは、その実 IP アドレスは応答パケット内の送信元 IP アドレスになります。 これには 1 つだけ例外があります。 特定の IP インターフェースにバインドされているアプリケーションの 場合、発信パケットには、それらの宛先となっているインターフェースの 発信元アドレスが入ります。

VIPA と経路指定プロトコル
gated デーモンは、インターフェース 経路を追加したり、公示を仮想インターフェースに 送信したりしないように、VIPA 用に修正されました。

gated でサポートされる **OSPF** プロトコルは、仮想インターフェースを 隣接するルーターに公示します。 ネットワーク上の他のホストは、最初にホップしたルーターを通して、VIPA ホストと会話できます。

複数の VIPA アドレス
複数の仮想インターフェースを 構成できます。 複数の VIPA インターフェースは、例えば、ある特定の VIPA アドレスに送信または受信される パケットに対して、ネットワーク・ルーターが優先的な待遇を与えることができれば役に立ちます。

あるいは、アプリケーションを特定の VIPA インターフェースにバインディングするならば 、複数の VIPA インターフェースを 使用するかもしれません。 例えば、1 つのマシンで複数の会社の複数の Web サーバーを実行するには、次のように構成することになります。

- vi0 200.1.1.1 www.companyA.com
- vi1 200.1.1.2 www.companyB.com
- vi2 200.1.1.3 www.companyC.com

イーサチャネル、IEEE 802.3ad リンク集約、チーミング

イーサチャネル、IEEE 802.3ad リンク集約、およびチーミングは、複数のイーサネット・アダプターを集合させて 1 つの疑似イーサネット・デバイスを形成できるようにするネットワーク・ポート集合テクノロジーです。

例えば、`ent0` と `ent1` を集合させて、`en3` というイーサチャネル・アダプターを形成することができます。この場合、インターフェース `en3` が IP アドレスを使用して構成されます。 システムは、これらの集合アダプターを 1 つのアダプターと見なします。 このため、他のイーサネット・アダプターと同様に、これらの集合アダプターにも IP が構成されます。 その上、イーサチャネルまたはリンク集約内のすべてのアダプターに同じハードウェア (MAC) アドレスが付与されるため、リモート・システムによって、1 つのアダプターかのように扱われます。スイッチ内ではイーサチャネルと IEEE 802.3ad リンク集約の両方のサポートが必要です。これにより、2 つのテクノロジーが、どのスイッチ・ポートを 1 つとして扱う必要があるかを認識できるようになります。

>|チーミング集約メカニズムでは、チャネルにおける各アダプターは、各アダプターのオリジナルのハードウェア (MAC) アドレスを保持します。 そのため、スイッチを構成する必要はありません。 イーサチャネルと IEEE 802.3ad リンク集約には、1 次チャネルとバックアップ・チャネルの両方を指定できます。 ただし、チーミング集約メカニズムでは 1 つのチャネルのみが使用されます。 チーミング・モードを選択した場合は、チャネルがパケットを送信する仕組みを制御できません。 そのチャネルは自動的にパケット送信に標準モードを使用します。 チーミング・モードは、イーサチャネルにおいてどのアダプターがトラフィックを受信するのかを制御するため、スイッチ構成は不要となります。|<

注 : イーサチャネル・ドライバーは、イーサチャネル構成の非アクティブなチャネルのホスト・イーサネット・アダプター (HEA) ポートに無効なメディア・アクセス制御 (MAC) アドレス `02:00:00:00:00:00` を割り当てます。この無効な MAC アドレスが割り当てられるのは、イーサチャネルの作成時、または HEA

ポートが実行時に非アクティブなチャネルに追加された場合です。イーサチャネルのフェイルオーバーまたはリカバリー時に無効な MAC アドレスが有効な MAC アドレスに交換され、実行時に有効な MAC アドレスが無効な MAC アドレスに交換されます。

イーサチャネルおよび IEEE 802.3ad リンク集約の主要な利点は、それらのすべてのアダプターのネットワーク帯域幅を単一ネットワーク内で保持することです。 いずれかのアダプターに障害が発生すると、既存のユーザー接続に影響を与えずに、ネットワーク・トラフィックは自動的に使用可能な別のアダプターに送信されます。 障害が生じたアダプターは、リカバリーが行われると、イーサチャネルまたはリンク集約でサービスを行うように自動的に戻されます。

イーサチャネル、IEEE 802.3ad リンク集約、およびチーミング集約メカニズムには、いくつかの違いがあります。 どのテクノロジーがお客様の要件に最適であるかを判別するには、426 ページの表 80 にリストされている相違点を考慮してください。

表 *80.* イーサチャネル、*IEEE 802.3ad* リンク集約、およびチーミングの違い

イーサチャネル	**IEEE 802.3ad リンク集約**	>|**チーミング**|<
スイッチの構成が必要です。	Link Aggregation Control Protocol Data Unit (LACPDU) 交換用のスイッチ構成が必要です。	>|スイッチ構成は不要です|<
スイッチ・ポートと隣接システム・ポートとの間でハートビートは交換されません。	IEEE 802.3ad 規格で定義されたインターバルで、ハートビート (LACPDU) が交換されます。ハートビートは障害における追加の保護を提供します。	>|スイッチ・ポートと隣接システム・ポートとの間でハートビートは交換されません。|<
1 次チャネルとバックアップ・チャネルの両方を使用できます。	1 次チャネルとバックアップ・チャネルの両方を使用できます。	>|単一 (1 次) チャネルのみが使用されます。|<

AIX オペレーティング・システムでは、動的アダプター・メンバーシップ機能が使用可能です。この機能を使用して、ユーザー接続を切断しなくても、イーサチャネルに対してアダプターの 追加や除去を行うことができます。

関連情報

動的アダプター・メンバーシップ

イーサチャネル

IEEE 802.3ad リンク集約の構成

インターオペラビリティーに関するシナリオ

イーサチャネル

イーサチャネルに属するアダプターは、イーサチャネルを使用できる同一のスイッチに接続しなければなりません。 アダプターが異なるスイッチに接続されている場合、それらのスイッチはスタックされて単一のスイッチとして機能する必要があります。

イーサチャネルに属するポートを 1 つの集合リンクとして扱うように、このスイッチを手動で構成しなければなりません。 スイッチについての文書では、 この機能のことをリンク集約 またはトランキング と呼んでいる場合があります。

イーサチャネルが正しく機能するためには、イーサチャネルを作成する前に、リンクの状況を定期的に検査するリンク・ポーリング・メカニズムを、アダプターごとに使用可能にしておく必要があります。トラフィックは、アダプターに、標準的な方法 (そこではパケットを送信するアダプターはアルゴリズムに 応じて選択される)、またはラウンドロビンをベースにした方法 (そこではパケットはすべてのアダプターに 等しく送信される) のどちらかで送信されます。 着信トラフィックはスイッチ構成に応じて配布され、イーサチャネル動作モードでは制御されません。

システムごとに複数のイーサチャネルを構成できます。 1 つのイーサチャネル上にあるすべてのリンクが単一のスイッチに接続される場合、しかも、スイッチのプラグが抜かれているかスイッチに障害がある場合は、イーサチャネル全体が失われます。 この問題を解決するために、メインのイーサチャネルに障害が発生してもサービスをアクティブに保持するバックアップ・オプションが使用可能です。バックアップ・

アダプターとイーサチャネル・アダプターは異なるネットワーク・スイッチに接続されている必要があり、このセットアップが適切に作動するには、それらのスイッチが相互接続されている必要があります。イーサチャネル内のすべてのアダプターに障害が発生すると、バックアップ・アダプターがすべてのトラフィックの送受信に使用されます。イーサチャネル内のリンクが 1 つでも復元されると、サービスはイーサチャネルに戻されます。

例えば、en3 というイーサチャネルを作成して、ent0 と ent1 をメインのイーサチャネル・アダプターとして構成し、ent2 をバックアップ・アダプターとして 構成できます。イーサチャネルを使用できる同一のスイッチに ent0 と ent1 を接続し、別のスイッチに ent2 を接続するのが理想です。この例では、en3 (イーサチャネルのインターフェース) を介して送信されるすべてのトラフィックは、デフォルトでは ent0 または ent1 (イーサチャネルのパケット配布方式に応じて異なる) を介して送信されますが、ent2 はアイドル状態になります。ent0 と ent1 の両方に障害が起こると、すべてのトラフィックはバックアップ・アダプター ent2 を介して送信されます。ent0 または ent1 がリカバリーされると、再度これらのアダプターを使用してすべてのトラフィックが送信されます。

Network Interface Backup (ネットワーク・インターフェース ・バックアップ) は、イーサチャネルに使用できる操作のモードの 1 つで、イーサネット・ネットワークのある 1 点での障害に対する保護を行います。Network Interface Backup (ネットワーク・インターフェース ・バックアップ) を使用する際には特別なハードウェアは必要ありませんが、信頼性を最大限にするには、バックアップ・アダプターを 個別のスイッチに接続する必要があります。 Network Interface Backup (ネットワーク・インターフェース ・バックアップ) モードでは、ネットワーク・トラフィックに対し、ある時点で 1 つのアダプターのみアクティブに 使用できます。 イーサチャネルは、現在アクティブなアダプターと 、オプションでユーザー指定ノードへのネットワーク・パスをテストします。 障害が検出されると、次のアダプターを 使用してすべてのトラフィックが送信されます。 Network Interface Backup (ネットワーク・インターフェース ・バックアップ) によって、ユーザー接続を切断せずに、検出およびフェイルオーバーが実行されます。 ネットワーク・インターフェース・バックアップは、元々イーサチャネルのシステム 管理インターフェース・ツール (SMIT) メニュー内のモードとして実装されました。バックアップ・アダプターによって 同等の機能が提供されるため、このモードは SMIT メニューから除去されました。 ネットワーク・インターフェース ・バックアップを構成するには、432 ページの『ネットワーク・インターフェース ・バックアップの構成』を参照してください。

イーサチャネル構成における考慮事項

イーサチャネルを構成する前に、以下の推奨事項を確認してください。

- イーサチャネル当たり最大 8 つの 1 次イーサネット・アダプターと 最大 8 つのバックアップ・イーサネット・アダプターを 使用できます。
- 1 つのシステムに複数のイーサチャネルを構成できますが、各イーサチャネルは追加のイーサネット・インターフェースを 構成します。 各アダプターのイーサネット ・インターフェースだけでなく 、構成するイーサチャネルもすべて含めるために、**no** コマンドの **ifsize** オプションを大きくすることが必要になることもあります。 AIX 5.2 以前では、デフォルトの **ifsize** は 8 です。 デフォルトのサイズは 256 です。
- サポートされているイーサネット・アダプターは、いずれもイーサチャネル内で使用できます (444 ページの『サポートされているアダプター』を参照)。 しかし、イーサネット・アダプターは、イーサチャネルをサポートするスイッチに接続されている必要があります。 ユーザーのスイッチがイーサチャネルをサポートするかどうかは、スイッチに付いてくる資料を参照して判断してください (ご使用のスイッチの資料では、この機能のことをリンク集約またはトランキング機能と言及している可能性もあります)。
- イーサチャネル内のすべてのアダプターは、同じ速度 (例えば 100 Mbps) に構成する必要があり、さらに全二重モードである必要があります。
- イーサチャネルで使用されるアダプターは、イーサチャネルの構成後にシステムがアクセスすることはできません。 メディア・スピードや送信または受信キューのサイズなどの属性を変更するには、イーサチャネルに組み込む前に変更しなければなりません。
- この手順を始める前に、イーサチャネルで使用する予定のアダプターに、構成された IP アドレスがあってはなりません。 以前に IP アドレスを指定して構成されたアダプターを 指定してイーサチャネルを構成する際には、インターフェースが detach 状態であることを確認してください。 イーサチャネルに追加されるアダプターは、オブジェクト・データ・マネージャー (ODM) 内で up 状態で構成されたインターフェースをもつことはできません。これは IP アドレスが SMIT を使用して構成された場合に生じる可能性があります。 これにより、マシンのリブート時にイーサチャネルの始動に問題が生じることがあります。これは、基礎となるインターフェースが ODM で検出された情報を使用してイーサチャネルの始動前

に構成されるためです。 したがって、イーサチャネルの構成時に、イーサチャネルは、そのアダプターのいずれかが既に使用中であることを検出します。 この状態を変更するには、イーサチャネルを作成する前に、`smitty chinet` と入力し、イーサチャネルに組み込まれるアダプターの 各インターフェースを選択し、さらに「**state**」の値を `detach` に変更します。 これにより、マシンのリブート時にエラーを生じさせずにイーサチャネルを構成できるようになります。

ODM の詳細については、プログラミングの一般概念*:* プログラムの作成およびデバッグ の『Object Data Manager (ODM)』を参照してください。

- AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) より前の AIX バージョンのイーサチャネルで 10/100 イーサネット・アダプターを 使用する場合は、これらのアダプターをイーサチャネルに追加する前に、アダプター上でリンク・ポーリングを使用可能にする必要が生じることがあります。 コマンド・ラインに `smitty chgenet` と入力します。 「**Enable Link Polling (リンク・ポーリングを使用可能にする)**」の値を「`yes` (はい)」に変更して、Enter キーを押します。

  注 : AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) 以降では、リンク・ポーリング機構を使用可能にする必要はありません。 リンク・ポーリング・プログラムは自動的に始動されます。
- ジャンボ・フレームを使用する予定がある場合は、イーサチャネルの作成前のすべてのアダプターならびにイーサチャネル自体でこの機能を使用可能にする必要が生じることがあります。 コマンド・ラインに `smitty chgenet` と入力します。 「**Enable Jumbo Frames (ジャンボ・フレームを使用可能にする)**」の値を 「`yes` (はい)」に変更して、Enter キーを押します。 ジャンボ・フレームを使用可能にしたいアダプターごとに、上記のことを行います。 あとで、イーサチャネル自体でジャンボ・フレームを使用可能にします。

  注 : イーサチャネル自体を使用可能化にすると、すべての基礎となるアダプター 中でジャンボ・フレームを使用可能にする必要はありません。 「**Enable Jumbo Frames (ジャンボ・フレームを使用可能にする)**」属性を「`yes` (はい)」に設定すると、このフィーチャーは自動的に使用可能になります。
- AIX 5.3 および AIX 6.1 のレベルでは、以下のホスト・イーサネット・アダプター (HEA) の構成がサポートされます。
  - 手動集約と LACP 集約の両方で、専用 HEA ポートと PCI/PCI-E アダプター間のリンク集約がサポートされます。
  - 非専用 HEA ポート・サポート、PCI/PCI-E として構成されたバックアップ・アダプターのあるイーサチャネル、仮想イーサネットのいずれかが組み込まれているイーサチャネル構成。

    注 : イーサチャネル構成内の非専用 HEA ポートには、リンク集約に関する制限が適用されます。
- AIX バージョン 6.1 (6100-06 テクノロジー・レベル適用) 以降では、スタックされたスイッチ上のイーサチャネルをサポートします。
- ネットワーク・インストール管理 (NIM) クライアントにおける、イーサチャネル経由のネットワーク・ブートもネットワーク・インストールもサポートされません。

イーサチャネルの構成

イーサチャネルを構成するには、以下の手順を使用します。

1. コマンド・ラインに `smitty etherchannel` と入力します。
2. リストから「**Add an EtherChannel/Link Aggregation (イーサチャネル/リンク集約の追加)**」を選択して、Enter キーを押します。
3. イーサチャネルで 1 次イーサネット・アダプターとするものを選択して、Enter キーを押します。

   イーサチャネルのバックアップを使用する予定であれば、ここにバックアップに使用する予定のアダプターを選択しないでください。

   注 : 「**Available Network Adapters (使用可能なネットワーク・アダプター)**」に、すべてのイーサネット・アダプターが 表示されます。 既に使用されている (インターフェースが 定義されている) イーサネット・アダプターを 選択すると、エラー・メッセージが出されます。 使用されているインターフェースを使用する場合は、最初に、それを切り離す必要があります。
4. 次の説明に従って、フィールドに情報を入力します。
   - **Parent Adapter (親アダプター):** イーサチャネルの親デバイスの情報を示します (イーサチャネルが共用イーサネット・アダプターに属する場合など)。 イーサチャネルが別のアダプター内に含まれてい

ない場合 (デフォルト)、このフィールドには「NONE」の値が表示されます。 イーサチャネルが別のアダプター内に含まれている場合、このフィールドには親アダプターの 名前が表示されます (「ent6」など)。 このフィールドは情報フィールドであるため、変更することはできません。 親アダプター・オプションは、AIX 5.3 以降で使用可能です。

- **EtherChannel/Link Aggregation Adapters (イーサチャネル/リンク集約のアダプター):** イーサチャネル内で使用するすべての 1 次アダプターが 表示されています。 前のステップで、これらのアダプターを選択しました。
- **Enable Alternate Address (代替アドレスを使用可能にする):** このフィールドはオプションです。 これを「yes （はい)」に設定すると、イーサチャネルが使用する MAC アドレスを指定できます。 このオプションを「no （いいえ)」に設定すると、イーサチャネルは最初のアダプターの MAC アドレスを使用します。
- **Alternate Address (代替アドレス):**「**Enable Alternate Address (代替アドレスを使用可能にする)**」を「yes （はい)」に設定した場合に、使用する MAC アドレスをここに指定します。 指定するアドレスは、0x で始まる 12 桁の 16 進数のアドレスでなければなりません (0x001122334455 など)。
- **Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする):** このフィールドはオプションです。 これを使用するには、スイッチがジャンボ・フレームをサポートしている必要があります。 これは、標準イーサネット (en) インターフェース (IEEE 802.3 (et) インターフェースではない) とだけ連動します。 これを使用可能にするには、「yes （はい)」に設定します。
- **Mode (モード):** 次のモードから選択できます。
  - **standard (標準)**: このモードでは、イーサチャネルは、アルゴリズムを使用してパケットを送出するアダプターを 選択します。 このアルゴリズムでは、データ値を取り出し、その値をイーサチャネル内のアダプター 数で除算し、(「モジュラス」演算子を使用して) 剰余を使用して、発信リンクを識別します。 「Hash Mode (ハッシュ・モード)」値は、このアルゴリズムに送られるデータ値を判別します (さまざまなハッシュ・モードの説明については、「Hash Mode (ハッシュ・モード)」属性を参照してください)。 例えば、「Hash Mode (ハッシュ・モード)」が「standard （標準)」の場合は、パケットの宛先 IP アドレスが使用されます。 この宛先 IP が 10.10.10.11 で、イーサチャネルに 2 つのアダプターがある 場合、(1/2) = 0 残り 1 となり、2 番目のアダプターが 使用されます (アダプターの番号は 0 から始まります)。 アダプターは、SMIT メニューでリストされた順番に番号が付けられます。 これはデフォルトの動作モードです。
  - **round_robin (ラウンドロビン)**: このモードでは、イーサチャネルは、各アダプターに 1 つのパケットを与えながら、アダプターを 巡回します。 パケットはイーサチャネルに到達した順番と多少異なった順番で送信されるかもしれませんが、その帯域幅を十分に使用します。 このモードと「default （デフォルト)」以外の「Hash Mode (ハッシュ・モード)」を選択するのは、無効な組み合わせです。 ラウンドロビン・モードを選択する場合は、「Hash Mode (ハッシュ・モード)」値を「default （デフォルト)」のままにしてください。
  - **netif_backup**: ネットワーク・インターフェース ・バックアップ・モードを使用可能にするために、1 次イーサチャネルとバックアップ・イーサチャネルで 1 つ以上のアダプターを 構成できます。 詳しくは、432 ページの『ネットワーク・インターフェース ・バックアップの構成』を参照してください。
  - **8023ad**: このオプションにより、自動リンク集約に IEEE 802.3ad Link Aggregation Control Protocol (LACP) が使用可能になります。 このフィーチャーに関する詳細は、439 ページの『IEEE 802.3ad リンク集約の構成』を参照してください。
- **IEEE 802.3ad Interval (IEEE 802.3ad 間隔)**: 以下の値から選択できます。
  - **long**: これは間隔のデフォルト値です。 この値を選択すると、イーサチャネルは、プロトコルで指定された long 間隔値でパートナーに LACP パケットを要求します。
  - **short**: この値を選択すると、イーサチャネルは、プロトコルで指定された short 間隔値でパートナーに LACP パケットを要求します。

    注 : この間隔値は、イーサチャネルが IEEE 802.3ad モードで作動中の場合にのみ使用されます。 それ以外の場合は、この値は無視されます。

    注 : AIX は、long および short 両方の間隔要求をパートナーから受け取ります。

- **Hash Mode (ハッシュ・モード)**: 以下のハッシュ・モードから選択します。これにより、アルゴリズムで発信アダプターを判別するために使用するデータ値が判別されます。
  - **default (デフォルト)**: パケットの宛先 IP アドレスを使用して発信アダプターが判別されます。 非 IP のトラフィック (ARP など) の場合、宛先 MAC アドレスの最後のバイトがこの計算に使用されます。 このモードでは、パケットはイーサチャネルに到達した順にイーサチャネルから送信されることが保証されますが、帯域幅をいっぱいには使用していません。
  - **src_port (ソース・ポート)**: パケットのソース UDP または TCP ポート値を使用して発信アダプターが判別されます。 パケットが UDP または TCP トラフィックでない場合は、宛先 IP アドレスの最後のバイトが使用されます。 パケットが IP トラフィックでない場合は、宛先 MAC アドレスの最後のバイトが使用されます。
  - **dst_port (宛先ポート)**: パケットの宛先 UDP または TCP ポート値を使用して発信アダプターが判別されます。 パケットが UDP または TCP トラフィックでない場合は、宛先 IP の最後のバイトが使用されます。 パケットが IP トラフィックでない場合は、宛先 MAC アドレスの最後のバイトが使用されます。
  - **src_dst_port (ソースおよび宛先ポート)**: パケットのソースおよび宛先の両方の UDP または TCP ポート値を使用して発信アダプターが判別されます (特に、ソースおよび宛先ポートが加算されて 2 で除算されてからアルゴリズムに送られる)。 パケットが UDP または TCP トラフィックでない場合は、宛先 IP の最後のバイトが使用されます。 パケットが IP トラフィックでない場合は、宛先 MAC アドレスの最後のバイトが使用されます。 クライアントとサーバーの 両方において、ほとんどの状態で、このモードでのパケット配布は良好です。

    注 :「default（デフォルト）」以外の「Hash Mode (ハッシュ・モード)」と「round_robin（ラウンドロビン）」モードを選択するのは、無効な組み合わせです。

  パケット配布とロード・バランシングについて学習するには、434 ページの『イーサチャネルのロード・バランシング・オプション』を参照してください。
- **Backup Adapter (バックアップ・アダプター):** このフィールドはオプションです 。 イーサチャネルのバックアップとして使用するアダプターのリストを 入力します。
- **Internet Address to Ping (ping するインターネット・アドレス):** このフィールドはオプションで 、「**ネットワーク・インターフェース ・バックアップ**」モードで実行している場合、またはイーサチャネルとバックアップ・リストにそれぞれ 1 つ以上のアダプターがある 場合にのみ有効です。 イーサチャネルは、指定された IP アドレスまたはホスト名を ping します。 イーサチャネルが IP アドレスに対して、「**再試行回数**」フィールドで指定された回数を、「**再試行タイムアウト**」フィールドで指定された時間内に ping できない場合、イーサチャネルはバックアップ・リスト内の他のアダプターに 切り替えます。 このフィールドをリセットして、このオプションに対して以前に入力した値が削除されるようにするには、「**ping するインターネット・アドレス:**」オプションの値を 0 で入力します。
- **Number of Retries (再試行回数):** 試行する ping の最大回数を入力します。この回数 ping が失敗すると、イーサチャネルは次のアダプターに 切り替えます。 デフォルトは 3 回です。 このフィールドはオプションで、「**Internet Address to Ping (ping するインターネット・アドレス)**」を設定した場合にのみ有効です。
- **Retry Timeout (再試行タイムアウト (秒)):** イーサチャネルが、「**Internet Address to Ping (ping するインターネット・アドレス)**」に指定した IP アドレス に ping するインターバルを秒数で入力します。デフォルトは 1 秒です。 このフィールドはオプションで 、「**Internet Address to Ping (ping するインターネット・アドレス)**」を設定した場合にのみ有効です。

5. 必要なフィールドを変更したあと、Enter キーを押してイーサチャネルを作成します。
6. コマンド・ラインに `smitty chinet` と入力して、新たに作成したイーサチャネル・デバイスに IP を構成します。
7. リストから新規イーサチャネル・インターフェースを 選択します。
8. すべての必要なフィールドに値を入れて、Enter キーを押します。

イーサチャネルの構成後に実行できる追加のタスクについては、436 ページの『イーサチャネルまたはリンク集約のリスト』を参照してください。

リカバリーおよびフェイルオーバー・オプション
イーサチャネルまたは IEEE 802.3ad リンク集約では、リカバリーとフェイルオーバーの機能を使用することができます。

これらの機能を使用すると、以下の点で向上します。

- リカバリー中のパケット・ロスを回避できます
- フェイルオーバーを即時に発生させるように設定できます
- 自動リカバリーをオフにして、バックアップ・アダプターによる 操作を続行できます
- リンク集約が 1 次チャネルからバックアップに強制的にフェイルオーバーできるようにします (あるいは、その逆の操作を行います)

無損失リカバリー
無損失リカバリー機能により、バックアップ・アダプターから 1 次チャネルへのリカバリーで損失するパケットをできるだけ少なくすることができます。

無損失リカバリーの前に、イーサチャネルまたは IEEE 802.3ad がいずれかの 1 次アダプターのリカバリーを検出すると同時に 1 次チャネルにリカバリーします。 場合によっては、アダプター・スイッチがデータを送受信できる状態に入らず、リカバリーの直後にパケットの一部が失われることがあります。

無損失リカバリーでは、イーサチャネルまたは IEEE 802.3ad アダプターは実際にトラフィックを受信できている場合のみ 1 次チャネルにリカバリーします。 これにより、スイッチ・ポートが完全に初期化されて、パケットが失われません。

無損失フェイルオーバー
無損失フェイルオーバー機能は、無損失リカバリー機能の動作を修正します。

ping 障害によりフェイルオーバーが発生すると、デフォルトで無損失リカバリーが確認されます。 これは、フェイルオーバーをファイナライズする 前に、非アクティブのアダプター ・スイッチがトラフィックを受信するまで待機する時間を要します。 ただし、**noloss_failover** 属性が no に設定されている場合は、ping フェイルオーバーは直ちに発生します。

自動リカバリー
1 次チャネルからバックアップ・アダプターへのフェイルオーバーの 後、イーサチャネルと IEEE 802.3ad リンク集約は、少なくともそのアダプターの 1 つが回復すると、失敗した 1 次チャネルに対するリカバリーを自動的に開始します。

このリカバリー・オプションは IEEE 802.3ad モードではサポートされず、バックアップ・アダプターへのフェイルオーバーは Link Aggregation Control Protocol (LACP) の障害によるものです。LACP 障害は、1 次チャネル内のすべてのアダプターが、タイムアウト期間内に LACP データ単位 (LACPDU) を受信しなかったときに発生します。タイムアウト期間は、IEEE 802.3ad モード用に構成されたインターバルに基づいた IEEE 標準によって判別されます。

このデフォルトの動作を変更するには、**auto_recovery** 属性を no に設定します。この設定では、イーサチャネルまたは IEEE 802.3ad リンク集約は、フェイルオーバー後もバックアップ・アダプターで 操作を続行します。 バックアップ・アダプターでの 操作は、以下のいずれかのイベントが発生するまで続行されます。

- フェイルオーバーが強制される。
- バックアップ・アダプターが失敗する。
- バックアップ・アダプターで ping の失敗が検出される。

強制的なフェイルオーバー
イーサチャネルまたは IEEE 802.3ad リンク集約を強制的に 1 次チャネルからバックアップ・アダプターに、またはバックアップ・アダプターから 1 次チャネルにフェイルオーバーさせることができます。

強制的なフェイルオーバーが機能するのは、バックアップ・アダプターが定義されている場合、また非アクティブ・チャネルが稼働中の場合のみです。 例えば、1 次チャネルからバックアップ・アダプターに強制的にフェイルオーバーさせるには、そのバックアップ・アダプターが実行されていなければなりません。

この機能を使用するには、`smitty etherchannel` と入力し、「**Force A Failover In An EtherChannel / Link Aggregation (イーサチャネル/リンク集約でフェイルオーバーを強制する)**」オプションを画面から選

択します。 次に、強制的にフェイルオーバーさせるイーサチャネルまたは IEEE 802.3ad リンク集約を選択します。

ネットワーク・インターフェース・バックアップの構成
Network Interface Backup (ネットワーク・インターフェース・バックアップ) は、ユーザー接続を中断せずに、障害の検出とフェイルオーバーを提供することによって、ネットワークのある 1 点での障害に対する保護を行います。 このモードで動作しているとき、ある時点で、1 つのアダプターのみアクティブです。

アクティブ・アダプターに障害が発生すると、イーサチャネル内の別のアダプターがすべてのトラフィックの対応に使用されます。 Network Interface Backup (ネットワーク・インターフェース・バックアップ) モードで動作している場合、イーサチャネル可能スイッチに接続されている必要はありません。

Network Interface Backup (ネットワーク・インターフェース・バックアップ) セットアップは、すべてのアダプターを 1 つのスイッチに接続するより冗長度が大きくなるため、アダプターを異なるネットワーク・スイッチに接続するときに最も効率的です。 異なるスイッチに接続する場合、スイッチの間に接続があることを確認してください。 これは、現在アクティブなアダプターに 常に経路があることを保証することで、1 つのアダプターからもう 1 つのアダプターへのフェイルオーバーの 機能を提供します。

1 次イーサチャネルで構成されたアダプターは、バックアップ・アダプターより 優先されます。1 次アダプターが機能している限り、それが使用されます。 これは、1 次アダプターが回復したかどうかにかかわらず、バックアップ・アダプターに障害が発生するまでバックアップ・アダプターが 使用されていた、以前のリリースでのネットワーク・インターフェース・バックアップ・モードでの動作と対照的です。

例えば、`ent3` というイーサチャネルを作成して、`ent0` をメイン・アダプターとして 構成し、`ent2` をバックアップ・アダプターとして 構成できます。 2 つの別のスイッチに `ent0` と `ent2` を接続するのが理想です。 この例では、`ent3` (イーサチャネルのインターフェース) を介して送信されるすべてのトラフィックは、デフォルトでは `ent0` を介して送信されますが、`ent2` はアイドル状態になります。 `ent0` に障害が起こると、すべてのトラフィックはバックアップ・アダプター `ent2` を介して送信されます。 `ent0` がリカバリーされると、再度このアダプターを 使用してすべてのトラフィックが送信されます。

バックアップ・アダプターを 含む複数のイーサチャネルに対して、イーサチャネルがリンク障害とネットワークの到達不能を検出するように構成することが可能です。 この構成を行うには、**netaddr** 属性を使用して、常に接続されているはずのリモート・ホストの IP アドレスまたはホスト名を指定します。 イーサチャネルは周期的にこのホストに ping して、このホストへのネットワーク・パスがまだ存在しているかどうか確認します。 指定された回数まで ping しても応答がない場合、他のアダプターを 通してリモート・ホストへのネットワーク・パスが存在することを見越して、イーサチャネルは他のアダプターにフェイルオーバーします。 このセットアップでは、すべてのアダプターを 異なるスイッチに接続するというだけでなく、それぞれのスイッチが ping されるホストへの異なる経路を持つ必要があります。

この ping 機能は、バックアップ・アダプターを 含む 1 つ以上のイーサチャネルで使用可能です。 しかし、1 次アダプター 上で ping しても応答がないためにフェイルオーバーする 場合、バックアップ・アダプターが作動中はアクティブ・チャネルになります。 バックアップ・アダプター 上での操作中は、1 次アダプターから ping されるホストに到達できるかどうか知る方法はありません。 1 次とバックアップの間で双方向にフェイルオーバーしないように、バックアップ上で操作し続けます (バックアップ・アダプター 上でも ping した後に応答がない場合、またはバックアップ・アダプター 自体に障害が起こった場合は、1 次アダプターにフェイルオーバーします)。 しかし、(ping しても応答がないためではなく) 1 次アダプターの 障害のためにフェイルオーバーが発生した場合、イーサチャネルは、1 次アダプターが回復するとすぐに、通常どおり 1 次アダプターに戻します。

新しいバージョンでネットワーク・インターフェース・バックアップを構成するには、432 ページの『ネットワーク・インターフェース・バックアップの構成』を参照してください。

ネットワーク・インターフェース・バックアップの構成
新しいバージョンでネットワーク・インターフェース・バックアップを構成するには、この手順を使用します。

1. root 権限で、コマンド・ラインに `smitty etherchannel` と入力します。
2. リストから「**Add an EtherChannel/Link Aggregation (イーサチャネル/リンク集約の追加)**」を選択して、Enter キーを押します。
3. 1 次イーサネット・アダプターを 選択して、Enter キーを押します。 このアダプターが、障害が発生するまで使用されます。

注 :「**Available Network Adapters (使用可能なネットワーク・アダプター)**」フィールドに、すべてのイーサネット・アダプターが表示されます。 既に使用されているイーサネット・アダプターを選択すると、エラー・メッセージが出されます。そのアダプターを使用するには、このインターフェースを切り離す必要があります。 インターフェースを切り離す方法については、439 ページの『5200-01 以前でのイーサチャネルへの変更』 を参照してください。

4. 次のガイドラインに従って、以下のフィールドに情報を入力します。
   - **Parent Adapter (親アダプター):** このフィールドはイーサチャネルの親デバイスの情報を示します (イーサチャネルが共用イーサネット・アダプターに属する場合など)。 イーサチャネルが別のアダプター内に含まれていない場合 (デフォルト)、このフィールドには「NONE」の値が表示されます。 イーサチャネルが別のアダプター内に含まれている場合、このフィールドには親アダプターの名前が表示されます (「ent6」など)。 このフィールドは情報フィールドであるため、変更することはできません。 バックアップ・アダプター・オプションは、AIX オペレーティング・システムで使用可能です。
   - **EtherChannel/Link Aggregation Adapters (イーサチャネル/リンク集約のアダプター)**: 前のステップで選択した 1 次アダプターが表示されています。
   - **Enable Alternate Address (代替アドレスを使用可能にする):** このフィールドはオプションです。 これを「yes (はい)」に設定すると、イーサチャネルが使用する MAC アドレスを指定できます。 このオプションを「no (いいえ)」に設定すると、イーサチャネルは 1 次アダプターの MAC アドレスを使用します。
   - **Alternate Address (代替アドレス):**「**Enable Alternate Address (代替アドレスを使用可能にする)**」を「yes (はい)」に設定した場合に、使用する MAC アドレスをここに指定します。 指定するアドレスは、0x で始まる 12 桁の 16 進数のアドレスでなければなりません (0x001122334455 など)。
   - **Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする)**: このフィールドはオプションです。 これを使用するには、スイッチがジャンボ・フレームをサポートしている必要があります。 これは、標準イーサネット (en) インターフェース (IEEE 802.3 (et) インターフェースではない) とだけ連動します。 これを使用するには、「yes (はい)」に設定します。
   - **Mode (モード)**: メイン・イーサチャネルには 1 つしかアダプターがないため、どの動作モードを選択するかは関係ありません。 すべてのパケットは、そのアダプターに障害が発生するまで、そのアダプターで送信されます。 そのモードはバックアップ・アダプターを使用してエミュレートできるため、netif_backup モードはありません。
   - **Hash Mode (ハッシュ・モード)**: メイン・イーサチャネルには 1 つしかアダプターがないため、どのハッシュ・モードを選択するかは関係ありません。 すべてのパケットは、そのアダプターに障害が発生するまで、そのアダプターで送信されます。
   - **バックアップ・アダプター**: イーサチャネル・バックアップ・グループに組み込む、1 つ以上のアダプターのリストを入力します。 1 次イーサチャネル・グループの欠落によるフェイルオーバー・イベント後、その 1 次イーサチャネル・グループがリカバリーされるまで、バックアップ・アダプターが使用されます。
   - **Internet Address to Ping (ping するインターネット・アドレス)**: このフィールドはオプションです。 イーサチャネルは、ここで指定された IP アドレスまたはホスト名を ping します。 イーサチャネルが「**Number of Retries (再試行回数)**」フィールドの回数内に、また「**Retry Timeout (再試行タイムアウト)**」フィールドのインターバルで、このアドレスに ping できなければ、イーサチャネルは次のアダプターに切り替えます。
   - **Number of Retries (再試行回数)**: 試行する ping の最大回数を入力します。この回数 ping が失敗すると、イーサチャネルは次のアダプターに切り替えます。 デフォルトは 3 回です。 このフィールドはオプションで、「**Internet Address to Ping (ping するインターネット・アドレス)**」を設定した場合にのみ有効です。
   - **Retry Timeout (再試行タイムアウト (秒))**: イーサチャネルが、「**Internet Address to Ping (ping するインターネット・アドレス)**」に指定した IP アドレス に ping するインターバルを秒数で入力します。 デフォルトは 1 秒です。 このフィールドはオプションで、「**Internet Address to Ping (ping するインターネット・アドレス)**」を設定した場合にのみ有効です。
5. 必要なフィールドを変更したあと、Enter キーを押してイーサチャネルを作成します。

6. コマンド・ラインに smitty chinet と入力して、新たに作成したインターフェースに IP を構成します。
7. リストから新規イーサチャネル・インターフェースを 選択します。
8. すべての必要なフィールドに値を入れて、Enter キーを押します。

ネットワーク・インターフェース ・バックアップの構成が完了しました。

イーサチャネルのロード・バランシング・オプション

イーサチャネルでの発信トラフィックに関するロード・バランシングには、次の 2 つの方式があります。1 つはラウンドロビン方式で、これは発信トラフィックをすべてのアダプターに 均等に分散されます。 もう 1 つは標準方式で、これはアルゴリズムを使用するアダプターを 選択します。

Hash Mode パラメーターは、アルゴリズムに送る数値を判別します。

以下の表には、備えられている有効なロード・バランシング・オプションの組み合わせが要約されています。

表 *81.* 「*Mode (*モード*)*」および「*Hash Mode (*ハッシュ・モード*)*」の組み合わせと、個々の組み合わせで行われる発信トラフィックの配布

Mode (モード)	Hash Mode (ハッシュ・モード)	発信トラフィックの配布
standard (標準) または 8023ad	デフォルト	従来の AIX の動作。 アダプター選択アルゴリズムは、宛先 IP アドレス (**TCP/IP** トラフィックの場合) または MAC アドレス (ARP などの非 IP トラフィックの場合) の最後のバイトを使用します。 多数のクライアントがあるサーバーの場合、普通はこのモードを初期項目として選択するのが適切です。
standard (標準) または 8023ad	src_dst_port (ソースおよび宛先ポート)	アルゴリズムにより、ソースおよび宛先の TCP または UDP ポート値の組み合わせを使用して、発信アダプター・パスが選択されます。接続ごとに固有の TCP または UDP ポートが使用されるので、1 対の IP アドレスの間に数種類の TCP または UDP 接続がある場合には、3 つのポートがベースになっているハッシュ・モードによりアダプター配布の柔軟性が向上します。
standard (標準) または 8023ad	src_port (ソース・ポート)	アダプター 選択アルゴリズムは、ソースの TCP または UDP ポート値を使用します。 「**netstat -an**」コマンド出力の Local 列の TCP/IP アドレス・サフィックス値が、ポートになります。
standard (標準) または 8023ad	dst_port (宛先ポート)	アルゴリズムにより、宛先のシステム・ポート値を使用して、発信アダプター・パスが選択されます。「**netstat -an**」コマンド出力の Foreign 列の TCP/IP アドレス・サフィックスが、TCP または UDP 宛先ポート値になります。

表 81. 「*Mode* (モード)」および「*Hash Mode* (ハッシュ・モード)」の組み合わせと、個々の組み合わせで行われる発信トラフィックの配布 (続き)

Mode (モード)	Hash Mode (ハッシュ・モード)	発信トラフィックの配布
round-robin (ラウンドロビン)	デフォルト	発信トラフィックは、イーサチャネル中のすべてのアダプター・ポートに均等に分散されます。 2 つのホストが (スイッチを介さずに) バックツーバック接続している場合は、このモードを選択するのが一般的です。

ラウンドロビン配布

すべての発信トラフィックが、イーサチャネル中のすべてのアダプターに均等に分散されます。 AIX サーバー・システムで、帯域幅の最適化が最大になります。 すべてのリンクを均等に使用するにはラウンドロビン配布が理想的な方法ですが、受信側システムで無秩序パケットが生じる可能性があることも考慮してください。

通常、ラウンドロビン・モードは、ジャンボ・フレームを実行するバックツーバック接続の場合に理想的です。 この環境ではスイッチが介在しないので、スイッチでの処理によりパケット送達の時刻、順序、またはアダプター・パスを変更する機会がありません。 この直接ケーブル・ネットワーク・パスでは、パケットは送信したとおりに正確に受信されます。 ジャンボ・フレーム (9000 バイト MTU) は従来の 1500 バイト MTU より、ファイル転送のパフォーマンスが常に向上します。 しかし、ラウンドロビン・モードの場合、ジャンボ・フレームには別の利点があります。 ラウンドロビン・モードのパケットの方が大きく送信する時間が長くかかるので、ジャンボ・フレームにより受信側のホストが無秩序パケットに連続的に割り込まれる可能性が低くなります。

その他の環境でもラウンドロビン・モードをインプリメントできますが、受取側システムでの無秩序パケットのリスクが高くなります。 このリスクは特に少数の存続時間が長いストリーミング TCP 接続がある場合に高くなります。 1 対のホストの間にこの種の接続が多数ある場合は、さまざまな接続からのパケットが混在する可能性が高いので、同一の接続からのパケットが無秩序に着信する可能性は低くなります。 無秩序パケットの統計情報は、「**netstat -s**」コマンド出力の `tcp` セクションで検査してください。 値が継続的に大きくなっている場合は、イーサチャネルから送信されたトラフィックに問題が生じている可能性があることを示します。

無秩序パケットの問題により、システムで従来のイーサネット MTU を使用しなければならず、スイッチを介して接続しなければならない場合は、標準モード操作で使用できるさまざまなハッシュ・モードを試してください。 モードごとに特定の利点がありますが、デフォルト・モードと src_dst_port モードは適用範囲が広いので論理的な開始点になります。

標準アルゴリズムまたは *802.3ad* アルゴリズム

イーサチャネルの標準アルゴリズムを使用する利点は多くあります。

標準アルゴリズムは、標準型と IEEE 802.3ad 型のリンク集約の両方で使用されます。 AIX は、「数値」の最後のバイトをイーサチャネル内のアダプター数で除算し、その剰余を使用して、発信リンクを識別します。 剰余がゼロの場合は、イーサチャネル中の最初のアダプターが選択されます。 剰余が 1 の場合は 2 番目のアダプターが選択され、以下同様です (**adapter_names** 属性にリストされている順序でアダプターが選択されます)。

「Hash Mode (ハッシュ・モード)」の選択項目は、計算で使用される数値を判別します。 デフォルトでは宛先 IP アドレスまたは MAC アドレスの最後のバイトが計算に使用されますが、ソースまたは宛先の TCP または UDP ポート値も使用できます。 これらの代替値を使用すると、イーサチャネル中の実際のアダプター間への発信トラフィックの配布を微調整できます。

デフォルトのハッシュ・モードでは、アダプター選択アルゴリズムは、IP トラフィックの宛先 IP アドレスの最後のバイトに適用されます。 ARP などの他の非 IP トラフィックの場合、宛先 MAC アドレスの最後のバイトに同じ公式が適用されます。 アダプター障害のためにフェイルオーバーしていなければ、デフォルトの標準モードの 1 対のホスト間のトラフィックはすべて同じアダプターを介します。 デフォルトのハッシュ・モードは、ローカル・ホストで多種の IP アドレスへの接続が確立される場合に理想的です。

しかし、ローカル・ホストで少数の IP アドレスへの長期接続が確立される場合は、特定の宛先に送信されるトラフィックがすべて同一のアダプターを 介して送信されるので、アダプター間で負荷の大きさが異なることに気付くはずです。 この場合、パケットの無秩序着信は生じませんが、いかなる場合でも最も効率の良い方法で帯域幅を使用できない可能性があります。 ポート・ベースのハッシュ・モードは、この場合も順序どおりにパケットを送信しますが、パケットが属する UDP または TCP 接続が異なると (同じ宛先に送信される場合でも)、送信の際に介するアダプターが 異なるので、すべてのアダプターの 帯域幅を使用する方が効率的です。

src_dst_port (ソースおよび宛先ポート) ハッシュ・モードでは、発信パケットの TCP または UDP のソースと宛先のポート値が加算されてから、2 で除算されます。 計算結果の整数 (小数部なし) は、標準アルゴリズムに挿入されます。 TCP または UDP トラフィックは、標準アルゴリズムおよび選択済みのハッシュ・モード値によって選択されたアダプター 上で送信されます。 非 TCP または UDP トラフィックは、デフォルトのハッシュ・モードにフォールバックします。 この場合、宛先 IP アドレスまたは MAC アドレスの最後のバイトが使用されます。 **src_dst_port (ソースおよび宛先ポート)** ハッシュ・モード・オプションは、ソースと宛先の両方の TCP または UDP のポート値を考慮します。 このモードでは、1 つの TCP または UDP 接続中のすべてのパケットは単一のアダプターを 介して送信されるので、順序どおりに着信することが保証されます。しかし、接続ごとに送信時に介するアダプターが 異なる可能性がある (同一ホストに送信する場合でも) のでこの場合もトラフィックは分散されます。 接続確立指示ではソースと宛先の両方の TCP または UDP のモード値が使用されるので、このハッシュ・モードの結果は接続確立指示によってスキューされません。

src_port (ソース・ポート) ハッシュ・モードでは、発信パケットのソースの TCP または UDP ポート値が使用されます。 **dst_port (宛先ポート)** ハッシュ・モードでは、発信パケットの宛先の TCP または UDP ポート値が使用されます。 ポート値がある接続から別の接続に変更され、**src_dst_port (ソースおよび宛先ポート)** オプションで希望どおりに配布できない場合は、**src_port (ソース・ポート)** または **dst_port (宛先ポート)** ハッシュ・モード・オプションを使用してください。

イーサチャネルまたはリンク集約のリスト

イーサチャネルまたはリンク集約をリストするには、以下の手順を使用します。

1. コマンド・ラインで、`smitty etherchannel` と入力します。
2. 「**List All EtherChannels/Link Aggregations (イーサチャネル/リンク集約をすべてリスト)**」を選択して、Enter キーを押します。

代替アドレスの変更

イーサチャネルまたはリンク集約の MAC アドレスを指定するには、以下の手順を使用します。

1. 実行している AIX バージョンによっては、インターフェースを 切り離す必要があります。
   - AIX 5.2 (5200-01 適用) 以前では、`smitty chinet` と入力し、ご使用のイーサチャネルに属するインターフェースを 選択します。 「**Current STATE (現在の状態)**」属性を「**detach (切り離し)**」に変更し、Enter キーを押します。
   - AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) 以降では、インターフェースを切り離さずにイーサチャネルの代替アドレスを変更できます。
2. コマンド・ラインで、`smitty etherchannel` と入力します。
3. 「**Change/Show Characteristics of an EtherChannel (イーサチャネルの特性の変更/表示)**」を選択して、Enter キーを押します。
4. 複数のイーサチャネルがある場合は、代替アドレスを作成するイーサチャネルを選択します。
5. 「**Enable Alternate EtherChannel Address (代替イーサチャネル・アドレスを使用可能にする)**」の値を、「yes (はい)」に変更します。
6. 「**Alternate EtherChannel Address (代替イーサチャネル・アドレス)**」フィールドに、代替アドレスを入力します。 アドレスは、0x で始まる 12 桁の 16 進数のアドレスでなければなりません (0x001122334455 など)。
7. Enter キーを押すと、処理が完了します。

   注 : 実行時にイーサチャネルの MAC アドレスを変更すると、接続が一時的に失われることがあります。その理由は、アダプターをリセットして新しいハードウェア・アドレスを認識させる必要があり、初期化するのに数秒かかるアダプターもあるからです。

動的アダプター・メンバーシップ
AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) より前は、イーサチャネルからアダプターを追加したり除去したりするには、最初にインターフェースを 切り離して、一時的にすべてのユーザー・トラフィックを中断する必要があります。 この制限を解消するために、AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) には動的アダプター・メンバーシップ (DAM) が追加されました。

DAM を使用すると、ユーザー接続を切断せずに、イーサチャネルからアダプターの 追加や除去を行えます。バックアップ・アダプターの 追加や除去も行えます。最初にバックアップ・アダプターのないイーサチャネルを作成し、後日必要に応じて追加できます。

ユーザー接続を切断せずにアダプターの 追加や除去を行えるだけでなく、実行時にほとんどのイーサチャネル属性を変更できます。 例えば、イーサチャネルの使用中にネットワーク・インターフェース・バックアップの「ping」フィーチャーを使用し始めたり、いつでも ping されるリモート・ホストを変更したりできます。

正規のイーサチャネルを IEEE 802.3ad リンク集約に変え (またはその逆を行い)、イーサチャネルを除去して再作成しなくてもユーザーがこのフィーチャーを使用できるようにすることも可能です。

さらに DAM を使用すると、単一アダプター・イーサチャネルの作成を選択することもできます。 単一アダプター・イーサチャネルは、正規のアダプターとまったく同じ働きをします。しかし、このアダプターに障害が起こったら、実行時に接続を失わずに交換できます。 交換するには、一時アダプターをイーサチャネルに追加し、障害のあるアダプターをイーサチャネルから 除去し、ホット・プラグを使用して障害のあるアダプターを作業アダプターに 交換し、新しいアダプターをイーサチャネルに 追加してから、一時アダプターを除去します。 このプロセス中に接続が失われたことに気付くことは決してありません。 しかし、このアダプターがスタンドアロン・アダプターとして 機能していた場合は、ホット・プラグを使用して除去する前に切り離す必要があり、その間はこのアダプターを 介するトラフィックは失われます。

イーサチャネルまたはリンク集約内のアダプターの追加、除去、または変更
イーサチャネルまたはリンク集約内でアダプターの 追加、除去、または変更を行う方法は 2 つあります。

1 つの方式ではイーサチャネルまたはリンク集約インターフェースを 切り離す必要があり、もう 1 つの方式ではその必要はありません (AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) 以降で使用できる動的アダプター・メンバーシップを使用します)。

動的アダプター・メンバーシップを使用してイーサチャネルに変更を加える
動的アダプター・メンバーシップを使用して変更を加える際には、インターフェースを 切り離して、イーサチャネルを介するトラフィックをすべて停止する必要はありません。

作業を続行する前に、以下のことを検討してください。

1. 実行時にアダプターを 追加する際には、イーサネット・アダプターごとにサポートする 機能が異なることに注意してください (例えば、チェックサム・オフロードを行う機能、専用セグメントを使用する機能、大規模送信を行う機能など)。 同一のイーサチャネル中でさまざまなタイプのアダプターを 使用する場合には、すべてのアダプターでサポートされている 機能がインターフェース 層に報告されます (例えば、1 つのアダプターのみ 専用セグメントの使用をサポートしていない場合は、イーサチャネルは専用セグメントをサポートしていないことを知らせます。すべてのアダプターが 大規模送信をサポートしている場合は、チャネルは大規模送信をサポートしていることを知らせます)。 実行時にアダプターをイーサチャネルに追加する際には、少なくともイーサチャネル中に既にある他のアダプターと 同じ機能をサポートしていることを確認してください。 イーサチャネルがサポートしているすべての機能をサポートしていないアダプターを 追加しようとすると、その追加作業は失敗します。 しかし、イーサチャネルのインターフェースを 切り離すと、どのアダプターでも (サポートしている機能に関係なく) 追加でき、インターフェースを 再びアクティブ化する際に、新しいアダプターのリストに 基づいてサポートする機能が再計算されることに注意してください。
2. 代替アドレスを使用しておらず、イーサチャネルで MAC アドレスが使用されているアダプターを 削除する予定の場合は (イーサチャネルで使用される MAC アドレスは 1 つのアダプターに「所有」されます)、イーサチャネルは次の使用可能なアダプターの MAC アドレスを使用します。 すなわち、削除後に最初のアダプターになるアダプター 、またはメイン・アダプターがすべて削除された場合はバックアップ・アダプターです。 例えば、イーサチャネルにメイン・アダプター `ent0` および `ent1` とバックアップ・アダプター `ent2` がある場合は、デフォルトではイーサチャネルは `ent0` の MAC アドレスを使用します (この場合、`ent0` が MAC アドレスを「所有」しているといいます)。 `ent0` が削除された場合は、

イーサチャネルは ent1 の MAC アドレスを使用します。 その後 ent1 が削除された場合は、イーサチャネルは ent2 の MAC アドレスを使用します。 その後 ent0 がイーサチャネルに再追加された場合は、ent2 がこの時点で MAC アドレスの所有者なので、引き続き ent2 の MAC アドレスを使用します。その後 ent2 がイーサチャネルから削除された場合は、再び ent0 の MAC アドレスを使用し始めます。

イーサチャネルで MAC アドレスが使用されているアダプターを削除すると、一時的に接続が失われることがあります。その理由は、イーサチャネル中のすべてのアダプターをリセットして、新しいハードウェア・アドレスを認識させる必要があるからです。 初期化に数秒かかるアダプターもあります。

イーサチャネルが代替アドレス (ユーザー指定の MAC アドレス) を使用している場合は、アダプターの追加や削除に関係なくこの MAC アドレスが引き続き使用されます。 さらに、どのアダプターもイーサチャネルの MAC アドレスを「所有」していないので、アダプターの 追加や削除に関係なく一時的に接続が失われないことも意味します。

3. 実行時にほとんどすべてのイーサチャネル属性を変更できるようになりました。 「**Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする)**」のみ例外です。 「**Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする)**」属性を変更するには、最初にイーサチャネルのインターフェースを 切り離してからこの値の変更を試行しなければなりません。
4. 実行時に変更できない属性 (現在は「**Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする)**」のみ) の場合、「**Apply change to DATABASE only (データベースだけに変更を適用する)**」というフィールドがあります。 この属性を「yes （はい)」に設定すると、普通は実行時に変更できない属性の値を、実行時に変更できます。「**Apply change to DATABASE only (データベースだけに変更を適用する)**」フィールドを「yes （はい)」に設定すると、属性は ODM 中でのみ変更され、メモリーに再ロードされる (インターフェースを 切り離し、rmdev -l EtherChannel_device コマンドを使用してから、mkdev -l EtherChannel_device コマンドを使用することにより) かマシンがリブートされるまで実行中のイーサチャネルには反映されません。 この方法は、実行中のイーサチャネルを中断せずに、次回のマシン・ブート時に属性を確実に変更させるのに便利です。
5. 論理区画では、アダプターをイーサチャネルから 削除する場合、関連するスイッチ・ポートもスイッチのイーサチャネルから削除する必要があります。 これを行わないと、スイッチが通信に同じスイッチ・ポートを使用している可能性があるため、接続が失われることがあります。

動的アダプター・メンバーシップを 使用してイーサチャネルまたはリンク集約に変更を加えるには、次のステップを実行してください。

1. コマンド・ラインで、smitty etherchannel と入力します。
2. 「**Change/Show Characteristics of an EtherChannel/Link Aggregation (イーサチャネル/リンク集約の特性の変更/表示)**」を選択します。
3. 変更するイーサチャネルまたはリンク集約を選択します。
4. 次の説明に従って、必要なフィールドに入力します。
   - 「**Add adapter (アダプターの 追加)**」または「**Remove adapter (アダプターの 除去)**」フィールドで、追加または除去したいイーサネット・アダプターを 選択します。
   - 「**Add backup adapter (バックアップ・アダプターの追加)**」または「**Remove backup adapter (バックアップ・アダプターの 除去)**」フィールドで、バックアップとしての使用を開始または停止したいイーサネット・アダプターを 選択します。
   - 実行時にほとんどすべてのイーサチャネル属性を変更できますが、「**Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする)**」属性は変更できません。
   - 正規のイーサチャネルを IEEE 802.3ad リンク集約に変えるには、「**Mode (モード)**」属性を「8023ad」に変更します。 IEEE 802.3ad リンク集約をイーサチャネルに変えるには、「**Mode (モード)**」属性を「standard （標準)」または「round_robin （ラウンドロビン)」に変更します。
5. 必要なデータを入れて、Enter キーを押します。

5200-01 以前でのイーサチャネルへの変更

インターフェースを 切り離し、5200-01 以前でイーサチャネルに変更を加えるには、この手順を使用します。

1. `smitty chinet` と入力し、ご使用のイーサチャネルに属するインターフェースを 選択します。「**Current STATE (現在の状態)**」属性を「**detach (切り離し)**」に変更し、Enter キーを押します。
2. コマンド・ラインで、`smitty etherchannel` と入力します。
3. 「**Change/Show Characteristics of an EtherChannel/Link Aggregation (イーサチャネル/リンク集約の特性の変更/表示)**」を選択して、Enter キーを押します。
4. 変更するイーサチャネルまたはリンク集約を選択します。
5. イーサチャネルまたはリンク集約で変更したい属性を修正して、Enter キーを押します。
6. 必要なフィールドに値を入れて、Enter キーを押します。

イーサチャネルまたはリンク集約の除去

イーサチャネルまたはリンク集約を除去するには、以下の手順を使用します。

1. `smitty chinet` と入力し、ご使用のイーサチャネルに属するインターフェースを 選択します。「**Current STATE (現在の状態)**」属性を「**detach (切り離し)**」に変更し、Enter キーを押します。
2. コマンド・ラインで、`smitty etherchannel` と入力します。
3. 「**Remove an EtherChannel (イーサチャネルの除去)**」を選択し、Enter キーを押します。
4. 除去するイーサチャネルを選択して、Enter キーを押します。

既存のイーサチャネルまたはリンク集約のバックアップ・アダプターの構成または除去

次の手順を使用して、イーサチャネルまたはリンク集約のバックアップ・アダプターを 構成または除去します。

1. `smitty chinet` と入力し、ご使用のイーサチャネルに属するインターフェースを 選択します。「**Current STATE (現在の状態)**」属性を「**detach (切り離し)**」に変更し、Enter キーを押します。
2. コマンド・ラインで、`smitty etherchannel` と入力します。
3. 「**Change/Show Characteristics of an EtherChannel/Link Aggregation (イーサチャネル/リンク集約の特性の変更/表示)**」を選択します。
4. バックアップ・アダプターを 追加または変更するイーサチャネルまたはリンク集約を選択します。
5. バックアップ・アダプターとして 使用するアダプターを 「**Backup Adapter (バックアップ・アダプター)**」フィールドに 入力します。あるいは、バックアップ・アダプターの 使用を停止する場合は、「**NONE (なし)**」を選択します。

IEEE 802.3ad リンク集約の構成

IEEE 802.3ad は、リンク集約を行う標準の方法です。 概念上は、複数のイーサネット・アダプターが 集められて 1 つの仮想アダプターが形成される (これにより帯域幅が大きくなり、障害からの保護がよりなされるようになる) という点で、IEEE 802.3ad はイーサチャネルと同じ働きをします。

例えば、`ent0` と `ent1` を集合させて、`ent3` という IEEE 802.3ad リンク集約を形成することができます。この場合、インターフェース `ent3` が IP アドレスを使用して構成されます。 システムは、これらの集合アダプターを 1 つのアダプターと 見なします。 このため、他のイーサネット・アダプターと 同様に、これらの集合アダプターにも IP が構成されます。

IEEE 802.3ad にはスイッチのサポートが必要です。

イーサチャネルでなく IEEE 802.3ad リンク集約を使用する利点は、IEEE 802.3ad 規格をサポートしていてイーサチャネルをサポートしていないスイッチを使用でき、アダプターの 障害に対する保護があることです。

IEEE 802.3ad 集約が構成されると、サーバー・マシン (ホスト・システム) と隣接スイッチとの間で Link Aggregation Control Protocol Data Unit (LACPDU) が交換されます。 アクティブ・チャネル (1 次チャネルの場合もバックアップ・アダプターの 場合もありますが) のみが隣接スイッチと LACPDU を交換します。

複数のアダプターを集合できるようにするには (これは、スイッチにより複数のアダプターが同じ集合体に属することができるようにすることを意味する)、これらのアダプターは同じ回線速度 (例えば、すべて 100 Mbps またはすべて 1 Gbps) でなければならず、さらにすべて全二重モードでなければなりません。 異なる回線速度または異なる二重モードのアダプターを配置しようとする場合は、AIX システム上での集合体の作成には成功しますが、スイッチはアダプターを集合しない可能性があります。 スイッチが正常にアダプターを集合しない場合は、ネットワークのパフォーマンスが低下していることがあります。 スイッチ上の集合が成功したかどうかを判別する方法については、442 ページの『IEEE 802.3ad リンク集約のトラブルシューティング』を参照してください。

IEEE 802.3ad 仕様によれば、同じ IP アドレスへ送信されるパケットは、同じアダプター上で送信されます。このため、802.3ad モードで動作する場合、パケットは常に標準方式で配布され、ラウンドロビン方式で配布されることはありません。

バックアップ・アダプター・フィーチャーは、イーサチャネルと同様に、IEEE 802.3ad リンク集約で使用可能です。 バックアップ・アダプターは IEEE 802.3ad LACP にも従います。バックアップ・アダプターに接続したスイッチ・ポートは、IEEE 802.3ad にも対応している必要があります。

注 : IEEE 802.3ad を使用可能にするステップは、スイッチごとに異なります。 スイッチ内で LACP を使用可能にするために、どのような初期ステップ (ある場合) を実行する必要があるかを判別するには、ご使用のスイッチの資料を参照する必要があります。

IEEE 802.3ad の集合体を構成する方法については、440 ページの『IEEE 802.3ad リンク集約の構成』を参照してください。

IEEE 802.3ad リンク集約を構成する前に、以下のことを検討してください。

- 公式にはサポートされていませんが、IEEE 802.3ad の AIX 実装では、リンク集約に異なる回線速度のアダプターを含めることができます。ただし、同じ回線速度に設定され、全二重に設定されているアダプターだけを集約する必要があります。これは、スイッチ上にリンク集約を構成する際の潜在的な問題を回避するのに役立ちます。ご使用のスイッチで許可される集約のタイプについて詳しくは、スイッチの資料を参照してください。
- リンク集約で 10/100 イーサネット・アダプターを使用する場合は、これらのアダプターをリンク集約に追加する前に、アダプター上でリンク・ポーリングを使用可能にする必要があります。コマンド・ラインに `smitty chgenet` と入力します。「**Enable Link Polling (リンク・ポーリングを使用可能にする)**」の値を「yes (はい)」に変更して、Enter キーを押します。 このアクションを、リンク集約に追加する 10/100 イーサネット・アダプターごとに実行します。

  注 : AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) 以降では、リンク・ポーリング機構を使用可能にする必要はありません。 リンク・ポーリング・プログラムは自動的に始動されます。

IEEE 802.3ad リンク集約の構成

IEEE 802.3ad リンク集約を構成する手順は、次のとおりです。

1. コマンド・ラインに `smitty etherchannel` と入力します。
2. リストから「**Add an EtherChannel/Link Aggregation (イーサチャネル/リンク集約の追加)**」を選択して、Enter キーを押します。
3. リンク集約で 1 次イーサネット・アダプターとするものを選択して、Enter キーを押します。

   バックアップ・アダプターを使用する予定であれば、ここにバックアップに使用する予定のアダプターを選択しないでください。

   注 : 「**Available Network Adapters (使用可能なネットワーク・アダプター)**」に、すべてのイーサネット・アダプターが表示されます。 既に使用されている (インターフェースが定義されている) イーサネット・アダプターを選択すると、エラー・メッセージが出されます。 使用されているインターフェースを使用する場合は、最初に、それを切り離す必要があります。
4. 次の説明に従って、フィールドに情報を入力します。

   - **Parent Adapter (親アダプター):** イーサチャネルの親デバイスの情報を示します (イーサチャネルが共用イーサネット・アダプターに属する場合など)。 イーサチャネルが別のアダプター内に含まれていない場合 (デフォルト)、このフィールドには「NONE」の値が表示されます。 イーサチャネルが別のアダプター内に含まれている場合、このフィールドには親アダプターの名前が表示されます (「ent6」な

ど)。 このフィールドは情報フィールドであるため、変更することはできません。 親アダプター・オプションは、AIX 5.3 以降で使用可能です。

- **EtherChannel/Link Aggregation Adapters (イーサチャネル/リンク集約のアダプター):**リンク集約内で使用するすべての 1 次アダプターが表示されています。 前のステップで、これらのアダプターを選択しました。
- **Enable Alternate Address (代替アドレスを使用可能にする):** このフィールドはオプションです。 これを「yes （はい)」に設定すると、リンク集約が使用する MAC アドレスを指定できます。 このオプションを「no （いいえ)」に設定すると、リンク集約は最初のアダプターの MAC アドレスを使用します。
- **Alternate Address (代替アドレス):**「**Enable Alternate Address (代替アドレスを使用可能にする)**」を「yes （はい)」に設定した場合に、使用する MAC アドレスをここに指定します。 指定するアドレスは、0x で始まる 12 桁の 16 進数のアドレスでなければなりません (0x001122334455 など)。
- **Enable Gigabit Ethernet Jumbo Frames (ギガビット・イーサネット・ジャンボ・フレームを使用可能にする):** このフィールドはオプションです。 これを使用するには、スイッチがジャンボ・フレームをサポートしている必要があります。 これは、標準イーサネット (en) インターフェース (IEEE 802.3 (et) インターフェースではない) とだけ連動します。 これを使用可能にするには、「yes （はい)」に設定します。
- **Mode (モード)**: 8023ad と入力します。
- **Hash Mode (ハッシュ・モード)**: 以下のハッシュ・モードから選択できます。この選択項目により、アルゴリズムで発信アダプターを 判別するのに使用するデータ値が判別されます。
  - **default (デフォルト)**: このハッシュ・モードでは、パケットの宛先 IP アドレスを使用して発信アダプターが判別されます。 非 IP のトラフィック (ARP など) の場合、宛先 MAC アドレスの最後のバイトがこの計算に使用されます。 このモードでは、パケットはイーサチャネルに到達した順にイーサチャネルから送信されることが保証されますが、帯域幅をいっぱいには使用していません。
  - **src_port (ソース・ポート)**: パケットのソース UDP または TCP ポート値を使用して発信アダプターが判別されます。 パケットが UDP または TCP トラフィックでない場合は、宛先 IP アドレスの最後のバイトが使用されます。 パケットが IP トラフィックでない場合は、宛先 MAC アドレスの最後のバイトが使用されます。
  - **dst_port (宛先ポート)**: パケットの宛先 UDP または TCP ポート値を使用して発信アダプターが判別されます。 パケットが UDP または TCP トラフィックでない場合は、宛先 IP の最後のバイトが使用されます。 パケットが IP トラフィックでない場合は、宛先 MAC アドレスの最後のバイトが使用されます。
  - **src_dst_port (ソースおよび宛先ポート)**: パケットのソースおよび宛先の両方の UDP または TCP ポート値を使用して発信アダプターが判別されます (特に、ソースおよび宛先ポートが加算されて 2 で除算されてからアルゴリズムに送られる)。 パケットが UDP または TCP トラフィックでない場合は、宛先 IP の最後のバイトが使用されます。 パケットが IP トラフィックでない場合は、宛先 MAC アドレスの最後のバイトが使用されます。 クライアントとサーバーの 両方において、ほとんどの状態で、このモードでのパケット配布は良好です。

  パケット配布とロード・バランシングについて学習するには、434 ページの『イーサチャネルのロード・バランシング・オプション』を参照してください。
- **Backup Adapter (バックアップ・アダプター):** このフィールドはオプションです。 バックアップとして使用するアダプターを 入力します。
- **Internet Address to Ping (ping するインターネット・アドレス):** このフィールドはオプションで、メイン集約に 1 つ以上のアダプターと 1 つのバックアップ・アダプターが 含まれている場合にのみ使用可能です。 リンク集約は、ここで指定された IP アドレスまたはホスト名を ping します。 リンク集約が「**Number of Retries (再試行回数)**」フィールドの回数内に、また「**Retry Timeout (再試行タイムアウト)**」フィールドのインターバルで 、このアドレスに ping できなければ、リンク集約は次のアダプターに切り替えます。
- **Number of Retries (再試行回数):** 試行する ping の最大回数を入力します。この回数 ping が失敗すると、リンク集約は次のアダプターに切り替えます。 デフォルトは 3 回です。 このフィールドはオプションで、「**Internet Address to Ping (ping するインターネット・アドレス)**」を設定した場合にのみ有効です。

- **Retry Timeout (再試行タイムアウト (秒)):** リンク集約が、「**Internet Address to Ping (ping するインターネット・アドレス)**」に指定した IP アドレスに ping するインターバルを秒数で入力します。 デフォルトは 1 秒です。 このフィールドはオプションで、「**Internet Address to Ping (ping するインターネット・アドレス)**」を設定した場合にのみ有効です。

5. 必要なフィールドを変更したあと、Enter キーを押してリンク集約を作成します。
6. コマンド・ラインに `smitty chinet` と入力して、新たに作成したリンク集約デバイスに IP を構成します。
7. リストから新規リンク集約インターフェースを 選択します。
8. すべての必要なフィールドに値を入れて、Enter キーを押します。

IEEE 802.3ad リンク集約のトラブルシューティング

IEEE 802.3ad リンク集約をトラブルシューティングするには、**entstat** コマンドを使用します。

IEEE 802.3ad リンク集約で問題が生じる場合は、次のコマンドを使用してリンク集約の操作のモードを検査してください。

```
entstat -d device
```

ここで、*device* はリンク集約デバイスです。

これにより、スイッチから受信した LACPDU に基づいて LACP の進行状況について最大限の配慮がなされます。 使用可能な状況値は次のとおりです。

- `Inactive`: LACP が開始されていません。 これは、リンク集約がまだ構成されていない場合の状況です。 リンク集約にまだ IP アドレスが割り当てられていないか、またはそのインターフェースが 切り離されているためです。
- `Negotiating`: LACP は進行中ですが、スイッチがまだアダプターを 集合していません。 リンク集約が 1 分以上この状況のままである場合は、スイッチが正しく構成されているか検査してください。 例えば、LACP が各ポート上で使用可能になっているか検査する必要があります。
- `Aggregated`: LACP が成功し、スイッチがアダプターを 集合しました。
- `Failed`: LACP が失敗しました。 考えられる原因としては、集合体の中の各アダプターが 異なる回線速度または二重モードに設定されているか、または異なるスイッチに接続されていることなどです。 アダプターの構成を検査してください。

  さらに、スイッチの中には、隣接するポートのみを集合することができ、集合することができるアダプターの数が制限されている可能性があるものがあります。 スイッチに設けられている可能性がある制限を判別するためにスイッチに関する資料を調べてから、スイッチの構成を確認してください。

注 : リンク集約の状況は診断値であり、構成の AIX サイドには影響を及ぼしません。 この状況値は、最善の試行によって得られたものです。 集合に関する問題をすべてデバッグするには、スイッチの構成を検査するのが最もよい方法です。

以下の IEEE 802.3ad リンク集約の統計情報は、集約の各ポートにおける LACP の状況を示します。

これらの情報は、Actor (アクター) (IEEE 802.3ad リンク集約) と Partner (パートナー) (スイッチ・ポート) の両方に対して表示されます。

`System Priority` (システムの優先順位) : priority value for this system (このシステムの 優先順位の値)
`System` (システム) : value that uniquely identifies this system (このシステムを一意的に識別する値)
`Operational Key` (操作キー) : value that denotes which ports may be aggregated together
(集約されるポートを指示する値)
`Port Priority` (ポートの優先順位) : priority value for this port (このポートの優先順位の値)
`Port` (ポート) : unique value that identifies this port in the aggregation
(集約内のこのポートを識別する固有値)
State (状態):
`LACP activity` (LACP 活動状況) : Active or Passive (アクティブまたはパッシブ)-
whether to initiate sending LACPDUs always, or only in response to
another LACPDU (LACPDU の送信を常に開始するか、または別の
LACPDU に応じてのみ開始するか):

the IEEE 802.3ad Link Aggregation will always operate in Active mode
(IEEE 802.3ad リンク集約は常にアクティブ・モードで動作します)
LACP timeout（LACP タイムアウト）: Long or Short (長期または短期)-
time to wait before sending LACPDUs
(LACPDU を送信するまでに待機する時間):
the IEEE 802.3ad Link Aggregation will always use the long timeout
(IEEE 802.3ad リンク集約は常に長期のタイムアウトを使用します)Aggregation（集約）:
Individual or Aggregatable (個別または集計可能) - whether this
port can form an aggregation with other ports, or
if it can only form an aggregation with itself:
the port in a one-adapter IEEE 802.3ad Link Aggregation
will be marked as Individual, or Aggregatable
if there are more than one ports (このポートが他のポートとの集約を形成できるかどうか、または
このポート自体での集約のみを形成できるかどうか: 単一アダプター IEEE 802.3ad リンク集約でのポート
は、「個別」として、または「集計可能」(ポートが複数の場合) としてマークされます)
Synchronization（同期）: IN_SYNC or OUT_OF_SYNC (IN_SYNC または OUT_OF_SYNC)-
whether the aggregation has determined it has reached synchronization
with the partner
(集約がパートナーとの同期に到達したと判別したか)Collecting（収集）: Enabled or Disabled
(使用可能または使用不可)-
whether the IEEE 802.3ad Link Aggregation is collecting (receiving) packets
(IEEE 802.3ad リンク集約がパケットを収集 (受信) するか)Distributing（配布）: Enabled or
Disabled (使用可能または使用不可)-
whether the IEEE 802.3ad Link Aggregation is distributing (sending) packets
(IEEE 802.3ad リンク集約がパケットを配布 (送信) するか)Defaulted（デフォルト）: True or
False (真または偽)-
whether the IEEE 802.3ad Link Aggregation is using default values for the
partner's information
(IEEE 802.3ad リンク集約がパートナーの情報に対してデフォルト値を
使用するか)Expired（期限切れ）: True or False (真または偽)-
whether the IEEE 802.3ad Link Aggregation is operated in expired mode
(IEEE 802.3ad リンク集約が期限切れモードで動作するか)

以下の統計情報は、ポートごとと集約ベースの両方で表示されます。

Received LACPDUs（受信 LACPDU）: LACPDU packets received (受信された LACPDU パケット)
Transmitted LACPDUs（送信 LACPDU）: LACPDU packets sent out (送信された LACPDU パケット)
Received marker PDUs（受信マーカー PDU）: marker PDUs received (受信されたマーカー PDU)
Transmitted marker PDUs（送信マーカー PDU）: marker PDUs sent (送信されたマーカー PDU):
this version of the protocol does not implement the
marker protocol, so this statistic will always be zero
(このバージョンのプロトコルはマーカー・プロトコルを実装しないため、
この統計情報は常にゼロです)
Received marker response PDUs（受信マーカー応答 PDU）:
marker PDUs received (受信されたマーカー PDU)
Transmitted marker response PDUs（送信マーカー応答 PDU）: marker response PDUs sent
(送信されたマーカー応答 PDU):
this version of the protocol does not implement
the marker protocol, so this statistic will always be zero
(このバージョンのプロトコルはマーカー・プロトコルを実装しないため、
この統計情報は常にゼロです)
Received unknown PDUs（受信不明 PDU）: PDUs received of unknown type
(受信された不明なタイプの PDU)
Received illegal PDUs（受信不正 PDU）: PDUs received of known type, but which were
malformed, of unexpected length, or unknown subtype
(受信された既知のタイプの PDU。ただし、誤った形式、予期しない長さ、
または不明なサブタイプのもの)

インターオペラビリティーに関するシナリオ
イーサチャネルまたは IEEE 802.3ad リンク集約を構成する際に、以下のインターオペラビリティーに関するシナリオについて検討してください。

各シナリオの追加の説明は、この表の後に示されています。

表 *82. AIX* とスイッチ構成の異なる組み合わせ、および各組み合わせによって生じる結果

イーサチャネルのモード	スイッチ構成	結果
8023ad	IEEE 802.3ad LACP	OK - AIX は LACPDU を開始します。これにより、スイッチ上に IEEE 802.3ad リンク集約が設定されます。
standard (標準) または round_robin (ラウンドロビン)	イーサチャネル	OK - 従来のイーサチャネル動作が生じます。
8023ad	イーサチャネル	望ましくない - AIX とスイッチは集合を行うことができません。 AIX は LACPDU を開始しますが、スイッチはそれらを無視し、LACPDU を AIX に送信しません。 LACPDU が欠落しているため、AIX はそのリンクまたはポートでパケットを配布しません。 その結果、ネットワーク接続が失われます。
standard (標準) または round_robin (ラウンドロビン)	IEEE 802.3ad LACP	望ましくない - スイッチが集合を行うことができません。 スイッチがスイッチ・ポート間で MAC アドレスを移動させるので、結果としてパフォーマンスが低下することがあります。

それぞれの構成の組み合わせについて簡潔に説明します。

- イーサチャネルを使用した 8023ad:

  この場合、AIX は LACPDU を送信しますが、スイッチがイーサチャネルとして動作するので、LACPDU には応答がありません。 その結果、LACPDU が欠落しているため、AIX はそのリンクまたはポートをパケット配布のために使用しません。 これにより、ネットワーク接続が失われます。

  注: この場合、`entstat -d` コマンドは常に、集合が Negotiating (ネゴシエーション中) 状態にあると報告します。 さらに、entstat の出力の IEEE 802.3ad Port Statistics (IEEE 802.3ad ポート統計情報) セクションに、**Actor (アクター)** の **Distributing (配布)** が使用不可であることが示されます。
- イーサチャネルを使用した standard (標準) または round_robin (ラウンドロビン)

  これは最もよく使用されるイーサチャネル構成です。
- IEEE 802.3ad LACP を使用した standard (標準) または round_robin (ラウンドロビン)

  この設定は無効です。 スイッチが集合体を作成するために LACP を使用する場合は、AIX が LACPDU に応答することはないため、集合が起こることはありません。 これを正しく機能させるには、AIX 上のモードを 8023ad に設定する必要があります。

サポートされているアダプター
イーサチャネルおよび IEEE 802.3ad リンク集約は、IBM Power Systems Peripheral Component Interconnect-X (PCI-X) および PCI Express (PCIe) イーサネット・アダプター上でサポートされています。

その他の考慮事項は、次のとおりです。

- 仮想 I/O Ethernet アダプター

仮想入出力イーサネット・アダプターがサポートされるのは、以下の 2 つのイーサチャネル構成のみです。

- 1 次の仮想入出力イーサネット・アダプターが 1 つと、バックアップの仮想入出力イーサネット・アダプターが 1 つ。 この構成では、「**Internet Address to Ping (ping するインターネット・アドレス)**」属性を使用可能にして、イーサチャネルがリモート接続の障害を検出できるようにする必要があります。 Virtual I/O Server (VIOS) 2.2.3.0 以降、および AIX バージョン 7.1 (テクノロジー・レベル 3 以降適用) では、仮想イーサネット・デバイスの **poll_uplink** 属性を yes に設定することにより、仮想イーサネット・アップリンク・ステータス機能を使用して、サービスを提供している VIOS または共用イーサネット・アダプター (SEA) の障害を検出できます。
- 1 次のサポートされた物理イーサネット・アダプターが 1 つと、バックアップの仮想入出力イーサネット・アダプターが 1 つ。 この構成では、「**Internet Address to Ping (ping するインターネット・アドレス)**」属性を使用可能にして、イーサチャネルがリモート接続の障害を検出できるようにする必要があります。

- ホスト・イーサネット・アダプター (HEA)

  HEA 論理ポートは、イーサチャネル内のすべてのアダプターが HEA 論理ポートである場合には、イーサチャネルの下でサポートされます。 専用 HEA ポートに対しては、PCI/PCI-E アダプター付きのリンク集約がサポートされます。 さらに、バックアップ・アダプターとして PCI/PCI-E および仮想イーサネット・アダプターもサポートされます (基本アダプターに HEA が含まれる場合)。

  1 つのイーサチャネルで複数の HEA 論理ポートを基本アダプターとして使用する場合は、HEA 論理ポートに関連付けられた物理ポートも、イーサネット・スイッチで 1 つのイーサチャネルに入れる必要があります。 したがって、同じ HEA 物理ポートが関連付けられた HEA 論理ポートを使用するすべての区画も、1 つのイーサチャネルに入れる必要があります。

  例えば、区画 1 が次のように構成されているとします。

  - 1 つの論理 HEA ポートに物理 HEA ポート 0 が関連付けられている。
  - 1 つの論理 HEA ポートに物理 HEA ポート 1 が関連付けられている。
  - 上記の論理 HEA ポートを使用して 1 つのイーサチャネルが作成されている。

  同じシステム上の別の区画で、物理 HEA ポート 0 または物理 HEA ポート 1 が関連付けられた論理 HEA ポートを使用する必要がある場合、区画 1 と同様の構成で、両方の論理 HEA ポートにまたがるイーサチャネルをその別の区画用に作成する必要があります。これらの論理 HEA ポートのいずれかを別の区画でスタンドアロン・ポートとして使用しようとすると、パケットが正しい論理 HEA ポートに送信されないために接続の問題が生じる可能性があります。

  ネットワーク・インターフェース ・バックアップ構成 (1 つの基本アダプターと 1 つのバックアップ・アダプター) で論理 HEA ポートを使用する場合は、物理 HEA ポートがイーサネット・スイッチ上で固有の構成を必要としないため、この制限は存在しません。

  注 : 物理 HEA ポートに関連付けられた論理ポートが LACP 集約 (802.3ad) の一部として構成される場合は、これらの物理ポートはその LPAR に対して排他的でなければなりません。 HMC では、ポートがその他の LPAR に割り当てられなくなることはありませんが、この構成はサポートされません。

- Fibre Channel over Ethernet コンバージド・ネットワーク・アダプター

  共用ポート (イーサネットおよびファイバー・チャネル・トラフィックの両方に使用されるポート) およびその他のサポートされるアダプターの間のリンク集約がサポートされるのは、共用ポートに接続されているスイッチがファイバー・チャネル・トラフィックに影響を与えずにリンク集約をサポートできる場合のみです。

- Single Root I/O Virtualization (SR-IOV) アダプター

  SR-IOV 論理ポートを使用したリンク集約は、以下のいずれかの方法で着手できます。

  - IEEE 802.3ad リンク集約 (Link Aggregation Control Protocol (LACP) とも呼ばれる)
  - ネットワーク・インターフェース ・バックアップ (NIB)
  - LACP と NIB の両方

  単一ポートよりも大きな帯域幅が必要なネットワーク・アプリケーションでは、IEEE 802.3ad リンク集約を使用して、複数の SR-IOV 論理ポートを集約できます。 物理ポート用に構成される論理ポートは、

IEEE 802.3ad リンク集約を使用して集約される SR-IOV 論理ポートのみでなければなりません。 同じ物理ポート (複数の SR-IOV 論理ポートのいずれかが IEEE 802.3ad リンク集約構成の一部になるよう構成されている) 用に構成される複数の SR-IOV 論理ポートは、複数の LACP パートナーが物理ポートを介して通信している可能性があるため、スイッチで正しく管理されない場合があります。 IEEE 802.3ad リンク集約構成内の SR-IOV 論理ポートと同じ物理ポート上で 2 つ目の SR-IOV 論理ポートが構成されないようにするには、論理ポートの構成時に論理ポートの容量値を 100 (100%) に設定する必要があります。

単一のネットワーク障害に対する保護とともに単一ポートよりも小さい帯域幅が必要なネットワーク・アプリケーションの場合、SR-IOV 論理ポートを NIB 構成の一部にすることができます。 SR-IOV 論理ポートが、NIB 構成内の 1 次アダプターまたはバックアップ・アダプターのどちらかとして構成されている場合、物理ポートを他の SR-IOV 論理ポートで共有することができます。 この構成では、「**ping するインターネット・アドレス**」属性を使用可能にして、リモート接続の障害を検出することができます。 SR-IOV 論理ポートは、別の SR-IOV 論理ポートの 1 次アダプターまたはバックアップ・アダプター、仮想イーサネット・アダプター、あるいは物理アダプター・ポートのいずれにもすることができます。

新しいアダプターに関する追加のリリース情報については、ご使用の AIX レベルに該当する AIX リリース情報を参照してください。

重要 : 同一のイーサチャネルでさまざまなスピードのアダプターを 混在させることは、それらのアダプターの 1 つがバックアップ・アダプターとして 作動する場合でも、サポートされていません。 ただし、この種の構成が作動しないという意味ではありません。 さまざまなスピードが混在するシナリオであっても、イーサチャネル・ドライバーは、妥当な作業をすべて試行します。

関連情報

シングル・ルート I/O 仮想化

イーサチャネルのトラブルシューティング

イーサチャネルに問題が発生した場合は、検討すべきシナリオが多くあります。

問題を診断する際は、トレースと統計情報を使用することができます。これらの問題は、フェイルオーバーとジャンボ・フレームの問題に関係している可能性があります。

イーサチャネルのトレース

tcpdump および **iptrace** を使用して、イーサチャネルのトラブルシューティングを 行います。

送信パケットのトレース・フック ID は 2FA で、他のイベントのトレース・フック ID は 2FB です。 イーサチャネル全体として受信パケットをトレースすることはできませんが、各アダプターの受信トレース・フックをトレースできます。

イーサチャネル統計情報

entstat コマンドを使用して、イーサチャネル内のすべてのアダプターの 集約された統計情報を取得します。

例えば、`entstat ent3` では、ent3 の集約された統計情報が表示されます。 **-d** フラグを付けると、それぞれのアダプターの 統計情報が個々に表示されます。 例えば、`entstat -d ent3` と入力すると、イーサチャネルの集約された統計情報だけでなく、イーサチャネル内の個々のアダプターの 統計情報も表示されます。

注 : 「General Statistics (一般統計情報)」セクションで、「Adapter Reset Count (アダプターのリセット・カウント)」に表示される数は、フェイルオーバーの数です。 イーサチャネル・バックアップで、バックアップ・アダプターからメインのイーサチャネルに戻ることは、フェイルオーバーとは見なされません。 メイン・チャネルからバックアップへのフェイルオーバーのみカウントされます 。

「Number of Adapters (アダプター数)」フィールドに表示される数に、バックアップ・アダプターも入っています。

スロー・フェイルオーバー
ネットワーク・インターフェース・バックアップ・モードまたはイーサチャネル・バックアップを使用しているときに、フェイルオーバーに時間がかかる場合は、スイッチがスパン・ツリー・プロトコル (STP、Spanning Tree Protocol) を実行していないか確認してください。

スイッチは自己のスイッチ・ポートから MAC アドレスへのマッピング内に変更があったことを検出すると、スパン・ツリー・アルゴリズムを実行してネットワーク内にループがないか確認します。 ネットワーク・インターフェース・バックアップとイーサチャネル・バックアップは、ポートから MAC アドレスへのマッピングに変更を加える可能性があります。

スイッチ・ポートは、初期化されたあと、どのくらいでそれぞれのポートがパケットの転送または送信を開始するかを決定する、転送遅延カウンターを持っています。 このため、メイン・チャネルが再度使用可能にされたとき接続が再確立されるまでに遅延があるのに対して、バックアップ・アダプターへのフェイルオーバーの方がより高速です。 スイッチに構成された転送遅延カウンターをチェックし、それをできるだけ小さくして、可能な限り迅速にメイン・チャネルに戻ることができるようにします。

具体的には、イーサチャネル・バックアップ機能を正しく機能させるために、転送遅延カウンターが 10 秒を超えないようにします。そうでないと、メイン・イーサチャネルに正しく戻ることができないことがあります。 転送遅延カウンターを、スイッチに指定できる最も低い値に設定することをお勧めします。

フェイルオーバーを行わないアダプター
AIX 5.2 (5200-01 適用) の実行時に、アダプター障害によりフェイルオーバーが引き起こされていない場合は、アダプター・カードがリンク障害を検出できるようにリンク・ポーリングを使用可能にする必要があるかどうかチェックします。

アダプターによっては、リンク状況を自動的に検出できないものもあります。 当該の条件を検出するには、これらのアダプターは、タイマーを始動させることによって周期的にリンク状況を確認するリンク・ポーリング・メカニズムが使用可能である必要があります。 リンク・ポーリングは、デフォルトでは使用不可です。 しかし、これらのアダプターが含まれたイーサチャネルを正しく機能させるためには、イーサチャネルを作成する前に、各アダプターのリンク・ポーリング・メカニズムを使用可能にしておく必要があります。 AIX 5L バージョン 5.2 (5200-03 推奨メンテナンス・パッケージ適用) 以降を実行している場合は、リンク・ポーリングは自動的に開始されるので、この問題は起こりません。

リンク・ポーリング・メカニズムを持つアダプターは、**poll_link** と呼ばれる ODM 属性を持っています。それを「yes (はい)」に設定して、リンク・ポーリングを使用可能にしておく必要があります。 イーサチャネルを作成する前に、このチャネルに含めるすべてのアダプターに対して、次のコマンドを使用します。

```
smitty chgenet
```

「**Enable Link Polling (リンク・ポーリングを使用可能にする)**」の値を「yes (はい)」に変更して、Enter キーを押します。

ジャンボ・フレーム
イーサチャネルの **use_jumbo_frame** 属性を使用可能にするだけでなく、イーサチャネルを作成する前に、各アダプターのジャンボ・フレームを使用可能にする必要もあります。

この操作を行うには、次のコマンドを実行します。

```
smitty chgenet
```

イーサチャネルの **use_jumbo_frame** 属性を「yes (はい)」に設定すると、下層のすべてのアダプターでジャンボ・フレームが自動的に使用可能になります。

リモート・ダンプ
イーサチャネルではリモート・ダンプはサポートされません。

InfiniBand (IPoIB) を介したインターネット・プロトコル

インターネット・プロトコル (IP) パケットは、InfiniBand (IB) インターフェースを 介して 送信することができます。 このトランスポートは、ネットワーク・インターフェースを 使用して IB パケットの IP パケットをカプセル化することによって行われます。

IB を介して IP を使用するためには、InfiniBand 接続マネージャー (ICM) ドライバーと、少なくとも 1 つの IB デバイスをシステムにインストールして 構成する必要があります。 IB デバイスが既にインストール済みであるかどうか調べるには、`lsdev -C | grep iba` コマンドを使用します。 IB インターフェースが入っているファイルセットの名前は、`devices.common.IBM.ib` です。 現在サポートされているアダプター・ファイルセットの例としては、`devices.chrp.IBM.lhca` ファイルセットがあります。

ICM ドライバーを構成するには、450 ページの『InfiniBand コミュニケーション・マネージャー・ドライバーの構成』を参照してください。

InfiniBand インターフェース (IB IF) を作成するには、その IB IF によって既存のブロードキャスト・マルチキャスト・グループとユーザー提供の PKEY (または、ユーザーが提供していない場合はデフォルトの PKEY = 0xFFFF が使用される) およびユーザー提供の Q_Key (または、ユーザーが提供していない場合はデフォルトの Q_Key = 0x1E が使用される) を結合できなければなりません。 ブロードキャスト・マルチキャスト・グループは、ブロードキャストと **ARP** パケットを送信するために、 インターフェースが 結合する必要のあるマルチキャスト・グループです。 このようなブロードキャスト・マルチキャスト・グループが存在しない場合、またはこのグループをインターフェースによって 作成できない場合は、IB IF の作成は失敗します。

ユーザーは、コマンド・ライン・インターフェースまたは SMIT ユーザー・インターフェースを 使用して IB IF を作成または変更することができます。 IB IF の作成に必要なパラメーターは、以下のとおりです。

- *interface name (インターフェース 名)*
- *adapter name (アダプター 名)*
- *port number (ポート番号)*
- *interface IP address (インターフェース IP アドレス)*

IB IF の変更に使用するパラメーターは、以下のとおりです。

- *internet address (インターネット・アドレス)*
- *network mask (ネットワーク・マスク)*
- *MTU size (MTU サイズ)* (必要な MTU と等しい。IB ヘッダーの場合は 4 バイト未満)
- *state (状態)*
- *Send and Receive queue size (送信キューと受信キューのサイズ)* (デフォルトは 4000)
- *Multicast Queue Key (マルチキャスト・キュー・キー)*
- *Super packet on or off (スーパー・パケットのオン/オフ)*

以下は、コマンド・ラインから IB IF を作成するために使用するコマンドの例です。

```
$ /usr/sbin/mkiba -i ib0 -p 1 -A iba0 -a 1.2.3.8 [-P -1 -S "up" -m "255.255.254.0" -M 2044]
```

この場合、

項目	説明
`-M 2044`	最大伝送単位
`-m "255.255.254.0"`	ネットマスク
`-p 1`	ポート番号 (指定しない場合は、デフォルトにより *1*)
`-A iba0`	IB デバイス名
`-a 1.2.3.8`	IF IP アドレス
`-i ib0`	インターフェース 名

項目	説明
-P -1	区画キー (指定しない場合は、デフォルトで *PKEY*。 一度インターフェースが 作成されると、PKEY は変更できません。 ユーザーはネットワーク管理者からデフォルト以外の PKEY をもらう必要があります。)
-S "up"	インターフェースの 状態
-q 8000	受信キューと送信キューのサイズ (各サイズ)。
-Q 0x1E	マルチキャスト・グループに割り当てられているマルチキャスト・キュー・キー (このキーが提供されていない場合のデフォルトは Q_KEY = 0x1E です)。
-k "on"	スーパーパケットにより 、インターフェース TCP/IP MTU の最大値は 64K まで可能です。 スーパーパケットは 、リモート・ホストでも機能できるように使用可能にする必要があります。

以下は、SMIT ユーザー・インターフェースから IB IF を作成するために使用するコマンドの例です。

```
$ smitty inet
```

「Network Interface Selection (ネットワーク・インターフェースの 選択)」メニューが表示されたら、以下の手順を実行します。

1. 「**Add a Network Interface (ネットワーク・インターフェースの 追加)**」または「**Change / Show Characteristics of a Network Interface (ネットワーク・インターフェースの 特性の変更/表示)**」を選択します。 「Add a Network Interface (ネットワーク・インターフェースの 追加)」メニューが 表示されます。
2. 「Add a Network Interface (ネットワーク・インターフェースの 追加)」メニューで、「**Add an IB Network Interface (IB ネットワーク・インターフェースの 追加)**」を選択する。 「Add an IB Network Interface (IB ネットワーク・インターフェースの 追加)」メニューが 表示されます。
3. 「Add an IB Network Interface (IB ネットワーク・インターフェースの 追加)」 メニューで、必要な変更を行い、Enter キーを押す。

ARP エントリーの作成、表示、追加、および削除、ならびに ARP タイマーの変更

「**アドレス解決プロトコル (ARP)**」エントリーは、 インターフェースが 、同じマルチキャスト・グループ内にない別のインターフェースとも 通信が できるようにします。

ARP エントリーは、`arp -t ib` コマンドを使用して 手動で作成することができます。

すべての **ARP** エントリーを表示するには、`$ arp -t ib -a` コマンド を実行します。 特定数の **ARP** エントリーを表示したい場合は、その数を指定することができます。 例えば `$ arp -t ib -a 5` と指定すると、5 つの **ARP** エントリーが表示されます。

次のコマンドは、**ARP** エントリーを 1 つ追加します。

```
$ arp -t ib -s IB interface name dlid <16 bits DLID> dqp
16 bits hex Destination Queue Pair Number
ipaddr <Destination IP Address>
```

この場合、

項目	説明
DLID	宛先ローカル ID
DGID	宛先グローバル ID

次のコマンドは **ARP** エントリーを 1 つ除去します。

```
$ arp -t ib -d IP Address
```

以下のコマンドを使用して、完全および不完全な ARP エントリーに関する ARP エントリーのタイマー値を変更します。 これらの値を使用して、一定期間の経過後に ARP エントリーを除去します。

```
arp –t ib –i <number in complete minutes to remove incomplete ARP entries>
-c <number in complete minutes to remove complete ARP entries>
```

除去対象となる不完全な ARP エントリーの現行デフォルト時間は 3 分です。 完全な ARP エントリーの場合、デフォルト時間は 24 時間です。 これらの値を変更する必要がある場合は、このコマンドの実行により、構成済み (または定義済みの状態) のすべての現行インターフェースの 値のみが変更されます。 新規インターフェースが 構成される場合は、このコマンドを再度実行する必要があります。 これらの値は ODM でも変更されます。

ifconfig コマンドを実行して、これらの値を 1 つの特定のインターフェースに 動的に変更することができます。

```
不完全な ARP エントリー・タイマーを変更するには、次のようにします。
ifconfig ib0 inc_timer 4
ifconfig ib0 com_timer 60
```

InfiniBand インターフェースのパラメーターの 変更

IB IF のパラメーターを変更するには、SMIT ユーザー・インターフェースまたはコマンド ・ライン・コマンドを使用します。

SMIT を使用して IB IF のパラメーターを変更するには、以下のようにします。

1. `$ smitty inet` コマンドを実行する。

   「Network Interface Selection (ネットワーク・インターフェースの 選択)」メニューが 表示されます。
2. 「Network Interface Selection (ネットワーク・インターフェースの 選択)」メニュー で、「**Change/Show Characteristics of a Network Interface (ネットワーク・インターフェースの 特性の変更/表示)**」を選択する。

   「Available Network Interfaces (使用可能なネットワーク・インターフェース)」メニューが表示されます。
3. 「使用可能なネットワーク・インターフェース 」メニューで、 「**InfiniBand インターフェース** 」を選択する。

   「Change/Show an IB Interface (IB インターフェースの 変更/表示)」メニューが表示されます。
4. 必要なパラメーターを変更する。

コマンド・ラインで IB IF パラメーターを変更するには、`$ ifconfig` コマンドを実行します。 次のコマンドは、コマンド・ラインから IB IF パラメーターを変更します。

```
$ ifconfig ib0 [ib_port port number mtu maximum transmission unit p_key
16 bits hex partition key  ib_adapter InfiniBand adapter name netmask
dotted decimals]
```

```
$ ifconfig ib0 inc_timer  3  com_timer 60
```

- *inc_timer* は、不完全な ARP エントリーの有効期限が切れる時間 (分単位) です。 デフォルトは 2 分です。
- *com_timer* は、完全な ARP エントリーの有効期限が切れる時間 (分単位) です。 デフォルトは 24 時間です。

InfiniBand コミュニケーション・マネージャー・ドライバーの構成

InfiniBand コミュニケーション・マネージャーを構成するには、以下の手順を使用します。

1. `$ smitty icm` コマンドを実行する。

   「InfiniBand Communication Manager (InfiniBand コミュニケーション・マネージャー)」メニューが表示されます。
2. 「InfiniBand Communication Manager (InfiniBand コミュニケーション・マネージャー)」メニューで、「**Add an InfiniBand Communication Manager (InfiniBand コミュニケーション・マネージャーの追加)**」を選択する。
3. 「Add an InfiniBand Communication Manager (InfiniBand コミュニケーション・マネージャーの追加)」メニューで、 「**Add an InfiniBand Communication Manager (InfiniBand コミュニケーション・マネージャーの追加)**」を選択する。

「Name of IB Communication Manager to Add (追加する IB コミュニケーション・マネージャー名)」メニューが表示されます。

4. 「追加する IB コミュニケーション・マネージャー名」メニューで、「**management icm InfiniBand**」を選択する。
5. デフォルトを使用するか、必要なパラメーターがあればそれを変更して、Enter キーを押す。

iSCSI ソフトウェア・イニシエーターとソフトウェア・ターゲット

iSCSI ソフトウェア・イニシエーターにより、イーサネット・ネットワーク・アダプター上で TCP/IP を使用して AIX からストレージ・デバイスにアクセスできます。 iSCSI ソフトウェア・ターゲットにより、AIX は、RFC 3720 で定義されている iSCSI プロトコルを使用して、他の iSCSI イニシエーターがアクセスするローカル・ストレージをエクスポートできます。

iSCSI テクノロジーは、IP を介した SAN テクノロジーと言及されることもよくありますが、このテクノロジーを使用すると IP ネットワークを介した Storage Area Network を取り入れることができます。 iSCSI は、**TCP/IP** によって SCSI 情報をカプセル化し、イーサネットおよびギガビット・イーサネット・ネットワークを介して転送できるようにするオープン標準ベースの方法です。 既存のイーサネット・ネットワークで iSCSI を使用すると、場所全体の独立性を確保して SCSI のコマンドやデータを転送できます。 iSCSI ソリューションでは、以下のさまざまな完全に関連しているコンポーネントを使用します。

- イニシエーター

  クライアント上にあるデバイス・ドライバーです。 SCSI コマンドをカプセル化し、IP ネットワークを介してターゲット・デバイスに経路指定します。
- ターゲット・ソフトウェア

  このソフトウェアは、IP ネットワークを介して、カプセル化された SCSI コマンドを受信します。 またこのソフトウェアは、構成サポートとストレージ管理サポートを備えています。
- ターゲット・ハードウェア

  このハードウェアは、組み込みストレージのあるストレージ装置です。 またこのハードウェアは、独自の内部ストレージのないゲートウェイやブリッジ製品でもかまいません。

iSCSI ソフトウェア・イニシエーターの構成

ソフトウェア・イニシエーターを SMIT を使用して構成する手順は、次のとおりです。

1. 「**Devices (デバイス)**」を選択します。
2. 「**iSCSI**」を選択します。
3. 「**Configure iSCSI Protocol Device (iSCSI プロトコル・デバイスの構成)**」を選択します。
4. 「**Change/Show Characteristics of an iSCSI Protocol Device (iSCSI プロトコル・デバイスの特性の変更/表示)**」を選択します。
5. 「**Initiator Name (イニシエーター名)**」の値が正しいことを確認します。

   「**Initiator Name (イニシエーター名)**」の値は、ログイン操作中に iSCSI ターゲット・デバイスによって使用されます。

   注 : ソフトウェアのインストール時に、デフォルトのイニシエーター名が割り当てられます。 ローカル・ネットワークの命名規則と一致するように、イニシエーター名を変更することができます。 複数の iSCSI イニシエーター・デバイスを使用する場合、それらのイニシエーター・デバイスごとに固有の名前を割り当てる必要があります。
6. 「**Maximum Targets Allowed (使用できる最大ターゲット数)**」フィールドは、構成できる iSCSI ターゲット・デバイスの最大数に対応しています。

   この数値を小さくする場合は、構成時に iSCSI プロトコル・ドライバー用に事前割り振りされるネットワーク・メモリーの量も少なくします。
7. iSCSI ターゲット・デバイスをディスカバーするために、「**Discovery Policy (ディスカバリー・ポリシー)**」 フィールドを使用して、iSCSI ディスカバリー・メソッドを構成します。

   iSCSI イニシエーター・ソフトウェアでは以下のディスカバリー・メソッドがサポートされます。

   file
   ターゲット・デバイスに関する情報は、構成ファイルに保管されます。

注 :「**ディスカバリー・ポリシー (Discovery Policy)**」フィールドが `file` に設定されていると、iSCSI ターゲット・デバイスの名前は、「**ディスカバリー・ファイル名 (Discovery file name)**」属性に指定されたファイルから読み取られます。 iSCSI ソフトウェア・イニシエーターの複数のインスタンスを使用する場合、イニシエーター・デバイスが異なる iSCSI ターゲット・デバイスにアクセスするのであれば、複数のディスカバリー・ファイルを作成する必要があります。

odm
ターゲットに関する情報は、オブジェクト・データ・マネージャー (ODM) オブジェクトに保管されます。 iSCSI ディスクをブート・ディスクとして使用している場合、または **rootvg** ブートの一部として使用している場合は、**odm** ディスカバリー・メソッドを使用する必要があります。 静的にディスカバーされた iSCSI ターゲットの ODM への追加を参照してください。

isns
ターゲットに関する情報は internet Storage Name Service (iSNS) サーバーに保管され、iSCSI イニシエーター構成の間に自動的に取り出されます。

slp
ターゲットに関する情報は、Service Location Protocol (SLP) サービス・エージェントまたはディレクトリー・エージェントに保管され、iSCSI イニシエーターの構成時に自動的に取り出されます。

8. **isw_err_recov** 属性の値 (**delayed_fail** または **fast_fail**) を選択します。 この属性では、iSCSI イニシエーターがネットワーク・エラーからの復旧を試行する回数を設定できます。 **delayed_fail** 値は、iSCSI ソフトウェア・イニシエーターが使用するデフォルト動作を表します。 **delayed_fail** 値は、デバイスへの iSCSI パスが 1 つである環境にお勧めします。 **fast_fail** 値は、iSCSI ソフトウェア・イニシエーターで使用される各種タイムアウト値と再試行値を削減します。 この値は、iSCSI デバイスへの複数のパス (MPIO) を使用している場合、または iSCSI デバイスが LVM ミラーリングを利用している場合にお勧めします。 これらの状況で **fast_fail** 値を使用すると、AIX オペレーティング・システムは、1 つのパスにネットワーク障害が発生したときに、より迅速に作動中のパスに切り替えます。
9. **initial_r2t** 属性および **immediate_data** 属性として必要な値を選択します。 デフォルトで、**initial_r2t** 属性は yes に、**immediate_data** 属性は no に設定されています。これらの値は切り替えることができます。

   intial_r2t 属性を no に設定すると、iSCSI ソフトウェア・イニシエーター・デバイスは、データを iSCSI ディスクに書き込むのと同時に最初の転送可能プロトコル・データ単位 (PDU) を使用しないよう、iSCSI ターゲット・デバイスとネゴシエーションを行うことができます。 **immediate_data** 属性を yes に設定すると、iSCSI ソフトウェア・イニシエーター・デバイスは、データを直接 iSCSI ディスクに送信するよう、iSCSI ターゲット・デバイスとネゴシエーションを行うことができます。 **intial_r2t** 属性および **immediate_data** 属性として適切な値を選択すると、データを iSCSI ディスクに書き込むときのパフォーマンスが向上します。

ソフトウェア・イニシエーターを構成したら、次の手順を実行します。

1. ディスカバリー・ポリシーが **file** の場合、`/etc/iscsi/targets` ファイルを編集して、デバイスの構成時に必要な iSCSI ターゲットを組み込みます。

   ファイル中の個々のコメント行にされていない行は、iSCSI ターゲットを表します。 詳しくは、ファイル参照 の targets File を参照してください。

   ディスカバリー・ポリシーが **odm** の場合、**mkiscsi** コマンドまたは SMIT パネルを使用して、ODM 内にターゲット定義を作成します。 詳しくは、静的にディスカバーされた iSCSI ターゲットの ODM への追加を参照してください。

   ディスカバリー・ポリシーが **isns** または **slp** の場合、iSNS サーバーまたは SLP サーバーが適切に構成されており、iSCSI イニシエーターによってアクセス可能であることを確認してください。

   iSCSI デバイス構成では、適切に構成されたネットワーク・インターフェースを 介して iSCSI ターゲットに達することができる必要があります。 iSCSI ソフトウェア・イニシエーターは、10/100 イーサネット LAN を使用して作動できますが、他のネットワーク・トラフィックから分離されたギガビット・イーサネット・ネットワークで使用するように設計されています。
2. ターゲットを定義した後で、次のコマンドを入力します。

```
cfgmgr -l iscsi0
```

このコマンドは、ソフトウェア・イニシエーター・ドライバーを再構成します。 これにより、ドライバーは /etc/iscsi/targets ファイルにリストされているターゲットと通信し、検出されたターゲット上の LUN ごとに新しい hdisk を定義するよう試みます。 詳しくは、**cfgmgr** コマンドの説明を参照してください。

注: 該当するディスクが定義されない場合は、イニシエーター、ターゲット、および iSCSI ゲートウェイの構成を検査し、正しいことを確認してから、**cfgmgr** コマンドを再実行してください。

iSCSI ソフトウェア・イニシエーター・デバイスのパラメーターをさらに構成したい場合は、次のように SMIT を使用します。

1.「**Devices (デバイス)**」を選択します。
2.「**Fixed Disk (ディスク)**」を選択します。

典型的なソフトウェア・イニシエーター・デバイスは次の例のように表示されます。

```
hdisk2   Available              Other iSCSI Disk Drive
```

iSCSI ディスクが制御バイト中にコマンド・タグ・キューイングと NACA=1 をサポートしている場合は、このディスクのキューの深さの設定を大きな値に変更することを考慮してください。 値を大きくすると、デバイスのパフォーマンスを改善するのに役立つことがあります。 最適なキューの深さの設定は、ドライブ上の実際のキュー・サイズを超えない値です。 ドライブのキュー・サイズより大きな値にキューの深さを設定すると、パフォーマンスが低下する可能性があります。 ドライブのキュー・サイズを調べるには、ドライブの資料を参照してください。

複数の iSCSI ソフトウェア・イニシエーター・デバイスの構成

インターネット Small Computer Systems Interface (iSCSI) ソフトウェアは、iSCSI ソフトウェア・イニシエーターを表すデバイスを作成します。 デフォルトで、デバイスの名前は `iscsi0` になります。 このデバイスは、それ自体に関連付けられている iSCSI 名が入った **initiator_name** 属性を持ちます。

複数の iSCSI ソフトウェア・イニシエーター・デバイスを、1 つの AIX オペレーティング・システム・インスタンスに作成することができます。 複数の iSCSI ソフトウェア・イニシエーター・デバイスが存在することによって、以下の利点があります。

- マルチパス入出力 (MPIO) をサポートする iSCSI ディスク用の MPIO を簡単に作成できます。 それぞれの MPIO パスは、それ自体の TCP/IP ソケット接続を作成します。そのため、iSCSI データ・トラフィックは複数の接続に広がり、並行処理がいくつも行われることによってパフォーマンスが向上します。
- 複数の iSCSI 入出力要求を論理的に分離することができます。 `iscsi0` デバイス上のディスクが、あるアプリケーションによって使用され、同時に `iscsi1` デバイス上のディスクが別のアプリケーションによって使用される、ということが可能です。 このようなケースでは、入出力要求を分離することによって、アプリケーション間の入出力要求の競合の可能性は低くなります。

iSCSI ソフトウェア・イニシエーター・デバイスを作成するには、次のコマンドを実行します。

```
mkdev -c driver -s node -t iscsi -d
```

このコマンドは、iSCSI ソフトウェア・イニシエーター・デバイスを作成し、そのデバイスの名前を出力します。 **mkdev** コマンドを使用して、複数のイニシエーター・デバイスを作成することができます。 8 個を超える iSCSI イニシエーター・デバイスを作成も使用もしないようお勧めします。

デバイスが作成された後、そのデバイスに固有の名前を割り当てる必要があります。 **chdev** コマンドを使用すると、新しいデバイスの **initiator_name** 属性を設定できます。 また、`iscsi0` デバイスの構成と同じように、特定の iSCSI ターゲット・デバイスにアクセスするよう、新しいデバイスを構成する必要があります。 複数の iSCSI ソフトウェア・イニシエーター・デバイスを作成する場合、それらのデバイスを、同じ iSCSI ターゲット・デバイスにアクセスするよう構成することも、異なる iSCSI ターゲット・デバイスにアクセスするよう構成することもできます。

2 つのイニシエーター・デバイスが同じターゲット・デバイスにアクセスし、ファイル・ディスカバリー・ポリシーを使用している (つまり、 **disc_policy** 属性が `file` に設定されている) 場合、これらのイニシエーター・デバイスは同じファイル名を指す必要があります。 2 つのイニシエーター・デバイスが異なるターゲット・デバイスにアクセスしている場合、これらのイニシエーター・デバイスは異なるファイル名

を指す必要があります。 2 つのイニシエーター・デバイスが同じターゲット・デバイスにアクセスし、Object Data Manager (ODM) ディスカバリー・ポリシーを使用している (つまり、 **disc_policy** 属性が odm に設定されている) 場合、最初のイニシエーター・デバイスの ODM 項目を、2 番目のイニシエーター・デバイス用にリストされるよう複製する必要があります。 **cpiscsi** コマンドを使用すると、iSCSI ターゲット・デバイス構成を複製することができます。

iSCSI ソフトウェア・ターゲットの構成

iSCSI ソフトウェア・ターゲット・ドライバーにより、AIX は、1 つの iSCSI ターゲット・デバイスまたは複数の iSCSI ターゲット・デバイスとして機能することができます。 iSCSI ターゲット・ドライバーは、ローカル・ディスク、論理ボリューム、またはローカル・ファイルを iSCSI イニシエーターにエクスポートします。このイニシエーターは、iSCSI プロトコルおよび TCP/IP を使用して AIX に接続します。

ターゲット・デバイスには、それぞれ 1 つの iSCSI 修飾名と 1 組の論理装置番号 (LUN) があり、仮想 iSCSI ターゲットに接続するイニシエーターが使用できるようになっています。 ユーザーは、着信接続を受け入れるためにターゲット・ドライバーが使用するネットワーク・インターフェースおよび TCP/IP ポート番号を、ターゲット・デバイスごとに指定できます。

注 : iSCSI ターゲット・ファイルセットがインストールされている必要があります。 ファイルセット名は devices.tmiscsw.rte で、ファイルセットは AIX Expansion pack に含まれています。

iSCSI ターゲット・ドライバーを構成するには、次の手順を実行します。

1. 次の SMIT パスを使用して、iSCSI ターゲット・ドライバーの単一インスタンスを作成します。 このインスタンスは他の iSCSI オブジェクトのコンテナーとして働きます。

   「**Devices (デバイス)**」 > 「**iSCSI**」 > 「**iSCSI Target Device (iSCSI ターゲット・デバイス)**」 > 「**iSCSI Target Protocol Device (iSCSI ターゲット・プロトコル・デバイス)**」 > 「**Add an iSCSI Target Protocol Device (iSCSI ターゲット・プロトコル・デバイスの追加)**」
2. iSCSI ターゲット・ドライバーによって割り当てられる仮想 iSCSI ターゲットごとに 1 つの iSCSI ターゲット・デバイスを作成します。 次の SMIT パスを使用して、iSCSI ターゲット・デバイスをそれぞれ作成します。

   「**Devices (デバイス)**」 > 「**iSCSI**」 > 「**iSCSI Target Device (iSCSI ターゲット・デバイス)**」 > 「**iSCSI Targets (iSCSI ターゲット)**」 > 「**Add an iSCSI Targets (iSCSI ターゲットの追加)**」
3. 次の SMIT パスを使用して、ターゲット・デバイスごとに 1 つ以上の LUN を作成します。

   注 : LUN は仮想ターゲットに接続するイニシエーターからアクセスできます。 iSCSI ターゲット上で、各 LUN は、前もって物理ボリュームで定義された論理ボリューム、あるいはローカル・ファイルシステム上で前もって作成されたファイルのいずれかに関連付けることができます。 iSCSI ターゲット論理装置に関連付けられた物理ボリュームは、iSCSI ターゲット・ドライバーが稼働する AIX システムではこれ以外の方法では使用できません。

   「**Devices (デバイス)**」 > 「**iSCSI**」 > 「**iSCSI Target Device (iSCSI ターゲット・デバイス)**」 > 「**iSCSI Target LUN (iSCSI ターゲット)**」

   通常はこれで構成手順が完了します。 ただし、チャレンジ・ハンドシェーク認証プロトコル (CHAP) を使用する場合、またはアクセス制御リスト (ACL) を使用して、どのイニシエーターがどの LUN にアクセスできるかを指示する場合は、ターゲット構成を完了するのに追加のステップが必要です。

   - イニシエーターの CHAP 承認を使用する場合は、`/etc/tmiscsi/autosecrets` ファイルを編集し、イニシエーターがログインに使用する機密事項を追加します。 `/etc/tmiscsi/autosecrets` ファイルには、ターゲットごとに 1 つのエントリーが含まれます。 各エントリーのフォーマットは次のとおりです。

     target_name chap_name chap_secret
   - ACL を使用して、どのイニシエーターがどの LUN にアクセスできるかを指示する場合は、`/etc/tmiscsi/access_lists` ファイルを編集し、ターゲットごとに 1 つのエントリーを追加します。 各エントリーのフォーマットは次のとおりです。

     target_name|lun_name iSCSI_name, iSCSI_name,...

関連情報
/etc/tmiscsi/autosecrets
/etc/tmiscsi/access_lists
/etc/tmiscsi/isns_servers

iSCSI ソフトウェア・イニシエーターの考慮事項
iSCSI ソフトウェア・イニシエーターを扱う場合は、以下の事項を考慮します。

- ターゲット・ディスカバリー

  iSCSI ソフトウェア・イニシエーターは、以下の 4 つの形式のターゲット・ディスカバリーをサポートします。

  file
  テキスト・ファイルを使用して個々のターゲットが構成されます。

  odm
  ODM オブジェクトを使用して個々のターゲットが構成されます。 iSCSI ディスクをブート・ディスクとして使用している場合、または rootvg ブートの一部として使用している場合は、**odm** ディスカバリー・メソッドを使用する必要があります。

  isns
  個々のターゲットが 1 つ以上の Internet Storage Name Service (iSNS) サーバーに登録されます。

  slp
  個々のターゲットが 1 つ以上の Service Location Protocol (SLP) サービス・エージェントまたはディレクトリー・エージェントに登録されます。

- iSCSI 認証

  iSCSI ソフトウェア・イニシエーターは、チャレンジ・ハンドシェーク認証プロトコル (CHAP) としてローカル iSCSI 修飾名を使用します (CHAP 名が指定されていない場合)。 ターゲット・ファイルまたは ODM 構成で代替 CHAP 名を指定できます。 CHAP 名の指定について詳しくは、ターゲット・ファイルまたは mkiscsi コマンドの参照情報を確認してください。

  イニシエーター認証の構成に使用できるのは、CHAP(MD5) のみです。 ターゲット認証はインプリメントされていません。

- iSCSI ソフトウェア・イニシエーター MPIO

  AIX iSCSI ソフトウェア・イニシエーターではマルチパス入出力 (MPIO) がサポートされます。 ファイルまたは ODM ディスカバリー・ポリシーのいずれかを使用している場合は、AIX 内で複数のネットワーク・インターフェースを 使用することにより、ストレージ・デバイスの複数のポートに接続するためのエントリーを追加してください。 iSCSI ドライバーは、同一デバイスへのマルチパスを 認識し、MPIO 構成でそれらのパスを構成します。 これは、他のストレージ・デバイス・プロトコル用の MPIO 構成に似ています。

  サード・パーティーの iSCSI ディスク ODM 定義を使用している場合は、iSCSI ディスクへのマルチパスを使用するために、それらの定義の入手可能な最新版がインストールされていることを確認してください。

- 構成される LUN 数

  iSCSI ソフトウェア・イニシエーターを使用してテストされた、構成される LUN の最大数は、iSCSI ターゲット当たり 128 です。 ソフトウェア・イニシエーターは、iSCSI ターゲットごとに 1 つの TCP 接続 (iSCSI セッション当たり 1 つの接続) を使用します。 この TCP 接続は、ターゲット用に構成されるすべての LUN の間で共有されます。 ソフトウェア・イニシエーターの TCP ソケットの送受信スペースは、両方ともシステム・ソケット・バッファーの最大値に設定されます。 この最大値は、**sb_max** ネットワーク・オプションによって設定されます。 デフォルトは、1 MB です。

- ボリューム・グループ

  構成上の問題とエラー・ログ項目を避けるには、iSCSI デバイスを使用してボリューム・グループを作成する際に、次の説明に従ってください。

- リブート後に非アクティブ状態になるように、iSCSI デバイスを使用して作成するボリューム・グループを構成します。 iSCSI デバイスの構成後に、iSCSI によって戻されたボリューム・グループを手動で活動化します。 それから、関連したファイルシステムをマウントします。

  ボリューム・グループは、iSCSI ソフトウェア・ドライバーとは違うブート・フェーズ時に活動化されます。 そのため、ブート・プロセス時に iSCSI ボリューム・グループを活動化できません。
- 非 iSCSI デバイス間にボリューム・グループをスパンしないでください。

- 入出力障害

  iSCSI ターゲット・デバイスへの接続が失われると、入出力障害が起こります。 入出力障害およびファイルシステムの破損が起こらないようにするには、入出力アクティビティーをすべて 停止し、iSCSI によって戻されたファイルシステムをアンマウントしてから、アクティブな iSCSI ターゲットに対する接続が長期間失われる作業を行ってください。

  アプリケーションが iSCSI デバイスを使用して入出力アクティビティーを 試行している際に iSCSI ターゲットへの接続が失われると、最終的に入出力エラーが起こります。 基礎となる iSCSI デバイスは使用中のままなので、iSCSI によって戻されたファイルシステムをアンマウントできなくなる 可能性があります。

  アクティブな iSCSI ターゲットへの接続が失われたために入出力障害が起こった場合は、ファイルシステム保守を実行しなければなりません。 ファイルシステム 保守を行うには、**fsck** コマンドを実行してください。

- AIX iSCSI ソフトウェア・イニシエーターまたは AIX iSCSI ソフトウェア・ターゲットをループバック・インターフェース (`lo0`) で使用しないでください。ループバック・インターフェースの 割り込みの処理は、物理または仮想のイーサネット・アダプター・ネットワーク・インターフェースの 割り込みの処理と異なります。ループバック・インターフェースが iSCSI ソフトウェア・ドライバーで使用された場合、AIX オペレーティング・システムは操作を停止する可能性があります。

関連情報

静的に発見された iSCSI ターゲットの ODM への追加

iSCSI セキュリティーの考慮事項

`/etc/iscsi` ディレクトリー、`/etc/tmiscsi` ディレクトリー、およびそれらのディレクトリー内のファイルは、ファイルの許可と所有権により、特権のないユーザーから保護されています。

CHAP 機密事項は、`/etc/iscsi/targets` ファイルおよび `/etc/tmiscsi/autosecrets` ファイル中に平文として保管されています。

注 : これらのファイルのオリジナルのファイル許可と所有権を変更しないでください。

iSCSI パフォーマンスの考慮事項

iSCSI から最高のパフォーマンスを得るには、以下の構成を設定します。

パフォーマンスを最高にするには、次のようにします。

- AIX Gigabit Ethernet アダプターおよび iSCSI ターゲット・インターフェースの TCP 大容量送信、TCP 送受信フロー制御、およびジャンボ・フレーム・フィーチャーを使用可能にします。
- AIX システム上で最大の iSCSI 入出力スループットを得られるように、ネットワーク・オプションとインターフェースのパラメーターを 次のように調整します。
  - RFC 1323 ネットワーク・オプションを使用可能にします。
  - **tcp_sendspace**、**tcp_recvspace**、**sb_max**、および **mtu_size** ネットワーク・オプションと、ネットワーク・インターフェース ・オプションを該当する値にセットアップします。

    iSCSI ソフトウェア・イニシエーターの最大転送サイズは 256 KB です。 **tcp_sendspace** および **tcp_recvspace** のシステム最大値が 262144 バイトに設定されていると想定すると、ギガビット・イーサネットの構成に使用する **ifconfig** コマンドは、次のようになります。

    ```
    ifconfig en2 10.1.2.216 mtu 9000 tcp_sendspace 262144 tcp_recvspace 262144
    ```
  - **sb_max** ネットワーク・オプションを 524288 以上に設定します。 推奨値は 1048576 です。

- **mtu_size** を 9000 に設定します。
- 一部の iSCSI ターゲットでは、最高のパフォーマンスを得るために TCP Nagle アルゴリズムを使用不可にする必要があります。 Nagle アルゴリズムを使用不可にするには、**no** コマンドを使用して **tcp_nagle_limit** パラメーターを 0 に設定します。

調整パラメーターに関する詳細と、追加の調整パラメーターについては、TCP および UDP パフォーマンスの調整を参照してください。

iSCSI ソフトウェア・ターゲットの考慮事項
iSCSI ソフトウェア・ターゲットを定義し、論理装置番号 (LUN) をエクスポートする場合は、以下の事項を考慮してください。

- 各仮想ターゲットの iSCSI 修飾名 (IQN) は、ソフトウェア・ターゲットの定義時に SMIT で指定します。 SMIT パネルでは、名前のフォーマットの制限はありません。 ただし、iSCSI イニシエーターによっては、iSCSI プロトコルで定義されたフォーマットで IQN を指定する必要があります。名前のフォーマットが間違っていると、イニシエーターがターゲットにログインできず、ターゲットによってエクスポートされたディスクにアクセスできない場合があります。

  iSCSI ターゲット・デバイス現在の名前を表示するには、次の手順を実行します。

  1. 次のようなコマンドを実行します。 この例では、iSCSI ターゲット・デバイスが target0 であると想定しています。

     ```
     lsattr -E -l target0
     ```

  2. iscsi_name 属性を確認します。
- エクスポートされた LUN について戻される照会データの値は次のとおりです。
  - ベンダー ID: AIX
  - 製品 ID: iSCSI_VDASD
  - ANSI バージョン番号: 3
- AIX iSCSI ソフトウェア・イニシエーターまたは AIX iSCSI ソフトウェア・ターゲットをループバック・インターフェース (lo0) で使用しないでください。ループバック・インターフェースの 割り込みの処理は、物理または仮想のイーサネット・アダプター・ネットワーク・インターフェースの 割り込みの処理と異なります。ループバック・インターフェースが iSCSI ソフトウェア・ドライバーで使用された場合、AIX オペレーティング・システムは操作を停止する可能性があります。

ストリーム制御伝送プロトコル

ストリーム制御伝送プロトコル (SCTP) は、TCP に類似したコネクション指向のプロトコルですが、**UDP** と同様にメッセージ指向のデータ転送を提供します。 AIX オペレーティング・システムは RFC 4960 に準拠しています。

次の表は、SCTP と既存の伝送プロトコル TCP および UDP との動作の一般的な相違点を強調しています。

表 *83. TCP*、*UDP*、および *SCTP* の相違点

属性	**TCP**	**UDP**	**SCTP**
信頼性	信頼できる	信頼できない	信頼できる
接続管理	コネクション指向	コネクションレス	コネクション指向
伝送	バイト指向	メッセージ指向	メッセージ指向
フロー制御	はい	いいえ	はい
輻輳制御	はい	いいえ	はい
フォールト・トレランス	いいえ	いいえ	はい
データ配信	厳密な順序	順序付けなし	部分的な順序

表 *83. TCP*、*UDP*、および *SCTP* の相違点 *(*続き*)*			
属性	**TCP**	**UDP**	**SCTP**
セキュリティー	はい	はい	改良済み

一般的には、**SCTP** は信頼性が高く、かつメッセージ指向のデータ転送を必要とする特定のアプリケーション (**Voice over IP (VoIP)** など) に対して、より多くの柔軟性を提供する場合があります。 このカテゴリーのアプリケーションの場合は、**TCP** や **UDP** よりも **SCTP** の方が適しています。

- **TCP** は信頼性が高く、厳密な順序で伝送データ配信を提供します。 信頼性は必要でも、順序が関係ないか、部分的に順序が決まっていればよいデータ配信を容認できるアプリケーションの場合、**TCP** では優先ブロッキングのために不必要な遅延が生じることがあります。 単一接続内のマルチストリームの概念では、**SCTP** はデータを別のストリームから論理的に分離しながら、ストリーム内で厳密な配信を行うことができます。
- **SCTP** はバイト指向の **TCP** とは異なり、メッセージ指向です。 **TCP** のバイト指向の性質のため、アプリケーションは独自のレコード・マーキングを追加してメッセージ境界を保守することが必要です。
- **SCTP** は、マルチホーミング機能を使用して、ある程度のフォールト・トレランスを提供します。 ホストは、複数のネットワーク・インターフェースが 接続されている場合、同じネットワークでも別のネットワークでもマルチホームであると見なされます。 2 つのマルチホーム・ホスト間に **SCTP** アソシエーションを確立することができます。 この場合、両方のエンドポイントのすべての IP アドレスがアソシエーション起動時に交換されます。これにより、代替インターフェースを 介してピアに到達可能である限り、インターフェースの 1 つが何らかの理由でダウンしても、各エンドポイントが接続の存続期間中にこれらのアドレスのいずれかを使用できます。
- **SCTP** は、**TCP** および **UDP** が提供しない、付加的なセキュリティー機能を提供します。 **SCTP** では、アソシエーションのセットアップ中のリソース割り振りを、cookie 交換メカニズムを使用してクライアントの ID を検査できるまで遅らせるので、サービス妨害攻撃の可能性も低くなります。

SCTP アソシエーションの開始およびシャットダウン
ここでは、**SCTP** アソシエーションの開始とシャットダウンに関するガイドラインを示します。

SCTP アソシエーションは、次の順序で実行される、4 方向のハンドシェークから構成されます。

1. アソシエーションを開始するため、クライアントがサーバーに **INIT** シグナルを送信します。
2. **INIT** シグナルの受信時に、サーバーは **INIT-ACK** 応答をクライアントに送信します。 この **INIT-ACK** シグナルに状態 cookie が含まれます。 この状態 cookie には、メッセージ確認コード (MAC) と共に、cookie の作成に対応したタイム・スタンプ、状態 cookie の存続期間のスパン、およびアソシエーション確立に必要な情報が入っているはずです。 MAC は、あるサーバーのみに認知されている秘密鍵に基づいて、そのサーバーにより計算されます。
3. この **INIT-ACK** シグナルを受信すると、クライアントは、状態 cookie を単にエコーする **COOKIE-ECHO** 応答を送信します。
4. 秘密鍵を使って状態 cookie の認証性を検査した後、サーバーはアソシエーションにリソースを 割り振り、**COOKIE-ACK** 応答を送信して **COOKIE-ECHO** シグナルを認識し、アソシエーションを **ESTABLISHED** 状態にします。

さらに **SCTP** は、**SCTP** ユーザーからの要求に応じてアクティブ状態のアソシエーションを正常終了します。 イベントは、次の順序で発生します。

1. クライアントが **SHUTDOWN** シグナルをサーバーに送信し、これによってサーバーにクライアントが接続を閉じる準備ができたことを通知します。
2. サーバーは **SHUTDOWN-ACK** 確認通知を送信することによって応答します。
3. 次にクライアントが **SHUTDOWN-COMPLETE** シグナルをサーバーに送り返します。

また **SCTP** は、**SCTP** クライアントの要求に応じて、または **SCTP** スタックのエラーのために発生するアクティブ状態のアソシエーションの突然のクローズ (**ABORT** シグナル) もサポートします。 ただし、**SCTP** はハーフ・オープン接続はサポートしません。 プロトコルとその内部に関する詳細は、RFC 4960 に説明されています。

上記で指定される、**SCTP** と既存の伝送プロトコル間の相違点に加え、**SCTP** は次の機能も提供します。

- **ストリーム内の順序配信**: **SCTP** コンテキストのストリームは、エンドポイント間で転送されるユーザー・メッセージの順序を参照します。 **SCTP** アソシエーションは、複数のストリームをサポートできます。 アソシエーションのセットアップ時に、ユーザーはストリームの数を指定することができます。 ストリームの数の有効値は、ピアとのネゴシエーション後に固定されます。 各ストリーム内で、データ配信の順序は厳密に保守されています。 しかし、ストリーム間ではデータ配信は独立しています。 したがって、あるストリームからデータ損失があっても、データが別のストリームで配信されるのを妨げることはありません。 これにより、ユーザー・アプリケーションは論理的に独立しているデータに別のストリームを使用することができます。 また、データは特殊オプションを使用して順序なしの形式で配信することも可能です。 これは、緊急なデータを送信する場合に役立ちます。
- **ユーザー・データのフラグメント化**: **SCTP** はユーザー・メッセージをフラグメント化して、低位層に渡されるパケット・サイズがパス MTU を超えないようにすることができます。 受信時に、フラグメントは完全なメッセージに再度組み立てられ、ユーザーに渡されます。 フラグメント化もネットワーク・レベルで実行できますが、トランスポート層のフラグメント化には IP 層のフラグメント化と比較してさまざまな利点があります。 これらの利点の中には、フラグメントがネットワークで失われた場合にメッセージ全体を再送しなくてもよいのでルーターの負担を削減できるということがあります。トランスポート層のフラグメント化をしないと、IP フラグメント化を実行する必要が生じるかもしれません。
- **確認と輻輳制御**: パケット確認は、信頼できるデータ配信に必要です。 **SCTP** は指定時間内に送信するパケットの確認を取得しない場合、同じパケットの再送を起動します。 **SCTP** は、**TCP** が使用するものに類似した輻輳制御アルゴリズムに従います。 **TCP** と同様に累積確認通知を使用することに加え、**SCTP** は選択確認通知 (SACK) メカニズムを使用して、パケットを選択的に確認することができます。
- **チャンク・バンドル**: チャンクには、ユーザー・データまたは **SCTP** 制御情報が入っていることがあります。 複数のチャンクを、同じ **SCTP** ヘッダーの下で一緒にバンドルすることができます。 チャンク・バンドルには、送信側の末端の **SCTP** パケットに対するチャンクのアセンブリー、続いて受信側の末端のチャンクに対するパケットの逆アセンブリーが必要です。
- **パケット妥当性検査**: 各 **SCTP** パケットには、各エンドポイントによるアソシエーション起動の時間中に設定される、検査タグ・フィールドがあります。 すべてのパケットは、アソシエーションのライフタイムを通じて同じ検査タグ付きで送信されます。 アソシエーションのライフタイム 中に予期しない検査タグが付いたパケットを受信すると、そのパケットは廃棄されます。 また、ネットワークでのデータ破損に対する保護を強化するため、各 **SCTP** パケットの送信者により CRC-32 チェックサムを設定する必要もあります。 無効な CRC-32 チェックサムと共に受信されたパケットは廃棄されます。
- **パス管理**: アソシエーション・セットアップ時に、各エンドポイントは、持っているトランスポート・アドレスのリストを通知することがあります。 しかし、**SCTP** アソシエーションに定義される基本パスは 1 つだけで、通常のデータ転送に使用されます。 基本パスがダウンした場合、他のトランスポート・アドレスが使用されます。 アソシエーションのライフタイム 中、すべてのパスにおいて、定期的なインターバルでハートビートが送信され、パスの状況をモニターします。

SCTP ソケット API

SCTP ソケット API には、整合性、アクセス可能性、および互換性の特徴があります。

SCTP ソケット API は、次の機能を提供するために設計されました。

- 既存のソケット API との整合性を維持する
- 新しい **SCTP** 機能にアクセスのベースを提供する
- 既存の **TCP** および **UDP** アプリケーションのほとんどが、わずかな変更のみで **SCTP** に移行できるようにするための互換性を提供する

既存の **TCP** および **UDP** アプリケーションを容易に移行するため、**SCTP** API には 2 つの異なるスタイルが用意されています。

- **UDP** スタイルの API – セマンティクスが **UDP** のようなコネクションレスのプロトコル用に定義されたものに類似している。
- **TCP** スタイルの API – セマンティクスが **TCP** のようなコネクション指向のプロトコル用に定義されたものに類似している。

SCTP ではソケット API の **TCP** および **UDP** スタイルの両方を定義かつ使用できますが、AIX 5.3 では、**UDP** スタイルのソケット構文だけがサポートされます。これは、**UDP** スタイルの API の方が **SCTP** の新機能にアクセスする上で柔軟性がより高いためです。 **UDP** スタイルの API を使用すると、標準的なサーバーはアソシエーションのライフタイム 中に次の呼び出しのシーケンスを使用します。

1. socket()
2. bind()
3. listen()
4. recvmsg()
5. sendmsg()
6. close()

標準的なクライアントは次のソケット API 呼び出しのシーケンスを使用します。

1. socket()
2. sendmsg()
3. recvmsg()
4. close()

上記の呼び出しシーケンスを使用して作成されるアソシエーションは、明示的に作成されたアソシエーションと呼ばれます。 アソシエーションは、単に sendmsg()、 recvmsg() または sendto() および recvto() を呼び出すことで、ソケットの作成後に暗黙的に作成することができます。 暗黙アソシエーションの場合、bind() および listen() 呼び出しは必要ありません。 これらのすべてのシステム呼び出しの構文は、**UDP** ソケットで使用されるものと類似しています。 ソケット・サブルーチンの場合、「**Type (タイプ)**」フィールドは SOCK_SEQPACKET に設定されるべきであり、「Protocol (プロトコル)」フィールドは IPPROTO_SCTP でなければなりません。 これらの標準ソケット API に加え、**SCTP** は 2 つの新しい API、sctp_peeloff() および sctp_opt_info() を提供します。 **SCTP** 用のソケット API の詳細は、SCTP Socket API Draft にあります。 **SCTP** は、AIX 5.3 ではカーネル・エクステンションとしてインプリメントされています。 ユーザーは **sctpctrl** コマンドを使用して SCTP カーネル・エクステンションをロードしたりアンロードしたりできます。

さらに、このコマンドは、get や set など異なるオプションを使用して、**SCTP** カーネル・エクステンションの他の統計情報およびチューナブル・パラメーターを表示したり変更したりする際に使用できます。

sctp_bindx サブルーチン

ソケットのバインド・アドレスを追加または削除します。

ライブラリー

```
/usr/lib/libsctp.a
```

構文

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/sctp.h>

int sctp_bindx(int sd, struct sockaddr * addrs, int addrcnt, int flags);
```

説明

sctp_bindx サブルーチンは、ソケット **sd** との間で **addrs** 配列に入れて受け渡されるバインド・アドレスのセットを追加または削除します。 **addrcnt** パラメーターは、配列内のアドレスの数を示し、**flags** パラメーターは、アドレスを追加する必要があるのか、削除する必要があるのかを指定します。

ソケット **sd** が IPv4 ソケットの場合、受け渡されるアドレスは IPv4 アドレスでなければなりません。 ソケット **sd** が IPv6 ソケットの場合、受け渡されるアドレスは IPv4 アドレスでも IPv6 アドレスでも構いません。

addrs パラメーターは、1 つ以上のソケット・アドレスの配列へのポインターです。 各アドレスは、適切な構造体、つまり **struct sockaddr_in** または **struct sockaddr_in6** に入っています。 アドレス・タイプの

ファミリーを使用して、アドレスの長さを区別する必要があります。 呼び出し元は、**addrcnt** と同様に、配列内にアドレスの数を指定する必要があります。

flags パラメーターは、**SCTP_BINDX_ADD_ADDR** または **SCTP_BINDX_REM_ADDR** に設定できます。 アプリケーションは、**bind** コマンドを呼び出した後、SCTP_BINDX_ADD_ADDR を使用して、追加のアドレスをエンドポイントに関連付けることができます。 **SCTP_BINDX_REM_ADDR** パラメーターは、指定されたアドレスをアソシエーションから削除することを SCTP に指示します。 呼び出し元は、1 つのアソシエーションからすべてのアドレスを削除できない場合があります。 コマンドは失敗し、**EINVAL** エラー・コードが出されます。

戻り値

正常終了時には、**sctp_bindx()** コマンドは 0 を返します。失敗時には、**sctp_bindx()** コマンドは -1 を返し、**errno** パラメーターを該当するエラー・コードに設定します。

エラー・コード

エラー	説明
EINVAL	**EINVAL** エラー・コードは、ポートまたはアドレスが無効であるか、コマンドが 1 つのアソシエーションからすべてのアドレスを削除しようとしていることを示します。
EOPNOTSUPP	**EOPNOTSUPP** エラー・コードは、このコマンドが、接続済みアソシエーションにアドレスを追加または削除しようとしていることを示しています。

sctp_getladdrs and sctp_freeladdrs サブルーチン

ソケット上でローカルにバインドされたすべてのアドレスを返します。

ライブラリー

```
/usr/lib/libsctp.a
```

構文

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/sctp.h>

int sctp_getladdrs(int sd, sctp_assoc_t assoc_id, struct sockaddr **addrs);
void sctp_freeladdrs(struct sockaddr *addrs);
```

説明

sctp_getladdrs サブルーチンは、ソケット上でローカルにバインドされたすべてのアドレスを返します。戻り時に **addrs** パラメーターは、各ローカル・アドレスに対して動的に割り当てられた、適切なタイプの **sockaddr** 構造体のパック配列を指しています。 **sctp_freeladdrs** パラメーターを使用して、メモリーを解放する必要があります。

注 : 入力または出力パラメーター **addrs** は、NULL であってはなりません。

sd パラメーターが IPv4 ソケットの場合、返されるアドレスはすべて IPv4 アドレスです。 **sd** パラメーターが IPv6 ソケットの場合、返されるアドレスは、IPv4 アドレスと IPv6 アドレスの混合であることがあります。

1 対多スタイルのソケットの場合、**id** フィールドは、照会するアソシエーションを指定します。 1 対 1 スタイルのソケットの場合、**id** フィールドは無視されます。 **id** フィールドが 0 に設定されている場合、特定のアソシエーションに関係なく、ローカルでバインドされたアドレスが返されます。

sctp_freeladdrs サブルーチンは、**sctp_getladdrs** サブルーチンによって割り当てられたすべてのリソースを解放します。

戻り値

成功時には、**sctp_getladdrs** サブルーチンはソケットにバインドされたローカル・アドレスの数を返します。 ソケットがアンバインドされている場合は 0 が返され、***addrs** フィールドの値は未定義です。 エラー時には、**sctp_getladdrs** サブルーチンは -1 を返し、***addrs** フィールドの値は未定義です。

sctp_getpaddrs and sctp_freepaddrs サブルーチン

1 つのアソシエーション内のすべてのピア・アドレスを返します。

ライブラリー

```
/usr/lib/libsctp.a
```

構文

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/sctp.h>

int sctp_getpaddrs(int sd, sctp_assoc_t assoc_id, struct sockaddr **addrs);
void sctp_freepaddrs(struct sockaddr *addrs);
```

説明

sctp_getpaddrs サブルーチンは、1 つのアソシエーション内のすべてのピア・アドレスを返します。 戻り時に **addrs** パラメーターは、各アドレスに対して動的に割り当てられた、適切なタイプの **sockaddr** 構造体のパック配列を指しています。 **sctp_freepaddrs** サブルーチンを使用して、メモリーを解放する必要があります。

注 : 入力または出力パラメーター **addrs** は、NULL であってはなりません。

sd パラメーターが IPv4 ソケットの場合、返されるアドレスはすべて IPv4 アドレスです。 **sd** パラメーターが IPv6 ソケットの場合、返されるアドレスは、IPv4 アドレスと IPv6 アドレスの混合であることがあります。 1 対多スタイルのソケットの場合、**id** フィールドは、照会するアソシエーションを指定します。 1 対 1 スタイルのソケットの場合、**id** フィールドは無視されます。

sctp_freepaddrs サブルーチンは、**sctp_getpaddrs** サブルーチンによって割り当てられたすべてのリソースを解放します。

戻り値

成功時には、**sctp_getpaddrs** サブルーチンはアソシエーション内のピア・アドレスの数を返します。 このソケット上にアソシエーションが存在しない場合は、0 が返され、***addrs** フィールドの値は未定義です。 エラー時には、**sctp_getpaddrs** サブルーチンは -1 を返し、***addrs** フィールドの値は未定義です。

パス MTU ディスカバリー

複数のネットワークのパスを介して通信する 2 つのホストの場合、送信されるパケットのサイズがパス内のいずれかのネットワークの最小 MTU より大きければ、そのパケットはフラグメント化されます。 パケットのフラグメント化はネットワークのパフォーマンスを低下させる可能性があるので、フラグメント化を回避するために、送信するパケットのサイズがなるべくネットワーク・パス内の最小 MTU より大きくならないようにしてください。 この最小 MTU サイズは、*パス MTU* と呼ばれます。

このオペレーティング・システムでは、RFC 1191 に記述されているパス MTU ディスカバリー・アルゴリズムがサポートされています。 パス MTU ディスカバリーを **TCP** および **UDP** アプリケーションに対して使用可能にするには、**no** コマンドの **tcp_pmtu_discover** および **udp_pmtu_discover** オプションを変更します。 **TCP** に対して使用可能にした場合、パス MTU ディスカバリーは、**TCP** アプリケーションによって送信されるすべてのパケットのサイズがパス MTU を超えないように自動的に強制します。 **UDP** アプリケーションはそれ自体が送信するパケットのサイズを決定するため、**udp_pmtu_discover no** オプションが使用可能になっていても、**IP_FINDPMTU** ソケット・オプションを使用して、パス MTU 情報を使用するように **UDP** アプリケーションを明確に指定しなければなりません。 デフォルトでは、**tcp_pmtu_discover** と **udp_pmtu_discover** は使用可能になります。

宛先に対してパス MTU ディスカバリーが試みられる場合、pmtu エントリーがパス MTU (PMTU) テーブルで作成されます。 このテーブルは、**pmtu** 表示コマンドを使用して表示できます。 pmtu エントリーが累積されるのを防ぐには、未使用の pmtu エントリーに有効期限を設定し、それを過ぎたら削除されるようにします。 PMTU エントリーの有効期限は、**no** コマンドの pmtu_expire オプションによって制御されます。 pmtu_expire は、デフォルトでは 10 分に指定されます。

経路は動的に変化する可能性があるので、あるパスのパス MTU 値も時間の経過とともに変化する場合があります。 パス MTU 値が減少するとパケットのフラグメント化が起きるため、発見されたパス MTU 値は減少の有無について定期的に検査されます。 デフォルトでは、10 分ごとに減少の有無が検査されますが、この値は **pmtu_default_age** (**no** コマンドのオプション) の値を修正することで変更できます。

UDP アプリケーションは常に **IP_DONTFRAG** ソケット・オプションを設定し、PMTU における減少を検出する必要があります。 これは、pmtu_default_age 分ごとに減少を検査するのではなく、パス MTU における減少の直接検出を可能にします。

パス MTU 値の増大はネットワーク・パフォーマンスの潜在的な向上をもたらす可能性があるため、発見されたパス MTU 値は、増大の有無について定期的に検査されます。 デフォルトでは、30 分ごとに増大の有無が検査されますが、この値は **pmtu_rediscover_interval** (**no** コマンドのオプション) の値を修正することで変更できます。

そのネットワーク・パス内の一部のルーターが RFC 1191 をサポートしていない場合は、正確なパス MTU 値を判別できない場合があります。 そのような場合には、**mmtu** コマンドを使用して、試行用のパス MTU 値を追加したり削除したりできます。

注 :

1. パス MTU ディスカバリーは、グループ経路指定用に構成された重複経路も含め、重複経路では使用できません (415 ページの『経路の使用制限』を参照してください)。 パス MTU ディスカバリーを重複経路上で使用できます。
2. パス MTU ディスカバリーを使用可能にすると、**no** コマンドの **arpqsize** オプションの値が最小値の 5 に設定されます。 この値は、それ以後にパス MTU ディスカバリーが使用不可にされた場合も減少しません。

TCP/IP のサービス品質

サービス品質 (QOS) とは、特定のタイプの IP トラフィックを優先的に処理する方法を提供する、進化し続けているインターネット標準のファミリーのことです。

経路上で QoS が適切にサポートされるため、ネットワークのパフォーマンスを低下させる要因となっているさまざまなキュー遅延および輻輳 (ふくそう) の影響を抑えることができます。 このオペレーティング・システムは、アウトバウンド・トラフィックをそれぞれ別個のサービス・クラスに分類し、クライアント・アプリケーションからの要求に応じてリソースの予約を告知および確立するために QoS に対してホスト・サポートを提供します。

QoS は、公共性のある機関がネットワークの帯域幅の使用を管理しながらネットワーク・ポリシーを展開および実施するために使用することができます。 以下に、QoS を使用する場合に、ホストでできることを示します。

- 特定のタイプのトラフィックのネットワークへの流入量の規制。
- 後続の各ルーターが指示されたサービスを送達することができるようにするための、選択されたパケットに対するなんらかのポリシーに従ったマーク付け。
- 経路上での適切な QoS サポートを伴う、仮想専用回線サービスなどの各種のサービスのサポート。
- 受信側からのリソース予約要求への関与およびリソース予約要求に使用可能な送信側セッションの告知。

QoS サポートでは、以下の機能が提供されます。

- RFC 2474 で定義されている Differentiated Services
- トラフィックのポリシング
- プロファイル内パケットおよびプロファイル外パケットのマーク付け
- トラフィック・シェーピング

- 課金
- RFC 1633 で定義されている、クライアント・アプリケーションおよびサーバー・アプリケーションに対する Integrated Services
- RSVP のシグナル方式 (RFC 2205)
- 保証付きサービス (RFC 2212)
- 管理されたロード・サービス (RFC 2211)
- ポリシーに基づくネットワーキング
- アプリケーション用の RAPI 共用ライブラリー

QoS サブシステムは、以下の 4 つのコンポーネントから成っています。

QoS カーネル・エクステンション機能 (/usr/lib/drivers/qos)/usr/lib/drivers/qos)
QoS カーネル・エクステンション機能は、/usr/lib/drivers/qos に常駐しており、構成メソッド **cfgqos** および **ucfgqos** を使用してロードおよびアンロードされます。 このカーネル・エクステンション機能により、QoS サポートが使用可能となります。

ポリシー・エージェント (/usr/sbin/policyd)
ポリシー・エージェントは、/usr/sbin/policyd に常駐するユーザー・レベルのデーモンです。 このデーモンは、ポリシー管理に対するサポートと、ポリシー規則をインストール、変更、および削除するための QoS カーネル・エクステンション機能とのインターフェースに対するサポートを提供します。 ポリシー規則は、ローカル構成ファイル (/etc/policyd.conf) の中で定義するか、LDAP を使用して中央ネットワーク・ポリシー・サーバーから検索することもできます。また、この両方を併用しても構いません。

RSVP エージェント (/usr/sbin/rsvpd)
RSVP エージェントは、/usr/sbin/rsvpd に常駐するユーザー・レベルのデーモンです。 このデーモンは、RSVP のシグナル方式プロトコルのセマンティクスをインプリメントしています。

RAPI 共用ライブラリー (/usr/lib/librapi.a)
アプリケーションは、RSVP API (RAPI) を使用して、Integrated Services インターネット QoS モデルによって定義されている拡張サービス品質を要求することができます。 このライブラリーは、RSVP プロトコルを使用してローカル RSVP エージェントと対話し、QoS 要求をデータ・フローの経路に沿って伝搬させます。 この API はオープン標準の 1 つです。

注 : この QoS のインプリメンテーションは、進化し続けているインターネット標準および Internet Engineering Task Force (IETF) と IETF のさまざまな作業グループによって現在作成中のドラフト標準のセットに基づいています。 IETF 内部でこれらの標準化作業が進められるにつれて、このテクノロジーはより整合性のとれたものとなり、その定義もより詳細になっていくことでしょう。 また、QoS がインターネット内で展開され始めたばかりの新生のインターネット・テクノロジーであることに注目することも重要です。 QoS の利点は、その展開のすべてのステージに数多く存在します。 しかし、本当の終端間サービスは、QoS サポートが特定の経路上のいたるところに存在してはじめて実現します。

QoS モデル
インターネットのための QoS モデルは、IETF によって定義されたオープン標準です。

現在 IETF 内において、*Integrated Services* と *Differentiated Services* という 2 つのインターネット QoS モデルの標準化が行われています。 これらの 2 つのインターネット QoS モデルは、RFC 1812 に記述されている従来のベストエフォート型のサービス・モデルを増補するものです。

Integrated services
Integrated Services (IS) は、RFC 1633 に記述されているインターネット用の動的リソース予約モデルです。

各ホストは、リソース予約プロトコル (RSVP: Resource ReSerVation Protocol) と呼ばれるシグナル方式プロトコルを使用して、ネットワークに対して特定のサービス品質を動的に要求します。 QoS パラメーターは、これらの RSVP メッセージに入れられて運ばれます。経路上の各ネットワーク・ノードでは、これらのパラメーターをインストールして、要求されたサービス品質を獲得します。 これらの QoS パラメーターは、保証付きサービスと制御されたロード・サービスという、現在定義中の 2 つのサービスのうちの 1 つを記述します。 IS の重要な特性の 1 つは、このシグナル方式がそれぞれのトラフィック・フローごとに使用され、予約が経路上の各ホップのすべてにインストールされることです。 このモデルはアプリケーショ

ンの動的に変化するニーズを満たすのに非常に適しているのですが、重大なスケーリング上の問題がいくつかあり、それらによって、各ルーターそれぞれが多くの流れを同時処理しているネットワークにはこのモデルを展開できないことが暗黙的に示されています。

Differentiated services

Differentiated Services (DS) では、フローごとおよびホップごとの拡張容易性を求める代わりにパケットを分類するための単純化された機構を採用することにより、上記の拡張容易性に関する問題を除去しています。

動的シグナル方式によるアプローチとは異なり、DS では IP サービス・タイプ (TOS) バイトの中のビットを使用してパケットをクラス分けします。 IP TOS バイト内の特定のビット・パターンは DS コード・ポイントと呼ばれ、当該の特定のホップにおいて送達されるサービス品質を定義するためにルーターによって使用されます。その方法は、ルーターがルーティング・テーブル検索を使用して IP 転送を行う方法とほとんど同じです。 特定の DS コード・ポイントを持つパケットに対する処理は、ホップごとの動作 (PHB: per-hop behavior) と呼ばれ、各ネットワーク・ノードごとにそれぞれ独立して管理されます。 これらの個別の独立した PHB の成果が連結されれば、結果的に終端間サービスが実現されます。

現在、Differentiated Services の標準化が IETF の 1 つの作業グループによって行われています。このグループにより、Expedited Forwarding PHB、Assured Forwarding (AF) PHB グループ、およびデフォルト (DE) PHB という 3 種類の PHB が定義されています。 EF PHB を使用すると、仮想専用回線 (VLL) のように、待ち時間、ジッター、および損失が少ない終端間サービスをインプリメントすることができます。 AF は、PHB グループと呼ばれる PHB のファミリーであり、パケットをさまざまなドロップ優先順位レベルに分類するのに使用されます。 パケットに割り当てられるドロップ優先順位により、そのパケットの AF クラス内における相対的な重要度が判別されます。 これを使用すると、銅、銀、および金という 3 つのクラスから構成される、いわゆるオリンピック・サービスをインプリメントすることができます。 DE PHB は、RFC 1812 で標準化されている従来のベストエフォート型のサービス・モデルです。

サポートされる標準およびドラフト標準

以下の RFC およびインターネット・ドラフトには、この QoS のインプリメンテーションが準拠している標準について記述されています。

項目	説明
RFC 2474	IPv4 ヘッダーおよび IPv6 ヘッダー内の「Differentiated Services」フィールド (DS フィールド) の定義
RFC 2475	Differentiated Services のアーキテクチャー
RFC 1633	インターネット・アーキテクチャーにおける Integrated Services : 概要
RFC 2205	リソース予約プロトコル (RSVP: Resource ReSerVation Protocol)
RFC 2210	IETF Integrated Services での RSVP の使用方法
RFC 2211	管理されたロード機能を提供するネットワーク・エレメントのサービス仕様
RFC 2212	保証付きサービス品質の仕様
RFC 2215	統合サービスを提供するネットワーク・エレメントの特長を表す汎用パラメーター

項目	説明
draft-ietf-diffserv-framework-01.txt, October 1998	DiffServ 用のフレームワーク
draft-ietf-diffserv-rsvp-01.txt, November 1998	DIFF-serv ネットワークで RSVP を使用するためのフレームワーク
draft-ietf-diffserv-phb-ef-01.txt	Expedited Forwarding PHB
draft-ietf-diffserv-af-04.txt	Assured Forwarding PHB グループ
draft-rajan-policy-qosschema-00.txt, October 1998	ネットワークにおける Differentiated Services および Integrated Services の図式

項目	説明
draft-ietf-rap-framework-01.txt, November 1998	ポリシーに基づくアドミッション制御のフレームワーク [25]
draft-ietf-rap-rsvp-ext-01.txt, November 1998	ポリシー制御用の RSVP 拡張機能

注 : QoS は新出のインターネット・テクノロジーです。 QoS の利点は、その展開のすべてのステージに数多く存在します。 しかし、本当の終端間サービスは、QoS サポートが特定の経路上のいたるところに存在してはじめて実現します。

QoS のインストール

QoS は `bos.net.tcp.server` に同梱されています。 QoS を使用するには、このファイルセットがインストールされている必要があります。

また、RAPI 共用ライブラリーを使用するには `bos.adt.include` もインストールされている必要があります。

QoS サブシステムの停止と始動

QoS は、SMIT を介して `smit qos` 高速パスあるいはコマンド **mkqos** および **rmqos** を使用して始動または停止することができます。

1. 現時点および次のシステム再始動時に QoS サブシステムを 使用不可にするには、次のように入力します。

   ```
   /usr/sbin/rmqos -B
   ```

2. 今だけ QoS サブシステムを 使用可能にするには、次のように入力してください。

   ```
   /usr/sbin/mkqos -N
   ```

コマンド・フラグの始動および除去については、**mkqos** および **rmqos** に関するコマンド説明を参照してください。

デーモン **policyd** および **rsvpd** は、それぞれ構成ファイル `/etc/policyd.conf` および `/etc/rsvpd.conf` を使用して構成します。 これらの構成ファイルを編集し、ローカル環境に合わせて QoS サブシステムをカスタマイズすることが必要 です。 提供される構成例をそのまま使用したのでは、QoS は正しく作動しません。

RSVP エージェントの構成

ホストが RSVP プロトコルをサポートするようにするには RSVP エージェントが必要です。

RSVP エージェントの構成は、`/etc/rsvpd.conf` 構成ファイルを使用して行います。 この構成ファイルの構文については、`/etc/rsvpd.conf` にインストールされているサンプル構成ファイルの中で説明しています。

次の例では、ホストが 4 つの IP アドレス (`1.2.3.1`、`1.2.3.2`、`1.2.3.3`、および `1.2.3.4`) で示される 4 つの (仮想または物理) インターフェースを 備えている場合の可能な RSVP 構成を示します。

```
interface 1.2.3.1
interface 1.2.3.2 disabled
interface 1.2.3.3 disabled
interface 1.2.3.4
{
  trafficControl
}

rsvp 1.2.3.1
{
  maxFlows 64
}

rsvp 1.2.3.4
{
  maxFlows 100
}
```

インターフェース 1.2.3.1 は RSVP に対して使用可能にされています。 しかし、トラフィック制御が指定されていないため、着信 RSVP RESV メッセージが受信されても TCP サブシステム内でのリソース予約は行われません。 このインターフェースは、最大 64 個までの RSVP セッションを同時にサポートすることができます。

インターフェース 1.2.3.2 および 1.2.3.3 は使用不可にされています。 RSVP エージェントは、このインターフェースを 使用して RSVP メッセージを送信または受信することはできません。

インターフェース 1.2.3.4 は RSVP に対して使用可能にされています。 それに加え、このインターフェースでは、RSVP RESV メッセージに応じてリソース予約を **TCP** サブシステムにインストールすることができます。 このインターフェースは 、最大 100 個までの RSVP セッションをサポートすることができます。

ホスト上に存在していても /etc/rsvpd.conf の中で明示的に記述されていない他のインターフェースは、すべて使用不可にされます。

ポリシー・エージェントの構成

ポリシー・エージェントは、QoS サブシステムに必須のコンポーネントです。

ポリシー・エージェントの構成は、/etc/policyd.conf 構成ファイルを使用して行います。 この構成ファイルの構文については、/etc/policyd.conf にインストールされているサンプル構成ファイルの中で説明しています。

ポリシー・エージェントは、/etc/policyd.conf を編集することにより構成できます。 また、次のコマンドがポリシーの構成に役立ちます。

- **qosadd**
- **qosmod**
- **qoslist**
- **qosremove**

以下の例では、premium サービス・カテゴリーを作成し、それを tcptraffic ポリシー規則の中で使用しています。 このサービス・カテゴリーでは、最大速度は 110000 Kbps、トークン・バケットの奥行きは 10000 ビット、発信 IP TOS 値は 2 進数で 11100000 となっています。 tcptraffic ポリシー規則により、1.2.3.6 で表される送信元 IP アドレス、宛先アドレス 1.2.3.3、および 0 から 1024 の範囲の宛先ポートが指定されているすべてのトラフィックに対してこの premium サービスが適用されます。

```
ServiceCategories  premium
{
   PolicyScope     DataTraffic
   MaxRate         110000
   MaxTokenBucket  10000
   OutgoingTOS     11100000
}

ServicePolicyRules   tcptraffic
{
   PolicyScope   DataTraffic
   ProtocolNumber 6  # tcp
   SourceAddressRange      1.2.3.6-1.2.3.6
   DestinationAddressRange 1.2.3.3-1.2.3.3
   DestinationPortRange    0-1024
   ServiceReference        premium
}
```

以下のステートメントでは、デフォルトのサービス・カテゴリーをセットアップし、それを使用して UDP トラフィックの流れをインターフェース 1.2.3.1 から 1.2.3.4 から IP アドレス 1.2.3.6 から 1.2.3.10、ポート 8000 までに制限しています。

```
ServiceCategories  default
{
   MaxRate         110000
   MaxTokenBucket  10000
   OutgoingTOS     00000000
}

ServicePolicyRules   udptraffic
{
```

```
        ProtocolNumber 17  # udp
        SourceAddressRange      1.2.3.1-1.2.3.4
        DestinationAddressRange 1.2.3.6-1.2.3.10
        DestinationPortRange    8000-8000
        ServiceReference        default
    }
```

以下の構成例を使用すると、サブツリーの識別名を使用して LDAP サーバーから規則をダウンロードし、LDAP サーバー・ホストでポリシーをルックアップできます。

```
    ReadFromDirectory
    {
      LDAP_Server      1.2.3.27
      Base             ou=NetworkPolicies,o=myhost.mydomain.com,c=us
    }
```

QoS のトラブルシューティング

qosstat コマンドを使用すると、QoS サブシステムにインストール 済みでアクティブになっているポリシーに関する状況情報を表示することができます。 この情報は、QoS 構成のトラブルシューティング 時に、どこに問題があるのかを判別するのに役立つ可能性があります。

以下に、**qosstat** を使用して生成することができるレポートを示します。

```
Action:
  Token bucket rate (B/sec): 10240
  Token bucket depth (B): 1024
  Peak rate (B/sec): 10240
  Min policied unit (B): 20
  Max packet size (B): 1452
  Type: IS-CL
  Flags: 0x00001001 (POLICE,SHAPE)

  Statistics:
    Compliant packets: 1423 (440538 bytes)

  Conditions:
    Source address        Dest address          Protocol
    192.168.127.39:8000   192.168.256.29:35049  tcp       (1 connection)

Action:
  Token bucket rate (B/sec): 10240
  Token bucket depth (B): 1024
  Peak rate (B/sec): 10240
  Outgoing TOS (compliant): 0xc0
  Outgoing TOS (non-compliant): 0x00
  Flags: 0x00001011 (POLICE,MARK)
  Type: DS

  Statistics:
    Compliant packets: 335172 (20721355 bytes)
    Non-compliant packets: 5629 (187719 bytes)

  Conditions:
    Source address        Dest address          Protocol
    192.168.127.39:80     *:*                   tcp       (1 connection)
    192.168.127.40:80     *:*                   tcp       (5 connections)
```

QoS ポリシーの仕様

ここでは、ポリシー・エージェントが発信トラフィックでサービス品質 (QoS) に関するポリシーを指定するときに使用するオブジェクト・クラスと属性について説明します。

オブジェクト・クラスと属性の定義の後、マーキング、ポリシングおよびシェーピングを使用可能にする場合のガイドラインについて説明します。

以下の規則は、以後の説明の中で使用されます。

```
      p  : choose one in the allowed parameter set
      B  : integer value of a byte (i.e., 0 =< B =< 255)
      b  : bit string starting with left most bit (e.g., 101 is
           equivalent 10100000 in a byte field)
      i  : integer value
      s  : a character string
      a  : IP address format B.B.B.B
```

```
    (R) : Required parameter
    (O) : Optional parameter
```

ReadFromDirectory ステートメント

このステートメントは、LDAP セッションを確立するためのパラメーターを指定します。

ReadFromDirectory ステートメントは、LDAP セッションの確立のために /etc/policyd.conf ファイルで使用します。

```
        ReadFromDirectory
        {
          LDAP_Server    a   # IP address of directory server running LDAP
          LDAP_Port      i   # Port number LDAP server is listening to
          Base     s         # Distinguished Name for LDAP usage
          LDAP_SelectedTag s # Tag to match SelectorTag in object classes
        }
```

この場合、

```
    LDAP_Server (R): IP address of LDAP server
    LDAP_Port   (0): Unique port number, default port is 389
    Base        (R): Example is o=ibm, c=us where o is your organization and c is country
    LDAP_SelectedTag (R): Unique string matching SelectorTag attribute in the object class
```

ServiceCategories ステートメント

このステートメントは、IP パケットのフロー (例えば、**TCP** 接続または **UDP** データからの) が、ネットワークを横断するときに終端間で受け取るサービスのタイプを指定します。

ServiceCategories は、後に参照できるように、異なる名前を指定し、繰り返し使用します。ServiceCategories オブジェクトを使用する場合は、ServicePolicyRules でポリシーを完全に定義する必要があります。

```
        ServiceCategories   s
        {
           SelectorTag    s     # Required tag for LDAP Search
           MaxRate        i     # Target rate for traffic in this service class
           MaxTokenBucket i     # The bucket depth
           OutgoingTOS    b     # TOS value of outbound traffic for this service class
           FlowServiceType   p  # Type of traffic

        }
```

この場合、

```
        s           (R)     : is the name of this service category
        SelectorTag (R)     : Required only for LDAP to Search object classes
        MaxRate     (O)     : in Kbps (K bits per second), default is 0
        MaxTokenBucket(O)   : in Kb, default is system defined maximum
        OutgoingTOS (O)     : default is 0
        FlowServiceType  (O): ControlledLoad | Guaranteed, default is ControlledLoad
```

ServicePolicyRules ステートメント

このステートメントは、対応するサービス・カテゴリーへの突き合わせに使用する IP パケットの特性を指定します。

つまり、このステートメントは、特定のサービスを受け取る IP データグラム・セットを定義します。ServicePolicyRules は、ServiceReference 属性によって ServiceCategories に関連付けられます。 2 つの規則が同じ ServiceCategory を参照する場合、それぞれの規則は、ServiceCategory の固有のインスタンスに関連付けられます。

```
        ServicePolicyRules   s
        {
           SelectorTag       s   # Required tag for LDAP Search
           ProtocolNumber    i   # Transport protocol id for the policy rule
           SourceAddressRange      a1-a2
           DestinationAddressRange a1-a2
           SourcePortRange         i1-i2
           DestinationPortRange    i1-i2
```

```
            PolicyRulePriority  i       # Highest value is enforced first
            ServiceReference        s  # Service category name which for this policy rule
         }
```

この場合、

```
         s           (R): is the name of this policy rule
         SelectorTag (R): required only for LDAP to Search object class
         ProtocolNumber     (R): default is 0 which causes no match, must explicity specify
         SourceAddressRange (O): from a1 to a2 where a2 >= a1, default is 0, any source address
         SourcePortRange    (O): from i1 to i2 where i2 >= i1, default is 0, any source port
         DestinationAddressRange (O): same as SourceAddressRange
         DestinationPortRange    (O): same as SourcePortRange
         PolicyRulePriority    (O): Important to specify when ovelapping policies exist
         ServiceReference   (R): service category this rule uses
```

DiffServ 環境に関するガイドライン

次に、DiffServ 環境でのマーキング、シェーピング、ポリシングについてのポリシーを指定するときのガイドラインを示します。

1. **マーキングのみ**

   ```
       OutgoingTOS     : Desired Type Of Service
       FlowServiceType : ControlledLoad
       MaxRate         : Take default of 0
   ```

2. **シェーピングのみ**

   ```
       OutgoingTOS     : Take default of 0
       FlowServiceType : Guaranteed
       MaxRate         : Target rate desired for traffic as a positive integer
   ```

3. **マーキングおよびポリシング (「注」を参照)**

   ```
       OutgoingTOS     : Desired Type of Service
       FlowServiceType : ControlledLoad
       MaxRate         : Target rate desired for traffic as a positive integer
   ```

4. **マーキングおよびシェーピング**

   ```
       OutgoingTOS     : Desired Type of Service
       FlowServiceType : Guaranteed
       MaxRate         : Target rate desired for traffic as a positive integer
   ```

注 : ポリシングの場合、プロファイル外パケット用に設定された Type of Service (ToS) はゼロに設定されます。

policyd 構成ファイルのサンプル

`/etc/policyd.conf` 構成ファイルの完全な例を次に示します。

```
#loglevel  511    # Verbose logging

###############################################################################
#
# Mark rsh traffic on TCP source ports 513 and 514.
ServiceCategories       tcp_513_514_svc
{
        MaxRate                 0               # Mark only
        OutgoingTOS             00011100        # binary
        FlowServiceType         ControlledLoad
}

ServicePolicyRules      tcp_513_514_flt
{
        ProtocolNumber          6  # TCP
        SourceAddressRange      0.0.0.0-0.0.0.0 # Any IP src addr
        DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
        SourcePortRange         513-514
        DestinationPortRange    0-0             # Any dst port
        ServiceReference        tcp_513_514_svc
}
#
```

```
###############################################################################
#
# Shape connected UDP traffic on source port 9000.
ServiceCategories       udp_9000_svc
{
        MaxRate                 8192     # kilobits
        MaxTokenBucket          64       # kilobits
        FlowServiceType         Guaranteed
}

ServicePolicyRules      udp_9000_flt
{
        ProtocolNumber          17  # UDP
        SourceAddressRange      0.0.0.0-0.0.0.0 # Any IP src addr
        DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
        SourcePortRange         9000-9000
        DestinationPortRange    0-0             # Any dst port
        ServiceReference        udp_9000_svc
}
#
###############################################################################
#
# Mark and police finger traffic on TCP source port 79.
ServiceCategories       tcp_79_svc
{
        MaxRate                 8         # kilobits
        MaxTokenBucket          32        # kilobits
        OutgoingTOS             00011100  # binary
        FlowServiceType         ControlledLoad
}

ServicePolicyRules      tcp_79_flt
{
        ProtocolNumber          6  # TCP
        SourceAddressRange      0.0.0.0-0.0.0.0 # Any IP src addr
        DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
        SourcePortRange         79-79
        DestinationPortRange    0-0             # Any dst port
        ServiceReference        tcp_79_svc
}
#
###############################################################################
#
# Mark and shape ftp-data traffic on TCP source port 20.
ServiceCategories       tcp_20_svc
{
        MaxRate                 81920        # kilobits
        MaxTokenBucket          128          # kilobits
        OutgoingTOS             00011101     # binary
        FlowServiceType         Guaranteed
}

ServicePolicyRules      tcp_20_flt
{
        ProtocolNumber          6  # TCP
        SourceAddressRange      0.0.0.0-0.0.0.0 # Any IP src addr
        DestinationAddressRange 0.0.0.0-0.0.0.0 # Any IP dst addr
        SourcePortRange         20-20
        DestinationPortRange    0-0             # Any dst port
        ServiceReference        tcp_20_svc
}
#
###############################################################################
#
# LDAP server entry.
#ReadFromDirectory
#{
#   LDAP_Server             9.3.33.138  # IP address of LDAP server
#   Base                    o=ibm,c=us  # Base distinguished name
#   LDAP_SelectedTag        myhost      # Typically client hostname
#}
#
###############################################################################
```

IBM SecureWay Directory サーバーのポリシー・ロード

IBM SecureWay Directory LDAP サーバーでポリシー・デーモンを使用する場合は、LDAP サーバーを始動する前に、/etc/ldapschema/V3.modifiedschema を更新するためのガイドとしてこのスキーマを使用してください。

詳しくは、210 ページの『LDAP ネーム・レゾリューションの計画と構成 (IBM SecureWay ディレクトリー・スキーマ)』を参照してください。

```
objectClasses {
( ServiceCategories-OID NAME 'ServiceCategories' SUP top MUST
( objectClass $ SelectorTag $ serviceName ) MAY
( description $ FlowServiceType $ MaxRate $ MaxTokenBucket $ OutgoingTos ) )
( ServicePolicyRules-OID NAME 'ServicePolicyRules' SUP top MUST
( objectClass $ PolicyName $ SelectorTag ) MAY
( description $ DestinationAddressRange $ DestinationPortRange $
ProtocolNumber $ ServiceReference $ SourceAddressRange $ SourcePortRange ) )
}
attributeTypes {
( DestinationAddressRange-OID NAME 'DestinationAddressRange' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( DestinationPortRange-OID NAME 'DestinationPortRange' SYNTAX
 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( FlowServiceType-OID NAME 'FlowServiceType'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( MaxRate-OID NAME 'MaxRate' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( MaxTokenBucket-OID NAME 'MaxTokenBucket' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( OutgoingTos-OID NAME 'OutgoingTos' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( PolicyName-OID NAME 'PolicyName' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( ProtocolNumber-OID NAME 'ProtocolNumber' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( SelectorTag-OID NAME 'SelectorTag' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( ServiceReference-OID NAME 'ServiceReference' SYNTAX
 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( SourceAddressRange-OID NAME 'SourceAddressRange' SYNTAX
 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( SourcePortRange-OID NAME 'SourcePortRange' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
}

IBMattributeTypes {
( DestinationAddressRange-OID DBNAME ( 'DestinationAddressRange' 'DestinationAddressRange' ) )
( DestinationPortRange-OID DBNAME ( 'DestinationPortRange' 'DestinationPortRange' ) )
( FlowServiceType-OID DBNAME ( 'FlowServiceType' 'FlowServiceType' ) )
( MaxRate-OID DBNAME ( 'MaxRate' 'MaxRate' ) )
( MaxTokenBucket-OID DBNAME ( 'MaxTokenBucket' 'MaxTokenBucket' ) )
( OutgoingTos-OID DBNAME ( 'OutgoingTos' 'OutgoingTos' ) )
( PolicyName-OID DBNAME ( 'PolicyName' 'PolicyName' ) )
( ProtocolNumber-OID DBNAME ( 'ProtocolNumber' 'ProtocolNumber' ) )
( SelectorTag-OID DBNAME ( 'SelectorTag' 'SelectorTag' ) )
( ServiceReference-OID DBNAME ( 'ServiceReference' 'ServiceReference' ) )
( SourceAddressRange-OID DBNAME ( 'SourceAddressRange' 'SourceAddressRange' ) )
( SourcePortRange-OID DBNAME ( 'SourcePortRange' 'SourcePortRange' ) )
}

ldapSyntaxes {
}

matchingRules {
}
```

QoS システムの構成

重複しているポリシーが QoS マネージャーにインストールされている場合、施行順序は決まっていません。重複ポリシーの場合、ポリシーの施行順序を決定するために ServicePolicyRules の PolicyRulePriority 属性を指定する必要があります。 PolicyRulePriority 属性のパラメーターは整数であり、重複ポリシーの場合は最も高い整数値を持つ規則から施行されます。

QoS には、接続されている **UDP** ソケットのみがサポートされます。

ポリシーと RSVP エージェントは相互に独立しています。 したがって、既存の RSVP 予約と対立しているか、あるいは既存の RSVP 予約の範囲に含まれているポリシーを指定しないように注意する必要があります。 このような対立が存在している場合、システムは最初のポリシーまたは予約を受け入れて、その他のポリシーまたは予約については違反のフラグを立てます。

システムの正しい運用のためには、MaxTokenBucket 属性は、少なくとも、システム内で構成されているすべてのインターフェースの 最大 MTU に設定する必要があります。

ポリシーの変更は、ポリシー・エージェントが自動的に、既存のポリシーを削除し、新規のポリシーをインストールする方法で行われます。 したがって、対応するトラフィックがデフォルトの (通常はベストエフォートの) サービスを受ける間、経過を示す一時ウィンドウが短時間表示される場合があります。

IntServ モデルと DiffServ モデルの IETF 規格準拠

このリリースは、インターネットに関する Differentiated Services (DiffServ) および Integrated Services (IntServ) の、変化し続ける Internet Engineering Task Force (IETF) 規格に準拠しています。

次の RFC では、IntServ モデルのさまざまなコンポーネントについて説明しています。

- IETF Integrated Services での RSVP の使用方法 (RFC 2210)
- 管理されたロード機能を提供するネットワーク・エレメントのサービス仕様 (RFC 2211)
- 保証付きサービス品質の仕様 (RFC 2212)

次の RFC では、DiffServ モデルのさまざまなコンポーネントについて説明しています。

- IPv4 ヘッダーおよび IPv6 ヘッダー内の「Differentiated Services」フィールド (DS フィールド) の定義 (RFC 2474)
- Differentiated Services のアーキテクチャー (RFC 2475)

次の RFC では、IP TOS オクテットの現在の使用法を示しています。

- Internet Protocol Suite 内の Type of Service (ToS) (RFC 1349)

次の RFC では、IP TOS オクテットの使用法に関する将来の実施要領を示しています。

- IPv4 ヘッダーおよび IPv6 ヘッダー内の「Differentiated Services」フィールド (DS フィールド) の定義 (RFC 2474)
- Assured Forwarding PHB グループ (RFC 2597)
- An Expedited Forwarding PHB (RFC 2598)

IPv6 サポート

QoS でサポートされているのは IPv4 だけです。IPv6 はサポートされていません。

ポリシー・デーモンの制御

ポリシー・デーモンは、システム・リソース・コントローラー (SRC) を使用して制御できます。

次に例を示します。

```
startsrc -s policyd -a "-i 60"
```

このコマンドは、60 秒のリフレッシュ・インターバルでポリシー・エージェントを始動します。

次のコマンドは、

```
stopsrc -s policyd
```

ポリシー・デーモンを停止します。

注 : ポリシー・デーモンを停止することが、カーネル内にインストールされているポリシーを除去することにはなりません。 再度ポリシー・デーモンを始動すると、古いポリシー (前にカーネルにインストールされていたも) が削除され、`/etc/policyd.conf` ファイルに定義されているポリシーが再インストールされます。

refresh SRC コマンドは、現在はサポートされていません。

QoS コマンドおよびメソッド

ここでは、**TCP/IP** サービス品質 (QoS) のコマンドとメソッドを示します。

本書の重要な更新内容については、`/usr/samples/tcpip/qos` 内の README ファイルをご覧ください。

以下の QoS コマンドがサポートされています。

- **qosadd**

- **qoslist**
- **qosmod**
- **qosremove**
- **qosstat**
- **mkqos**
- **rmqos**

以下の QoS メソッドがサポートされています。

- **cfgqos**
- **ucfgqos**

TCP/IP のトラブルシューティング

netstat コマンドは、**伝送制御プロトコル/インターネット・プロトコル (TCP/IP)** ネットワーク環境に共通する問題を診断する際に有効なツールです。

netstat コマンドは、ネットワーク内の問題が存在する領域を判別するのに役立ちます。 問題の領域を分離した後、さらに高度なツールを使用して作業を進めることができます。 例えば、**netstat -i** と **netstat -v** を使用して特定のハードウェア・インターフェースに 問題があるかどうかを判別し、その後、診断機能を実行してさらに問題を分析することができます。 あるいは、**netstat -s** コマンドでプロトコル・エラーの存在が示された場合は、**trpt** コマンドか **iptrace** コマンドを使用できます。

通信に関する問題

一般的な **TCP/IP** 通信問題には、ネットワーク上のホストと通信できない問題や経路指定の問題などがあります。 以下に解決策を示します。

ネットワーク上のホストと通信できない場合は、次のようにします。

- **ping** コマンドを使用して、ホストに接続してみます。 ローカル・ホスト上で **ping** コマンドを実行し、ネットワークへのローカル・インターフェースが 始動し、作動していることを検証します。
- **host** コマンドを使用して、ホスト名を解決してみます。 名前が解決されない場合は、ネーム・レゾリューションに関する問題が存在します。 詳しくは、474 ページの『ネーム・レゾリューションに関する問題』を参照してください。

名前が解決し、別のネットワーク上のホストへ接続しようとしている場合、経路指定の問題が存在する可能性があります。 詳しくは、476 ページの『TCP/IP の経路指定に関する問題』を参照してください。

- トークンリング・ネットワークを使用している場合は、ターゲット・ホストが別のリング上にあるかどうかを検査します。 別のリング上にある場合は、「**allcast**」フィールドが正しく設定されていない可能性があります。 「ネットワーク・インターフェース」メニューにアクセスする場合は、システム管理インターフェース・ツール (SMIT) 高速パスの `smit chinet` を使用してください。その後、トークンリング・ダイアログで「Confine Broadcast to Local Ring (ブロードキャストをローカル・リングに制限)」フィールドを **no** に設定します。
- ネットワーク上に大量の**アドレス解決プロトコル (ARP)** パケットが存在する場合は、サブネット・マスクが正しく設定されていることを検査します。 この状態はブロードキャスト・ストームとして知られており、システム・パフォーマンスに影響を及ぼす可能性があります。

ネーム・レゾリューションに関する問題

TCP/IP を実行しているホスト上の resolver ルーチンは、次の送信元をここに示す順に使用して名前を解決しようとします。

1. DOMAIN ネーム・サーバー (**named**)
2. ネットワーク情報サービス (NIS)
3. ローカル `/etc/hosts` ファイル

クライアント・ホストの解決に関する問題

ホスト名を解決できず、(/etc/hosts ファイルを使用する) フラット・ネーム・レゾリューションを使用している場合は、ホストの名前と正しいインターネット・プロトコル (IP) アドレス情報が /etc/hosts ファイルに入っていることを検証します。

ホスト名を解決できず、ネーム・サーバーを使用している場合は、以下の手順を実行します。

1. ネーム・サーバーのドメイン名と IP アドレスを指定した resolv.conf ファイルが存在するかどうかを検査します。
2. ローカル・ネーム・サーバーが始動しているかどうかを検査するために、ローカル resolv.conf ファイルに入っているネーム・サーバーの IP アドレスを指定して **ping** コマンドを実行します。
3. ローカル・ネーム・サーバーが始動している場合は、ローカル・ネーム・サーバー上で **named** デーモンがアクティブになっているかどうかを検査するために、ネーム・サーバー上で **lssrc -s named** コマンドを実行します。
4. **syslogd** を実行している場合は、ログに記録されているメッセージを調べます。
   このメッセージの出力は、/etc/syslog.conf ファイルの中で定義されています。

これらのステップを実行しても問題を識別できない場合は、ネーム・サーバー・ホストを調べてください。

ネーム・サーバー・ホストの解決に関する問題

ホストのネーム・サーバー問題を解決するには、以下の手順を使用します。

ホスト名を解決できない場合は、次の操作を行います。

1. 次のコマンドを入力して、**named** デーモンがアクティブであるかどうかを検査します。

   ```
   lssrc -s named
   ```

2. ターゲット・ホストのアドレスが存在するかどうか、また、そのアドレスがネーム・サーバーのデータベース内で正しく指定されていることを検証します。
   SIGINT シグナルを **named** デーモンに送信してデータベースをダンプし、/var/tmp/named_dump.db ファイルにキャッシュします。 解決しようとしているアドレスがそこにあるかどうか、また正しいかどうかを検査します。

   ドメインのマスター・ネーム・サーバー用の **named** ホスト・データ・ファイルに、「名前からアドレスへのレゾリューション情報」を追加するか訂正します。 その後、次の **SRC** コマンドを入力してデータ・ファイルを再読み取りします。

   ```
   refresh -s named
   ```

3. ネーム・レゾリューション要求が処理されているかどうかを検査します。
   そのためには、コマンド・ラインから **named** デーモンを入力し、デバッグ・レベルを指定します。 有効なデバッグ・レベルは 1 から 9 です。このレベルが高くなるほど、デバッグ機構によって記録される情報が多くなります。

   ```
   startsrc -s named -a "-d DebugLevel"
   ```

4. **named** データ・ファイルの中に構成に関する問題がないかどうかを検査します。
   詳しくは、188 ページの『ネーム・サーバー・レゾリューション』を参照してください。 また、ファイル参照 の『DOMAIN Data File Format』、『DOMAIN Reverse Data File Format』、『DOMAIN Cache File Format』、および『DOMAIN Local Data File Format』を参照してください。

   注 : 共通エラーは DOMAIN データ・ファイル内での . (ピリオド) および @ (アットマーク) の誤った使用です。

外部のユーザーがドメインに到達できない場合は、マスター・ネーム・サーバー以外のすべてのネーム・サーバー (スレーブ、ヒント) が、DOMAIN データ・ファイル内に同じ存続時間 (TTL) 情報を備えていることを確認します。

外部のリゾルバーがサーバーを常に照会している場合は、サーバーが配布している DOMAIN データ・ファイルに、合理的な TTL 値が指定されていることを確認します。 TTL がゼロかその他の小さな値になってい

ると、転送したデータはすぐに時間切れになってしまいます。 この問題を解決するため、開始権限 (SOA) レコード内の最小値を 1 週間またはそれ以上に設定してください。

TCP/IP の経路指定に関する問題
宛先ホストに到達できない場合は、以下の状況に対する解決策を検討してください。

- 「Network Unreachable」というエラー・メッセージが表示された場合は、ゲートウェイ・ホストまでの経路が正しく定義されていることを確認します。 これを検査するには、**netstat -r** コマンドを使用してカーネル・ルーティング・テーブルを表示します。
- 「No route to host」というエラー・メッセージが表示された場合は、ローカル・ネットワーク・インターフェースが 始動しているかどうかを検査するために、**ifconfig** interface_name コマンドを実行します。 この出力により、インターフェースが UP 状態になっているかどうかが示されます。 **ping** コマンドを使用して、ネットワーク上の別のホストへの到達を試行してください。
- 「Connection timed out」というエラー・メッセージが表示された場合は、次のようにします。
  - ローカル・ゲートウェイが始動しているかどうかを検査するために、ゲートウェイの名前または IP アドレスを指定して、**ping** コマンドを実行します。
  - ゲートウェイ・ホストまでの経路が正しく定義されていることを確認します。 これを検査するには、**netstat -r** コマンドを使用してカーネル・ルーティング・テーブルを表示します。
  - 通信したい相手側ホストが、こちら側のマシンに戻るためのルーティング・テーブル・エントリーを備えていることを確認します。
- 静的経路指定を使用する場合は、ターゲット・ホストとゲートウェイ・ホストへの経路が定義されていることを確認します。 これを検査するには、**netstat -r** コマンドを使用してカーネル・ルーティング・テーブルを表示します。

  注: 通信したい相手側ホストが、こちら側のマシンまでのルーティング・テーブル・エントリーを備えていることを確認してください。
- 動的経路指定を使用している場合は、**netstat -r** コマンドを入力して、カーネル・ルーティング・テーブルの中にゲートウェイが正しく指定されていることを検証します。
- ゲートウェイ・ホストが **Routing Information Protocol** (RIP) と **routed** デーモンを使用している場合は、ターゲット・ホストまでの静的経路が /etc/gateways ファイルにセットアップされていることを確認します。

  注: この操作を行う必要があるのは、経路指定デーモンが他のゲートウェイへの照会によって遠隔ホストまでの経路を識別できない場合だけです。
- ゲートウェイ・ホストが **RIP** と **gated** デーモンを使用している場合は、ターゲット・ホストまでの静的経路が gated.conf ファイルにセットアップされていることを確認します。
- 動的経路指定と **routed** デーモンを使用している場合は、次のようにします。
  - **routed** が照会によって経路を識別できない場合 (例えば、ターゲット・ホストが **RIP** を実行していない場合) は、/etc/gateways ファイルを調べて、ターゲット・ホストまでの経路が定義されていることを検査します。
  - ホストへのパケット転送を担当するゲートウェイが **RIP** を実行して いることを確認します。 そうでない場合は、静的経路を定義する必要があります。
  - デバッグ・オプションを使用して **routed** デーモンを実行し、正しくないパケットの受信などの情報を記録させます。 デーモンをコマンド・ラインから起動するには、次のコマンドを使用します。

    ```
    startsrc -s routed -a "-d"
    ```

  - **routed** デーモンを **-t** フラグ付きで実行します。これにより、送受信されたすべてのパケットが標準出力に書き出されます。 **routed** を、このモードで実行した場合には、これを始動した端末の制御を受けたままになります。 したがって、制御する端末からの割り込みがあると、このデーモンが強制終了されます。
- 動的経路指定と **gated** デーモンを使用している場合は、次のようにします。

- /etc/gated.conf ファイルが正しく構成されているかどうかと、正しいプロトコルが実行されていることを検証します。
- 送信元ネットワーク上のゲートウェイが、宛先ネットワーク上のゲートウェイと同じプロトコルを使用していることを確認します。
- 通信先にしようとしている相手側マシンが、こちら側のマシンまでの経路を備えていることを確認します。
- gated.conf ファイル内のゲートウェイ名が、/etc/networks ファイルに指定されているゲートウェイ名に対応していることを検証します。

- **RIP** プロトコルまたは **HELLO** プロトコルを使用していて、宛先までの経路を経路照会を介して識別できない場合は、gated.conf ファイルを調べて、ターゲット・ホストまでの経路が定義されていることを検査します。 次の条件下では静的経路を設定してください。
  - 宛先ホストが送信元ホストと同じプロトコルを実行していないため、経路指定情報の交換ができない場合。
  - 遠くのゲートウェイ (送信元ホストと異なる自律システム上にあるゲートウェイ) からホストへ到達しなければならない場合。 **RIP** は、同じ自律システム上にあるホスト間でだけ使用できます。

他の方法がすべて失敗してしまう場合は、経路指定デーモン (**routed** または **gated**) のトレースをオンにしても構いません。 コマンド・ラインから SRC の **traceson** コマンドを使用するか、デーモンにシグナルを送信して、さまざまなレベルのトレースを指定してください。 デーモンにシグナルを送信する方法の詳細については、**gated** デーモンまたは **routed** デーモンのセクションを参照してください。

SRC サポートに関する問題の解決

システム・リソース・コントローラーに共通する問題を解決するには、以下の推奨事項を参照してください。

- /etc/inetd.conf ファイルに加えた変更が有効にならない場合:

  inetd デーモンを更新するために、**refresh -s inetd** コマンドまたは **kill -1** *InetdPID* コマンドを入力します。

- **startsrc -s** [*subsystem name*] で次のエラー・メッセージが戻される場合:

  ```
  0513-00  The System Resource Controller is not active.
  ```

  システム・リソース・コントローラー・サブシステムがアクティブになっていません 。 **srcmstr &** コマンドを入力して SRC を始動したあと、**startsrc** コマンドを再入力してください。

  SRC サポートなしでコマンド・ラインからデーモンの始動を試みる方法もあります。

- **refresh -s** [*subsystem name*] または **lssrc -ls** [*subsystem name*] で次のエラー・メッセージが戻される場合:

  ```
  [subsystem name] does not support this option.
  ```

  サブシステムが、実行された SRC オプションをサポートしていません。 サブシステムの 資料を調べて、このサブシステムがサポートするオプションを 確認してください。

- 次のメッセージが表示される場合:

  ```
  SRC was not found, continuing without SRC support.
  ```

  デーモンが、**startsrc** コマンドを使用せずにコマンド・ラインから直接始動されました。 これは、問題ではありません。 ただし、**stopsrc** や **refresh** などの SRC コマンドは、直接起動したサブシステムを操作しません。

inetd デーモンを正しく起動し、実行しており、該当するサービスが正しいと思われるが、接続できない場合は、デバッガーを介して **inetd** デーモン・プロセスを実行してみてください。

1. 次のコマンドで **inetd** デーモンを一時的に停止します。

```
stopsrc -s inetd
```

stopsrc コマンドは、**inetd** デーモンのように、サブシステムを停止します。

2. syslog.conf ファイルの最後に、デバッグ行を追加します。 以下に例を示します。

```
vi /etc/syslog.conf
```

   a. ファイルの最後に「*.debug /tmp/myfile」という行を追加して、ファイルを閉じます。
   b. 指定したファイルが存在していなくてはなりません (この例の場合は、/tmp/myfile 内)。 **touch** コマンドを使用して、ファイルを作成できます。

3. 次の方法で、ファイルをリフレッシュします。
   - SRC を使用している場合は、次のように入力します。

```
refresh -s syslogd
```

   - SRC を使用していない場合は、次のコマンドで **syslogd** デーモンを kill します。

```
kill -1 `ps -e | grep /etc/syslogd | cut -c1-7`
```

4. 次のコマンドで、デバッグを使用可能にして **inetd** デーモンのバックアップを開始します。

```
startsrc -s inetd -a "-d"
```

-d フラグによってデバッグが可能になります。

5. /tmp/myfile デバッグ・ファイル内のログ・エラーへの接続を試行してください。 以下に例を示します。

```
tn bastet
Trying...
connected to bastet
login:>
Connection closed
```

6. デバッグ・ファイルに、問題が示されているかどうか調べます。 例えば、次のとおりです。

```
tail -f /tmp/myfile
```

telnet または rlogin に関する問題の解決

ここでは、**telnet** コマンドまたは **rlogin** コマンドに関する問題の解決に役立つ情報を記載します。

フルスクリーン・アプリケーションで画面のひずみによる問題がある場合は、次のようにします。

1. 次のどちらかのコマンドを入力して TERM 環境変数を検査します。

```
env
```

```
echo $TERM
```

2. TERM 変数が、使用している端末ディスプレイのタイプと 一致する値に設定されていることを検証します。

問題のデバッグに役立つことがある **telnet** サブコマンドは次のとおりです。

項目	説明
display	設定値とトグル値を表示します。
toggle	16 進によるすべてのネットワーク・データの表示を切り替えます。
toggle options	内部 **telnet** プロセス・オプションの表示を切り替えます。

inetd デーモンを使用して **telnet** サービスを実行できるが、**telnet** コマンドを使用して接続できない場合は、**telnet** インターフェースに 問題がある可能性があります。

1. **telnet** が正しい端末タイプを使用しているかどうか検査してください。
   a. 使用中のマシンで、次のコマンドを使って $TERM 変数をチェックします。

   ```
   echo $TERM
   ```

   b. 接続先となるマシンにログインし、次のコマンドで $TERM 変数をチェックします。

   ```
   echo $TERM
   ```

2. フラグを指定しないで **telnet** コマンドを入力し、**telnet** インターフェースのデバッグ 機能を使用します。

   ```
   telnet
   tn>
   ```

   a. open *host* コマンドを入力します。*host* にはマシン名を指定します。
   b. Ctrl-T キーを押して、tn%gt; プロンプトを表示します。
   c. tn> プロンプトに、デバッグ・モードを示す debug を入力します。
3. 次のように入力し、**telnet** インターフェースを 使用して接続先マシンへの接続を試行します。

   ```
   telnet bastet
   Trying...
   Connected to bastet
   Escape character is '^T'.
   ```

   画面をスクロールアップするさまざまなコマンドを確認します。 以下に例を示します。

   ```
   SENT do ECHO
   SENT do SUPPRESS GO AHEAD
   SENT will TERMINAL TYPE (reply)
   SENT do SUPPORT SAK
   SENT will SUPPORT SAK (reply)
   RCVD do TERMINAL TYPE (don't reply)
   RCVD will ECHO (don't reply)
   RCVD will SUPPRESS GO AHEAD (don't reply)
   RCVD wont SUPPORT SAK (reply)
   SENT dont SUPPORT SAK (reply)
   RCVD do SUPPORT SAK (don't reply)
   SENT suboption TELOPT_NAWS Width 80, Height 25
   RCVD suboption TELOPT_TTYPE SEND
   RCVD suboption TELOPT_TTYPE aixterm
   ...
   ```

4. /etc/termcap または /usr/lib/terminfo で aixterm 定義をチェックします。 以下に例を示します。

   ```
   ls -a /usr/lib/terminfo
   ```

5. aixterm 定義が存在していない場合は、ibm.ti ファイルを作成してこの定義を追加します。 以下に例を示します。

   ```
   tic ibm.ti
   ```

   tic コマンドは、端末情報コンパイラーです。

拡張 curses を使用するプログラムと一緒に **rlogin** および **telnet** コマンドを使用すると、ファンクション・キーと矢印キーに問題が起こる可能性があります。 ファンクション・キーと矢印キーはエスケープ・シーケンスを生成し、それらのエスケープ・シーケンスは、そのキー・シーケンス全体に割り当てられた時間が短すぎると分割されます。 curses は特定の時間だけ待って、Esc がエスケープ・キーだけを示しているのか、それともカーソル・キー、アクション・キー、ファンクション・キーなど、その他のキーによって生成される複数バイト・エスケープ・シーケンスの始まりを示しているのかを判断します。

割り当てられた時間内に Esc に続くデータが存在しないか有効でない場合、curses はその Esc がエスケープ・キーであり、そのキー・シーケンスは分割されていると判断します。 **rlogin** コマンドまたは **telnet** コマンドによって発生する遅延は、ネットワークによって異なります。 接続するネットワークの速度に応じて、矢印キーとファンクション・キーが機能する場合と機能しない場合があります。 この問題は、ESCDELAY 環境変数を大きな値 (1000 から 1500) に設定すると解決します。

TCP/IP の構成に関する問題
ネットワーク・インターフェースは、アダプター・カードをインストールして最初にシステムを始動したときに自動的に構成されます。 しかし、そのほかに、ホスト名、インターネット・アドレス、サブネット・マスクなど、**TCP/IP** の初期値をいくつか設定する必要があります。

これを行うには、次の方法で SMIT インターフェースを 使用します。

- `smit mktcpip` 高速パスを使用して、ホスト名、IP アドレス、およびサブネット・マスクの初期値を設定します。
- `smit mktcpip` 高速パスを使用して、ネーム・レゾリューション・サービスを提供するネーム・サーバーを指定します。 (`smit mktcpip` は、1 つのネットワーク・インターフェースのみを 構成するということに注意してください。)
- `smit chinet` 高速パスを使用して、その他のネットワーク属性を設定します。

また、ローカル・ゲートウェイへの経路など、ホストが送信情報を送信するために必要な静的経路をセットアップすることもできます。 これらの経路を構成データベースに永続的にセットアップするには、SMIT 高速パスの `smit mkroute` を使用します。

そのほかに構成に関する問題がある場合、詳細については 108 ページの『TCP/IP の構成』 を参照してください。

ネットワーク・インターフェースに 共通する TCP/IP 問題
ネットワーク・インターフェースは、アダプター・カードをインストールして最初にシステムを始動したときに自動的に構成されます。 しかし、**TCP/IP** を始動するために設定しなければならない特定の値があります。 これらの値にはホスト名とインターネット・アドレスが含まれ、SMIT 高速パスの `smit mktcpip` を使用して設定できます。

SMIT メソッドを選択した場合、これらの値を構成データベース内に永続的に設定するには、`smit mktcpip` 高速パスを使用します。 稼働中のシステムで値を変更する場合は、`smit chinet` および `smit hostname` 高速パスを使用してください。 `smit mktcpip` 高速パスは、**TCP/IP** を最小限に構成します。 アダプターを 追加する場合、「Further Configuration (追加構成)」メニューを使用すれば、`smit tcpip` 高速パスで実行できます。

これらの値が正確であることを確認済みであるにもかかわらず、情報の送受信に問題が生じる場合は、次の検査を行ってください。

- ネットワーク・アダプターにネットワーク・インターフェースがあるかどうかを 検査するために、**netstat -i** コマンドを実行します。 出力の Name 列に **tr0** などのインターフェースが 表示されます。 表示されない場合は、SMIT 高速パスの `smit mkinet` を入力して、ネットワーク・インターフェースを作成します。
- インターフェースの IP アドレスが正しいかどうかを検査するために、**netstat -i** コマンドを実行します。 出力の Network 列に IP アドレスが表示されます。 その IP アドレスが正しくない場合は、SMIT 高速パスの `smit chinet` を入力して、IP アドレスを設定します。
- **arp** コマンドを使用して、ターゲット・マシンの完全な IP アドレスを保持していることを確認します。 以下に例を示します。

  ```
  arp -a
  ```

  arp コマンドは、物理アダプター・アドレスを探します。 このコマンドを実行した場合、不完全なアドレスが戻される場合があります。 例えば、次のとおりです。

  ```
  ? (192.100.61.210) at (incomplete)
  ```

原因として、マシンのプラグがはずれている、特定のアドレスにマシンが存在しない、あるいはハードウェアに問題がある (例えば、マシンが、接続してパケットを受信できるが、パケットを返送できない) などが考えられます。

- アダプター・カードのエラーを探してください。 以下に例を示します。

```
netstat -v
```

  netstat -v コマンドを実行すると、イーサネット、トークンリング、X.25 および 802.3 アダプター・デバイス・ドライバーに関する統計が表示されます。 また、このコマンドにより、インターフェース上でアクティブなすべてのデバイス・ドライバーに関するネットワークおよびエラー・ログのデータ (「No Mbufs Errors」、「No Mbuf Extension Errors」および「Packets Transmitted and Adapter Errors Detected」) も表示されます。

- **errpt** コマンドを実行してエラー・ログを検査し、アダプターに問題がないことを確認します。
- 診断機能を実行し、アダプター・カードに異常がないかどうか検査します。 `smit diag` 高速パス、または **diag** コマンドを使用してください。

SLIP ネットワーク・インターフェースに関する *TCP/IP* 問題

一般に、**シリアル回線インターフェース・プロトコル (SLIP)** インターフェースに 関する問題のデバッグに最も効果的な方法は、構成をトレースし直して各手順を検査することです。

しかし、次のようにすることもできます。

- **slattach** プロセスが実行されていて、正しい tty ポートが使用されていることを検証するために、**ps -ef** コマンドを入力します。 実行されていない場合は、**slattach** コマンドを実行します。 (使用すべき正確な構文については、686 ページの『モデムによる SLIP の構成』または 688 ページの『null モデム・ケーブルによる SLIP の構成』を参照してください。)
- PPP アドレスが正しく指定されているかどうかを検査するために、`smit chinet` 高速パスを入力します。

  SLIP インターフェースを 選択します。 「**INTERNET ADDRESS (インターネット・アドレス)**」フィールドと「**DESTINATION ADDRESS (宛先アドレス)**」フィールドが正しいことを確認してください。

モデムが正しく機能しない場合は、次のようにします。

- モデムが正しくインストールされていることを確認します。 モデムのインストール・マニュアルを参照してください。
- モデムによるフロー制御がオフになっていることを検証します。

tty が正しく機能していない場合は、`smit tty` 高速パスを入力して、tty のボー・レートとモデムの特性が構成データベースに正しく設定されていることを検証してください。

イーサネット・ネットワーク・インターフェースに関する *TCP/IP* 問題

イーサネット・ネットワーク・インターフェースで **TCP/IP** 問題が発生する場合は、以下のチェックリストを参照してください。

ネットワーク・インターフェースが 初期化され、アドレスが正しく指定されていて、アダプター・カードに異常がないことが検査済みである場合は、次のようにします。

- インボード/アウトボード・トランシーバーに T コネクターのプラグを直接差し込んで使用していることを検証します。
- イーサネット・ケーブルを使用していることを確認します。 (イーサネット・ケーブルは 50 オームです。)
- イーサネット・ターミネーターを使用していることを確認します。 (イーサネット・ターミネーターは 50 オームです。)
- イーサネット・アダプターは、カード上のトランシーバーと、外部トランシーバーのどちらにも使用できます。 アダプター上に、どちらを使用するかを指定するジャンパーがあります。 ジャンパーが正しく設定されているかどうかを検査してください (方法については、アダプターの 解説書を参照してください)。
- 正しいタイプのイーサネット・コネクター (THIN は BNC、THICK は DIX) を使用していることを検証します。 このコネクター・タイプを変更する場合は、SMIT 高速パスの **smit chgenet** を使用して、「デー

タベースだけに変更を適用する」フィールドを設定します (SMIT で「Yes」に設定してください。) マシンを再始動して、構成変更を適用します。 (167 ページの『アダプターの 管理および構成』を参照してください。)

トークンリング・ネットワーク・インターフェースに関する *TCP/IP* 問題

ネットワーク・インターフェースに 関する通信問題を解決するには、以下のガイドラインを参照してください。

ネットワーク・インターフェースが 初期化され、アドレスが正しく指定されていて、アダプター・カードに異常がないことが検査済みであるにもかかわらず、ネットワーク上の一部のマシンと通信できない場合は、検査を行ってください。

- 通信できない相手側ホストが別のリング上にあるかどうかを検査します。 別のリング上にある場合、SMIT 高速パスの `smit chinet` を使用して、「ブロードキャストをローカル・トークンリングに制限」フィールドをチェックします。SMIT には No を設定しないでください。
- トークンリング・アダプターが正しいリング速度で実行されるように構成されていることを検証します。正しく構成されていない場合は、SMIT を使用して、アダプター・リング速度属性を変更します (167 ページの『アダプターの 管理および構成』を参照)。**TCP/IP** を再始動すると、トークンリング・アダプターのリング速度がネットワークのその他の部分と同じになります。

トークンリング/イーサネットのブリッジに関する *TCP/IP* 問題

トークンリング・ネットワークとイーサネット・ネットワークの間でブリッジを使用した通信ができず、かつそのブリッジが正しく機能していることが検査済みである場合は、イーサネット・アダプターがパケットをドロップしている可能性があります。

マシンがパケットをドロップするのは、着信パケット (ヘッダーを含む) がネットワーク・アダプターの最大伝送単位 (MTU) 値より大きい場合です。 例えば、トークンリング・アダプターからブリッジを 使用して送信された 1500 バイトのパケットには 8 バイトの論理リンク制御 (LLC) ヘッダーが付き、合計パケット・サイズが 1508 バイトになります。 受信側イーサネット・アダプターの MTU が 1500 に設定されていると、そのパケットはドロップされます。

両方のネットワーク・アダプターの MTU 値を検査してください。 8 バイトの LLL ヘッダーを受け入れるために、トークンリング・アダプターを 発信パケットに接続し、トークンリング・アダプターの MTU 値をイーサネット・アダプターの MTU 値より少なくとも 8 バイト小さく設定してください。 例えば、MTU が 1500 のイーサネット・アダプターと 通信するには、トークンリング・アダプターの MTU を 1492 に設定します。

トークンリング/トークンリングのブリッジに関する *TCP/IP* 問題

ブリッジを介して操作する場合は、最大伝送単位 (MTU) のデフォルト値 1500 を、ブリッジが経路指定制御フィールドで通知した最大情報フィールド (最大 I フレーム) より 8 だけ小さい値に変更してください。

経路指定制御フィールドの値を検索するには、**iptrace** デーモンを使用して着信パケットを調べます。 バイト 1 のビット 1、2、3 は、最大フレーム・ビットであり、特定の経路を通じて 2 つの通信ステーション間で送信できる情報フィールドの最大値を指定します。 経路指定制御フィールドのフォーマットは、次の図をご覧ください。

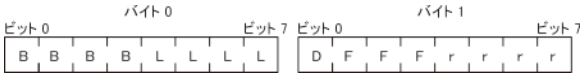


図 *25.* 経路指定制御フィールド

この図には、経路指定制御フィールドのバイト 0 とバイト 1 が示されています。 バイト 0 の 8 ビットは B、B、B、B、L、L、L、L です。バイト 1 の 8 ビットは D、F、F、F、r、r、r、r です。

最大フレーム・ビットの値は次のとおりです。

項目	説明
000	情報フィールドの最大値が 516 バイトであることを指定します。

項目	説明
001	情報フィールドの最大値が 1500 バイトであることを指定します。
010	情報フィールドの最大値が 2052 バイトであることを指定します。
011	情報フィールドの最大値が 4472 バイトであることを指定します。
100	情報フィールドの最大値が 8144 バイトであることを指定します。
101	予約済み
110	予約済み
111	全経路ブロードキャスト・フレーム内で使用されます。

例えば、最大 I フレーム値が経路指定制御フィールド内で 2052 である場合、MTU サイズは 2044 に設定しなければなりません。 これは、トークンリング・ネットワーク・インターフェースの 場合だけです。

注 : iptrace を使用する場合、出力ファイルをネットワーク・ファイルシステム (NFS) 上に置くことはできません 。

リモート・ホストとの通信における TCP/IP の問題
リモート・ホストと通信できない場合は、次の推奨事項を行ってください。

- ローカル・ホスト上で **ping** コマンドを実行し、ネットワークへのローカル・インターフェースが 始動し、作動していることを検証します。
- 通信に障害が起きている点を判別するために、ローカル・ホストからホップするホストとゲートウェイに対して、**ping** コマンドを使用します。

パケットが失われたり、パケット送達が遅れたりする場合は、次の操作を試みてください。

- **trpt** コマンドを使用して、ソケット・レベルでパケットをトレースします。
- **iptrace** コマンドを使用して、すべてのプロトコル層をトレースします。

トークンリング・ネットワークとイーサネット・ネットワークの間でブリッジを使用して通信ができず、そのブリッジが正しく機能していることが検査済みである場合は、次のようにします。

- 両方のアダプターの MTU 値を検査してください。 通信できるようにするには、両方の MTU 値に互換性がなければなりません。 着信パケット (ヘッダーを含む) がアダプターの MTU 値より大きい場合、マシンはパケットをドロップします。 例えば、ブリッジを使用して送信された 1500 バイトのパケットには 8 バイトの LLC ヘッダーが付き、合計パケット・サイズが 1508 バイトになります。 受信側マシンの MTU が 1500 に設定されていると、1508 バイトのパケットはドロップされます。

照会への snmpd 応答に関する TCP/IP 問題
snmpd が照会に応答せず、ログ・メッセージも受信していない場合、パケットがカーネルの **ユーザー・データグラム・プロトコル (UDP)** パケット・ハンドラーにとって大きすぎる可能性があります。

その場合には、カーネル変数の **udp_sendspace** と **udp_recvspace** を変更するため、次のコマンドを入力します。

```
no -o udp_sendspace=64000
no -o udp_recvspace=64000
```

UDP パケットの最大サイズは 64K です。 64K を上回る照会は拒否されます。 この問題を回避するために、そのようなパケットは複数の小さなパケットに分割してください。

動的ホスト構成プロトコルに関する TCP/IP 問題

構成データを取得できない場合は、以下の解決策を試してください。

IP アドレスまたはその他の構成パラメーターを取得できない場合は、次のようにします。

- 構成したいインターフェースを 指定してあるかどうかを検査します。 これは、SMIT 高速パス `smit dhcp` を使用すると実行できます。
- ローカル・ネットワーク上のサーバーまたはリレー・エージェントが、こちら側の要求をローカル・ネットワークの外へ送り出すように構成されていることを検証します。
- **dhcpcd** プログラムが実行されているかどうかを検査します。 実行されていない場合は、**startsrc -s dhcpcd** コマンドを使用します。

TCP/IP コマンド

TCP/IP は、システムの基礎となる構造の部分です。 TCP/IP により、コマンドまたはプログラムを実行するだけで別の端末またはシステムと 通信できます。

TCP/IP は、システムの基礎となる構造の部分です。 TCP/IP により、コマンドまたはプログラムを実行するだけで別の端末またはシステムと 通信できます。 後のことは、システムが処理します。

項目	説明
chnamsv	ホスト上の**伝送制御プロトコル/インターネット・プロトコル (TCP/IP)** ベースのネーム・サービス構成を変更します。
chprtsv	クライアント・マシンまたはサーバー・マシン上で表示サービスの構成を変更します。
hostent	システム 構成データベース内でアドレス・マッピング・エントリーを直接操作します。
ifconfig	**TCP/IP** を使用して、ネットワークのネットワーク・インターフェース ・パラメーターを構成または表示します。
mknamsv	ホスト上でクライアント用の **TCP/IP** ベースのネーム・サービスを構成します。
mkprtsv	ホスト上で **TCP/IP** ベースの印刷サービスを構成します。
mktcpip	ホスト上で **TCP/IP** を始動するために必要な値を設定します。
no	ネットワーク・オプションを構成します。
rmnamsv	ホスト上で **TCP/IP** ベースのネーム・サービスを構成解除します。
rmprtsv	クライアント・マシンまたはサーバー・マシン上で表示サービスを構成解除します。
slattach	シリアル回線をネットワーク・インターフェースとして 接続します。
arp	**アドレス解決プロトコル (ARP)** が使用するハードウェア・アドレス変換テーブルに対する IP アドレスを表示または変更します。
gettable	ホストからネットワーク情報センター (NIC) フォーマットのホスト・テーブルを取得します。
hostid	現行ローカル・ホストの ID を設定または表示します。
hostname	現行ホスト・システムの名前を設定または表示します。
htable	ホスト・ファイルをネットワーク・ライブラリー・ルーチン用のフォーマットに変換します。
ipreport	指定されたパケット・トレース・ファイルからパケット・トレース・レポートを生成します。
iptrace	インターネット・プロトコルにインターフェース ・レベルのパケット・トレース機能を提供します。

項目	説明
lsnamsv	データベース内に保管されているネーム・サービス情報を表示します。
lsprtsv	データベース内に保管されている表示サービス情報を表示します。
mkhosts	ホスト・テーブル・ファイルを生成します。
namerslv	システム構成データベース内で、ローカル resolver ルーチン用のドメイン・ネーム・サーバー・エントリーを直接操作します。
netstat	ネットワーク状況を表示します。
route	ルーティング・テーブルを手作業で操作します。
ruser	3 つのシステム・データベース内で、外部ホストによるプログラムへのアクセスを制御するエントリーを直接操作します。
ruptime	ネットワーク上の各ホストの状況を表示します。
securetcpip	ネットワーク・セキュリティー機能を使用可能にします。
setclock	ネットワーク上のホストの時刻と日付を設定します。
timedc	**timed** デーモンに関する情報を戻します。
trpt	**伝送制御プロトコル (TCP)** ソケット上でプロトコル・トレースを実行します。

SRC コマンド

SRC コマンドは、1 つのデーモン、デーモンのグループ、デーモンとそのデーモンが制御する複数のデーモン (サブシステムと 複数のサブサーバー) に対して効果があります。

また、すべての SRC コマンドに応答しない **TCP/IP** デーモンもあります。 **TCP/IP** デーモンとその例外を制御するために使用できる SRC コマンドを次に示します。

項目	説明
startsrc	すべての **TCP/IP** サブシステムと **inetd** サブサーバーを始動します。 **startsrc** コマンドは、すべての **TCP/IP** サブシステムと **inetd** サブサーバーに機能します。
stopsrc	すべての **TCP/IP** サブシステムと **inetd** サブサーバーを停止します。 このコマンドは、**stop normal** コマンドとも呼ばれます。 **stop normal** コマンドを使用すると、サブシステムはすべての 未解決作業を処理して、サブシステムを 正常に終了することができます。 **inetd** サブサーバーの場合、保留中のすべての接続を開始でき、既存のすべての接続を終了できます。 **stop normal** コマンドは、すべての **TCP/IP** サブシステムと **inetd** サブサーバーに機能します。
stopsrc -f	すべての **TCP/IP** サブシステムと **inetd** サブサーバーを停止します。 このコマンドは、**stop force** コマンドとも呼ばれます。 **stop force** コマンドを使用すると、すべてのサブシステムがすぐに 終了します。 **inetd** サブサーバーの場合は、保留中のすべての接続と既存のすべての接続がすぐに終了します。
refresh	**inetd**、**syslogd**、**named**、**dhcpsd** および **gated** サブシステムとサブサーバーをリフレッシュします。
lssrc	サブシステムに関する簡略状況、つまり、指定したサブシステムの 状態 (アクティブか作動不能) を提供します。 また、**inetd** サブサーバーに関する簡略状況も提供します。 **inetd** サブサーバーに関する簡略状況には、サブサーバー名、状態、サブサーバーの説明、コマンド名、そのサブサーバーを起動したときの引数が入っています。

項目	説明
lssrc -l	l 次のサブシステムについて、簡略状況と追加情報 (長時間状況) を提供します。 **gated** デバッグまたはトレースの状態、アクティブにされた経路指定プロトコル、ルーティング・テーブル、受信したシグナルとその機能 **inetd** デバッグの状態、アクティブなサブサーバーのリストとそれらのサブサーバーの簡略状況、受信したシグナルとその機能 **named** デバッグの状態、named.conf ファイル情報 **dhcpsd** デバッグの状態、制御されているすべての IP アドレスおよびその現在の状態 **routed** デバッグとトレースの状態、経路指定情報の供給状態、ルーティング・テーブル **syslogd** **syslogd** 構成情報 **lssrc -l** コマンドは **inetd** サブサーバーに関する長時間状況も提供します。 長時間状況には、簡略状況の情報とアクティブな接続に関する情報が入っています。一部のサブサーバーは、追加情報を提供します。 サブサーバーによる追加情報は次のとおりです。 **ftpd** デバッグとロギングの状態 **telnetd** 端末エミュレーションのタイプ **rlogind** デバッグの状態 **fingerd** デバッグとロギングの状態 **rwhod** サブサーバーと **timed** サブサーバーは、長時間状況を提供しません。
traceson	ソケット・レベルのデバッグをオンにします。 出力をフォーマット設定するには、**trpt** コマンドを使用します。 **timed** サブシステムと **iptraced** サブシステムは **traceson** コマンドをサポートしていません。
tracesoff	ソケット・レベルのデバッグをオフにします。 出力をフォーマット設定するには、**trpt** コマンドを使用します。 **timed** サブシステムと **iptraced** サブシステムは **tracesoff** コマンドをサポートしていません。

これらのコマンドの使用例については、個々のコマンドに関するトピックを参照してください。システム・リソース・コントローラーついて詳しくは、オペレーティング・システムおよびデバイスの 管理のシステム・リソース・コントローラーの概要を参照してください。

ファイル転送コマンド

ここでは、ファイル転送コマンドについて簡潔に説明します。

項目	説明
ftp *hostname*	ローカル・ホストとリモート・ホスト間でファイルを転送します。
rcp*file host:file*	ローカル・ホストとリモート・ホストの間、または 2 つのリモート・ホスト間でファイルを転送します。
tftp	ホスト間でファイルを転送します。

リモート・ログイン・コマンド

ここでは、**TCP/IP** リモート・ログイン・コマンドについて簡潔に説明します。

項目	説明
rexec *host command*	リモート・ホスト上で一度に 1 つずつコマンドを実行します。
rlogin *remotehost*	ローカル・ホストをリモート・ホストに接続します。
rsh および **remsh** *remotehost command*	指定されたコマンドをリモート・ホスト上で実行するか、またはリモート・ホストにログインします。
telnet、**tn**、および **tn3270** *hostname*	**TELNET** インターフェースを 使用して、ローカル・ホストをリモート・ホストに接続します。

状況コマンド

ここでは、**TCP/IP** 状況コマンドについて簡潔に説明します。

項目	説明
finger または **f** *user@host*	ユーザー情報を表示します。
host *hostname*	ホスト名を IP アドレスに変えるか、または IP アドレスをホスト名に変えます。
ping *hostname*	エコー要求をネットワーク・ホストに送信します。
rwho	ローカル・ネットワーク上のホストにログインしているユーザーを表示します。
whois *name*	ユーザー ID または別名でユーザーを識別します。

リモート通信コマンド

TCP/IP リモート通信コマンド **talk** *User@Host* を使用すると、他のユーザーと会話できます。

項目	説明
talk *User@Host*	別のユーザーと会話します。

印刷コマンド

ここでは、**TCP/IP** 印刷コマンドについて簡潔に説明します。

項目	説明
enq *file*	ファイルをキューに入れます。
refresh	1 サブシステムまたはサブシステムのグループのリフレッシュを 要求します。
smit	システム管理を実行します。

TCP/IP デーモン

サブシステム とは、SRC によって制御されるデーモン、すなわち、サーバーのことです。 サブシステムによって制御されるデーモンのことをサブサーバー といいます (デーモン・コマンドとデーモン名は通常、名前の末尾に **d** が付きます)。

サブシステムと サブサーバーのカテゴリーは、互いに排他的です。 つまり、デーモンはサブシステムとサブサーバーの両方としてリストされることはありません。 他のデーモンを制御する **TCP/IP** サブシステムは、**inetd** デーモンだけです。 したがって、すべての **TCP/IP** サブサーバーは、**inetd** サブサーバーでもあります。

SRC によって制御される **TCP/IP** のデーモンは次のとおりです。

サブシステム

項目	説明
gated	ゲートウェイ経路指定機能を提供するとともに、**Routing Information Protocol (RIP)**、**Routing Information Protocol Next Generation (RIPng)**、**外部ゲートウェイ・プロトコル (EGP)**、**ボーダー・ゲートウェイ・プロトコル (BGP)** と **BGP4+**、**DCN ローカル・ネットワーク・プロトコル (HELLO)**、**Open Shortest Path First (OSPF)**、Intermediate System to Intermediate System (IS-IS)、および**インターネット制御メッセージ・プロトコル (ICMP** と **ICMPv6)/Router Discovery 経路指定**プロトコルをサポートします。 さらに、**gated** デーモンは**シンプル・ネットワーク管理プロトコル (SNMP)** もサポートします。 **gated** デーモンはネットワーク・アドレスへの経路指定に使用できる 2 つの経路指定デーモンの 1 つであり、優先して使用される経路指定デーモンです。 **gated** デーモンは **routed** デーモンより優先して使用されます。その理由は、**gated** デーモンがより多くのゲートウェイ・プロトコルをサポートしていることにあります。
inetd	他のデーモンのサービスを求める要求を受信すると、それらのデーモンを起動して、スケジュールします。 このデーモンは、他のデーモンを始動することもできます。 **inetd** デーモンはスーパーデーモンとも呼ばれます。
iptrace	インターネット・プロトコルにインターフェース ・レベルのパケット・トレース機能を提供します。
named	**ドメイン・ネーム・サーバー**・プロトコル (DOMAIN) に命名機能を提供します。
routed	ネットワーク・ルーティング・テーブルを管理し、**Routing Information Protocol (RIP)** をサポートします。 **gated** デーモンは **routed** デーモンより優先して使用されます。その理由は、**gated** デーモンがより多くのゲートウェイ・プロトコルをサポートしていることにあります。
rwhod	他のすべてのホストに 3 分間隔でブロードキャストを送信し、ログインしているユーザーとネットワーク状況に関する情報を保管します。 **rwhod** デーモンは、マシンのリソースのかなりの量を使用することがあるので、十分に注意して使用してください。
timed	タイム・サーバー機能を提供します。

注 : routed デーモンと **gated** デーモンは、両方とも TCP/IP サブシステムとしてリストされます。 この両方の経路指定デーモンを開始する **startsrc -g tcpip** コマンドは、ほかのどの **TCP/IP** サブシステムとも一緒に実行しないでください。 1 台のマシン上で両方のデーモンを同時に実行すると、予測できない結果になる恐れがあります。

inetd サブシステムによって制御される TCP/IP デーモンは次のとおりです。

inetd サブサーバー

項目	説明
comsat	ユーザーにメールが着信したことを通知します。
fingerd	指定したリモート・ホストにログインしているすべてのユーザーと、そのホストのネットワーク状況に関するレポートを提供します。 このデーモンは、**Finger** プロトコルを使用します。
ftpd	**ファイル転送プロトコル (FTP)** を使用してクライアント・プロセスにファイル転送機能を提供します。
rexecd	**rexec** コマンドに外部ホスト・サーバー機能を提供します。
rlogind	**rlogin** コマンドにリモート・ログイン機能を提供します。
rshd	**rcp** コマンドと **rsh** コマンドにリモート・コマンド実行サーバー機能を提供します。
talkd	**talk** コマンドに会話機能を提供します。

項目	説明
syslogd	システム・メッセージを読み取って記録します。 このデーモンは、**リモート・アクセス・サービス** (RAS) サブシステム・グループに属します。
telnetd	**TELNET** プロトコル用のサーバー機能を提供します。
tftpd	**トリビアル・ファイル転送プロトコル (TFTP)** 用のサーバー機能を提供します。
uucpd	基本ネットワーク・ユーティリティー (BNU) と **TCP/IP** の間の通信を処理します。

デバイス・メソッド

デバイス・メソッドは、基本的なデバイス構成操作を実行するデバイス関連プログラムです。

TCP/IP メソッドについては、*Communications Programming Concepts* の List of TCP/IP Programming References を参照してください。

Request for Comments

AIX システムでは、以下の **TCP/IP** Request for Comments (RFC) がサポートされています。

このオペレーティング・システムでサポートされる RFC (Request For Comments) のリストについては、*Communications Programming Concepts* の List of TCP/IP Programming References を参照してください。

- RFC 1359 *Connecting to the Internet: What connecting institutions should anticipate*
- RFC 1325 *FYI on questions and answers: Answers to commonly asked 'new Internet user' questions*
- RFC 1244 *Site Security Handbook*
- RFC 1178 *Choosing a Name for Your Computer*
- RFC 1173 *Responsibilities of host and network managers: A summary of the `oral tradition' of the Internet*

基本ネットワーク・ユーティリティー

基本ネットワーク・ユーティリティー (BNU) とは、ローカル・ネットワークとリモート・ネットワークのコンピューター・システム間で通信を確立するプログラム、ディレクトリー、およびファイルのグループです。 BNU を使用すると、UNIX 間コピー・プログラム (UUCP) のいずれかのバージョンを実行している UNIX システムと通信することができます。 BNU は、基本オペレーティング・システムと共にインストールできる拡張サービス・プログラムの 1 つです。

UUCP (AT&T が開発し、後にバークレー・ソフトウェア・ディストリビューション (BSD) に組み込まれた、UNIX 間通信機能の総称) 関連のコマンド群も BNU に属します。 BNU はローカル・システムおよびリモート・システムへの接続のためのコマンド、プロセス、サポート用データベースを提供します。 ローカル・ネットワーク上のシステムに接続するには、トークンリングやイーサネットなどの通信ネットワークを使用します。 ローカル・ネットワークは、ケーブルまたは電話モデムによってリモート・システムに接続できます。 接続されると、ローカル・ネットワークとリモート・システムの間で、コマンドとファイルを交換できます。

システムで BNU プログラムを実行するには、その前に BNU をインストールして、構成する必要があります。

BNU は一連の構成ファイルによって制御されます。各構成ファイルは、リモート・システムがローカル・システムにログインできるかどうか、また、ログイン後にリモート・システムが何を実行できるかを決定します。 使用システムに必要な要件とリソースに応じて、これらの構成ファイルを設定する必要があります。

BNU を保守するには、ログ・ファイルを定期的に読み取って除去し、さらに BNU キューを調べて、ジョブがリモート・システムに正しく転送されていることを確認する必要があります。 また、構成ファイルを定期的に更新して、使用システムとリモート・システム内の変更内容を反映させなければなりません。

BNU の機能

BNU は、一連のハードウェア接続とソフトウェア・プログラムを使用して、システム間で通信します。

ディレクトリーとファイルの構造により、BNU のアクティビティーが 追跡されます。 この構造には、一連の公開ディレクトリー、管理ディレクトリーと管理ファイルのグループ、構成ファイル、およびロック・ファイルが含まれています。 BNU 用のほとんどのディレクトリーは、インストール・プロセス中に作成されます。 管理ディレクトリーと管理ファイルの一部は、各種の BNU プログラムによって作成されます。

リモート・ログイン・コマンドを除いて、BNU はバッチ・システムとして機能します。 ユーザーがジョブをリモート・システムに送信するように要求すると、BNU は、そのジョブを完了するのに必要な情報を格納します。 このことを、ジョブのキューイング と呼びます。 スケジュールした時刻、またはユーザーが指示した時刻に、BNU は各種のリモート・システムに接続し、キューに入る作業を転送してジョブを受け入れます。 これらの転送処理は、システム 上の構成ファイルとリモート・システム 上の構成ファイルによって制御されます。

BNU コマンドの各国語サポート

uucpadm 以外のすべての BNU コマンドは各国語サポートに対応しています。

ユーザー名は ASCII 文字でなくてもかまいません。 ただし、システム 名はすべて ASCII 文字でなければなりません。 ASCII 文字ではないシステム 名を伴う転送またはリモート・コマンドの実行をスケジュールしようとすると、BNU からエラー・メッセージが戻ります。

BNU のファイルとディレクトリーの構造

BNU は、ディレクトリーとファイルの構造を使用して、そのアクティビティーを 追跡します。

この構造には、公開ディレクトリー、構成ファイル、管理ディレクトリー、およびロック・ファイルが含まれます。

BNU 用のほとんどのディレクトリーは、インストール・プロセス中に作成されます。 管理ディレクトリーと管理ファイルの一部は、各種の BNU プログラムによって実行時に作成されます。

BNU 公開ディレクトリー

指定されている場合、BNU 公開ディレクトリー (`/var/spool/uucppublic`) は、他のシステムからローカル・システムに転送されたファイルを格納します。

そのファイルは、ユーザーが要求するまで、公開ディレクトリー内で待機します。 公開ディレクトリーは、BNU のインストール時に作成されます。 公開ディレクトリー内で、BNU は、ローカル・システムにファイルを送信するリモート・システムごとに 1 つずつ、サブディレクトリーを作成します。

BNU の構成ファイル

BNU 構成ファイルは、BNU サポート・データベースとも呼ばれ、`/etc/uucp` ディレクトリー内にあります。 このファイルは、使用システムに合わせて構成する必要があります。

各ファイルは、uucp のログイン ID によって所有され、root 権限を持つユーザーのみ編集できます。 この構成ファイルには、次のような情報が入っています。

- アクセス可能なリモート・システム
- リモート・システムへの接続用デバイス
- リモート・システムへの接続時間帯
- リモート・システムがシステム 上で実行可能なタスク

また、一部の構成ファイルでは、システムが過負荷にならないように BNU のアクティビティーに 関する制限を指定します。

BNU 構成ファイルには、次のファイルがあります。

項目	説明
`Devices`	モデムによる接続と直接接続など、使用可能なデバイスについての情報が入っています。

項目	説明
Dialcodes	Systems ファイル内の電話番号を短縮できるように、ダイヤル・コードの短縮形が入っています。
Dialers	特定のモデム・タイプ (自動ダイヤラー) をコールするためのコマンド構文を指定します。
Maxuuscheds	同時にスケジュールされるジョブの数を制限します。
Maxuuxqts	リモート・コマンドの同時実行数を制限します。
Permissions	アクセス権コードが入っています。 このファイルは、BNU のセキュリティーを判別するための 1 次ファイルです。
Poll	BNU プログラムがリモート・システムをポーリングしてタスクを開始する時刻を指定します。
Sysfiles	BNU 構成の Systems、Devices、および Dialers ファイルとして機能するファイルを出力します。 このファイルを使用しない場合、デフォルト・ファイルは /etc/uucp/Systems、/etc/uucp/Devices、および /etc/uucp/Dialers です。
Systems	アクセス可能なリモート・システムや、使用するデバイスとログインに必要なユーザー名とパスワードの組み合わせなどリモート・システムへの接続に必要な情報を出力します。 また、システムに接続可能な時間帯も指定します。

各構成ファイルは、BNU の使用中に相互参照します。 以下に例を示します。

- Devices ファイルには、Dialers ファイル内のエントリーを参照する *Token* フィールドがあります。
- Systems ファイルには、デバイスの *Class* のエントリーが入っています。 Systems ファイル内で参照される各 *Class* のデバイスは、Devices ファイル内で定義されていなければなりません。
- Poll ファイルには、システムのコール先のシステムに関するエントリーが入っています。 これらの各システムは Systems ファイル内で定義されていなければなりません。

BNU 構成ファイル内のエントリーは、システムと各リモート・システム間の接続のタイプによって異なります。 例えば、TCP/IP 接続または直接接続を使用して他のシステムに接続する場合は、特殊なエントリーを作成する必要があります。 モデムを使用して他のシステムに接続する場合は、そのモデムを Dialers ファイル内に定義する必要があります。

BNU を使用してリモート・システムに接続する前に、システム上で Systems、Devices、および Permissions ファイルを構成しなければなりません。 他の構成ファイルにより、自動ポーリングなどの BNU 機能が使用できるようになります。 多くの構成ファイルは定期的に変更して、使用のシステムや接続先のシステムの変更内容を反映させなければなりません。 Sysfiles ファイルを使用して、デフォルトの Systems、Devices、および Dialers ファイル以外のファイルを同等の機能を持つファイルとして指定することができます。

BNU の管理ディレクトリーと管理ファイル

BNU の管理ディレクトリーと管理ファイルは、/var/spool/uucp ディレクトリーのサブディレクトリーに入っています。

これらのディレクトリーとファイルには、次の 2 種類の情報が入っています。

- 他のシステムに転送されるまで待機中のデータ
- BNU のアクティビティーに関するログ情報とエラー情報

BNU は、/var/spool/uucp ディレクトリーの下に次のディレクトリーを作成します。

項目	説明
.Admin	次の 4 つの管理ファイルが入っています。 • audit • Foreign • errors • xferstats これらのファイルには、BNU のアクティビティーに 関するエラー情報とログ情報が入っています。
.Corrupt	BNU プログラムでは処理できないファイルのコピーが入っています。
.Log および .Old	BNU トランザクションからのログ・ファイルが入っています。
.Status	**uucico** デーモンが最後にリモート・システムに接続しようとした時刻が保管されています。
.Workspace	ファイル転送プログラムが内部で使用する一時ファイルが入っています。
.Xqtdir	実行ファイル、およびリモート・システムが実行できるコマンドのリストが入っています。
SystemName	ファイル転送プログラムが使用するファイルが入っています。 それらのファイルは次のとおりです。 • コマンド (**C.***) • データ (**D.***) • 実行 (**X.***) • 一時 (**TM.***) BNU は、接続先となるリモート・システムごとに *SystemName* ディレクトリーを作成します。

名前がピリオドで始まるディレクトリーは、隠し ディレクトリーです。 隠しディレクトリーは **-a** フラグを指定した場合を除き、**ls** コマンドや **li** コマンドでは表示されません。 **uucico** デーモンは、始動時に /var/spool/uucp ディレクトリーから作業ファイルを見つけ、隠しディレクトリー以外のディレクトリーにあるファイルをすべて転送します。 **uucico** デーモンは、*SystemName* ディレクトリーのみを参照し、その他の管理ディレクトリーは参照しません。

隠しディレクトリー内のファイルは、uucp ログイン ID によって所有されます。 これらのファイルには、root 権限をもつユーザー、すなわち UID が 5 のログイン ID を持つユーザーのみがアクセス可能です。

BNU 管理ディレクトリーの保守の詳細については、506 ページの『BNU の保守』 を参照してください。

BNU ロック・ファイル

BNU ロック・ファイルは、/var/locks ディレクトリーに格納されています。 BNU はデバイスを使用してリモート・コンピューターに接続するときに、そのデバイスのロック・ファイルを /var/locks ディレクトリーに入れます。

別の BNU プログラムや他のプログラムがそのデバイスを必要とする場合、そのプログラムは、/var/locks ディレクトリーを調べてロック・ファイルを探します。 ロック・ファイルが存在する場合、プログラムは、そのデバイスが使用可能になるまで待つか、または別のデバイスを通信に使用します。

また、**uucico** デーモンは、リモート・システム用のロック・ファイルを /var/locks ディレクトリーに入れます。 リモート・システムへ接続する前に、**uucico** デーモンは、/var/locks ディレクトリー内で、そのシステム用のロック・ファイルを探します。 これらのファイルにより、**uucico** デーモンの他のインスタンスは、同じリモート・システムへの重複接続を確立できなくなります。

注 : BNU 以外の ATE や TCP/IP などの他のソフトウェアも /var/locks ディレクトリーを使用します。

BNU の構成

この手順では、直接接続、モデム接続、および TCP/IP (伝送制御プロトコル/インターネット・プロトコル) 接続など、各種の接続用に基本ネットワーク・ユーティリティー (BNU) を構成する方法について説明します。

前提条件

- BNU がシステムにインストールされている必要があります。
- BNU 構成ファイルを編集するには、root ユーザー権限が必要です。
- BNU 通信用に直接接続を使用する場合、ローカル・システムとリモート・システム間で接続が正しく接続されている必要があります。
- BNU 通信用にモデムを使用する場合、各モデムをインストールし、構成する必要があります。
- 1 つ以上の接続で TCP/IP を使用する場合は、ローカル・システムと該当するリモート・システム間で TCP/IP が実行されている必要があります。
- BNU の構成に必要な情報を収集します (以下のリストを参照)。この情報には、リモート・システムのリストと、リモート・システムへの接続に使用するデバイスやモデムのリストが含まれます。

必要なシステム情報の収集

BNU を構成する前に、以下の情報を収集してください。

- システムがコールするリモート・システム ごとに、次の情報を収集します。
  - システム名
  - ローカル・システムがリモート・システム上で使用するログイン名
  - ログイン名のパスワード
  - リモート・システム上のログイン・プロンプトとパスワード・プロンプト。
  - リモート・システムに到達するために使用する接続のタイプ (直接、モデム、または TCP/IP)

  直接接続の場合、以下の情報を収集します。

  - 接続のビット伝送速度
  - ローカル・システム上の接続に使用されるポート

  モデムを介した接続 (電話接続) の場合、以下の情報を収集します。

  - リモート・システムの電話番号。
  - リモート・システムの速度と互換性のあるモデムの速度。

  注: リモート・システムがローカル・システムをコールする場合は、各リモート・システム上の BNU 管理者がローカル・システムについて上記の情報をすべて持っていることを確認してください。
- BNU 接続に使用するローカル・モデム ごとに、次の情報を収集します。
  - モデム用のチャット・スクリプト (モデムの資料を参照してください) 。

    注: モデムによっては、チャット・スクリプトが `/etc/uucp/Dialers` ファイルに入っている場合があります。
  - モデムのローカル・ポート。

システム・デバイスのリストの作成

収集した情報を使用して、リモート・システムに接続する必要がある各システム・デバイスのリストを作成します。ローカル・システム `morgan` のサンプル・リストを次に示します。

```
direct:
hera 9600 tty5
zeus& 2400 tty2
ariadne 2400 tty1
hayes modem (tty3):  apollo, athena
TCP/IP:  merlin, arthur, percy
```

上記の例では、システム hera に接続するには、9600 の速度でポート tty5 から direct 接続を使用します。システム apollo に接続するには、ポート tty3 に接続している hayes モデムを使用します。システム merlin、arthur および percy に接続するには、TCP/IP を使用します。

リモート通信機能の構成

BNU がサイトで正常に機能するためには、リモート通信機能を以下のように構成する必要があります。

- 直接、電話、またはモデムによる通信リンクの確立に使用するデバイスをリストする。
- 電話網を介してリモート・システムへの接続に使用するモデムをリストする。
- アクセス可能なリモート・システムをリストする。
- 指定したリモート・システムへの接続に使用する電話番号の接頭部を表す省略形をリストする (オプション)。
- ローカル・システムとリモート・システムの通信方法を指定することにより、アクセス権を設定する。
- ネットワーク化されたリモート・システムのモニター・スケジュールを作成する (オプション)。

上記のリスト、許可、およびスケジュールを作成するには、以下の手順を実行します。

- BNU 構成ファイルを変更します。
- /var/spool/cron/crontabs/uucp ファイルを編集し、自動保守ルーチンをスケジュールする行の先頭からコメント記号 (#) を除去します。

注 : BNU をサイトで正しく実行するには、Systems、Devices、および Permissions の各ファイルを構成する必要があります。ただし、これらの BNU 構成ファイルの変更順序は問いません。

上記の手順を完了した後、ローカル・システム上に BNU を構成できます。

ローカル・システム上での BNU の構成

BNU を構成するには、次の手順を実行します。

1. 次のコマンドを実行し、システムに BNU がインストールされていることを確認します。

   ```
   lslpp -h bos.net.uucp
   ```

   BNU がインストールされている場合は、出力に「bos.net.uucp」と表示されます。これが表示されない場合は、インストール・テープから BNU をインストールしてください。
2. ローカル・システムを呼び出すリモート・システムに適切なログイン ID とパスワードを設定し、各リモート・システムの BNU や UNIX 間コピー・プログラム (UUCP) の管理担当者に、ログインとパスワードを提供します。

   この手順を実行するには、/etc/passwd、/etc/group、/etc/security/login.cfg、および /etc/security/passwd の各ファイルを編集します。

   重要 : UUCP のログイン ID でリモート・システムがローカル・システムにログインするのを 許可すると、ローカル・システムのセキュリティーが 危険にさらされます。UUCP ID を使用してログインしたリモート・システムは、ローカルの Systems ファイルと Permissions ファイルを表示できるだけでなく、変更することも可能です。 そのようなリモート・システムのアクションは、Permissions ファイルの LOGNAME エントリーで指定された許可によって異なります。リモート・システム用として別途に BNU ログイン ID を作成し、BNU の管理担当者の UUCP ログイン ID はローカル・システム上に予約しておくことをお勧めします。セキュリティーを最善なものにするためには、ローカル・システムに接続する各リモート・システムには固有のユーザー ID (UID) 番号を含む固有のログイン ID を持たせる必要があります。これらのログイン ID は、グループ ID (GID) を 5 にする必要があります。デフォルトでは、オペレーティング・システムにはファイル転送用の NUUCP ログイン ID が組み込まれています。

   a) 個々のシステムのアクセスに対する完全な制御を維持する必要がある場合は、ログイン ID を別々に作成し、Permissions ファイル内で MACHINE エントリーと LOGNAME エントリーを組み合わせる必要があります。 個別のログインを使用するか、またはすべての BNU 接続に 1 つのログインを使用するかを選択できます。 以下に、/etc/passwd エントリーの例をいくつか示します。

```
Umicrtk:!:105:5:micrtk uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
Ufloydl:!:106:5:floydl uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
Uicus:!:107:5:icus uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
Urisctkr:!:108:5::/usr/spool/uucppublic:/usr/sbin/uucp/uucico
```

b) すべての UUCP 接続に対して別々の制御を維持するのではなく、1 つの許可セットだけを使用したい場合は、すべてのシステム用に単一のログインを持つことができます。そのようなシナリオのエントリーの例を以下に示します。

```
nuucp:!:6:5::/usr/spool/uucppublic:/usr/sbin/uucp/uucico
```

注:

- UID (コロンで区切られた 3 番目のフィールド) は、セキュリティー・リスクを回避するために、固有であることが必要です。
- GID (コロンで区切られた 4 番目のフィールド) は、UUCP と同じグループに含まれるよう、5 とする必要があります。
- ホーム・ディレクトリー (コロンで区切られた 6 番目のフィールド) は、任意の有効なディレクトリーに変更できます。
- ログイン・シェル (コロンで区切られた 7 番目のフィールド) は、常に /usr/sbin/uucp/uucico とする必要があります。

c) /etc/group ファイルに新しいユーザーが存在していることを 確認します。 そのようなエントリーの例を次に示します。

```
uucp:!:5:uucp,uucpadm,nuucp,Umicrtk,Uicus,Urisctakr
```

d) モデムを使用して、**cu** コマンド以外のプログラムで接続するユーザーを、UUCP グループに追加します。

e) 上記のファイルを root として編集した後、コマンド **passwd** *UserName* を使用して、新しいユーザーのパスワードを設定します。

注: root ログインからパスワードを変更すると、/etc/security/passwd ファイル内にあるそのユーザーのスタンザの flags エントリーに、次の行が含まれています。

```
flags = ADMCHG
```

上記の行を、次の例に示すように変更する必要があります。

```
flags =
```

変更しなかった場合、リモートの **uucico** プロセスがローカル・システムにログインするとき、システムはプロセスにプロンプトを出し、新しいパスワードの入力を求めます。 このアクションは不可能であり、そのためにログインは失敗します。

f) **uucico** プロセスによって起こされる、ログイン・プロセスでの割り込み (デフォルトのヘラルドによって、そのすべての Ctrl-J で開始される可能性がある) を回避するために、デフォルトのスタンザを (アスタリスクで) コメント化し、次の例に示すように、使用する tty のスタンザを定義します。

```
 /dev/tty0:
            herald = "¥nrisc001 login:"
```

g) ASCII テキスト・エディターまたは **uucpadm** コマンドを使用して、Poll ファイルを編集します。システムがポーリングするシステムごとに 1 つずつエントリーを追加します。

注: Poll ファイル内にリストしたシステムは、/etc/uucp/Systems ファイル内にもリストする必要があります。

h) ASCII テキスト・エディターを使用して、/var/spool/cron/crontabs/uucp ファイルを編集します。 **uudemon.hour** コマンドおよび **uudemon.poll** コマンドの実行を指定する行から、コメント記号 (#) を外します。

これらのコマンドを実行する回数を変更できます。 ただし、**uudemon.poll** コマンドの実行は、**uudemon.hour** コマンドの実行よりも約 5 分前になるようにスケジュールしてください。

i) 次のコマンドを実行して、変更が有効になったかどうかを確認します。

```
crontab -l uucp
```

j) BNU データ・ファイル (Systems、Permissions、Devices、Dialers、および Sysfiles) をセットアップします。

/usr/sbin/uucp/uucpadm コマンドを使用して、最初にファイルをセットアップし、その後、必要に合わせて編集してもかまいません。 BNU 構成用に /etc/uucp/Systems、/etc/uucp/Devices、および /etc/uucp/Dialers 以外のファイルを指定するには、Sysfiles ファイルを使用します。 詳しくは、Sysfiles を参照してください。

3. Systems ファイル内で電話番号にダイヤル・コード省略形を使用する場合は、省略形ごとに Dialcodes エントリーを設定します。詳しくは、『Dialcodes File Format for BNU』を参照してください。

   BNU 接続に TCP/IP を使用する場合、次のような **netstat** コマンドを入力して、**uucpd** デーモンが機能しているかどうかを調べます。

   ```
   netstat -a
   ```

   uucpd デーモンは、**inetd** デーモンによって始動されます。 **uucpd** デーモンが実行されない場合は、**inetd** デーモンを再構成して、**uucpd** デーモンを始動します。詳しくは、408 ページの『inetd デーモンの構成』を参照してください。
4. この手順を開始する前に収集したデバイスのリストを使用して、ご使用のシステム上の Devices ファイルを変更します。モデムごとに、また直接接続ごとにエントリーを作成します。 TCP/IP を使用する場合は、Devices ファイル内の TCP/IP エントリーのコメントを外します。

   /etc/uucp/Sysfiles ファイルを構成して、デバイスの構成に他のファイルを使用するよう指定できます。Devices ファイルの詳細については、『Devices File Format for BNU』を参照してください。

   また、TCP/IP を使用する場合は、/etc/services ファイルに次の行が含まれていることを確認します。

   ```
   uucp        540/tcp          uucpd
   ```

   含まれていない場合は、この行をファイルに追加してください。
5. この手順を開始する前に収集した各リモート・システムに関する情報を使用して、ご使用のシステム上の Systems ファイルを変更します。構成を指定するときには、Systems ファイル内でコメントされている例を参考にしてください。 TCP/IP を使用する場合は、/etc/hosts ファイル内のホスト名テーブルに、接続したいリモート・コンピューターの名前が含まれていることを確認します。 /etc/uucp/Sysfiles ファイルを構成して、システムの構成に他のファイルを使用するよう指定できます。
6. この手順を開始する前に収集したデバイスとモデムに関する情報を使用して、ご使用のシステム上の Dialers ファイルに各モデムのエントリーが含まれていることを確認します。TCP/IP 接続および直接接続を使用する場合は、TCP/IP エントリーと直接接続エントリーがファイル内に存在することを確認します。/etc/uucp/Sysfiles ファイルを構成して、ダイヤラーの構成に他のファイルを使用するよう指定できます。
7. ローカル・システムからのコール先あるいはローカル・システムへのコール元となる、各リモート・システムに対して、ローカル・システムへのアクセス権をどのように与えるかを決定します。 Permissions ファイル内で、各システムと各ログイン名に該当するエントリーを設定します。
8. **uucheck** コマンドを使用して、ディレクトリー、プログラム、およびサポート・ファイルが正しくセットアップされていることを確認します。

   ```
   /usr/sbin/uucp/uucheck -v
   ```

uucheck コマンドは、ディレクトリー、プログラム、およびサポート・ファイルが正しく設定されているかどうか、また Permissions ファイルのエントリーに整合性があるかどうかを検査します。 **uucheck** コマンドでエラーが報告された場合は、そのエラーを修正してください。

9. オプション: BNU 操作の自動モニターとリモート・システムの自動ポーリングをセットアップします。 詳しくは、497 ページの『BNU の自動モニター機能の設定』、および 497 ページの『リモート・システムの BNU ポーリング機能の設定』を参照してください。

BNU の自動モニター機能の設定

BNU は、**cron** デーモンを使用して BNU デーモンを始動し、BNU のアクティビティーをモニターします。

前提条件

- 493 ページの『BNU の構成』の手順を完了します。
- /var/spool/cron/crontabs/uucp ファイルを編集するためには、root ユーザー権限が必要です。

cron デーモンは、/var/spool/cron/crontabs/uucp ファイル内で、BNU のプロシージャーを始動する時点に関する指示を読み取ります。

BNU の自動モニターをセットアップするには、以下の手順を実行します。

1. root ユーザー権限を持つユーザーとしてログインします。
2. ASCII テキスト・エディターを使用して、/var/spool/cron/crontabs/uucp ファイルを編集します。
3. BNU 保守プロシージャーの uudemon.admin および uudemon.cleanup に関する行からコメント記号を外します。

   システム保守の実行回数を増減する必要がある場合、これらのプロシージャーの実行回数を変更できます。 ただし、uudemon.admin コマンドは少なくとも 1 日に 1 回、uudemon.cleanup コマンドは少なくとも 1 週間に 1 回は実行するのが最善の方法です。
4. crontabs/uucp ファイルを使用して、**uulog**、**uuclean**、 または **uucleanup** コマンドなど、他の BNU 保守コマンドをスケジュールすることができます。

   または、crontabs/uucp ファイルを使用すると、**cron** デーモンに対して、特定の時刻に **uucico**、**uuxqt**、または **uusched** デーモンを始動するように命令できます。

リモート・システムの BNU ポーリング機能の設定

BNU がジョブについてリモート・システムをポーリングできるようにするには、そのリモート・システムを /etc/uucp/Poll ファイル内に出力します。

前提条件

- 493 ページの『BNU の構成』にリストされている手順を実行します。
- /var/spool/cron/crontabs/uucp ファイルと /etc/uucp/Poll ファイルを編集するには、root 権限が必要です。

また、/etc/uucp/Poll ファイル内のシステムのリストに加えて、**uudemon.hour** コマンドと **uudemon.poll** コマンドを定期的に実行してください。

リモート・システムの BNU ポーリングをセットアップするには、以下の手順を実行します。

1. 自動的にポーリングするリモート・システムを決定します。 各システムをポーリングする 頻度を決定します。 各システムについて、Poll ファイルで、ポーリング回数を 1 日 1 回以上、必要な回数まで指定できます。
2. root 権限を持つユーザーとしてログインします。
3. ASCII テキスト・エディターまたは **uucpadm** コマンドを使用して、Poll ファイルを編集します。 ご使用のシステムでポーリングするようセットアップされているシステムごとに 1 つずつ、エントリーを追加します。

   注: Poll ファイル内にリストしたシステムは、/etc/uucp/Systems ファイル内にもリストする必要があります。

4. ASCII テキスト・エディターを使用して、/var/spool/cron/crontabs/uucp ファイルを編集します。 **uudemon.hour** コマンドおよび **uudemon.poll** コマンドの実行を指定する行から、コメント記号 (#) を外します。
   これらのコマンドの実行時刻を変更することもできます。 ただし、**uudemon.poll** コマンドの実行は、**uudemon.hour** コマンドの実行よりも約 5 分前になるようにスケジュールしてください。

これで、BNU は、Poll ファイル内にリストされているシステムに対して、指定された時刻に自動的にポーリングを行うようセットアップされました。

/etc/uucp/Systems ファイル

リモート・システムのリストは /etc/uucp/Systems ファイル内にあります。

システム管理者は、/etc/uucp/Systems 以外のファイルを Systems ファイルとして指定することもできます。 この指定は **/etc/uucp/Sysfiles** ファイルで行います。

Systems ファイル内の各エントリーには、以下の項目が入っています。

- リモート・システムの名前
- ユーザーがリモート・システムに接続できる時間
- リンクのタイプ (直接回線またはモデム)
- リンク上の送信速度
- リモート・システムへのログインに必要な情報

Systems ファイル内の各エントリーは、1 つのリモート・システムに対応しています。 通信を確立するためには、ローカルの Systems ファイル内にリモート・システムが登録されていることが必要です。
Systems ファイルは、BNU 機能を使用するすべてのシステム上になければなりません。 通常、Systems ファイルを読むことができるのは、root ユーザーのみです。 しかし、**uuname** コマンドを使用すれば、どのユーザーもリモートの BNU システムの名前を出力することができます。

直接接続用の Devices ファイルの編集

直接接続用の Devices ファイルを編集するには、/etc/uucp/Devices ファイルを編集するための root 権限、または /etc/uucp/Sysfiles ファイル内で Devices ファイルとして指定した別のファイルを編集するための root 権限を持っている必要があります。

ポートとリモート・システムを指定して直接接続を設定するには、以下のようにエントリーを作成します。

1. エントリーの 2 行目の「**Type (タイプ)**」フィールドに、直接回線を介してローカル・コンピューターを接続したいリモート・システムの名前を入力します。
2. エントリーの両方の行の「**Line (行)**」フィールドに、サイトで使用する直接接続に適したデバイス名を入力します。
3. エントリーの両方の行の「**Line2 (行)**」フィールドに、プレースホルダーを表すハイフン (-) を入力します。
4. エントリーの両方の行の「**Speed (速度)**」フィールドに、サイトで使用する直接接続に適した送信速度を入力します。
5. エントリーの両方の行の「**Dialer-Token Pairs**」フィールドに、direct (すべて小文字) と入力します。

   例えば、次のとおりです。

```
type device - speed direct
```

ローカル・システムをリモート・システムに直接接続する各デバイスをリストし終わるまで、Devices ファイルへのエントリーの追加を続行します。

相手固定非同期シリアル接続を使用する 2 つのシステム間の直接接続を設定するには、以下のように 1 行のエントリーを作成します。

1.「**Type (タイプ)**」フィールドに、リモート・システム名を入力します。
2.「**Line (回線)**」フィールドに tty デバイスの名前を入力します。

3.「**Line2 (行)**」フィールドに、プレースホルダーを表すハイフン (-) を入力します。
4. ご使用のサイトで使用する直接接続に適した伝送速度を、「**Class (クラス)**」フィールドに入力します。
5.「**Dialer-Token Pairs (ダイヤラー・トークン・ペア)**」フィールドに direct (すべて小文字) と入力します。
   例えば、次のとおりです。

```
type device - speed direct
```

ローカル・システムをリモート・システムに接続する各直接デバイスをリストし終わるまで、Devices ファイルへのエントリーの追加を続行します。

自動ダイヤラー接続用の Devices ファイルの編集
/etc/uucp/Devices ファイルを編集する場合は、以下の手順を実行します。

/etc/uucp/Devices ファイルまたは /etc/uucp/Sysfiles ファイル内に Devices ファイルとして指定した別のファイルを編集するには、root 権限が必要です。

電話接続エントリー内で、「**Type (タイプ)**」フィールドを自動コール装置 (ACU) として指定します。 電話回線を介して確立されるすべてのリモート接続内で、「**Type (タイプ)**」フィールドのエントリーとして ACU と入力します。 「Devices (デバイス)」ファイル・エントリーを自動ダイヤラー接続用にセットアップするには、モデムごとに 1 行のエントリーを作成します。

1.「**Type (タイプ)**」 フィールドに、ACU と入力します。
2.「**Line (回線)**」フィールドに、モデムに接続したデバイスの名前を入力します。
3. 自動ダイヤラーが標準の 801 ダイヤラーでなければ、「**Line2 (回線 2)**」フィールドに、プレースホルダーとしてハイフン (-) を入力します。自動ダイヤラーが標準 801 ダイヤラーの場合は、801 と入力します。
4.「**Speed (速度)**」フィールドに、使用するモデムと回線に適したボー・レートを入力するか、使用するモデムのクラス (例えば、D2400) を入力します。ボー・レートの値は、モデムに応じて、300、1200、2400、またはそれより大きな値とすることができます。

   注 : 指定した複数のボー・レートでモデムを使用できる場合は、各レートごとに Devices ファイル内に別々のエントリーを作成します。モデムをどのボー・レートでも使用できる場合は、「**Speed (速度)**」フィールドに「Any」という語を入力します。
5.「**Dialer-Token Pair (ダイヤラー・トークン・ペア)**」フィールドの「**Dialer (ダイヤラー)**」フィールド・エントリーとして、モデムの名前を入力します。

   /etc/uucp/Sysfiles ファイル内に指定した /etc/uucp/Systems ファイルまたは別の Systems ファイルに完全な電話番号を入れたい場合は、「**Token (トークン)**」フィールドをブランクにしておきます。ブランクは、デフォルトの ¥D トークンを使用することを BNU プログラムに指示します。/etc/uucp/Dialcodes ファイル内に指定したダイヤル・コードの省略形を使用したい場合は、「¥T」トークンを入力します。

   例えば、次のとおりです。

```
type line - speed dialer - token pair
```

電話回線とモデムを使用するローカル・システムとリモート・システムの間の各接続をリストし終わるまで、引き続き Devices ファイルにエントリーを追加します。

TCP/IP 用のデバイス・ファイルの編集
/etc/uucp/Devices ファイルを編集する場合は、以下の手順を実行します。

/etc/uucp/Devices ファイルまたは /etc/uucp/Sysfiles ファイル内に Devices ファイルとして指定した別のファイルを編集するには、root 権限が必要です。

ご使用のサイトでシステム 間の接続に TCP/IP を使用している場合は、関連する TCP/IP エントリーを Devices ファイル内に組み込んでください。ファイルを TCP/IP 用にセットアップするには、Devices ファイルに次の行を入力します。

```
TCP - - - TCP
```

例: TCP/IP 接続用の BNU 構成
この例のグループでは、BNU を TCP/IP 接続用に構成します。

以下のファイルは、システム zeus とシステム hera の間の TCP/IP 接続用にセットアップされます。ここで、zeus はローカル・システムで hera はリモート・システムです。

ローカル・システム・ファイルの *TCP/IP* 接続エントリー用の *BNU* ファイル
以下の BNU ファイルは、ローカル・システム zeus 上のエントリーです。

- **Systems ファイル:** システム zeus 上の Systems ファイルには、zeus がシステム hera に接続できるように次のエントリーが入っています。

  ```
  hera Any TCP,t - - in:--in: uzeus word: birthday
  ```

  この例は、システム zeus が、システム hera との通信に **t** プロトコルを使用し、hera を常時呼び出せることを指定します。システム zeus は、uzeus としてパスワード birthday を使用し、システム hera にログインします。

  注 : **t** プロトコルは **TCP** をサポートします。 したがって、TCP/IP 接続を介した BNU 通信には、必ず **t** プロトコルを使用してください。 ただし、**t** プロトコルは、「**Type (タイプ)**」フィールドが ACU (自動コール装置) である場合、またはモデム接続を使用している場合は使用できません。

  BNU は、Systems ファイル内の「**Type (タイプ)**」フィールドと「**Class (クラス)**」フィールドを使用して、接続に適したデバイスを見つけます。 Devices ファイル内でタイプ TCP のエントリーを調べます。
- **Devices ファイル:** システム zeus 上の **uucico** デーモンが使用する Devices ファイルには、TCP/IP 接続用に次のエントリーが含まれています。

  ```
  TCP  -  -  -  TCP
  ```

  デバイス・タイプは、TCP なので、*Class*、*Line*、または *Line2* の各エントリーはありません。 *Dialer* も TCP として指定されています。 BNU は、Dialers ファイル内で TCP エントリーを探します。
- **Dialers ファイル:** システム zeus 上の **uucico** デーモンが使用する Dialers ファイルには、次のような TCP/IP エントリーが含まれています。

  ```
  TCP
  ```

  このエントリーは、ダイヤラーの構成が必要でないことを指定します。

  注 : TCP/IP 接続では、ダイヤラーの構成はまったく必要ありません。
- **Permissions ファイル:** システム zeus 上の Permissions ファイルには、システム hera に対してシステム zeus へのアクセスを認可する以下のエントリーが含まれています。

  ```
  LOGNAME=uhera SENDFILES=yes REQUEST=yes ¥
  MACHINE=zeus:hera VALIDATE=uhera ¥
  READ=/var/spool/uucppublic:/home/hera ¥
  WRITE=/var/spool/uucppublic:/home/hera COMMANDS=ALL
  ```

  LOGNAME エントリーと MACHINE エントリーを組み合わせることにより、システム zeus とシステム hera が接続されたとき、システム hera に以下の許可が提供されます。

  - システム hera は、コールを開始したシステムに関係なく、ファイルの要求と送信ができます。
  - システム hera は、システム zeus 上で公開ディレクトリーと /home/hera ディレクトリーに対して読み取りと書き込みを行うことができます。
  - システム hera は、システム zeus 上ですべてのコマンドを実行できます。
  - システム hera は、システム zeus にユーザー uhera としてログインする必要があります。また、システム hera は、BNU トランザクションに他のログイン ID を使用することはできません。

注 : 与えられる許可は、どのシステムがコールを開始するかに関係なく同じであるため、前記の LOGNAME エントリーと MACHINE エントリーは組み合わされています。 システム hera とシステム zeus で許可が同じでない場合、LOGNAME エントリーと MACHINE エントリーは以下のようになります。

```
LOGNAME=uhera VALIDATE=hera SENDFILES=yes REQUEST=yes ¥
READ=/var/spool/uucppublic:/home/hera ¥
WRITE=/var/spool/uucppublic:/home/hera

MACHINE=zeus:hera REQUEST=yes COMMANDS=ALL¥
READ=/var/spool/uucppublic:/home/hera ¥
WRITE=/var/spool/uucppublic:/home/hera
```

リモート・システム・ファイルの *TCP/IP* 接続エントリー用の *BNU* ファイル

以下のファイルは、リモート・システム hera 上にあります。

- **Systems ファイル:** システム hera 上の Systems ファイルには、hera がシステム zeus に接続できるように次のエントリーが入っています。

  ```
  zeus Any TCP,t - - ogin:--ogin: uhera ord: lightning
  ```

  この例は、システム hera が、システム zeus との通信に **t** プロトコルを使用し、zeus を常時呼び出せることを指定します。システム hera は、パスワード lightning を使用してシステム zeus に、ユーザー uhera としてログインします。 この場合も、BNU は Devices ファイルにタイプ TCP のエントリーが入っているかどうかを検査します。

  注 : **t** プロトコルは **TCP** をサポートします。 したがって、TCP/IP 接続を介した BNU 通信には、必ず **t** プロトコルを使用してください。 ただし、**t** プロトコルは、「*Type (タイプ)*」フィールドが ACU である場合、またはモデム接続を使用している場合は使用できません。

- **Devices ファイル:** システム hera 上の **uucico** デーモンが使用する Devices ファイルには、TCP/IP 接続用に次のエントリーが含まれています。

  ```
  TCP  -  -  -  TCP
  ```

  デバイス・タイプは TCP なので、*Type*、*Line*、または *Line2* の各エントリーはありません。 *Dialer* も TCP として指定されています。 BNU は、Dialers ファイル内で TCP エントリーを探します。

- **Dialers ファイル**: システム hera 上の **uucico** デーモンが使用する Dialers ファイルには、次のような TCP/IP エントリーが含まれています。

  ```
  TCP
  ```

  このエントリーは、ダイヤラーの構成が必要でないことを指定します。

  注 : TCP/IP 接続では、ダイヤラーの構成はまったく必要ありません。

- **Permissions ファイル:** システム hera 上の Permissions ファイルには、システム zeus に対してシステム hera へのアクセスを認可する次のエントリーが含まれています。

  ```
  LOGNAME=uzeus SENDFILES=yes REQUEST=yes ¥
  MACHINE=hera:zeus VALIDATE=zeus COMMANDS=rmail:who:uucp
  ```

  LOGNAME エントリーと MACHINE エントリーを組み合わせることにより、システム zeus とシステム hera が接続されたとき、システム zeus に以下の許可が提供されます。

  – システム zeus は、コールを開始したシステムに関係なく、ファイルの要求と送信ができます。
  – システム zeus は、公開ディレクトリーに対してのみ読み取りと書き込みを実行できます (デフォルト)。
  – システム zeus は、**rmail**、**who**、および **uucp** コマンドのみを実行できます。
  – システム zeus は、システム hera にユーザー uzeus としてログインする必要があります。また、システム zeus は BNU トランザクションに他のログイン ID を使用することはできません。

注 : システム hera とシステム zeus で許可が同じでない場合、LOGNAME エントリーと MACHINE エントリーは以下のようになります。

```
LOGNAME=uzeus VALIDATE=zeus SENDFILES=yes REQUEST=yes
MACHINE=hera:zeus COMMANDS=rmail:who:uucp REQUEST=yes
```

例: 電話接続用の BNU 構成

これらのファイル例は、モデムを使用してシステム venus とシステム merlin を電話回線で接続するためにセットアップされます。

システム venus はローカル・システムで、システム merlin はリモート・システムです。

どちらのシステムでも、デバイス tty1 は 1200 ボーで Hayes モデムに接続されています。システム venus がシステム merlin へのログインに使用するログイン ID は uvenus であり、それに関連したパスワードは mirror です。 システム merlin がシステム venus へのログインに使用するログイン ID は umerlin であり、それに関連したパスワードは oaktree です。 venus に接続しているモデムの電話番号は 9=3251436 であり、merlin のモデムの電話番号は 9=4458784 です。どちらのコンピューターでも、部分的な電話番号がそれぞれの Systems ファイルに含まれ、ダイヤル・コードがそれぞれの Dialcodes ファイルに含まれています。

以下のファイル例は、システム venus とシステム merlin を接続するためにセットアップされます。

- **Systems ファイル:** システム venus 上の Systems ファイルには、電話番号やダイヤル接頭部など、システム merlin に関する次のエントリーが入っています。

  ```
  merlin Any ACU 1200 local8784 "" in:--in: uvenus word: mirror
  ```

  システム venus は、システム merlin をいつでもコールできます。そのために、ACU デバイスを 1200 ボーで使用し、uvenus としてパスワード mirror を使用してログインします。 電話番号は、Dialcodes ファイル内のコード local に基づいて展開され、使用されるデバイスは、*Type* エントリーと *Class* エントリーに基づいて判別されます。 BNU は、Devices ファイルにタイプが ACU でクラスが 1200 のデバイスがないかどうかを検査します。
- **Dialcodes ファイル:** システム venus 上の Dialcodes ファイルには、Systems ファイル内の番号とともに使用する次のダイヤル・コード接頭部が入っています。

  ```
  local 9=445
  ```

  このようなコードがあると、Systems ファイル内のシステム merlin の電話番号が 9=4458784 に展開されます。
- **Devices ファイル:** システム venus 上の Devices ファイルには、システム merlin に接続するための次のエントリーが入っています。

  ```
  ACU  tty1  -  1200  hayes ¥T
  ```

  使用されるポートは tty1 であり、「*Dialer-Token Pairs*」フィールド内の *Dialer* エントリーは hayes です。 *Token* エントリーの T は、電話番号が Dialcodes ファイルのコードを使用して展開されることを示します。 BNU は Dialers ファイルに hayes ダイヤラー・タイプがないかどうかを検査します。
- **Dialers ファイル:** システム venus 上の **uucico** デーモンが使用する Dialers ファイルには、hayes のモデム用の次のエントリーが含まれています。

  ```
  hayes =,-, "" ¥dAT¥r¥c OK ¥pATDT¥T¥r¥c CONNECT
  ```

  注 : 送信予期文字は、Dialers ファイルの形式で定義されています。
- **Permissions ファイル:** システム venus 上の Permissions ファイルには、以下のエントリーが入っています。これらのエントリーは、システム merlin がシステム venus との **uucico** トランザクションおよび **uuxqt** トランザクションを実行する方法を指定します。

```
LOGNAME=umerlin REQUEST=yes SENDFILES=yes ¥
READ=/var/spool/uucppublic:/home/merlin ¥
WRITE=/var/spool/uucppublic:/home/merlin
MACHINE=venus:merlin VALIDATE=umerlin REQUEST=yes SENDFILES=yes    ¥
COMMANDS=ALL ¥
READ=/var/spool/uucppublic:/home/merlin ¥
WRITE=/var/spool/uucppublic:/home/merlin
```

システム merlin は、システム venus に umerlin としてログインします。これは、システム merlin に固有のログインです。 システム merlin は、コールを開始したシステムに関係なく、ファイルの要求と送信ができます。また、システム merlin は、システム venus 上で /var/spool/uucppublic ディレクトリーと /home/merlin ディレクトリーに対して読み取りと書き込みを行うことができます。 システム merlin は、システム venus 上でデフォルト・コマンド・セットのすべてのコマンドを発行できます。

ローカル・システムの電話接続エントリーを含む *BNU* ファイル

以下のファイルには、ローカル・システム venus 上の電話接続エントリーが含まれています。

- **Systems ファイル:** システム venus 上の Systems ファイルには、電話番号やダイヤル接頭部など、システム merlin に関する次のエントリーが入っています。

```
merlin Any ACU 1200 local8784 "" in:--in: uvenus word: mirror
```

システム venus は、システム merlin をいつでもコールできます。そのために、ACU デバイスを 1200 ボーで使用し、ユーザー uvenus としてパスワード mirror を使用してログインします。 電話番号は、Dialcodes ファイル内のコード local に基づいて展開され、使用されるデバイスは、*Type* エントリーと *Class* エントリーに基づいて判別されます。 BNU は、Devices ファイルにタイプが ACU でクラスが 1200 のデバイスがないかどうかを検査します。

- **Dialcodes ファイル:** システム venus 上の Dialcodes ファイルには、Systems ファイル内の番号とともに使用する次のダイヤル・コード接頭部が入っています。

```
local 9=445
```

このようなコードがあると、Systems ファイル内のシステム merlin の電話番号が 9=4458784 に展開されます。

- **Devices ファイル:** システム venus 上の Devices ファイルには、システム merlin に接続するための次のエントリーが入っています。

```
ACU  tty1  -  1200  hayes ¥T
```

使用されるポートは tty1 であり、「*Dialer-Token Pairs*」フィールド内の **Dialer** エントリーは hayes です。 *Token* エントリーの T は、電話番号が Dialcodes ファイルのコードを使用して展開されることを示します。 BNU は、Dialers ファイルに hayes ダイヤラー・タイプのエントリーがないかどうかを検査します。

- **Dialers ファイル:** システム venus 上の **uucico** デーモンが使用する Dialers ファイルには、hayes のモデム用の次のエントリーが含まれています。

```
hayes  =,-,  ""  ¥dAT¥r¥c OK ¥pATDT¥T¥r¥c CONNECT
```

注: 送信予期文字は、Dialers ファイルの形式で定義されています。

- **Permissions ファイル:** システム venus 上の Permissions ファイルには、以下のエントリーが入っています。これらのエントリーは、システム merlin がシステム venus との **uucico** トランザクションおよび **uuxqt** トランザクションを実行する方法を指定します。

```
LOGNAME=umerlin REQUEST=yes SENDFILES=yes ¥
READ=/var/spool/uucppublic:/home/merlin ¥
WRITE=/var/spool/uucppublic:/home/merlin
MACHINE=venus:merlin VALIDATE=umerlin REQUEST=yes SENDFILES=yes    ¥
COMMANDS=ALL ¥
```

```
READ=/var/spool/uucppublic:/home/merlin ¥
WRITE=/var/spool/uucppublic:/home/merlin
```

システム merlin は、システム venus に umerlin としてログインします。これは、システム merlin に固有のログインです。 システム merlin は、コールを開始したシステムに関係なく、ファイルの要求と送信ができます。また、システム merlin は、システム venus 上で /var/spool/uucppublic ディレクトリーと /home/merlin ディレクトリーに対して読み取りと書き込みを行うことができます。 システム merlin は、システム venus 上でデフォルト・コマンド・セットのすべてのコマンドを発行できます。

リモート・システムの電話接続エントリーを含む *BNU* ファイル

以下のファイルには、リモート・システム merlin 上の電話接続エントリーが含まれています。

- **Systems ファイル:** システム merlin 上の Systems ファイルには、電話番号やダイヤル接頭部など、システム venus に関する次のエントリーが入っています。

```
venus Any ACU 1200 intown4362 "" in:--in: umerlin word: oaktree
```

システム merlin は、システム venus をいつでもコールすることができます。そのために、ACU デバイスを 1200 ボーで使用し、ユーザー umerlin としてパスワード oaktree を使用してログインします。電話番号は、Dialcodes ファイル内のコード intown に基づいて展開され、使用されるデバイスは、*Type* エントリーと *Class* エントリーに基づいて判別されます。 BNU は、Devices ファイルにタイプが ACU でクラスが 1200 のデバイスがないかどうかを検査します。

- **Dialcodes ファイル:** システム merlin 上の Dialcodes ファイルには、Systems ファイル内の番号とともに使用する次のダイヤル・コード接頭部が入っています。

```
intown 9=325
```

したがって、システム venus に到達するために展開された電話番号は、9=3254362 になります。

- **Devices ファイル:** システム merlin 上の Devices ファイルには、システム venus に接続するための次のエントリーが入っています。

```
ACU  tty1  -  1200  hayes ¥T
```

ACU は、ポート tty1 に接続され、ダイヤラーは hayes です。 電話番号は、Dialcodes ファイル内の情報を使用して展開されます。 BNU は Dialers ファイルに hayes モデムのエントリーがないかどうかを検査します。

- **Dialers ファイル:** システム merlin 上の **uucico** デーモンが使用する Dialers ファイルには、そのモデム用の次のエントリーが含まれています。

```
hayes  =,-,  ""  ¥dAT¥r¥c OK ¥pATDT¥T¥r¥c CONNECT
```

- **Permissions ファイル:** システム merlin 上の Permissions ファイルには、システム venus に対してシステム merlin へのアクセスを認可する以下のエントリーが含まれています。

```
LOGNAME=uvenus SENDFILES=call REQUEST=no ¥
WRITE=/var/spool/uucppublic:/home/venus ¥
READ=/var/spool/uucppublic:/home/venus
MACHINE=merlin:venus VALIDATE=uvenus  ¥
READ=/ WRITE=/ COMMANDS=ALL REQUEST=yes ¥
NOREAD=/etc/uucp:/usr/etc/secure ¥
NOWRITE=/etc/uucp:/usr/etc/secure
```

例: 直接接続用の BNU 構成

以下のファイル例は、システム zeus とシステム hera の間の直接接続用にセットアップされます。ここで、zeus はローカル・システムで hera はリモート・システムです。

システム zeus 上の直接デバイスは、tty5 です。システム hera では、直接デバイスは tty1 です。 接続の速度は、1200 bps です。 システム zeus のシステム hera に対するログイン ID は uzeus であり、それ

に関連するパスワードは thunder です。 システム hera のシステム zeus に対するログイン ID は uhera であり、それに関連するパスワードは portent です。

ローカル・システム・ファイルの直接接続を含む *BNU* ファイル

以下のファイルには、ローカル・システム zeus 上の電話接続エントリーが含まれています。

- **Systems ファイル:** システム zeus 上の Systems ファイルには、リモート・システム hera 用の次のエントリーが入っています。

```
hera Any hera 1200 - "" ¥r¥d¥r¥d¥r in:--in: uzeus word:  thunder
```

このエントリーは、システム hera がシステム zeus にいつでもログインできるように指定します。そのために、Devices ファイル内に指定した直接接続を使用します。 Devices ファイル内でこのエントリーを見つけるために、BNU は Systems エントリーの 3 番目と 4 番目のフィールドを使用します。 したがって、BNU は、Devices ファイルに *Type* が hera で *Class* が 1200 のエントリーがあるかどうかを検査します。 システム zeus は、システム hera に、ユーザー uzeus としてパスワード thunder を使用してログインします。

- **Devices ファイル:** システム zeus 上の Devices ファイルには、リモート・システム hera に接続するための次のエントリーが入っています。

```
hera    tty5  -  1200  direct
```

このエントリーは、システム zeus がデバイス tty5 を 1200 bps で使用して、システム hera と通信することを指定します。 両方の「*Dialer-Token Pairs*」フィールド内の **Dialer** が direct であることに注意してください。 システム hera に接続するとき、BNU は Dialers ファイルに direct エントリーがあるかどうかを検査します。

- **Dialers ファイル:** システム zeus 上の Dialers ファイルには、直接接続用に次のエントリーが入っています。

```
direct
```

このエントリーは、直接接続にハンドシェークが必要でないことを指定します。

- **Permissions ファイル:** ローカル・システム zeus 上の Permissions ファイルには、次のエントリーが入っています。このエントリーは、リモート・システム hera がシステム zeus との **uucico** トランザクションおよび **uuxqt** トランザクションを実行できる方法を指定します。

```
LOGNAME=uhera MACHINE=hera VALIDATE=uhera REQUEST=yes ¥
SENDFILES=yes MACHINE=zeus READ=/ WRITE=/ COMMANDS=ALL
```

このエントリーは、システム hera が uhera としてログインすることを指定します。 VALIDATE=uhera オプションが指定されているので、システム hera は、他のログイン ID を指定してシステム zeus にログインできず、他のリモート・システムが uhera の ID を使用することもできません。 システム hera は、システム zeus 上ですべてのディレクトリーに対して読み取りと書き込みを行うことができ、どのシステムがコールを開始したかに関係なく、ファイルの送信と要求ができます。 システム hera は、システム zeus 上ですべてのコマンドを開始することもできます。

注: どのシステムが接続を開始したかに関係なく、与えられる許可は同じなので、LOGNAME エントリーと MACHINE エントリーは組み合わされています。 システム hera とシステム zeus で許可が同じでない場合、LOGNAME エントリーと MACHINE エントリーは以下のようになります。

```
LOGNAME=uhera REQUEST=yes SENDFILES=yes READ=/ WRITE=/
MACHINE=zeus:hera VALIDATE=uhera READ=/ WRITE=/ REQUEST=yes ¥
COMMANDS=ALL
```

重要: この例で与える許可は、リモート・システム上のすべてのユーザーにローカル・システムのログイン ID を与えることに相当します。 そのような自由な許可は、セキュリティーを危険にさらす可能性があるので、同じサイトの十分に信頼できるリモート・システムにのみ与えられます。

リモート・システム・ファイルの直接接続を含む *BNU* ファイル

以下のファイルには、リモート・システム hera 上の電話接続エントリーが含まれています。

- **Systems ファイル:** システム hera 上の Systems ファイルには、システム zeus 用の次のエントリーが入っています。

```
zeus Any zeus 1200 - "" ¥r¥d¥r¥d¥r in:--in: uhera word: portent
```

このエントリーは、システム hera がシステム zeus にいつでもログインできるように指定します。そのために、Devices ファイル内に指定した直接接続を使用します。 Devices ファイル内でこのエントリーを見つけるために、BNU は Systems エントリーの 3 番目と 4 番目のフィールドを使用します。 したがって、BNU は、Devices ファイルに **Type** の値が zeus で **Class** の値が 1200 のエントリーがあるかどうかを検査します。 システム hera は、パスワード portent を使用してシステム zeus にユーザー uhera としてログインします。

- **Devices ファイル:** システム hera 上の Devices ファイルには、システム zeus との通信用に次のエントリーが入っています。

```
zeus    tty1  -  1200  direct
```

このエントリーは、システム hera がデバイス tty1 を 1200 bps で使用して、システム zeus と通信することを指定します。*Dialer* が direct として指定されているので、BNU は、Dialers ファイルに direct エントリーがあるかどうかを検査します。

- **Dialers ファイル:** システム hera 上の Dialers ファイルには、直接接続用に次のエントリーが入っています。

```
direct
```

このエントリーは、直接接続でダイヤラーの構成が必要ないことを指定します。

- **Permissions ファイル:** システム hera 上の Permissions ファイルには、以下のエントリーが入っています。これらのエントリーは、zeus がシステム hera との **uucico** トランザクションおよび **uuxqt** トランザクションを実行する方法を指定します。

```
LOGNAME=uzeus REQUEST=yes SENDFILES=yes READ=/ WRITE=/
MACHINE=hera:zeus VALIDATE=uzeus REQUEST=yes COMMANDS=ALL READ=/¥
WRITE=/
```

これらのエントリーは、システム zeus が uzeus としてシステム hera にログインすることを指定します。 VALIDATE=uzeus パラメーターが指定されているので、システム zeus は、他のログイン ID を使用してシステム hera にログインすることはできず、他のリモート・システムが uzeus の ID を使用することもできません。 システム zeus は、システム hera 上のすべてのディレクトリーに対して読み取りと書き込みを行うことができ、どのシステムがコールを開始したかに関係なくファイルの送信と要求ができます。 システム zeus は、システム hera 上ですべてのコマンドを開始することもできます。

重要: 上記の例のすべての許可を提供することは、リモート・システム上のすべてのユーザーにローカル・システムのログイン ID を与えることに相当します。そのような自由な許可は、セキュリティーを危険にさらす可能性があるので、同じサイトのリモート・システムにのみ与えられます。

BNU の保守

BNU がシステム上で正しく機能するためには、保守する必要があります。

BNU を保守するには、次のことを行います。

- ログ・ファイルの定期的な読み取りと除去を行います。
- **uuq** コマンドと **uustat** コマンドを使用して BNU キューを検査し、各ジョブがリモート・システムに正常に転送されていることを確認します。

- 自動コマンドをスケジュールしてリモート・システムのジョブをポーリングし、転送されなかったファイルをユーザーに戻し、BNU の状況に関する定期的メッセージをローカル・システムに送信するようにします。
- 構成ファイルを定期的に更新して、システム内の変更を反映させます。

また、ローカル・システムの構成に影響のある変更がリモート・システム上で行われていないかどうかを、リモート・システムの管理者に随時確認することも必要です。 例えば、リモート・システム venus の管理者がローカル・システムのパスワードを変更した場合は、/etc/uucp/Systems ファイル (または /etc/uucp/Sysfiles によって指定されている適切な Systems ファイル) にその新しいパスワードを設定してからでなければ、ローカル・システムはシステム venus にログインできません。

BNU ログ・ファイル

BNU は、ログ・ファイルとエラー・ファイルを作成して、自分が行った処理内容を記録します。

システム上のストレージがいっぱいにならないよう、これらのファイルを定期的に検査して除去する必要があります。 BNU には、次のようなログ・ファイルのクリーンアップ用のコマンドがあります。

- uulog
- uuclean
- uucleanup
- uudemon.cleanu.

これらのコマンドは手動で実行するか、または /var/spool/cron/crontabs/uucp ファイル内のエントリーを使用して **cron** デーモンによって実行します。

.Log および *.Old* ディレクトリー内のログ・ファイル

BNU は /var/spool/uucp/.Log ディレクトリー内に個々のログ・ファイルを作成します。

BNU は、**uucp**、**uucico**、**uux**、および **uuxqt** コマンドを使用して、アクセス可能なリモート・システムごとにこれらのログ・ファイルを作成します。 システム上で BNU が使用されるたびに、BNU は各トランザクションに関する状況情報を該当するログ・ファイルに入れます。 複数の BNU プロセスが実行中のときは、システムはログ・ファイルにアクセスできません。 代わりに、.LOG 接頭部の付いた別のファイルに状況情報を入れます。

uulog コマンドは、**uucp** 要求または **uux** 要求の要約情報をユーザー別またはシステム別に表示します。 **uulog** コマンドはこの情報をファイル名として表示します。 なお、該当のログ・ファイルすべてを自動的にまとめて 1 つの 1 次ログ・ファイルにするように BNU を設定することも可能です。 この処理はログ・ファイルの圧縮 と呼ばれ、**uudemon.cleanu** コマンドを使用して実行できます。通常このコマンドは、**cron** デーモンが実行します。

cron デーモンは、**uudemon.cleanu** コマンドを実行します。 この **uudemon.cleanu** コマンドは、ローカル・システム上で **uucico** ログ・ファイルと **uuxqt** ログ・ファイルを結合して、/var/spool/uucp/.Old ディレクトリーに格納します。 同時に、このコマンドは、以前に .Old ディレクトリーに格納された古いログ・ファイルを除去します。 デフォルトでは、**uudemon.cleanu** コマンドは、生成後 2 日経っているログ・ファイルを保存します。

ストレージに問題がある場合は、ファイルを保管する日数の短縮を検討してください。 BNU のトランザクションを長期間追跡するときは、ファイルを保管する日数の延長を検討してください。 ログ・ファイルのデフォルトの保管期間を変更するには、**uudemon.cleanu** コマンドのシェル・プロシージャーを変更します。 このスクリプトは、/usr/sbin/uucp ディレクトリーに格納されており、root 権限によって変更できます。

BNU /.Admin ログ・ファイル

BNU はまた、情報を収集して /var/spool/uucp/.Admin ディレクトリーに格納します。 このディレクトリーには、errors、xferstats、Foreign、および audit ファイルが入っています。

ストレージを節約するために、これらのファイルを時々調べて除去する必要があります。 BNU は、必要に応じて、各ファイルを作成します。

別のシステムが **uucico** デーモンのデバッグ・モードをオンにしてローカル・システムに接続すると、ローカル・システム上の **uucico** デーモンがデバッグの状態でコールされます。 ローカル・システム上のデーモンが生成したデバッグ・メッセージは、audit ファイルに格納されます。 このファイルは、極端に大きくなることがあります。 audit ファイルは、頻繁に調べて除去してください。

errors ファイルには、**uucico** デーモンが検出したエラーが記録されます。 このファイルを調べることにより、BNU 作業ファイルに関する不正な許可などの問題を解決できます。

xferstats ファイルには、すべてのファイル転送の状況に関する情報が入っています。 このファイルは、時々調べて除去してください。

Foreign ファイルは、システムのセキュリティー上、重要です。 未知のシステムがローカル・システムへのログインを試行すると BNU は remote.unknown シェル・プロシージャーをコールします。 このシェル・プロシージャーは、その試行を Foreign ファイルに記録します。 Foreign ファイルには、ローカル・システムをコールしようとして拒否されたシステムの名前が入っています。 あるシステムが頻繁にコールを試行した場合は、この情報を使用して、そのシステムにアクセスを許可するかどうか検討してください。

BNU が使用するシステム単位のログ・ファイル

多くの場合、BNU プロセスではタスクの実行に root 権限が必要です。したがって /var/spool/sulog log ログ・ファイル内には BNU によって多数のエントリーが作成されます。

同様に、**cron** デーモンを使って BNU タスクのスケジューリングを行った場合も /var/spool/cron/log ファイルに多数のエントリーが作成されます。 BNU を使用するときは、これらのファイルを検査し内容を整理してください。

BNU 保守コマンド

基本ネットワーク・ユーティリティーには、BNU のアクティビティーをモニターし、BNU のディレクトリーとファイルの内容を整理するためのコマンドがいくつかあります。

BNU クリーンアップ・コマンド

BNU には、ディレクトリーの内容を整理し、送信されなかったファイルを除去するための、次の 3 つのコマンドがあります。

項目	説明
uuclean	指定した時間数より古いすべてのファイルを、BNU の管理ディレクトリーから削除します。 **uuclean** コマンドを使用して、クリーンアップを実行するディレクトリーを指定するか、または削除するファイルのタイプを指定してください。 削除したファイルをオーナーに通知するよう、このコマンドに指示することもできます。 **uuclean** コマンドは、**uucleanup** コマンドと同等のバークレー版コマンドです。
uucleanup	**uuclean** コマンドと同様の機能を実行します。 ただし、**uucleanup** コマンドは、時間数ではなく、日数 に基づいて、ファイルの経過日数を調べます。 **uucleanup** コマンドを使用して、ユーザーのファイルがまだ転送されていない場合に、そのユーザーに警告メッセージを送信し、ファイルがキューに入ったままであることを通知できます。 **uucleanup** コマンドは、指定したリモート・システムに関連したファイルの除去も行います。
uudemon.cleanu	**uulog** コマンドと **uucleanup** コマンドを発行し、BNU ログ・ファイルを圧縮して、3 日以上経過したログ・ファイルと作業ファイルを除去するシェル・プロシージャーです。 **uudemon.cleanu** コマンドは、**cron** デーモンが実行します。

BNU 状況検査コマンド

BNU には、転送とログ・ファイルの状況を調べるコマンドもあります。

項目	説明
uuq	現在 BNU ジョブ・キューに入っているジョブを表示します。 **uuq** コマンドを使用すると、指定したジョブまたはすべてのジョブの状況が表示されます。 root 権限を持つユーザーは、**uuq** コマンドを使用して、ジョブをキューから削除できます。
uustat	**uuq** コマンドで提供される情報と同様の情報を、別のフォーマットで提供します。 **uustat** コマンドを使用すると、ジョブの状況を確認し、さらに自分が所有するジョブを削除できます。 root 権限を持つユーザーは、他のユーザーに属するジョブも削除できます。
uulog	ユーザー別またはシステム別に、**uucp** 要求または **uux** 要求の要約を表示します。 **uulog** コマンドは、ファイル名を表示します。 507 ページの『BNU ログ・ファイル』を参照してください。
uupoll	リモート・システムのポーリングを強制します。 リモート・システムの作業がキューで待機中に、そのシステムが自動的にコールされるようにスケジュールされる前に、その作業を転送する必要がある場合、このコマンドは有用です。
uusnap	BNU 状況の簡潔な要約を表示します。 リモート・システムごとに、このコマンドは、転送待機中のファイルの数を表示します。 ただし、それらのファイルの待機時間は表示しません。 **uusnap** コマンドは、**uustat** コマンドと同等のバークレー版のコマンドです。

BNU シェル・プロシージャー

BNU には、次の 2 つの保守用シェル・プロシージャーが付属しています。

項目	説明
uudemon.cleanu	508 ページの『BNU クリーンアップ・コマンド』で説明しています。
uudemon.admin	**uustat** コマンドを発行します。**uustat** コマンドは、BNU ジョブの状況を報告します。 このコマンドは、uucp のログイン ID にメールとして結果を送信します。 **uudemon.admin** シェル・プロシージャーを変更してメールを他の場所へ送信することも、メール・プログラムを使用して、uucp のログイン ID に対するすべてのメールを BNU 管理担当者に転送することもできます。

これらのシェル・プロシージャーは、/usr/sbin/uucp ディレクトリーに格納されています。 これらのプロシージャーの実行内容を変更したい場合は、プロシージャーのコピーを変更してください。 これらのプロシージャーは、コマンド・ラインから実行するか、または **cron** デーモンによって実行されるようスケジュールします。

uudemon.cleanu コマンドと **uudemon.admin** コマンドを自動的に実行するには、/var/spool/cron/crontabs/uucp ファイル内にある該当の行の先頭からコメント記号 (#) を外します。

BNU パス名

基本ネットワーク・ユーティリティー (BNU) コマンドに使用されるパス名は、 さまざまな方法で入力できます。

パス名は、ルート・ディレクトリーか、またはターゲット (リモート・システム名) までのショートカット・パスです。 各パスの変形は、特定のガイドラインに準拠しています。

絶対パス名

絶対パス名はルートで始まり、 ターゲット・ディレクトリーおよびファイルまでのすべてのディレクトリーをたどります。

例えば、/etc/uucp/Devices は、ルート・ディレクトリの下の etc の中の uucp ディレクトリー内の Devices ファイルを指します。

ルート・ディレクトリーを指定するスラッシュ (/) を必ず先頭に入力してください。 また、パス内の各エレメントは必ずスラッシュ (/) で区切ってください。

相対パス名
相対パス名では、 現行ディレクトリーを基準とするディレクトリーのみがリストされます。

例えば、 現行ディレクトリーが /usr/bin であり、 ターゲット・ディレクトリーが /usr/bin/reports であれば、 相対パス名 reports を (スラッシュを先頭に付けずに) 入力します。

相対パス名は、**cu**、**uucp**、**uux** の各コマンド、 および **uuto** コマンド内のソース・ファイル名に使用できます。

> **注:** 相対パス名はすべての BNU コマンドで機能するわけではありません。 相対パス名の使用時に問題が発生した場合は、 絶対パス名を使用してコマンドを再入力してください。

~ [option] パス名
~ [option] パス名は特定のユーザーのホーム・ディレクトリーを表します。

ティルド (~) は、特定のディレクトリーへのショートカット指定として使用できます。

例えば、~jane はユーザー jane のホーム・ディレクトリーを指します。 エントリー ~uucp、または ~ (ティルドのみ) は、リモート・システム上の BNU 共用ディレクトリーを指します。 BNU 共用ディレクトリーの絶対パス名は /var/spool/uucppublic です。

> **注:** このティルドの使用法を、BNU 内の他のティルドの使用法と混同しないでください。 ティルドは、リモート・システムへのログイン中に **cu** コマンドをローカル・システム上で実行したい時にも使用します。

system_name! パス名
system_name! パス名は、別のシステム上のファイルへのパスを識別します。

例えば、distant!/account/march は、リモート・システム distant 上の account ディレクトリー内の march ファイルを指します。

system_name!system_name! パス名
system_name!*system_name*! パス名は、複数のシステムを経由するパスを識別します。

例えば、システム near を通らなければシステム distant に到達できない場合、パス名は near!distant!/account/march となります。

システム名は感嘆符 (!) で区切ります。 複数システムのパス名がある場合は、各エレメントをスラッシュ (/) で区切るという規則は適用されません。 ただし、ディレクトリーおよびファイルが明記される終端のシステムの場合は、この規則が適用されます。

> **注:** Bourne シェルを使用する場合は、システム名を感嘆符 (!) で区切ります。 C または Korn シェルで BNU を使用する場合には、感嘆符の前に円記号 (¥) を付けます。 この円記号は、次の文字を特殊文字ではなく字句通りに解釈するために必要なエスケープ文字です。

BNU デーモン

BNU ソフトウェアには、/usr/sbin/uucp ディレクトリーに次の 4 つのデーモンが格納されています。

項目	説明
uucico	ファイル転送を容易にします (511 ページの『uucico デーモン』を参照)。
uusched	ローカル・スプール・ディレクトリー内のキューに入れられているファイルの作業要求に関するスケジューリングを容易にします (511 ページの『uusched デーモン』を参照)。
uuxqt	リモート・コマンドの実行を容易にします (512 ページの『uuxqt デーモン』を参照)。
uucpd	TCP/IP を使用した通信を容易にします (512 ページの『uucpd デーモン』を参照)。

uucico、**uusched**、**uuxqt** の各デーモンは、BNU 管理者が設定したスケジュールに従って **cron** デーモンにより始動されます。 また、root 権限を持つユーザーは、これらのデーモンを手動で始動することもできます。 **uucpd** デーモンは、TCP/IP の **inetd** デーモンによって始動されます。

uucico デーモン

uucico デーモンは、あるシステムから別のシステムへデータを送信するのに必要なファイルを転送します。

uucp コマンドと **uux** コマンドは、**uucico** デーモンを始動して、指定したシステムにコマンド、データ、実行ファイルを転送します。 また、**uucico** デーモンは、BNU スケジューラーである **uusched** デーモンによっても定期的に始動されます。 **uucico** デーモンは、**uusched** デーモンによって始動されると、他のシステムに接続してコマンド・ファイル内の命令を実行しようとします。

uucico デーモンはコマンド・ファイル内の命令を実行するために、最初に、/etc/uucp/Systems ファイル (または /etc/uucp/Sysfiles で指定したその他の 1 つ以上のファイル) 内でコールされるシステムを調べます。 次にこのデーモンは、Systems ファイル・エントリー内で有効なコール時間を調べます。 その時間が有効な場合、**uucico** デーモンは、「*Type (タイプ)*」フィールドと「*Class (クラス)*」フィールドを検査して、これらのフィールドに一致するデバイス用の /etc/uucp/Devices ファイル (または /etc/uucp/Sysfiles で指定したその他の 1 つ以上のファイル) にアクセスします。

uucico デーモンは、デバイスが見つかると、/var/locks ディレクトリー内で、そのデバイス用のロック・ファイルの有無を調べます。 存在する場合、このデーモンは、要求されたタイプと実行速度をもつ別のデバイスがないか調べます。

使用可能なデバイスがないときは、デーモンが Systems ファイルに戻り、リモート・システム用の別のエントリーを検索します。 別のエントリーが存在すると、デーモンはデバイスの検索処理を繰り返します。 別のエントリーが見つからない場合、デーモンはそのリモート・システム用のエントリーを /var/spool/uucp/.Status/*SystemName* ファイル内に作成し、次の要求に進みます。 コマンド・ファイルは、キューに残ります。 **uucico** デーモンは、その後、再度転送を試みます。 この動作を再試行 と呼びます。

uucico デーモンは、リモート・システムに到達すると、Systems ファイル内の命令を使用してログインします。 このため、リモート・システム上でも **uucico** デーモンのインスタンスが呼び出されます。

uucico デーモンは、各システムに 1 つずつあり、この 2 つのデーモンが共に機能して転送を行います。 コール側システム上の **uucico** デーモンはリンクを制御し、実行する要求を指定します。 リモート・システム上の **uucico** デーモンは、ローカル許可を検査して、要求が実行可能かどうかを調べます。 実行可能であれば、ファイル転送が開始されます。

コール側システム上の **uucico** デーモンは、リモート・システムに対して持っているすべての要求を転送し終わると、停止要求を送信します。 リモートの **uucico** デーモンは、コール側システムに送信するトランザクションを持っていれば、停止要求を拒否し、2 つのデーモンのロールが反転します。

注 : ローカル・システム上の /etc/uucp/Permissions ファイルまたはリモート・システム上の /etc/uucp/Permissions ファイルのどちらからも、デーモンのロール反転を禁止できます。 この場合、リモート・システムは、ローカル・システムをコールしてから、ファイルを転送する必要があります。

いずれの方向へも転送するデータが残っていなければ、2 つの **uucico** デーモンは停止します。 この場合は **uuxqt** デーモン (512 ページの『uuxqt デーモン』を参照) がコールされ、リモート・コマンド要求を実行します。

転送プロセス中に、両方のシステム上の **uucico** デーモンは BNU のログ・ファイルとエラー・ファイルにメッセージを記録します。

uusched デーモン

uusched デーモンは、ローカル・システム上のスプール・ディレクトリー内のキューに入っているファイルの転送スケジュールを作成します。

スプール・ディレクトリーは、/var/spool/uucppublic です。 **uusched** デーモンは、呼び出されるとスプール・ディレクトリー内でコマンド・ファイルを走査してファイルをランダム化し、**uucico** デーモンを始動します。 **uucico** デーモンは、ファイルを転送します。

uuxqt デーモン

uux コマンドを入力し、指定したシステム上で指定したコマンドを実行すると、**uuxqt** デーモンがそのコマンドを実行します。

uux コマンドは、必要なファイルを作成したあと、**uucico** デーモンを始動し、uucico デーモンはそれらのファイルを、指定したシステム上の共用スプール・ディレクトリーに転送します。

uuxqt デーモンは、スプール・ディレクトリー内で、接続されているすべてのシステムに関するコマンド実行要求を定期的に検索します。 要求が見つかると、**uuxqt** デーモンは、必要なファイルと許可の有無を検査します。 許可されていれば、このデーモンは指定されたコマンドを実行します。

uucpd デーモン

BNU が**伝送制御プロトコル/インターネット・プロトコル (TCP/IP)** によるリモート・コンピューターとの通信を確立するには、事前に、リモート・システム上で **uucpd** デーモンを実行可能にしておく必要があります。

uucpd デーモンは TCP/IP の **inetd** デーモンのサブサーバーであり、**inetd** デーモンによって始動されます。

デフォルトでは、`inetd.conf` ファイル内の **uucpd** デーモンはコメント化されています。 このデーモンを使用するには、コメント文字を削除し、**inetd** を再始動する必要があります。 しかし、この構成がシステム上で変更されている場合は、**inetd** デーモンを再構成して **uucpd** デーモンを始動しなければならない場合があります。

BNU のセキュリティー

ログイン、ファイル転送、コマンド入力など、いろいろな場面でリモート・システムからローカル・システムへの接続が行われます。セキュリティーを保証するための機能を用意しています。

BNU のセキュリティー機能を使うと、リモート・システムのユーザーがローカル・システムに対して実行できる操作を制限できます (当然ながら、この制限の設定はリモート・システムのユーザー側からローカル・システムに対して行うことも可能です)。 BNU は、複数のデーモンを使って必要な処理を実行し、さまざまな管理ディレクトリーを使用して必要なファイルを格納します。 また、BNU は、自分が行った処理内容の履歴をログに記録します。

BNU のセキュリティー機能は、複数のレベルで機能します。 BNU の構成時に、次の事柄を決定できます。

- システム上で BNU ファイルにアクセスできるユーザー
- システムが接続できるリモート・システム
- リモート・システム上のユーザーがローカル・システムにログインする 方法
- リモート・システム上のユーザーがローカル・システムにログイン後に実行できる操作

uucp のログイン ID

BNU をインストールすると、構成ファイル、デーモン、および多くのコマンドとシェル・プロシージャーは、すべて uucp のログイン ID に所有されます。

uucp のログイン ID は、ユーザー ID (UID) とグループ ID (GID) が共に 5 となります。**cron** デーモンは、`/var/spool/cron/crontabs/uucp` ファイルを読み取り、BNU 用の自動ジョブをスケジュールします。

通常、ユーザー uucp としてログインすることはできません。 uucp のログイン ID が所有するファイルを変更するには、root 権限を使用してログインします。

重要 : リモート・システムが uucp ログイン ID を使用してローカル・システムにログインすると、ローカル・システムのセキュリティーが 重大な危険にさらされます。 uucp の ID を使用してログインしたリモート・システムは、LOGNAME エントリー内で指定された他の許可に応じて、ローカルの `Systems` ファイルと `Permissions` ファイルを表示できるだけでなく、場合によっては変更することも可能です。 そのため、必ずリモート・システム用に他の BNU ログイン ID を作成し、BNU の管理担当者の uucp ログイン ID はローカル・システム上に保存してください。 セキュリティーを最善なものにするためには、ローカル・システムに接続する各リモート・システムには固有の UID 番号を含む固有のログイン ID を持たせる必要があります。

オペレーティング・システムには、ファイル転送用にデフォルトの nuucp ログイン ID が用意されています。

BNU のログイン ID

BNU のログイン ID 用の始動シェルは、**uucico** デーモン (/usr/sbin/uucp/uucico) です。

リモート・システムは、ローカル・システムをコールするときに、ローカル・システム上で自動的に **uucico** デーモンを始動します。 BNU のログイン ID は、uucp グループ ID が 5 になります。

リモート・システムが使用するログイン ID には、パスワードが必要です。 リモート・システムのログイン時に、新しい BNU ログイン ID に新しいパスワードの入力を求めるプロンプトがシステムに表示されないようにするには、アカウントの作成後、即座にパスワードを設定する必要があります。 そのためには、**passwd** コマンドに続けて **pwdadm** コマンドを使用します。 例えば、ログイン ID nuucp 用のパスワードを設定するには、root ユーザーとしてログインし、次のコマンドを入力します。

```
passwd nuucp
```

```
pwadm -f NOCHECK
nuucp
```

システムは、nuucp ログイン ID 用のパスワードを入力するよう求めるプロンプトを表示します。 以上の手順を完了すると、新しいパスワードの入力を求めるプロンプトが表示されなくなり、リモート・システムがログインできるようになります (バッチ型 nuucp ログイン ID の場合、パスワードの提示はありません)。

リモート・システム用のログイン ID を作成したあと、そのリモート・システムの BNU 管理者に、ローカル・システムにアクセスするためのログイン ID とパスワードを通知してください。

root 権限を持つユーザーは、BNU の管理ログイン ID を設定できます。 この ID は、root 権限を持たないユーザーに BNU の管理業務を委任したい場合に有効です。 BNU の管理ログイン ID にパスワード・セキュリティーを与え、UID と uucp グループ ID を共に 5 にしてください。管理ログイン用のログイン・シェルは、(**uucico** デーモンでなく) /usr/bin/sh プログラムにする必要があります。 BNU の管理ログインに UID として 5 を与えると、**uucp** のログイン ID と同じ特権を持つことになります。 したがって、セキュリティー上、リモート・システムが BNU 管理者としてログインできないようにしてください。

セキュリティーとシステム・ファイルおよび remote.unknown ファイル

ほとんどの BNU システム上では、/etc/uucp/Systems ファイルまたはその代替ファイル (Sysfiles ファイルで指定されている) にリストされているリモート・システムだけが、ローカル・システムにログインできます。

未知のシステムがローカル・システムをコールしようとするたびに、/usr/sbin/uucp/remote.unknown スクリプトが実行されます。 このスクリプトは、未知のシステムからのログインを拒否し、/var/spool/uucp/.Admin/Foreign ファイル内にエントリーを作成して、ログイン試行回数を記録します。

root 権限を所有しているか、または BNU 管理者であれば、remote.unknown シェル・プロシージャーを変更し、リモート・システムに関する詳細情報をログに記録したり、別のファイルにその情報を格納することができます。 例えば、このシェル・プロシージャーを変更すると、未知のシステムがログインしようとするたびにメールを BNU 管理者に送信できます。

remote.unknown シェル・プロシージャーの実行許可を取り消すと、未知のコンピューターがログインできるようになります。 その場合、/etc/uucp/Permissions ファイルに MACHINE=OTHER エントリーを追加し、未知のコンピューターに対する許可を確立する必要があります。

システムは、Systems ファイルに登録されているリモート・システムにだけ接続できます。 これにより、ローカル・システムのユーザーは未知のシステムに接続できなくなります。

セキュリティーと Permissions ファイル

Permissions ファイルを使用する場合は、以下のセキュリティー問題を考慮してください。

/etc/uucp/Permissions ファイルは、以下の内容を決定します。

- ローカル・システムにログインするためのリモート ・ログイン・ユーザー名

- ローカル・システムにログインするリモート ・システム用に承認されたコマンドと特権

/etc/uucp/Permissions ファイルには、次の 2 種類のエントリーがあります。

項目	説明
LOGNAME	ログイン名とそれに関連した特権を定義します。 リモート・システムがローカル・システムをコールしてログインしようとすると 、LOGNAME エントリーが有効になります。
MACHINE	マシン名とそれに関連した特権を定義します。 リモート・システムがローカル・システム上でコマンドを実行しようとすると、MACHINE エントリーが有効になります。

Permissions ファイル内のオプションを使用すると、リモート・システムごとに各種のセキュリティー・レベルを確立できます。 例えば、多数のリモート・システムがローカル・システム上で 1 つのログイン ID を共有する場合、VALIDATE オプションを使用して、各リモート・システムに固有のログイン ID を使用するように要求します。 SENDFILES、REQUEST、および CALLBACK オプションにより、どのシステムが制御権を持つかを指定できます。この結果、ローカル・システムは必要に応じてローカル・システムにトランザクションの管理下におくことができます。

READ、WRITE、NOREAD、および NOWRITE の各オプションは、ローカル・システム上の特定のディレクトリーへのアクセス権を定義します。 また、これらのオプションは、システム上でリモート・ユーザーがデータを置くことのできる位置も制御します。 COMMANDS オプションは、リモート・システム上のユーザーがローカル・システム上で実行できるコマンドの数を制限します。 COMMANDS=ALL オプションを使用すると、システムに密接に関連するシステムにすべての特権を与えることができます。

重要 : COMMANDS=ALL オプションを使用すると、システムをセキュリティー上、重大な危険にさらす可能性があります。

ローカル・システムとリモート・システム間の通信

リモート・システムとローカル・システム間で通信するには、リモート・システムにローカル・システムへのケーブルまたはモデム・リンクがあり、UNIX ベースのオペレーティング・システムがインストールされていて、BNU または別バージョンの UNIX 間コピー・プログラム (UUCP) が実行されている必要があります。

注: BNU を使用して UNIX 以外のシステムと通信できますが、 このような接続には追加のハードウェアまたはソフトウェアが必要となる場合があります。

BNU には、リモート・システムと通信するためのコマンドが 2 つあります。 **cu** コマンドは、 ケーブルまたは電話回線を通じてシステムに接続します。 **ct** コマンドは、 モデムを使用した電話回線によってのみシステムに接続します。

ターゲット・システムの電話番号または名前のいずれかがわかっている場合は、**cu** コマンドを使用してネットワーク間の通信を確立します。 **ct** コマンドを使用するには、ターゲット・システムの電話番号が必要です。

注: 第 3 のコマンド **tip** は、**cu** コマンドと同様に機能します。 ただし、**tip** コマンドは UUCP プログラムの Berkeley Software Distribution (BSD) バージョンのコンポーネントです。 このコンポーネントを BNU と共にインストールするには、特殊な構成が必要です。

別のシステムとのケーブルまたはモデムによる 通信

cu コマンドをローカル・システムから使用することで、次の通信タスクを実行できます。

- 指定したリモート・システムとの接続を確立する。
- リモート・システムにログインする 。
- リモート・システム上でタスクを実行する。
- 両方のシステムで並行して作業するため、相互に切り換える。

リモート・システムが同じオペレーティング・システムのもとで稼働している場合は、ローカル・システムから通常のコマンドを出すことができます。 例えば、ディレクトリーを変更し、ディレクトリー内容を

リストし、ファイルを表示し、 リモート・システム上の印刷キューにファイルを送信するコマンドを出すことができます。 ローカル・システムで使用するコマンドを出したり、リモート・コマンドおよびファイル交換を開始するには、ティルド (~) が前に付いた特殊な **cu** ローカル・コマンドを使用します。

別のシステムとのモデムによる通信

モデムによって別のシステムと通信するには、**ct** コマンドを出します。

リモート・モデムをコールするには、**ct** コマンドに続けて電話番号を入力します。 接続されると、画面にリモート・ログイン・プロンプトが表示されます。

ct コマンドは一定の条件下では便利です。 BNU **ct** コマンドの使用法の詳細は、次のセクションを参照してください。

- 515 ページの『接続されるまで 1 つの番号をダイヤルする』
- 515 ページの『接続されるまで複数の番号をダイヤルする』

接続されるまで 1 つの番号をダイヤルする

ここでは、**ct** コマンドを使用して、接続が成功するまで、または指定した時間が経過するまで、リモート・モデム番号をダイヤルし続けるための手順について説明します。

コールされるシステムで、 基本ネットワーク・ユーティリティー (BNU) または UNIX 間コピー・プログラム (UUCP) のいずれかのバージョンが稼働していること。

ローカル・システムで、コマンド・ラインに次のように入力します。

```
ct -w3 5550990
```

これにより、リモート・モデムの電話番号 555-0990 がダイヤルされます。 **-w3** のフラグと番号により、接続が成功するまで、または 3 分間経過するまで、リモート・モデムに 1 分間隔でダイヤルするよう **ct** コマンドに対して指示しています。

> **注: ct** コマンド・ラインでは、フラグの前または後ろにリモート・モデムの電話番号を入力してください。

接続されるまで複数の番号をダイヤルする

ここでは、**ct** コマンドを使用して、 接続が成功するまで、 または指定した時間が経過するまで、 複数のリモート・モデム番号をダイヤルし続けるプロシージャーを説明します。

コールされるシステムで、 基本ネットワーク・ユーティリティー (BNU) または UNIX 間コピー・プログラム (UUCP) のいずれかのバージョンが稼働していること。

ローカル・システムで、コマンド・ラインに次のように入力します。

```
ct -w6 5550990 5550991 5550992 5550993
```

これにより、リモート・モデムの電話番号 555-0990、555-0991、555-0992、および 555-0993 がダイヤルされます。 **-w6** のフラグと番号により、接続が成功するまで、または 6 分間経過するまで、リモート・モデムに 1 分間隔でダイヤルするよう **ct** コマンドに対して指示しています。

> **注: ct** コマンド・ラインでは、フラグの前または後ろにリモート・モデムの電話番号を入力してください。

ローカル・システムとリモート・システム間でのファイルの交換

システム間のファイル転送は、基本ネットワーク・ユーティリティー (BNU) の最も一般的なアプリケーションです。 BNU は **uucp**、**uusend**、**uuto**、**uupick** という 4 つのコマンドを使用して、 ローカル・システムとリモート・システム間でファイルを交換します。

uucp コマンドは、主要な BNU データ転送ユーティリティーです。 **uusend** コマンドは、BNU に組み込まれた Berkeley Software Distribution (BSD) の転送コマンドです。 **uuto** および **uupick** コマンドは、**uucp** コマンドと共に機能する特別な送受信コマンドです。

BNU コマンド **uuencode** および **uudecode** はファイル転送を援助します。 この 2 つのコマンドは、BNU のメール機能を通じて転送されたバイナリー・ファイルのエンコードとデコードを実行します。

ファイルの送受信

BNU 接続を介してファイルを送受信する際に使用するコマンドには、**uucp** コマンドと **uusend** コマンドがあります。

uucp コマンドとオプションは、ローカル・システム内、ローカル・システムとリモート・システム間、リモート・システム間でファイルを交換するのに使用します。 例えば、**uucp** オプションは、受信の終了時にファイルを入れるディレクトリーを作成するか、またはファイル転送の成否に関するメッセージをメールとして送信します。

uusend コマンドは、送信側システムに直接リンクされていないが、BNU 接続のチェーンを通じてアクセス可能なリモート・システムにファイルを送信する場合に使用します。 **uusend** には **uucp** コマンドよりもオプションが少ないですが、BSD UNIX 間コピー・プログラム (UUCP) ユーザーが使用する設定が BNU ユーティリティーに 組み込まれています。

特定のユーザーへのファイル送信

ファイルを特定のユーザーに送信するには、送信側システムおよび受信側システムで、基本ネットワーク・ユーティリティー (BNU) または UNIX 間コピー・プログラム (UUCP) のいずれかのバージョンが稼働していなければなりません。

uuto コマンドは、システム間でファイルを送信するのに使用します。 このコマンドは **uucp** コマンドの一部であり、送信側と受信側のファイル交換プロセスを簡略化します。 **uuto** コマンドは特定のユーザーにファイルを送信し、それをシステムの BNU 共用ディレクトリーのもとにあるユーザーの専用ディレクトリーに直接入れます。 そしてファイルが到着したことを受信者に通知します。 受信者は **uupick** コマンドを使用して、新しいファイルを処理します。

uuto コマンドによるファイル送信

uuto コマンドを使用してファイルを送信する場合、次のように送信したいファイル、リモート・システムの宛先、その宛先内のユーザーを指定してください。

例:

```
uuto /home/bin/file1 distant!joe
```

このコマンドはローカルの /home/bin ディレクトリーから、リモート・システム distant 上のユーザー joe に file1 を送信します。

uuto コマンドは **uucp** コマンドのもとで実行されます。 ファイルは、リモート・システムの /var/spool/uucppublic ディレクトリーに転送されます。 ファイルはリモート・システム上の /var/spool/uucppublic/receive/*user*/System ディレクトリーに置かれます。 ターゲット・ディレクトリーが存在しない場合は、ファイル交換中に作成されます。

BNU の **rmail** コマンドは、 受信側に対してファイルが到着したことを通知します。

> **注:** ローカル ・システム上のユーザーにファイルを送信する場合も、**uuto** コマンドを入力し、送信するファイル、ローカル・システムの宛先、ローカル宛先のユーザーを指定します。 以下に例を示します。
>
> ```
> uuto /home/bin/file2 near!nick
> ```
>
> これにより、ローカルの /home/bin ディレクトリーから、ローカル・システム near 上のユーザー nick に file2 が送信されます。

ファイルの受信

ファイルを受信して処理するには、送信側システムおよび受信側システムで、基本ネットワーク・ユーティリティー (BNU) または UNIX 間コピー・プログラム (UUCP) のいずれかのバージョンが稼働していなければなりません。

uupick コマンドは、**uuto** コマンドを使用して送信されたファイルを受信および操作するために使用します。 また、このコマンドには、受信側が送信されたファイルを検索し、ファイルを指定ディレクトリーに移動し、コマンドを実行し、あるいはファイルを削除できるようなファイル処理オプションがあります。

uupick コマンドによるファイルの受信

ファイルを受信するには、**uupick** コマンドを使用します。

例:

```
uupick
```

uupick コマンドは、 共用ディレクトリー内でパス名にリモート・ユーザー ID が含まれているファイルを検索します。 次に、**uupick** コマンドはリモート画面に次のようなメッセージを表示します。

```
from system base: file file1?
```

通知表示の 2 行目の ? (疑問符) は、BNU の共用ディレクトリー内のファイル処理に **uupick** オプションを使用するように受信側に求めるプロンプトです。

使用可能なすべてのオプションをリストするには、疑問符 (?) プロンプトの下の行にアスタリスク (*) を入力します。 表示、保存、終了のオプションは次の通りです。

項目	説明
p	ファイルの内容を表示します。
m *[Directory]*	*[Directory]* 変数に指定されたディレクトリーにファイルを保存します。 **m** オプションに宛先が指定されない場合、ファイルは現在の作業ディレクトリーに移動します。
q	**uupick** ファイル処理プロセスを終了します。

転送のためのエンコード・ファイルとデコード・ファイル

uuencode コマンドと **uudecode** コマンドは、モデムによる送信用にファイルを準備するために使用します。

これらのコマンドは、対として機能します。 **uuencode** コマンドはバイナリー・ファイルを ASCII ファイルに変換します。 これらのファイルは、メール機能でリモート・システムに送信できます。

uudecode コマンドを使用すると、受信側のユーザーは ASCII にエンコードされたファイルを元のバイナリー・フォーマットに変換できます。

コマンドおよびファイル交換の状況報告

ファイル交換の状況報告を表示するには、**uusnap**、**uuq**、および **uustat** の各コマンドを使用できます。

BNU によって接続されたシステムの状況の表示

uusnap コマンドは BNU によって接続されたすべてのシステムに関する情報のテーブルを表示します。

このテーブルでは 1 システムが 1 行で表され、システムのキュー内に保持されているコマンド・ファイル、データ・ファイル、 および実行されたリモート・コマンドの名前と数が表示されます。 各行の最終項目は状況メッセージです。 このメッセージは、正常な BNU 接続、または BNU がリンクを確立できなかった理由のいずれかを示します。

uusnap コマンドを参照してください。

BNU ジョブ・キューの表示

uuq コマンドは BNU ジョブ・キュー内のエントリーをリストします。

このリストのフォーマットは **ls** コマンドが表示するフォーマットに似ています。 各エントリーの表示内容は、 ジョブ番号と要約 (システム名、システムのジョブの数、および合計送信バイト数) であり、 これらは同一行に表示されます。 root 権限を持っているユーザーは、**uuq** コマンドを使用してキューに入れられた特定のジョブをそのジョブ番号で識別できます。

BNU 操作の状況

uustat コマンドは BNU システム内の特定のコマンドまたはファイル交換の状況を提供します。

フラグ・オプションを指定せずに **uustat** コマンドを入力すると、現行ユーザーから要求されたジョブの、次に示す情報がジョブごとに 1 行ずつ表示されます。

- ジョブ ID 番号
- 日時
- 状況 (送信または受信)
- システム名
- コマンドを出したユーザーのユーザー ID
- ジョブ・ファイルのサイズと名前

複数のフラグを指定して **uustat** コマンドを実行すると、キュー内のすべてのユーザーによるすべてのジョブ、またはネットワーク上の他のシステムが要求したジョブに関するレポートを作成できます。

uustat コマンドを使用すると、 ユーザーはリモート・コンピューター上で実行するためにキューに入れられたジョブに関して、 一定の制御を行うことができます。 他のシステムへの BNU 接続の状況を検査し、ファイルとコマンド交換をトラックできます。 例えば、**uucp** コマンドによって開始されたコピー要求を取り消すことができます。

uustat コマンドを参照してください。

ローカル・システムとリモート・システム間でのコマンドの交換

基本ネットワーク・ユーティリティー (BNU) を使用すると、 ユーザーはローカル・システムとリモート・システム間でコマンドを交換できます。

uux コマンドはリモート・システム上でコマンドを実行します。 **uupoll** コマンドはコマンド実行のタイミングを制御します。

リモート・システムでのコマンド実行要求

uux コマンドは、リモート・システムでのコマンドの実行を要求するために使用します。

uux コマンドは、そのコマンドをリモート・システムでは実行しません。 その代わりに、/var/spool/uucp において必要な制御とデータ・ファイルの準備をします。 転送を実行するために、**uucico** デーモンが起動されます。 転送が完了すると、リモート・システムの **uucico** はスプール・ディレクトリーに実行ファイルを作成します。

2 つの **uucico** デーモンが停止を合意すると、**uuxt** デーモンはスプール・ディレクトリーをスキャンして未処理の実行要求を探し、権限をチェックし、付加的な情報が必要かどうかを判断します。 次にコマンドを fork して要求を実行します。

> **注: uux** コマンドは、 指定したコマンドを実行するように構成された任意のシステム上で使用できます。 ただし、サイトの方針によりセキュリティー上の理由から、一部のコマンドの使用が制限されることがあります。 例えば、**mail** コマンドしか実行できないサイトがあります。

リモート・システム上でファイルが受信されると、**uuxqt** デーモンは指定されたコマンドをそのシステム上で実行します。 **uuxqt** デーモンはリモート・システムの共用スプール・ディレクトリーを定期的にスキャンし、**uux** 転送で受信したファイルを探します。 また、**uuxqt** デーモンは、送信されたファイルがアクセスするデータがリモート・システム上に存在するかどうかを検査します。 さらに、送信システムにデータへのアクセス権があるかどうかも検査します。 次に、**uuxqt** デーモンはコマンドを実行するか、またはコマンドを実行しなかったことを送信側システムに通知します。

BNU リモート接続のモニター

BNU リモート接続をモニターするには、以下の手順を使用します。

- BNU プログラムがシステムにインストールされている 必要があります。
- ローカル・システムとリモート ・システム間にリンク (ケーブル、モデム、または TCP/IP) が設定されていなければなりません。
- Systems ファイル、Permissions ファイル、Devices ファイル、および Dialers ファイル (場合により、Sysfiles ファイルも) などの BNU 構成ファイルが、ローカル・システムとリモート ・システムの間で通信ができるように設定されている必要があります。

  注 : BNU 構成ファイルを変更するには、root ユーザー権限が必要です。

Uutry コマンドを使用すると、そのサイトのユーザーがファイル転送上の問題を報告した場合、**uucico** デーモン・プロセスをモニターすることができます。

1. 次のように **uustat** コマンドを入力し、現行のキューに入っているすべての転送ジョブの状況を確認します。

   ```
   uustat -q
   ```

   システムは、次のような状況レポートを表示します。

   ```
   venus 3C (2) 05/09-11:02 CAN'T ACCESS DEVICE
   hera 1C 05/09-11:12 SUCCESSFUL
   merlin 2C 5/09-10:54 NO DEVICES AVAILABLE
   ```

   このレポートでは、リモート・システム venus 用の 3 つのコマンド・ファイル (C.*) がキューに 2 日間入っていたことが示されています。 この遅延については、何らかの理由が考えられます。 例えば、システム venus が保守のためにシャットダウンしていたり、モデムの電源が切れていた可能性があります。
2. より広範囲の問題解決作業を開始する前に、次のように **Uutry** コマンドを入力して、ローカル・システムがシステム venus に現在接続できるかどうか判別します。

   ```
   /usr/sbin/uucp/Uutry -r venus
   ```

   このコマンドは、デバッグ出力の最適化とデフォルトの再試行時間を無効にする命令を使用して、**uucico** デーモンを始動します。 **Uutry** コマンドは、デバッグ出力を一時ファイル /tmp/venus に送ります。
3. ローカル・システムがシステム venus への接続に成功した場合、デバッグ出力で大量の情報が表示されます。 しかし、最も重要なのは、以下のような、このスクリプトの最終行です。

   ```
   Conversation Complete: Status SUCCEEDED
   ```

   接続に成功した場合、一時的なファイル転送問題が解決されたと見なしてください。 **uustat** コマンドを再入力し、スプール・ディレクトリー内のファイルがリモート・システムに正常に転送されたことを確認します。 転送されていない場合は、520 ページの『BNU ファイル転送のモニター』 の手順に従って、ローカル・システムとリモート ・システムの間のファイル転送上の問題を調べてください。
4. ローカル・システムから目的のリモート・システムにアクセスできない場合、**Uutry** コマンドが生成するデバッグ出力情報の内容は次のようになります (出力のフォーマットの細部は、この例と異なることもあります)。

   ```
   mchFind called (venus)
   conn (venus)
   getto ret -1
   Call Failed: CAN'T ACCESS DEVICE
   exit code 101
   Conversation Complete: Status FAILED
   ```

   最初に、ローカル・システムとリモート ・システム間の物理接続を調べます。 リモート・コンピューターに電源が入っていてケーブルがすべて正しく接続されていること、該当の各システムでポートの使用

可能/使用不可が正しく設定されていることを確認します。モデムを使っている場合は、モデムが正しく機能していることを確認します。

物理接続が正しく、安全が確保されている場合は、ローカル・システムおよびリモート・システムの構成ファイルのうち該当するものをすべて検証します。

- **/etc/uucp** ディレクトリーにある Devices、Systems、および Permissions ファイル (場合により Sysfiles ファイルも) 内のエントリーが両方のシステム上で正しいことを確認します。
- モデムを使用する場合は、/etc/uucp/Dialers ファイル (または **/etc/uucp/Sysfiles** 内に指定した代替ファイル) に正しいエントリーが入っていることを確認します。 ダイヤル・コードの短縮形を使用している場合は、その短縮形が /etc/uucp/Dialcodes ファイル内に定義されていることを確認します。
- TCP/IP 接続を使用する場合は、**uucpd** デーモンをリモート・システム上で実行できること、および構成ファイルに正しい TCP エントリーが入っていることを確認します。

5. 物理接続と構成ファイルを調べたあと、**Uutry** コマンドを再入力してください。

   それでもデバッグ出力で接続の失敗が報告される場合、システム・サポート・チームのメンバーに相談しなければならないことがあります。 **Uutry** コマンドによって生成されたデバッグ出力は保管しておいてください。 これによって問題が診断しやすくなります。

リモート・システムへの印刷ファイルの転送

uux コマンドは、リモート・システムにファイルを転送して印刷するために使用します。

ファイルをリモート・システムに転送して印刷するには、以下の前提条件を満たす必要があります。

- リモート・システムへの基本ネットワーク・ユーティリティー (BNU) 接続が確立されていること。
- リモート・システム上で操作を実行する許可があること。

ローカル・システムのコマンド・ラインで、次のように入力します。

```
uux remote!/usr/bin/lpr local!filename
```

これにより、 リモート・システム上でローカル・ファイル *filename* が印刷されます。

BNU ファイル転送のモニター

以下の手順に従って、リモート・システムへのファイル転送をモニターします。

- BNU プログラムがシステムにインストールされ、使用システム用に設定されている必要があります。
- 519 ページの『BNU リモート接続のモニター』に示されている手順に従って、リモート・システムへの接続を確立します。

ファイル転送のモニターは、問題のリモート・システムへのファイル転送が何らかの原因で失敗に終わった場合に役に立ちます。 **uucico** デーモン (**Uutry** コマンドで呼び出される) によって生成されたデバッグ情報を使用すると、正常に機能していない部分を突き止めやすくなります。

Uutry コマンドを使用して、以下のようにファイル転送をモニターできます。

1. **uucp** コマンドに **-r** フラグを指定して次のように入力し、ファイルの転送を準備します。

   ```
   uucp -r test1 venus!~/test2
   ```

   -r フラグは、**uucico** デーモンを始動しないで、必要なすべての転送ファイルを作成してキューに入れるように UUCP プログラムに指示します。
2. **Uutry** コマンドに **-r** フラグを指定し、次のように入力します。デバッグ機能がオンの状態で **uucico** デーモンが始動されます。

   ```
   /usr/sbin/uucp/Uutry -r venus
   ```

   このコマンドは、**uucico** デーモンに対して、デフォルトの再試行時間を無効にしてリモート・システム venus に接続するように命令します。 このデーモンはシステム venus に接続し、ログインしてファイルを転送します。一方、**Uutry** コマンドは、**uucico** プロセスをモニターできるようにデバッグ出力

を生成します。 デバッグ出力を停止してコマンド・プロンプトに戻るには、Interrupt キー・シーケンスを押します。

また、**Uutry** コマンドは、デバッグ出力を /tmp/*SystemName* ファイルに格納します。 接続が完了する前にデバッグ出力を停止すると、この出力ファイルを参照することにより、その接続の結果を調べることができます。

スプール・ジョブの送信

uupoll コマンドは、ローカル・システムの共用スプール・ディレクトリーに格納されたジョブの転送を開始するために使用します。

uupoll コマンドはリモート・システム用の共用ディレクトリーに null ジョブを作成し、**uucico** デーモンを始動します。 これにより、**uucico** デーモンはすぐにリモート・システムに接続し、キューに入れられたジョブを転送します。

互換性のあるシステムの識別

ローカル・システムにアクセス可能なすべてのシステムのリストを表示するには、**uuname** コマンドを使用します。

例えば、コマンド・ラインに次のように入力すると、

```
uuname
```

次のようなリストが表示されます。

```
arthur
hera
merlin
zeus
```

この情報は、ファイルをコピーする前にアクセス可能なシステムの名前を判別するために使用されます。 **uuname** コマンドは、ローカル・システムの識別を確立するときにも使用します。 **uuname** コマンドは、/etc/uucp/systems ファイルを読み取ってその情報を獲得します。

tip コマンドを使用した接続済み UNIX システムとの通信

UNIX オペレーティング・システム を実行中の接続済みシステムにアクセスするには、**tip** コマンドを使用します。

tip コマンドは、基本ネットワーク・ユーティリティー (BNU) とともにインストールされ、BNU と同じ非同期接続を使用できます。

tip コマンドでは、変数、エスケープ・シグナル、およびフラグを使用して、このコマンドの動作を制御します。 フラグは、コマンド・ラインに入力できます。 リモート・システムとの接続を介してエスケープ・シグナルを使用すると、ファイル転送の開始と停止、ファイル転送方向の変更、およびサブシェルへ抜け出すことができます。

tip コマンドの変数

tip コマンドの変数は、行の終わり文字、ブレーク・シグナル、ファイル転送モードなどの設定を定義します。

変数の設定は、.tiprc ファイルを使用して実行時に初期化できます。 変数の設定は、~s エスケープ・シグナルを使用して実行中に変更することもできます。 一部の変数 (行の終わり文字など) は、個々のシステムごとに remote ファイルのシステム・エントリー内で設定できます。

tip コマンドは、phones、remote、 .tiprc の 3 つのファイルを読み取って、その変数の初期設定値を判別します。 .tiprc ファイルは、必ずユーザーのホーム・ディレクトリーに入っていなければなりません。 remote ファイルと phones ファイルの名前と位置は変更できます。 remote ファイルと phones ファイルの名前は、次の環境変数によって決定できます。

項目	説明
PHONES	ユーザーの電話ファイル名を指定します。 このファイルの名前は、有効な名前であればどのようなものでもかまいませんが、/usr/lib/phones-file ファイルのフォーマットで設定する必要があります。 デフォルトのファイルは etc/phones です。 PHONES 変数を使用してファイルを指定すると、そのファイルが /etc/phones ファイルの (追加ではなく) 代わりに使用されます。
REMOTE	ユーザーのリモート・システム定義ファイルの名前を指定します。 このファイルの名前は、有効な名前であればどのようなものでもかまいませんが、/usr/lib/remote-file ファイルのフォーマットで設定する必要があります。 デフォルトのファイルは /etc/remote です。 REMOTE 変数を使用してファイルを指定すると、そのファイルが /etc/remote ファイルの (追加ではなく) 代わりに使用されます。

環境変数を使用するときは、**tip** コマンドを実行する前に、その変数を設定します。 また、.tiprc ファイル内で **tip** コマンドの phones 変数と remote 変数を使用して、phones ファイルと remote ファイルの名前を決定する方法もあります。

注 : tip コマンドは、最後 に指定した remote ファイルまたは phones ファイルのみを読み取ります。 したがって、変数を使用して remote ファイルまたは phones ファイルを指定した場合、その新しいファイルが、以前に指定したファイルの (追加ではなく) 代わりに使用されます。

tip コマンドは、次の順に変数の設定値を使用します。

1. このコマンドは、PHONES 環境変数と REMOTE 環境変数の設定から、phones ファイルおよび remote ファイルとして使用するファイルの有無を検査します。
2. このコマンドは、.tiprc ファイルを読み取り、それに応じてすべての変数を設定します。 .tiprc ファイル内に phones 変数または remote 変数が設定されていると、その設定によって環境変数の設定がオーバーライドされます。
3. リモート・システムへの接続が開始されると、このコマンドは、そのシステムの remote ファイル・エントリーを読み取ります。 remote ファイル・エントリー内の設定によって、.tiprc ファイル内の設定がオーバーライドされます。
4. - BaudRate フラグを **tip** コマンドで使用すると、指定した速度によって、以前のボー・レートの設定がすべてオーバーライドされます。
5. ~s エスケープ・シグナルを使用して行った設定によって、以前の変数設定がすべてオーバーライドされます。

   注 : tip を使用すると .tiprc ファイルを作成することができ、このファイルを使用して、**tip** 変数の初期設定値を指定できます。 .tiprc ファイルは、ユーザーの $HOME ディレクトリーに入れる必要があります。

tip コマンドの構成ファイル

tip コマンドでリモート・システムに接続する前に、/etc/remote ファイルと /etc/phones ファイルが確立されている必要があります。

項目	説明
/etc/remote	システムに到達するために使用するポートとデバイスのタイプ、送信の開始と終了を示すために使用するシグナルなど、リモート・システムの属性を定義します。
/etc/phones	モデム回線を介してリモート・システムに接続するために使用する電話番号をリストします。

サンプルの remote ファイルと phones ファイルは、bos.net.uucp パッケージとともに配布されます。 サンプルの remote ファイルの名前は、/usr/lib/remote-file です。 サンプルの phones ファイルの名前は、/usr/lib/phones-file です。 /usr/lib/remote-file を /etc/remote にコピーし、/etc/remote を変更します。 これらのファイルのいずれかを確立するには、サンプル・ファイルを正しい名前にコピーし、サイトの必要性に合わせて変更します。

tip のユーザーは、カスタマイズした remote ファイルと phones ファイルを作成することもできます。個々の remote ファイルは、/usr/lib/remote-file ファイルのフォーマットでなければならず、remote 変数または REMOTE 環境変数を使用して指定する必要があります。 個々の phones ファイルは、/usr/lib/phones-file ファイルのフォーマットでなければならず、phones 変数または PHONES 環境変数を使用して指定する必要があります。 変数のいずれかを使用して、個々の phones ファイルまたは remote ファイルを指定すると、そのファイルが /etc/phones ファイルまたは /etc/remote ファイルの (追加ではなく) 代わりに読み取られます。

tip のユーザーは、個々の phones ファイルと remote ファイルを組み合わせて使用できます。 例えば、デフォルトの remote ファイルである /etc/remote を使用しても、個々の phones ファイルを phones 変数で指定して使用できます。

リモート・ジョブの取り消し

uustat コマンドを使用して、リモート・システムに対して出された BNU プロセスを取り消します。

リモート・ジョブを取り消すには、以下の前提条件を満たす必要があります。

- ターゲット・リモート・システムとの基本ネットワーク・ユーティリティー (BNU) 接続が確立されていること。
- ローカル・システムからリモート ・ジョブが出されていること。

1. リモート・キューにリストされているプロセスのジョブ ID 番号を判別します。 ローカル・システムのコマンド・ラインで、次のように入力します。

   ```
   uustat -a
   ```

   -a オプションにより、リモート・システムが保持しているキュー内のすべてのジョブと、そのシステム上の他の BNU ユーザーによるジョブ要求が表示されます。

   BNU は次のようなメッセージを出して応答します。

   ```
   heraC3113   11/06-17:47 S hera   you  289 D.venus471afd8
   merlinC3119 11/06-17:49 S merlin jane 338 D.venus471bc0a
   ```

2. その後、次のように入力します。

   ```
   uustat -k heraC3113
   ```

   -k オプションにより、heraC3113 ジョブ要求が取り消されます。

BNU のトラブルシューティング

対話の流れの中の特定のフェーズに BNU エラー・メッセージをリンクすることができます。「BNU 変換のフロー・チャート」と後続のエラーに関する説明は、BNU の問題を診断する際の参考にしてください。

以下のメッセージの中には BNU から送信されないものもあります。それらは別の UUCP バージョンが使用されている場合に送信されます。

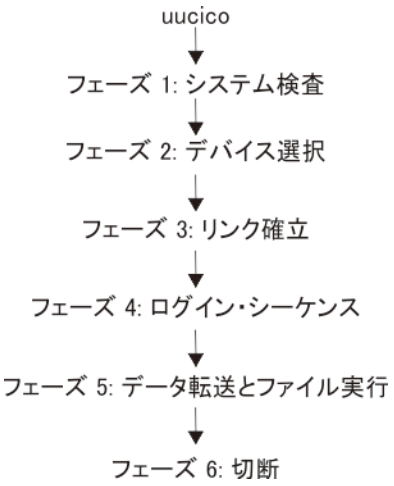


図 *26. BNU* 変換フロー・チャート

この図には、BNU 会話のフローと、そのそれぞれ異なるフェーズが示されています。 データは、いちばん上の uucico からフェーズ 1 (システム検査)、フェーズ 2 (デバイスの選択)、フェーズ 3 (リンクの確立)、フェーズ 4 (ログイン・シーケンス)、さらにフェーズ 5 (データの転送とファイルの実行) と順番に渡され、最後にフェーズ 6 (切断) へと渡されます。

BNU フェーズ 1 の状況メッセージ

BNU フェーズ 1 には、5 つの状況メッセージがあります。 次の表では、これらのメッセージについて説明します。

項目	説明
Assert Error	ローカル・システム装置に問題があります。 コマンド `errpt -a | pg` を実行し、エラー・レポートを調べて、考えられる原因を見つけてください。
System not in Systems	Systems ファイルの中にないリモート・システム名を指定すると、この状況メッセージが作成され、BNU が終了します。 **uuname** コマンドを使用して、システム名を調べ直してください。
Wrong time to call	Systems ファイルには、コールを許可する回数について制限があります。 BNU は、該当する回数に達するまで再試行を続けます。 Systems ファイルを調べてください。
Callback required	セキュリティー上または経費上の理由からネットワークの使用が制限されていて、現時点でのアクセスが拒否されています。
Cannot call No Call	これらのエラーは、最近、リモート・システムを BNU がコールしようとして失敗したことを意味します。 即座に再試行されることはありません。 また、これらのエラーは、古いシステム状況ファイルが保存されているために **uucico** デーモンが再試行できなくなっていることが原因で発生することもあります。

BNU フェーズ 2 の状況メッセージ

BNU フェーズ 2 には、4 つの状況メッセージがあります。 次の表では、これらのメッセージについて説明します。

項目	説明
Dialer Script Failed	Dialers ファイル・スクリプトが正常に完了しませんでした。

項目	説明
No Device Available Can't Access Device	モデムまたはシステムからの 発信電話回線がビジー状態です。 Systems ファイルのデバイス・エントリー内のエラーを調べてください。 また、Devices ファイルと Dialers ファイルを調べて、論理デバイスに物理デバイスが関連付けられていることを確認します。 ファイル /etc/uucp/Sysfiles 内で、正しく構成されていない代替の Systems、Devices、または Dialers ファイルを指定している可能性があります。 デバイスを他のプログラムが使用していないかどうか確認してください。 /var/locks ディレクトリー内でポートのロックを調べます。 ロック・ファイルが存在する場合 (LCK..TTY0 など)、そのロック・ファイル内の番号で識別されたプロセスがまだアクティブかどうか調べます。 アクティブでなければ、そのプロセスを除去できます (rm /var/locks/LCK..TTY0 のように)。 また、ポートに関する許可も調べてください。
Dial Failed Failed (call to system)	これらのエラー・メッセージは、システムから別のシステムへのダイヤル接続が成功したにもかかわらず、接続先システムが応答しなかった場合に表示されます。 原因としては、まず、Devices ファイルにエラーがあることが考えられます。 コマンド uucico -r1 -x6 -s *SystemName* を入力してください。 次に考えられる原因は、BNU に必要なファイルのうちいずれかが欠落していることです。 手作業で接続を行って Systems ファイル・エントリーに指定する必要のあるファイルを確認し、必要な設定を行ってください。 タイミングには十分な注意が必要です。モデムの電話番号を入力するときは特定の場所にポーズを入れる必要があります。 さらに、ポートがビジー、間違った番号のダイヤル、BNU にポートの所有権がない、などの原因も考えられます。
OK Auto Dial	これらは、単なる情報メッセージであり、エラーを示すものではありません。

BNU フェーズ 3 の状況メッセージ

BNU フェーズ 3 には、5 つの状況メッセージがあります。 次の表では、これらのメッセージについて説明します。

項目	説明
Handshake Failed (LCK)	デバイスを他のユーザーが使用しています。プロセスは、LCK ファイルを作成できません。 場合によっては、管理者は LCK ファイルを手動で除去する必要があります。 何回か再試行してから、システム管理者に問い合わせてください。 別のプロセス (例えば、**uucico** デーモンの別のインスタンスなど) がポートを制御していないかどうかを調べてください。
Login Failed	接続不良またはスロー・マシンが原因でログインに失敗しました。
Timeout	リモート・システムが一定期間内に応答しませんでした。 これは、チャット・スクリプトに関する問題を示すこともあります。
Succeeded (Call to System)	コールが完了しました。
BNU (continued)	これらは、単なる情報メッセージであり、エラーを示すものではありません。

BNU フェーズ 4 の状況メッセージ

BNU フェーズ 4 には、6 つの状況メッセージがあります。 次の表では、これらのメッセージについて説明します。

項目	説明
Startup Failed Remote reject after login	ログインすると、リモート・システム上で **uucico** デーモンが始動されます。 2 つのシステム間で会話を開始しようとして問題が起きた場合、これらのメッセージが作成されます。 また、誤った BNU アカウントにログインした可能性もあります。そうでなければ、初期ハンドシェークが失敗しました。
Wrong machine name	正しくないマシンをコールしたか、またはマシン名が変更されています。
Bad login/machine combination	リモート・システムにログインできませんでした。 問題としては、電話番号、ログイン、およびパスワードが正しくない、チャット・スクリプト内にエラーがあることなどが考えられます。
Remote has a LCK file for me	両方のシステムが同時に相手をコールしようとしました。 ローカル要求は、一時的に失敗します。
OK Talking	これらは、単なる情報メッセージであり、エラーを示すものではありません。
LOGIN: PASSWORD:	ログイン・プロンプトまたはパスワード・プロンプトがすべて大文字の場合、モデムがエコー・モード (Hayes 互換モデムの場合 E1) になっている可能性があります。 この結果、モデムは着信コールを受信するとローカル・システムに対してストリング RING をエコー・バック (送信) します。 **getty** コマンドがこのストリングを受信し、login: または password: のうち該当する方をすべて大文字に変更します。 モデム上でエコー・モードを off に変更してください (Hayes 互換モデムの場合は ATE0 を使用します)。 **注:** この変更を行った場合、Dialers ファイルのチャット・スクリプトに ATE1 を指定してください。 これを行わないと、モデムから OK の応答が戻ってきません。 リモート・ポートが delay または getty -r 用に設定されていて、チャット・スクリプトでキー入力が指定されている場合、delay 用に設定されたポートについては、ログインを開始する前に 1 つ以上の復帰の入力が必要になります。 ダイヤル接続を行う側のシステム上のチャット・スクリプトの先頭に次の指定を行ってください。 `"" ¥r¥d¥r¥d¥r¥d¥r in:--in:  ...` このチャット・スクリプトは「空白、リターン送信、遅延、リターン送信、遅延、リターン送信、遅延、リターン送信」を意味しています。

BNU フェーズ 5 の状況メッセージ

BNU フェーズ 5 には、5 つの状況メッセージがあります。 次の表では、これらのメッセージについて説明します。

項目	説明
Alarm	**uucico** デーモンに接続に関する問題が起こっています。 接続不良になっているか、またはモデム上で xon/xoff が yes に設定されています。
Remote access to path/file denied copy (failed)	これらのメッセージは、許可に関する問題を示します。ファイルとパスの許可を調べてください。

項目	説明
Bad read	リモート・システムがスペース (スプール領域内のスペースである可能性が最も高い) を使い尽くしました。あるいは、**uucico** デーモンがデバイスに対する読み取りまたは書き込みを実行できなかった可能性もあります。
Conversation failed	モデムのキャリア検知が失われました。 モデムの電源が切れていたか、ケーブルが緩んでいるか切断されている、または、リモート・システムがクラッシュしたかシャットダウンされた 可能性があります。電話の切断が原因でこのエラーが起きることもあります。
Requested Copy (succeeded)	これらは、単なる情報メッセージであり、エラーを示すものではありません。

BNU フェーズ 6 の状況メッセージ

BNU フェーズ 6 には、2 つの状況メッセージがあります。 次の表では、これらのメッセージについて説明します。

項目	説明
OK (Conversation Complete)	リモート・システムは、停止要求を拒否してロールを反転させることができます (つまり、ローカル・システムが実行する作業をリモート・システムが受け持つことを意味します)。 2 つの **uucico** デーモンは、他に作業が存在しないことに合意すると、停止します。
Conversation succeeded	これは、単なる情報メッセージであり、エラーを示すものではありません。

uucico デーモンを使用した BNU ログイン障害のデバッグ

BNU ログイン障害をデバッグするには、**uucico** デーモンを使用します。

- BNU がシステムにインストールされている 必要があります。
- ローカル・システムとリモート ・システム間にリンク (ケーブル、モデム、または TCP/IP) が設定されていなければなりません。
- Sysfiles ファイル (ある場合)、Systems ファイル、Permissions ファイル、Devices ファイル、Dialers ファイルなどの BNU 構成ファイルがローカル・システムとリモート・システム間で通信ができるように設定されている必要があります。

  注 : BNU 構成ファイルを変更するには、root ユーザー権限が必要です。
- デバッグ・モードで **uucico** デーモンを呼び出すには、root ユーザー権限が必要です。

1. ローカル・システムからリモート ・システムへの機能していない接続についてデバッグ情報を生成するには、次のように **-x** フラグを指定して、**uucico** デーモンを始動します。

   ```
   /usr/sbin/uucp/uucico -r 1 -s venus -x 9
   ```

   ここで、-r 1 は、マスター・モードまたはコール側モードを指定します。-s venus は、接続しようとしているリモート・システムの名前であり、-x 9 は、最も詳細なデバッグ情報が生成されるデバッグ・レベルです。
2. /etc/uucp/Systems フォーマットの Systems ファイルに次のような送信予期シーケンス・エントリーがある場合:

   ```
   venus Any venus 1200 - "" ¥n in:--in: uucp1 word:
    mirror
   ```

uucico デーモンは、ローカル・システムをリモート・システム venus に接続します。 デバッグ出力は、次のようになります。

```
expect: ""
got it
sendthem (^J^M)
expect (in:)^
M^Jlogin:got it
sendthem (uucp1^M)
expect (word:)^
M^JPassword:got it
sendthem (mirror^M)
imsg >^M^J^PShere^@Login Successful: System=venus
```

この場合、

項目	説明
expect: ""	ローカル・システムがリモート・システムからの情報を待たないことを指定します。
got it	メッセージを受信したことを確認します。
sendthem (^J^M)	ローカル・システムがリモート・システムに復帰と改行を送信することを指定します。
expect (in:)	ローカル・システムは、リモート・システムから文字列 in: で終わるログイン・プロンプトを受信することを予期します。
^M^Jlogin:got it	ローカル・システムがリモート・ログイン・スクリプトを受信したことを確認します。
sendthem (uucp1^M)	ローカル・システムがリモート・システムへ uucp1 ログイン ID を送信することを指定します。
expect (word:)	ローカル・システムは、リモート・システムから文字列 word: で終わるパスワード・プロンプトを受信することを予期します。
^M^JPassword:got it	ローカル・システムがリモート・パスワード・プロンプトを受信したことを確認します。
sendthem (mirror^M)	ローカル・システムがリモート・システムに uucp1 ログイン ID のパスワードを送信することを指定します。
imsg >^M^J^PShere^@Login Successful: System=venus	ローカル・システムがリモート・システム venus に正常にログインしたことを確認します。

注:

1. **uucico** コマンドが生成する送信予期デバッグ出力は、/etc/uucp/Dialers ファイル内の情報から、または /etc/uucp/Systems ファイル内の情報から得られます。 モデムとの通信に関する情報は、Dialers ファイルから得られますが、リモート・システムとの通信に関する情報は、Systems ファイルから得られます。(/etc/uucp/Systems と /etc/uucp/Dialers はデフォルトの BNU 構成ファイルです。 /etc/uucp/Sysfiles 内でその他のファイルを代替ファイルとして指定できます。)
2. リモート・システムとの接続を確立するには、そのシステムのログイン・シーケンスをよく理解しておく必要があります。

ネットワーク管理のための SNMP

ネットワーク管理機能では、ネットワーク・ホスト間の管理情報の交換を可能にする**シンプル・ネットワーク管理プロトコル (SNMP)** を使用して、システム・ネットワークを包括的に管理します。

SNMP は **TCP/IP** 準拠のインターネット用に設計されたインターネットワーク・プロトコルです。

AIX オペレーティング・システムがインストールされている場合、デフォルトで **SNMPv3** の非暗号化バージョンがインストールされ、それがシステムのブート時に始動されます。 `/etc/snmpd.conf` ファイル内にコミュニティー、トラップ、および SMUX のエントリーを独自に構成している場合は、それらを手動で `/etc/snmpdv3.conf` ファイルに移行する必要があります。 コミュニティーの移行の情報は、538 ページの『SNMPv1 から SNMPv3 への移行』を参照してください。

Communications Programming Concepts の SNMP Overview for Programmers も参照してください。

SNMP ネットワーク管理は、**TCP/IP** 準拠のネットワーク・アプリケーションで広く使用されている一般的なクライアント/サーバー・モデルに基づいています。 管理される各ホストでは、エージェント と呼ばれるプロセスを実行します。 エージェントとは、ホスト用の管理情報ベース (MIB) データベースを維持するサーバー・プロセスです。 ネットワーク管理の意思決定にかかわるホストは、マネージャーと呼ばれるプロセスを実行する場合があります。 マネージャー とは、MIB 情報を求める要求を生成し、応答を処理するクライアント・アプリケーションです。 また、マネージャーが MIB 情報の変更要求をエージェント・サーバーに送信する場合もあります。

SNMP は、AIX 内で次の RFC のサポートを提供します。

項目	説明
RFC 1155	**TCP/IP** 準拠のインターネットのための管理情報と構造の識別
RFC 1157	**シンプル・ネットワーク管理プロトコル (SNMP)**
RFC 1213	**TCP/IP** 準拠のインターネットのネットワーク管理用の管理情報ベース (MIB-II)
RFC 1227	**シンプル・ネットワーク管理プロトコル (SNMP)** の単一回線マルチプレクサー (SMUX) プロトコルと管理情報ベース (MIB)
RFC 1229	汎用インターフェースの 管理情報ベース (MIB) の拡張
RFC 1231	IEEE 802.5 トークンリングの管理情報ベース (MIB)
RFC 1398	イーサネット用管理オブジェクトの定義 (例えば、インターフェース ・タイプなど)
RFC 1512	FDDI 管理情報ベース
RFC 1514	ホスト・リソース MIB
RFC 1592	**シンプル・ネットワーク管理プロトコル**分散プログラム・インターフェース、バージョン 2
RFC 1905	シンプル・ネットワーク管理プロトコル、バージョン 2 (SNMPv2) のプロトコル操作
RFC 1907	シンプル・ネットワーク管理プロトコル、バージョン 2 (SNMPv2) の管理情報ベース
RFC 2572	**シンプル・ネットワーク管理プロトコル (SNMP)** のメッセージ処理とディスパッチ
RFC 2573	**SNMP** アプリケーション
RFC 2574	**シンプル・ネットワーク管理プロトコル、バージョン 3 (SNMPv3)** のユーザー・ベース・セキュリティー・モデル (USM)
RFC 2575	**シンプル・ネットワーク管理プロトコル (SNMP)** のビュー・ベース・アクセス制御モデル (VACM)

SNMPv3

以前のバージョンの AIX オペレーティング・システムでは、**SNMPv1** が AIX で使用可能な唯一の **SNMP** のバージョンでした。AIX オペレーティング・システムの **SNMPv3** は、メッセージ・セキュリティーとアクセス制御において強力で柔軟なフレームワークを提供します。

このセクションの情報は、**SNMPv3** にのみ適用されます。

メッセージ・セキュリティーでは、次のような検査が行われます。

- データが転送中に変更されなかったことを確認するデータ保全性検査
- 要求または応答がその発信元であるとされるソースから発信されることを確認するデータ発信元検査
- 盗聴防止のためのメッセージの適時性検査、およびオプションでデータの機密性検査

SNMPv3 アーキテクチャーでは、メッセージ・セキュリティーのためのユーザー・ベース・セキュリティー・モデル (USM) と、アクセス制御のためのビュー・ベース・アクセス制御モデル (VACM) が導入されました。 このアーキテクチャーでは、異なったセキュリティー・モデル、アクセス制御モデル、およびメッセージ処理モデルの同時使用をサポートします。 例えば、コミュニティー・ベース・セキュリティーを USM と並行して使用することもできます。

USM では、ユーザーのセキュリティー・パラメーター (セキュリティーのレベル、認証とプライバシー・プロトコル、およびキー) が エージェントとマネージャーの両方で構成されるという概念を使用します。 USM を使用して送信されるメッセージは、コミュニティー・ベース・セキュリティーを使用して送信される メッセージ (パスワードが暗号化されずトレースに表示される) よりも保護されています。 USM の場合、マネージャーとエージェントの間で交換されるメッセージは、データ保全性検査とデータ発信元認証が行われます。 コネクションレス転送プロトコルのために、通常発生するレベルを超えるメッセージ遅延とメッセージ再生は、時間インディケーターと要求 ID を使用することで回避されます。 また、データの機密性つまり暗号化も、輸出関連法規で許可されている国では、個別にインストール可能な製品として入手できます。 **SNMP** 暗号化バージョンは AIX 拡張パックにあります。

VACM を使用するには、データの集合 (ビューと呼ばれる)、データのユーザーのグループ、および、特定のユーザー・グループが読み取り、書き込み、またはトラップ内の受け取りに使用するビューを定義する アクセス・ステートメントを定義する必要があります。

また、**SNMPv3** では、エージェントの構成を表す MIB オブジェクトに対して **SNMP** SET コマンドを使用して、**SNMP** エージェントを動的に構成する機能も導入されました。 この動的構成のサポートによって、ローカルあるいはリモートで、構成エントリーを追加、削除、変更できるようになりました。

SNMPv3 アクセス・ポリシーとセキュリティー・パラメーターは、**SNMP** エージェント上の `/etc/snmpdv3.conf` ファイルと、**SNMP** マネージャー上の `/etc/clsnmp.conf` ファイルに指定されています。 これらのファイルの構成方法のシナリオは、541 ページの『SNMPv3 でのユーザーの作成』を参照してください。 また、ファイル参照 の `/etc/snmpdv3.conf` と `/etc/clsnmp.conf` のファイル・フォーマットを参照することもできます。

SNMPv3 アーキテクチャー

SNMPv3 アーキテクチャーには、4 つの主要な部分があります。

次の図では、これらのシステムがどのように相互に対話して、要求されたデータを提供するのかを説明します。

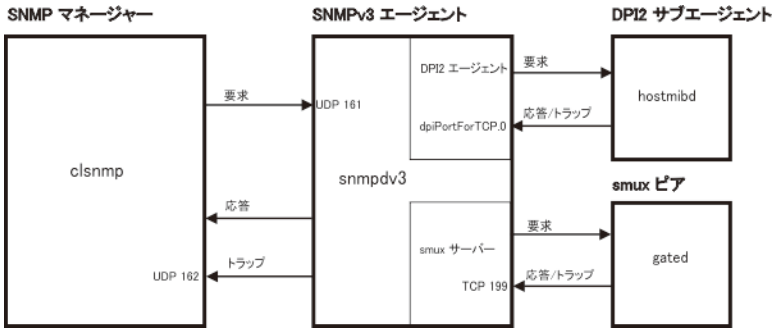


図 *27. SNMPv3* アーキテクチャーの基本部分

この図は、**SNMPv3** アーキテクチャーの例を示しています。 DPI2 サブエージェント、smux ピア、**SNMP** マネージャー、および **SNMP** エージェントが示されています。 さらに、これらの間の通信方法についても示されています。

SNMP エージェント

SNMP エージェントは、**SNMP** マネージャーから要求を受け取り、応答を戻します。

さらに、**SNMP** エージェントは、システム上のすべての DPI2 サブエージェントおよび SMUX ピアとの通信を行います。 **SNMP** エージェントは一部の MIB 変数を管理し、すべての DPI2 サブエージェントと SMUX ピアは MIB 変数を **SNMP** エージェントに登録します。

clsnmp (**SNMP** マネージャー) が要求を発行すると、その要求が **SNMP** エージェントにある UDP 161 に送信されます。 要求が **SNMPv1** または **SNMPv2c** 要求であれば、**SNMP** エージェントはコミュニティー名を

検査して、その要求を処理します。 要求が **SNMPv3** 要求であれば、**SNMP** エージェントは、データを要求しているユーザーを認証するために、認証キー (暗号化バージョンが稼働している場合はプライバシー・キー) を使用して、ユーザーが要求を満たすために必要なアクセス許可を持っていることを確認します。 **SNMP** エージェントがそのユーザーを認証できない場合、あるいはユーザーが要求を満たすために必要な正しいアクセス許可を持っていない場合、**SNMP** エージェントはその要求を承認しません。 **SNMPv3** でのユーザーの作成については、541 ページの『SNMPv3 でのユーザーの作成』を参照してください。

ユーザーが認証されて、正しいアクセス許可もある場合、**SNMP** エージェントは要求を満たします。 **SNMP** エージェントは要求されている MIB 変数を探し出します。 **SNMP** エージェント自体が要求された MIB 変数を管理している場合は、**SNMP** エージェントがその要求を処理して、応答を **SNMP** マネージャーに返します。 DPI2 サブエージェントまたは SMUX ピアが要求された MIB 変数を管理している場合、**SNMP** エージェントはその要求を MIB 変数を管理している DPI2 サブエージェントまたは SMUX ピアに転送して、そこで要求を処理させて、応答を **SNMP** マネージャーに返します。

DPI2 サブエージェント

hostmibd などの DPI2 サブエージェントは、DPI2 エージェントとの通信を行います。DPI2 エージェントは、**SNMPv3** では SNMP エージェントの一部です。

DPI2 サブエージェントは、応答とトラップを dpiPortForTCP.0 を介して DPI2 エージェントに送信します。これは予約済みポートではないため、DPI2 サブエージェントはまず dpiPortForTCP.0 のポート番号の要求を発行する必要があります。この要求は、**SNMP** エージェント上の UDP 161 に対して発行され、その後 **SNMP** エージェントは、DPI2 サブエージェントに dpiPortForTCP.0 のポート番号を返します。ポート番号を受け取った後、DPI2 サブエージェントはそのポート 番号を使用して DPI2 エージェントとの接続を確立します。 そして DPI2 サブエージェントは、その MIB サブツリーを DPI2 エージェントに登録します。

注 : SNMP エージェントが UDP 161 以外のポート上で listen できるようにするには、SNMP_PORT 環境を設定しなければなりません。 この変数を設定する方法は次の 2 つです。

- **メソッド 1**: DPI2 サブエージェントを停止し、次のコマンドを入力します。
  - `SNMP_PORT=<port_number> /usr/sbin/aixmibd -d 128`
  - `SNMP_PORT=<port_number> /usr/sbin/hostmibd -d 128`
  - `SNMP_PORT=<port_number> /usr/sbin/snmpmibd -d 128`

  port_number は、使用するポートの番号です。

  コマンドの実行が完了した後に、DPI2 サブエージェントを開始してください。
- **メソッド 2**: SNMP_PORT 変数を `/etc/environment` ファイルに組み込み、この変数に新しいポート値を割り当てます。 **aixmibd**、**hostmibd**、**snmpmibd**、および **snmpd** デーモンを現状のまま `/etc/rc.tcpip` から実行できます。 このメソッドでは、**aixmibd**、**hostmibd**、および **snmpmibd** コマンドをコマンド・ラインから実行する必要はありません。

接続が確立されて MIB サブツリーが登録されたあと、DPI2 サブエージェントは、DPI2 エージェントから受け取った要求に 応答することができます。 要求を受信すると、DPI2 サブエージェントは要求を処理して必要な情報を返します。

必要であれば DPI2 サブエージェントもトラップを送信できます。 トラップが送信されると、**SNMP** エージェントは `/etc/snmpdv3.conf` ファイルを検査して、トラップを転送する必要のある IP アドレス (複数の場合もある) を決定し、トラップをそれらのアドレスに送信します。

SMUX ピア

gated などの SNMP 多重化 (SMUX) ピアは、開始時に **TCP** 199 への接続を確立し、SMUX の関連付けを初期化します。

初期化の後、SMUX ピアは管理する MIB サブツリーを登録します。

登録が完了すると、SMUX ピアは SMUX サーバーから送られてくる要求を受け入れ、応答を返すことができます。 SMUX ピアは要求を受け取ると、その要求を処理して応答を SMUX サーバーに返します。

SMUX ピアもトラップを SMUX サーバーに送信できます。 トラップが送信されると、**SNMP** エージェントは `/etc/snmpdv3.conf` ファイルを検査して、トラップを転送する必要のある IP アドレス (複数の場合もある) を決定し、トラップをそれらのアドレスに送信します。

SNMP マネージャー

SNMP マネージャーは、**clsnmp** を実行します。**clsnmp** は、**SNMPv1**、**SNMPv2c**、および **SNMPv3** と互換性があります。

clsnmp コマンドを使用して、get、get-next、get-bulk、または set などの要求を発行してください。 要求は **SNMP** エージェント上の UDP 161 に送信され、その後、**SNMP** エージェントからの応答を待ちます。

注 : SNMP マネージャーが UDP 161 以外のポートを使用できるようにするには、/etc/clsnmp.conf ファイル内の **targetAgent** フィールドで使用するポート番号と IP アドレスを宣言する必要があります。 /etc/clsnmp.conf ファイルについては、ファイル参照 の clsnmp.conf File を参照してください。

これは UDP 162 上で **SNMP** トラップを listen することもできます。 **SNMP** マネージャーは、**SNMP** エージェント上の /etc/snmpdv3.conf ファイルで IP アドレスがそのように指定されていれば、トラップを受信します。

MIB 変数

MIB 変数に関する情報については、以下の場所から参照できます。

MIB 変数については、*Communications Programming Concepts* の Management Information Base、Terminology Related to Management Information Base Variables、Working with Management Information Base Variables、および Management Information Base Database を参照してください。

ユーザー独自の DPI2 サブエージェントまたは smux ピアを構成する場合は、/usr/samples/snmpd/smux および /usr/samples/snmpd/dpi2 ディレクトリーを参照してください。

SNMPv3 認証キー

SNMPv3 要求の処理には、通常、認証が必要です (要求されたセキュリティー・レベルが noAuth の場合を除く)。

要求を認証する場合、**SNMP** エージェントは、**SNMPv3** 要求で送信された認証キーが、ユーザーが定義した認証キーから作成されたメッセージ・ダイジェストと一致するメッセージ・ダイジェストの作成に 使用できるものであることを検査します。

SNMP マネージャーから要求が発行されると、**clsnmp** コマンドは、**SNMP** マネージャー上の /etc/clsnmp.conf ファイル内のエントリーにある認証キーを使用します。 このキーは、**SNMP** エージェントの /etc/snmpdv3.conf ファイル内の USM_USER エントリーに指定された、そのユーザー用の認証キーと関連している必要があります。 認証キーは **pwtokey** コマンドを使用して生成されます。

認証キーは、次の 2 つの情報から生成されます。

- 指定されたパスワード
- キーが使用される **SNMP** エージェントの ID。 エージェントが IBM エージェントであり、その engineID がベンダー固有の engineID 方式を使用して生成されたものである場合、エージェントは IP アドレスまたはホスト名によって識別されます。 そうでなければ、エージェント ID として engineID を提供する必要があります。

キーが使用されるエージェントの ID を含んだキーは、ローカライズ・キーと呼ばれます。 ローカライズ・キーは、そのエージェント内でのみ使用できます。 キーが使用されるエージェントの engineID を含まないキーは、非ローカライズ・キーと呼ばれます。

clsnmp コマンドの構成ファイル /etc/clsnmp.conf に保管されているキーは、非ローカライズと考えられます。 **SNMP** エージェントの構成ファイル /etc/snmpdv3.conf に保管されているキーは、ローカライズまたは非ローカライズのいずれかですが、ローカライズ・キーを使用する方がよりセキュアであると考えられています。

認証キーをクライアントの構成ファイルに保管する代わりに、**clsnmp** コマンドの場合は ユーザー・パスワードを保管できます。 構成ファイルに **clsnmp** コマンドのパスワードが 入っている場合、コードは、ユーザーの認証キーを生成します (暗号化バージョンがインストールされていて、要求があった場合は、プライバシー・キーも生成します)。 これらのキーは、エージェントの /etc/snmpdv3.conf ファイル内の USM_USER に構成されたキー、あるいは **SNMP** SET コマンドを使用して動的に構成されたキーと同じ認証

値を生成する必要があります。 しかし、クライアントの構成ファイル内のパスワードを使用する方が、構成ファイル内のキーを使用するよりも保護の度合いは低いと考えられています。

SNMPv3 プライバシー・キー
暗号化バージョンは、輸出関連法規で許可されている場合には AIX 拡張パックの別製品として入手可能です。 暗号化に使用されるキーは、認証に使用されるのと同じアルゴリズムを使用して生成されます。

ただし、キーの長さは異なります。 例えば、HMAC-SHA 認証キーは 20 バイト長ですが、HMAC-SHA と一緒に使用される ローカライズ暗号キーは、16 バイト長しかありません。

暗号化バージョンは、インストールされたあと自動的に活動化されます。 非暗号化バージョンに戻るには、**snmpv3_ssw** コマンドを使用します。

SNMPv3 キーの生成
AIX は **pwtokey** コマンドを使用して、認証キーと、該当する場合はプライバシー・キーを生成します。

pwtokey コマンドは、パスワードをローカライズおよび非ローカライズの認証キーとプライバシー・キーに変換できます。 **pwtokey** プロシージャーは、パスワードと、エージェントとしての ID を使用して、認証キーとプライバシー・キーを生成します。 **pwtokey** コマンドで使用されるプロシージャーは **clsnmp** コマンドで使用されるアルゴリズムと同じであるため、**SNMP** エージェントを構成している場合は、特定のパスワードと、ターゲットが稼働することになる IP アドレスがあれば、ユーザーの **SNMP** マネージャー上の /etc/clsnmp.conf ファイルに入れる適切な認証キー (とプライバシー・キー) を生成できます。

認証キー (暗号化バージョンが稼働している場合はプライバシー・キーも) を生成したら、それらのキーを **SNMP** エージェント上の /etc/snmpdv3.conf ファイルと **SNMP** マネージャー上の /etc/clsnmp.conf ファイルに入れる必要があります。

SNMPv3 では、9 つのユーザー構成が可能です。 次に、それぞれの構成を例とともに示します。 これらの特定のキーは、パスワードとして defaultpassword、IP アドレスとして 9.3.149.49 を使用して生成されました。 次のコマンドが使用されました。

```
pwtokey -u all -p all defaultpassword 9.3.149.49
```

次の認証キーとプライバシー・キーが生成されました。

```
Display of 16 byte HMAC-MD5 authKey:
  18a2c7b78f3df552367383eef9db2e9f

Display of 16 byte HMAC-MD5 localized authKey:
  a59fa9783c04bcbe00359fb1e181a4b4

Display of 16 byte HMAC-MD5 privKey:
  18a2c7b78f3df552367383eef9db2e9f

Display of 16 byte HMAC-MD5 localized privKey:
  a59fa9783c04bcbe00359fb1e181a4b4

Display of 20 byte HMAC-SHA authKey:
  754ebf6ab740556be9f0930b2a2256ca40e76ef9

Display of 20 byte HMAC-SHA localized authKey:
  cd988a098b4b627a0e8adc24b8f8cd02550463e3

Display of 20 byte HMAC-SHA privKey:
  754ebf6ab740556be9f0930b2a2256ca40e76ef9

Display of 16 byte HMAC-SHA localized privKey:
  cd988a098b4b627a0e8adc24b8f8cd02
```

これらのエントリーは、/etc/snmpdv3.conf ファイルに現れるものです。 次の 9 つの構成が可能です。

- HMAC-MD5 プロトコルを使用するローカライズ認証キーおよびローカライズ・プライバシー・キー:

  ```
  USM_USER user1 - HMAC-MD5 a59fa9783c04bcbe00359fb1e181a4b4 DES
  a59fa9783c04bcbe00359fb1e181a4b4 L - -
  ```

- HMAC-MD5 プロトコルを使用する非ローカライズ認証キーおよび非ローカライズ・プライバシー・キー:

```
USM_USER user2 - HMAC-MD5 18a2c7b78f3df552367383eef9db2e9f DES
18a2c7b78f3df552367383eef9db2e9f N  - -
```

- HMAC-MD5 プロトコルを使用するローカライズ認証キー:

```
USM_USER user3 - HMAC-MD5 a59fa9783c04bcbe00359fb1e181a4b4 - - L -
```

- HMAC-MD5 プロトコルを使用する非ローカライズ認証キー:

```
USM_USER user4 - HMAC-MD5 18a2c7b78f3df552367383eef9db2e9f - - N -
```

- HMAC-SHA プロトコルを使用するローカライズ認証キーおよびローカライズ・プライバシー・キー:

```
USM_USER user5 - HMAC-SHA cd988a098b4b627a0e8adc24b8f8cd02550463e3 DES
cd988a098b4b627a0e8adc24b8f8cd02 L -
```

- HMAC-SHA プロトコルを使用する非ローカライズ認証キーおよび非ローカライズ・プライバシー・キー:

```
USM_USER user6 - HMAC-SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 DES
754ebf6ab740556be9f0930b2a2256ca40e76ef9 N -
```

- HMAC-SHA プロトコルを使用するローカライズ認証キー:

```
USM_USER user7 - HMAC-SHA cd988a098b4b627a0e8adc24b8f8cd02550463e3 - - L -
```

- HMAC-SHA プロトコルを使用する非ローカライズ認証キー:

```
USM_USER user8 - HMAC-SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 - - N -
```

- 認証キーとプライバシー・キーのいずれも使用しない (**SNMPv1**)

```
USM_USER user9 - none - none - - -
```

SNMPv3 にユーザーを構成するには、/etc/snmpdv3.conf ファイルと /etc/clsnmp.conf ファイルの両方を構成する必要があります。 ユーザー・キーの生成と必要な構成ファイルの編集についてのシナリオは、541 ページの『SNMPv3 でのユーザーの作成』を参照してください。 また、/usr/samples/snmpdv3 ディレクトリーにあるサンプルの snmpdv3.conf 構成ファイルと clsnmp.conf 構成ファイルを参照することもできます。

SNMPv3 キーの更新

SNMPv3 では、新規パスワードを基に、ユーザー・キーを動的に更新する機能があります。

この更新を行うには、**pwchange** コマンドを使用して更新されたパスワードに基づいて新規ユーザー・キーを生成し、**clsnmp** コマンドを使用して /etc/snmpdv3.conf ファイル内のユーザー・キーを動的に更新し、/etc/clsnmp.conf を編集して新規キーを入れます。 この処理の間、新規パスワードがマシン間でやり取りされることはありません。

ユーザー・キーの更新のステップバイステップの説明は、534 ページの『SNMPv3 での認証キーとプライバシー・キーの動的更新』を参照してください。

SNMPv3 での認証キーとプライバシー・キーの動的更新

このシナリオでは、**SNMPv3** でユーザーの認証キーを動的に更新する方法を示します。

このシナリオでは、ユーザー u4 はユーザー u8 の認証キーを更新します。 ユーザー u4 と u8 の両方には、パスワード defaultpassword および IP アドレス 9.3.149.49 に基づいて作成された認証キーが既にあり、すべてが機能しています。

このシナリオでは、新規キーがユーザー u8 に対して作成され、/etc/snmpdv3.conf ファイルが動的に更新されます。 次に、マネージャー側の /etc/clsnmp.conf ファイルのユーザー u8 の認証キーは、新規キーを反映するために手動で編集する必要があります。

この手順を開始する前に、**SNMP** エージェント上の /etc/snmpdv3.conf ファイルのバックアップと、**SNMP** マネージャー上の /etc/clsnmp.conf ファイルのバックアップを取ります。

動的に更新される /etc/snmpdv3.conf ファイルを以下に示します。

```
USM_USER u4 - HMAC-MD5 18a2c7b78f3df552367383eef9db2e9f - - N -
USM_USER u8 - HMAC-SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 - - N -

VACM_GROUP group1 SNMPv1  public  -
VACM_GROUP group2 USM u4 -
VACM_GROUP group2 USM u8 -

VACM_VIEW defaultView        internet                    - included -

VACM_ACCESS  group1 - - noAuthNoPriv SNMPv1  defaultView - defaultView -
VACM_ACCESS  group2 - - noAuthNoPriv USM defaultView defaultView defaultView -
VACM_ACCESS  group2 - - AuthNoPriv USM defaultView defaultView defaultView -
VACM_ACCESS  group2 - - AuthPriv USM defaultView defaultView defaultView -

NOTIFY notify1 traptag trap -

TARGET_ADDRESS Target1 UDP 127.0.0.1       traptag trapparms1 - - -
TARGET_ADDRESS Target2 UDP 9.3.149.49       traptag trapparms2 - - -
TARGET_ADDRESS Target3 UDP 9.3.149.49       traptag trapparms3 - - -
TARGET_ADDRESS Target4 UDP 9.3.149.49       traptag trapparms4 - - -

TARGET_PARAMETERS trapparms1 SNMPv1  SNMPv1  public  noAuthNoPriv -
TARGET_PARAMETERS trapparms3 SNMPv2c  SNMPv2c  publicv2c  noAuthNoPriv -
TARGET_PARAMETERS trapparms4 SNMPv3  USM     u4 AuthNoPriv -
```

ユーザー u8 に対して更新される /etc/clsnmp.conf ファイルを以下に示します。

```
testu4 9.3.149.49 snmpv3 u4 - - AuthNoPriv HMAC-MD5 18a2c7b78f3df552367383eef9db2e9f - -
testu8 9.3.149.49 snmpv3 u8 - - AuthNoPriv HMAC-SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 - -
```

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

パスワードと認証キーの更新

/etc/snmpd.conf ファイル内のコミュニティー名は、/etc/snmpdv3.conf ファイル内の VACM_GROUP エントリーの一部になります。 各コミュニティーはグループに 入れる必要があります。 次に、必要とされるビューおよびアクセス権をこのグループに付与します。

1. **SNMP** マネージャー側で、**pwchange** コマンドを実行します。 このシナリオでは、次のコマンドが実行されました。

   ```
   pwchange -u auth -p HMAC-SHA defaultpassword newpassword 9.3.149.49
   ```

   このコマンドにより、新規認証キーが生成されます。

   - -u auth は、認証キーのみが作成されることを示します。 プライバシー・キーも更新する場合は、-u all を使用します。
   - -p HMAC-SHA は、認証キーの作成に使用されるプロトコルを示します。 プライバシー・キーも更新する場合は、-p all を使用します。
   - *defaultpassword* は、最新の認証キーの作成に使用されるパスワードです (例えば、最新の認証キーの作成に bluepen が使用された場合は、bluepen もここで使用されます)。
   - *newpassword* は、認証キーの生成に使用される新規パスワードです。 将来の参照のためにこのパスワードを保管してください。
   - 9.3.149.49 は、**SNMP** エージェントが実行中の IP アドレスです。

   このコマンドにより、以下の出力が生成されました。

   ```
   Dump of 40 byte HMAC-SHA authKey keyChange value:
     8173701d7c00913af002a3379d4b150a
     f9566f56a4dbde21dd778bb166a86249
     4aa3a477e3b96e7d
   ```

次のステップでこの認証キーを使用します。

注 : 使用する新規パスワードは安全な場所に保管してください。 将来の変更時にこのパスワードを再度使用する必要があります。

2. **SNMP** マネージャーで、ユーザー u4 は、次のコマンドを入力して、ユーザー u8 の認証キーを変更します。

```
clsnmp -h testu4 set usmUserAuthKeyChange.12.0.0.0.2.0.0.0.0.9.3.149.49.2.117.56
 ¥'8173701d7c00913af002a3379d4b150af9566f56a4dbde21dd778bb166a862494aa3a477e3b96e7d¥'h
```

- testu4 が使用されるのは、これが /etc/clsnmp.conf ファイルのユーザー u4 にマップされるためです。
- *usmUserAuthKeyChange* のインスタンス ID には、更新が行われる SNMP エージェントのエンジン ID および認証キーが更新されているユーザー名が 10 進値で組み込まれます。 エンジン ID は /etc/snmpd.boots ファイルの中で見つけることができます (/etc/snmpd.boots ファイルには数字からなる 2 つの文字列が含まれます。 エンジン ID は最初の文字列です。 2 番目にある、数字の文字列は無視してください)。

  エンジン ID は、ここで使用するために、16 進値から 10 進値に変換する必要があります。 16 進値のエンジン ID の数字は、2 桁ごとに 1 つの 10 進値に変換されます。 例えば、エンジン ID 0000000200000000009039531 は 00 00 00 02 00 00 00 00 09 03 95 31 として読み取られます。 これらの数値のそれぞれは、10 進値に変換する必要があります。その結果、0.0.0.2.0.0.0.0.9.3.149.49 が生じます (変換テーブルについては、ASCII、10 進、16 進、8 進、および 2 進の変換テーブルを参照してください)。文字列の最初の数字は、10 進値にした文字列でのバイト数です。 この場合、その数値は 12 です。その結果、12.0.0.0.2.0.0.0.0.9.3.149.49 が生じます。

  次の数字はユーザー名のバイト数で、その次に続く数字は、ユーザー名自体の 10 進値を示します。この場合、ユーザー名は u8 です。 10 進値に変換されると、u8 は 117.56 になります。 ユーザー名の長さは 2 バイトであるため、ユーザー名を表す値は 2.117.56 になります。 この値を 10 進値でのエンジン ID の末尾に追加します (変換テーブルについては、ASCII、10 進、16 進、8 進、および 2 進の変換テーブルを参照してください)。

  この場合、その結果は 12.0.0.0.2.0.0.0.0.9.3.149.49.2.117.56 です。
- コマンドの次の値は、前述のステップで **pwchange** コマンドを使用して生成された新規認証キーです。

  注 : そのユーザーにプライバシー・キーも構成されている場合は、プライバシー・キーを更新するために、この手順を繰り返す必要があります。 プライバシー・キーを更新するときは、usmUserAuthKeyChange 値ではなく usmUserPrivKeyChange 値を使用します。

  usmUserAuthKeyChange ではなく usmUserOwnAuthKeyChange を使用すると、ユーザーは、自身の認証キーを変更することができます。 例えば、ユーザー u4 は、usmUserOwnAuthKeyChange を使用して自身の認証キーを変更できます。

コマンドの出力は次のとおりです。

```
1.3.6.1.6.3.15.1.2.2.1.6.12.0.0.0.2.0.0.0.0.9.3.149.49.2.117.56 = '8173701d7c00913af002a3379
d4b150af9566f56a4dbde21dd778bb166a862494aa3a477e3b96e7d'h
```

このコマンドの完了後に、/etc/snmpdv3.conf ファイルは、**SNMP** エージェント側で 5 分後に自動的に更新されます。 このファイルは、**SNMP** デーモンを停止して開始することによっても更新することができます。 ユーザー u8 の次のエントリーは、/etc/snmpdv3.conf ファイルで動的に更新されます。

```
USM_USER u8 0000000200000000009039531 HMAC-SHA 4be657b3ae92beee322ee5eaeef665b338caf2d9
None - L nonVolatile
```

3. **SNMP** マネージャー側で、**pwtokey** コマンドを実行して、新規パスワードに基づいて新規認証キーを生成し、/etc/clsnmp.conf ファイルに入れます。 このシナリオでは、次のコマンドが実行されました。

```
pwtokey -u auth -p HMAC-SHA  newpassword 9.3.149.49
```

   - -u auth は、認証キーのみが作成されることを示します。 プライバシー・キーも更新する場合は、-u all を使用します。
   - -p HMAC-SHA は、認証キーの作成に使用されるプロトコルを示します。 プライバシー・キーも更新する場合は、-p all を使用します。
   - 使用されるパスワード (この場合は、*newpassword*) は、**pwchange** コマンドを使用した新規認証キーの生成時に使用されたパスワードと同じでなければなりません。
   - 使用される IP アドレス (この場合は、9.3.149.49) は、エージェントが実行中の IP アドレスでなければなりません。

   この結果、ローカライズおよび非ローカライズの認証キーが次のように表示されます。

```
Display of 20 byte HMAC-SHA authKey:
  79ce23370c820332a7f2c7840c3439d12826c10d

Display of 20 byte HMAC-SHA localized authKey:
  b07086b278163a4b873aace53a1a9ca250913f91
```

4. 任意のテキスト・エディターを使用して /etc/clsnmp.conf ファイルを開き、更新されるキーを保持するユーザーの行に非ローカライズ認証キーを入れます。 このシナリオでは、エントリーは次のとおりです。

```
testu8 9.3.149.49 snmpv3 u8 - - AuthNoPriv HMAC-SHA 79ce23370c820332a7f2c7840c3439d12826c10d
- -
```

   ファイルを保存してクローズします。
5. 次のコマンドを実行して、更新済み構成をテストします。

```
clsnmp -v -h testu8 walk mib
```

   ここで、*mib* は、ユーザー u8 が読み取り権限を持つ MIB 変数です。 この場合、ユーザー u8 は internet にアクセスできます。

SNMPv3 要求

clsnmp コマンドを使用して、ローカル・ホストまたはリモート・ホスト上の **SNMP** エージェントに **SNMP** 要求を送信します。

要求は、**SNMPv1**、**SNMPv2c**、または **SNMPv3** 要求です。 要求を処理するためには、/etc/clsnmp.conf ファイルが構成されていなければなりません。

clsnmp コマンドは、get、getnext、getbulk、set、walk、および findname 要求を発行できます。 次に、これらの要求をそれぞれ簡単に説明します。

get
ユーザーが 1 つの MIB 変数からデータを収集できるようにします。

getnext
MIB 変数サブツリー内の次の MIB 変数を指定します。

getbulk
複数の MIB サブツリーのすべての MIB 変数を指定します。

set
ユーザーが MIB 変数を設定できるようにします。

walk
1 つのサブツリーのすべての MIB 変数を指定します。

findname
OID を変数名にマップします。

trap
clsnmp がポート 162 上のトラップを listen できるようにします。

SNMPv1 から SNMPv3 への移行

このシナリオでは、**SNMPv1** から **SNMPv3** への標準的な移行を示します。

AIX オペレーティング・システムでは、システムのブート時に実行中のデフォルトの **SNMP** エージェントは、**SNMPv3** の非暗号化バージョンです。**SNMPv3** は、/etc/snmpdv3.conf ファイルをその構成ファイルとして使用します。 /etc/snmpd.conf ファイル (以前のバージョンの AIX オペレーティング・システムで **SNMPv1** によって使用される) で構成したすべてのパラメーターは、/etc/snmpdv3.conf ファイルに手動で移行する必要があります。

このシナリオでは、/etc/snmpd.conf ファイルで構成されたコミュニティーおよびトラップは、/etc/snmpdv3.conf ファイルに移行されます。 このシナリオが終了するまでに、**SNMPv3** は、**SNMPv1** が提供した機能と同じ機能を提供するようになります。 ユーザー独自の **SNMPv1** コミュニティーまたはトラップのいずれも構成しなかった場合は、この手順を実行する必要はありません。

このファイルには、**SNMPv3** で使用可能なフィーチャーに関するいずれの情報も含まれません。 **SNMPv1** では使用できない **SNMPv3** のフィーチャーを使用したユーザーの作成については、541 ページの『SNMPv3 でのユーザーの作成』を参照してください。

次のファイルは、移行される /etc/snmpd.conf ファイルの例です。 コミュニティー daniel、vasu、および david が構成されます。 これらのコミュニティーは、手動で移行する必要があります。

```
logging         file=/usr/tmp/snmpd.log         enabled
logging         size=0                          level=0

community       daniel       0.0.0.0     0.0.0.0         readWrite  1.17.35
community       vasu         9.3.149.49  255.255.255.255 readOnly   10.3.5
community       david        9.53.150.67 255.255.255.255 readWrite  1.17.35

view 1.17.35    udp  icmp  snmp 1.3.6.1.2.1.25
view 10.3.5     system interfaces tcp icmp

trap            daniel          9.3.149.49      1.17.35   fe
trap            vasu            9.3.149.49      10.3.5   fe
trap            david           9.53.150.67     1.17.35   fe

smux            1.3.6.1.4.1.2.3.1.2.3.1.1         sampled_password  # sampled
```

このシナリオの各ステップを実行するには、/etc/snmpd.conf ファイルを参照してください。 この手順を開始する際の準備としてそのファイルのコピーを保持してください。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ステップ 1. コミュニティー情報の移行

/etc/snmpd.conf ファイル内のコミュニティー名は、/etc/snmpdv3.conf ファイル内の VACM_GROUP エントリーの一部になります。 各コミュニティーはグループに 入れる必要があります。 次に、必要とされるビューおよびアクセス権をこのグループに付与します。

1. root 権限により、任意のテキスト・エディターを使用して /etc/snmpdv3.conf ファイルを開きます。 このファイル内の VACM_GROUP エントリーを見つけます。
2. 移行するコミュニティーごとに VACM_GROUP エントリーを作成します。 複数のコミュニティーが同じビューおよびアクセス権を共有する場合は、それらにグループを 1 つだけ作成する必要があります。 /etc/snmpd.conf ファイルのコミュニティー名は、VACM_GROUP エントリーの *securityName*

値になります。 このシナリオでは、以下のエントリーが vasu、daniel、および david に追加されています。

```
#-------------------------------------------------------------------
# VACM_GROUP entries
#    Defines a security group (made up of users or communities)
#    for the View-based Access Control Model (VACM).
# Format is:
#  groupName securityModel securityName storageType
VACM_GROUP group2 SNMPv1  vasu -
VACM_GROUP group3 SNMPv1  daniel -
VACM_GROUP group3 SNMPv1  david -
#-------------------------------------------------------------------
```

- *groupName* には、group1 を除く任意の選択値を指定できます。
- SNMPv1 コミュニティーを移行するので、*securityModel* は SNMPv1 のままになります。
- このシナリオでは、daniel および david は、/etc/snmpd.conf ファイルの同じビューおよびアクセス権を共有します。 したがって、この両方は /etc/snmpdv3.conf ファイルの group3 のメンバーです。 コミュニティー vasu は、そのビューおよびアクセス権が david および daniel のビューおよびアクセス権と異なるため、別のグループに入れられます。

これで、各コミュニティーはグループに 入れられました。

ステップ 2. ビュー情報の移行

/etc/snmpd.conf ファイル内のビュー情報は、/etc/snmpdv3.conf ファイル内の COMMUNITY、VACM_VIEW、および VACM_ACCESS エントリーになります。 これらのエントリーは、グループごとのビューおよびアクセス権を決定します。

1. daniel、vasu、および david 用に COMMUNITY エントリーを作成し、/etc/snmpd.conf ファイルに指定されたものと同じ IP アドレスを netAddr および netMask 用に維持します。

```
#-------------------------------------------------------------------------
# COMMUNITY
#    Defines a community for community-based security.
# Format is:
#  communityName securityName securityLevel netAddr netMask storageType
COMMUNITY public    public     noAuthNoPriv 0.0.0.0     0.0.0.0         -
COMMUNITY daniel    daniel     noAuthNoPriv 0.0.0.0     0.0.0.0         -
COMMUNITY vasu      vasu       noAuthNoPriv 9.3.149.49  255.255.255.255 -
COMMUNITY david     david      noAuthNoPriv 9.53.150.67 255.255.255.255 -
#-------------------------------------------------------------------------
```

2. 各グループがアクセスできる MIB オブジェクトまたは変数ごとに VACM_VIEW エントリーを作成します。 /etc/snmpd.conf ファイルに従って、daniel および david は udp、icmp、snmp、および 1.3.6.1.2.1.25 (RFC 1514 で定義されたホスト・サブツリー) にアクセスでき、vasu は system、interfaces、tcp、および icmp にアクセスできます。 これらのビュー・エントリーは、次のように /etc/snmpdv3.conf ファイルに移行されます。

```
#-------------------------------------------------------------------
# VACM_VIEW entries
#    Defines a particular set of MIB data, called a view, for the
#    View-based Access Control Model.
# Format is:
#  viewName viewSubtree viewMask viewType storageType

VACM_VIEW group2View        system                      - included -
VACM_VIEW group2View        interfaces                  - included -
VACM_VIEW group2View        tcp                         - included -
VACM_VIEW group2View        icmp                        - included -

VACM_VIEW group3View        udp                         - included -
VACM_VIEW group3View        icmp                        - included -
VACM_VIEW group3View        snmp                        - included -
VACM_VIEW group3View        1.3.6.1.2.1.25              - included -
#-------------------------------------------------------------------
```

3. VACM_ACCESS エントリーを追加して、VACM_VIEW エントリーで定義された MIB 変数にアクセス権を定義します。 /etc/snmpd.conf ファイルでは、daniel と david の両方が MIB 変数への readWrite アクセス権を持つのに対して、vasu は readOnly アクセス権を持ちます。

   VACM_ACCESS エントリーを追加して、これらのアクセス権を定義します。 このシナリオでは、group2 (vasu) に対して readView に group2View を付与し、writeView には - を付与しています。これは、vasu が /etc/snmpd.conf ファイルで readOnly アクセス権を付与されているためです。 group3 (daniel および david) に対しては、readView と writeView の両方に group3View を付与しています。これは、これらのグループが /etc/snmpd.conf ファイルで readWrite アクセス権を付与されているためです。 次の例を参照してください。

```
#-------------------------------------------------------------------------------------------------------------
--
# VACM_ACCESS entries
#    Identifies the access permitted to different security groups
#    for the View-based Access Control Model.
# Format is:
# groupName contextPrefix contextMatch securityLevel securityModel readView writeView notifyView
storageType
VACM_ACCESS  group1 - - noAuthNoPriv SNMPv1  defaultView - defaultView -
VACM_ACCESS  group2 - - noAuthNoPriv SNMPv1  group2View - group2View -
VACM_ACCESS  group3 - - noAuthNoPriv SNMPv1  group3View group3View group3View -
#-------------------------------------------------------------------------------------------------------------
--
```

ステップ 3. トラップ情報の移行

/etc/snmpd.conf ファイル内のトラップ・エントリーは、/etc/snmpdv3.conf ファイル内の NOTIFY、TARGET_ADDRESS、および TARGET_PARAMETERS エントリーになります。 ただし、TARGET_ADDRESS と TARGET_PARAMETERS のみを移行する必要があります。

1. /etc/snmpd.conf ファイル内のトラップ・エントリーにリストされている IP アドレスは、/etc/snmpdv3.conf ファイル内の TARGET_ADDRESS エントリーの一部になります。 この行は、トラップが送信されるホストを指定します。 targetParams エントリーを定義できます。 このシナリオでは、trapparms1、trapparms2、trapparms3、および trapparms4 が使用されます。これらは TARGET_PARAMETERS エントリーで定義されます。

```
#-------------------------------------------------------------------------------------
# TARGET_ADDRESS
#    Defines a management application's address and parameters
#    to be used in sending  notifications.
# Format is:
#  targetAddrName tDomain tAddress tagList targetParams timeout retryCount storageType
TARGET_ADDRESS Target1 UDP 127.0.0.1       traptag trapparms1 - - -
TARGET_ADDRESS Target2 UDP 9.3.149.49   traptag trapparms2 - - -
TARGET_ADDRESS Target3 UDP 9.3.149.49   traptag trapparms3 - - -
TARGET_ADDRESS Target4 UDP 9.53.150.67  traptag trapparms4 - - -
#-------------------------------------------------------------------------------------
```

2. /etc/snmpd.conf ファイル内のトラップ・エントリーで指定されているコミュニティー名は、/etc/snmpdv3.conf ファイル内の TARGET_PARAMETERS エントリーの一部になります。 コミュニティー名は、targetParams 値を使用して特定の TARGET_ADDRESS エントリーにマップする必要があります。 例えば、コミュニティー daniel は trapparms2 を使用してマップされます。この値は、TARGET_ADDRESS エントリーのもとで、IP アドレス 9.3.149.49 にマップされます。 コミュニティー daniel と IP アドレス 9.3.149.49 は元々、/etc/snmpd.conf ファイルの trap エントリーでした。 次の例を参照してください。

```
#-------------------------------------------------------------------------
# TARGET_PARAMETERS
#    Defines the message processing and security parameters
#    to be used in sending notifications to a particular management target.
# Format is:
#  paramsName mpModel securityModel securityName securityLevel storageType
TARGET_PARAMETERS trapparms1 SNMPv1  SNMPv1  public  noAuthNoPriv -
TARGET_PARAMETERS trapparms2 SNMPv1  SNMPv1  daniel  noAuthNoPriv -
TARGET_PARAMETERS trapparms3 SNMPv1  SNMPv1  vasu    noAuthNoPriv -
TARGET_PARAMETERS trapparms4 SNMPv1  SNMPv1  david   noAuthNoPriv -
#-------------------------------------------------------------------------
```

3. /etc/snmpd.conf ファイルの trapmask 情報は、/etc/snmpd.conf ファイルには移行されません。

ステップ 4. smux 情報の移行

移行する必要がある smux 情報がある場合は、これらの行を新規ファイルに直接コピーできます。 このシナリオでは、sampled smux エントリーは、/etc/snmpd.conf ファイルに構成されています。 この行は、/etc/snmpdv3.conf ファイルにコピーする必要があります。

```
#------------------------------------------------------------------------------------------
#       smux <client OIdentifier> <password> <address> <netmask>
smux            1.3.6.1.4.1.2.3.1.2.3.1.1         sampled_password  # sampled
#------------------------------------------------------------------------------------------
```

ステップ 5. snmpd デーモンの停止および開始

/etc/snmpd.conf ファイルの /etc/snmpdv3.conf ファイルへの移行が完了した後で、**snmpd** デーモンを停止してから開始します。 /etc/snmpdv3.conf ファイルに変更を加えるたびに、**snmpd** デーモンを停止してから開始する必要があります。

1. このデーモンを停止するには、次のコマンドを入力します。

   ```
   stopsrc -s snmpd
   ```

2. このデーモンを再始動するには、次のコマンドを入力します。

   ```
   startsrc -s snmpd
   ```

注 : SNMPv3 エージェントを単にリフレッシュしても、**SNMPv1** で機能したようには機能しません。 /etc/snmpdv3.conf ファイルに変更を加える場合は、上述のようにデーモンを停止してから開始する必要があります。 **SNMPv3** でサポートされている動的構成機能を使用してリフレッシュを行うことはできません。

SNMPv3 でのユーザーの作成

このシナリオでは、/etc/snmpdv3.conf および /etc/clsnmp.conf ファイルを手動で編集して、SNMPv3 でユーザーを作成する方法を示します。

ユーザー u1 は、このシナリオで作成されます。 ユーザー u1 には許可キーが付与されますが、プライバシー・キーは付与されません (プライバシー・キーは、snmp.crypto ファイルセットがインストールされている場合にのみ使用可能です)。 HMAC-MD5 プロトコルは、u1 の許可キーを作成するために使用されます。 u1 の構成後に、u1 はグループに入れられます。その後、そのグループにはビューおよびアクセス権が定義されます。 最後に、u1 のトラップ・エントリーが作成されます。

/etc/snmpdv3.conf および /etc/clsnmp.conf ファイルで使用されるそれぞれの個別の値は、32 バイトを超えてはなりません。

考慮事項

- ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ステップ 1. ユーザーの作成

1. 使用するセキュリティー・プロトコル (HMAC-MD5 または HMAC-SHA) を決定します。 このシナリオでは、HMAC-MD5 が使用されます。
2. pwtokey コマンドを使用して、認証キーを生成します。 使用している認証プロトコルによって、また、プライバシー・キーを使用しているかどうかによって出力結果の表示が異なることがあります。 これらのキーは、/etc/snmpdv3.conf および /etc/clsnmp.conf ファイルで使用されます。 ユーザー u1 に使用されるコマンドは、次のとおりです。

   ```
   pwtokey -p HMAC-MD5 -u auth anypassword 9.3.230.119
   ```

指定された IP アドレスは、エージェントが実行中の IP アドレスです。 パスワードは任意のパスワードで構いませんが、将来使用するためにそのパスワードを安全な場所に確実に保管してください。 出力は以下のようになります。

```
Display of 16 byte HMAC-MD5 authKey:
  63960c12520dc8829d27f7fbaf5a0470

Display of 16 byte HMAC-MD5 localized authKey:
  b3b6c6306d67e9c6f8e7e664a47ef9a0
```

3. root 権限により、任意のテキスト・エディターを使用して /etc/snmpdv3.conf ファイルを開きます。
4. ファイルで指定されている形式に従って USM_USER エントリーを追加して、ユーザーを作成します。 *authKey* 値は、**pwtokey** コマンドを使用して生成されたローカライズ認証キーになります。 ユーザー u1 のエントリーは、次のとおりです。

```
#-------------------------------------------------------------------------
# USM_USER entries
#    Defines a user for the User-based Security Model (USM).
# Format is:
#  userName engineID authProto authKey privProto privKey keyType storageType
#
USM_USER u1 - HMAC-MD5 b3b6c6306d67e9c6f8e7e664a47ef9a0 - - L -
#-------------------------------------------------------------------------
```

   - *userName* は、ユーザーの名前です。 この場合は u1 です。
   - *authProto* は、キーの作成時に使用したプロトコルでなければなりません。 この場合は HMAC-MD5 です。
   - *authKey* は、**pwtokey** コマンドを使用して作成されたローカライズ認証キーです。
   - このシナリオではプライバシー・キーを使用していないため、*privProto* および *privkey* は指定されていません。
   - ローカライズ認証キーを使用しているため、*keyType* は L です。
5. /etc/snmpdv3.conf ファイルを保存して閉じます。
6. 任意のテキスト・エディターを使用して SNMP マネージャー上で /etc/clsnmp.conf ファイルを開きます。
7. ファイルで指定された形式に従って新規ユーザーを追加します。 u1 のエントリーは、次のとおりです。

```
#-------------------------------------------------------------------------------------------------
--------
#
# Format of entries:
# winSnmpName targetAgent admin secName password context secLevel authProto authKey
privProto privKey
#
user1  9.3.230.119  SNMPv3  u1  -  -  AuthNoPriv  HMAC-MD5
63960c12520dc8829d27f7fbaf5a0470  -  -
#-------------------------------------------------------------------------------------------------
--------
```

   - *winSnmpName* には任意の値を指定できます。 この値は、**clsnmp** コマンドを使用して SNMP 要求を作成する際に使用されます。
   - *targetAgent* は、エージェントが実行中の IP アドレスです。この値は、認証キーの作成時にも使用されました。
   - SNMPv3 要求を送信しているため、*admin* は SNMPv3 に設定されます。
   - *secName* は、作成するユーザーの名前です。 この場合は u1 です。
   - *seclevel* は、認証を使用するがプライバシーは使用しないように構成されているため、AuthNoPriv に設定されます (結果として、*privProto* および *privKey* には値がありません)。
   - *authproto* は、認証キーの作成時に使用された認証プロトコルに設定されます。
   - *authKey* は、**pwtokey** コマンドによって生成された非ローカライズ・キーです。

8. /etc/clsnmp.conf ファイルを保存して閉じます。

ステップ 2. グループの構成

ここで、ユーザーをグループに入れる必要があります。 このユーザーに付与したいビューおよびアクセス権のすべてを用いて構成されるグループが既にある場合は、このユーザーをそのグループに入れることができます。 他のどのグループも保持していないビューおよびアクセス権をこのユーザーに付与したい場合、またはどのグループも構成されていない場合は、グループを作成し、このユーザーをそのグループに追加します。

このユーザーを新規グループに追加するには、/etc/snmpdv3.conf ファイルに新規 VACM_GROUP エントリーを作成します。 u1 のグループ・エントリーは、次のとおりです。

```
#--------------------------------------------------------------
# VACM_GROUP entries
#    Defines a security group (made up of users or communities)
#    for the View-based Access Control Model (VACM).
# Format is:
#  groupName securityModel securityName storageType
VACM_GROUP group1 USM u1 -
#--------------------------------------------------------------
```

- *groupName* には、任意の名前を指定できます。 これがユーザーのグループの名前になります。 この場合は group1 です。
- *securityModel* は USM に設定されます。これは SNMPv3 セキュリティー・フィーチャーを利用します。
- *securityName* は、ユーザーの名前です。 この場合は u1 です。

ステップ 3. ビューおよびアクセス権の構成

ビューおよびアクセス権は、作成したばかりの新規グループに対して設定する必要があります。 これらのアクセス権は、VACM_VIEW および VACM_ACCESS エントリーを /etc/snmpdv3.conf ファイルに追加することにより設定されます。

1. 新規グループに保持させたいビューおよびアクセス権を決定します。
2. 新規グループがアクセスできる MIB オブジェクトを定義するために、VACM_VIEW エントリーを /etc/snmpdv3.conf ファイルに追加します。 このシナリオでは、group1 は、interfaces、tcp、icmp、および system MIB サブツリーにアクセスできます。 ただし、group1 のアクセスをシステム MIB サブツリー内の sysObjectID MIB 変数に制限します。

```
#--------------------------------------------------------------
# VACM_VIEW entries
#    Defines a particular set of MIB data, called a view, for the
#    View-based Access Control Model.
# Format is:
#  viewName viewSubtree viewMask viewType storageType
VACM_VIEW group1View        interfaces       - included -
VACM_VIEW group1View        tcp              - included -
VACM_VIEW group1View        icmp              - included -
VACM_VIEW group1View        system           - included -
VACM_VIEW group1View        sysObjectID       - excluded -
#--------------------------------------------------------------
```

   - *viewName* は、ビューの名前です。 このシナリオでは group1View です。
   - *viewSubtree* は、アクセス権を付与したい MIB サブツリーです。
   - *viewType* は、定義された MIB サブツリーをビューに組み込むかどうかを決定します。 この場合、すべてのサブツリーが組み込まれますが、system サブツリーの一部である MIB 変数 sysObjectID は除外されます。

3. このグループが上で指定した MIB オブジェクトに対して持つアクセス権を定義するために、VACM_ACCESS エントリーを /etc/snmpdv3.conf ファイルに追加します。 group1 の場合は、読み取り専用アクセス権が付与されます。

```
#-------------------------------------------------------------------------------------------------------------
--
# VACM_ACCESS entries
#    Identifies the access permitted to different security groups
#    for the View-based Access Control Model.
# Format is:
# groupName contextPrefix contextMatch securityLevel securityModel readView writeView notifyView
storageType
VACM_ACCESS  group1 - - AuthNoPriv USM group1View - group1View -
#-------------------------------------------------------------------------------------------------------------
--
```

- *groupName* は、グループの名前です。 この場合は group1 です。
- *securityLevel* は、使用されるセキュリティーのレベルです。 このシナリオでは、認証キーが使用されますが、プライバシー・キーは使用されません。 したがって、この値は AuthNoPriv に設定されます。
- *securityModel* は、使用するセキュリティー・モデル (SNMPv1、SNMPv2c、または USM) です。 このシナリオでは、SNMPv3 セキュリティー・フィーチャーを使用できるように、この値は USM に設定されます。
- *readView* は、グループがどの VACM_VIEW に対する読み取り権限を持つかを決定します。 このシナリオでは、group1View が指定されており、これにより、group1 に group1View VACM_VIEW エントリーへの読み取り権限が付与されます。
- *writeView* は、グループがどの VACM_VIEW に対する書き込み権限を持つかを決定します。 このシナリオでは、group1 に付与されている書き込み権限はありません。
- *notifyView* は、トラップがアクセス・テーブル内のエントリーの制御下で実行されるときに適用されるビューの名前を指定します。

注: 場合によっては、1 つのグループに対して複数の VACM_ACCESS エントリーが必要になることがあります。 グループ内の各ユーザーの認証およびプライバシーの設定値 (noAuthNoPriv、AuthNoPriv、または AuthPriv) が異なる場合は、それに応じて設定された securityLevel パラメーターに複数の VACM_ACCESS エントリーが必要になります。

ステップ 4. ユーザーのトラップ・エントリーの構成

SNMPv3 でのトラップ・エントリーは、NOTIFY、TARGET_ADDRESS、および TARGET_PARAMETERS エントリーを /etc/snmpdv3.conf ファイルに追加することにより作成されます。 TARGET_ADDRESS エントリーはトラップの送信先を指定し、TARGET_PARAMETERS エントリーは TARGET_ADDRESS 情報を group1 にマップします。

NOTIFY エントリーは、デフォルトで構成されています。 デフォルトの NOTIFY エントリーを次に示します。

```
NOTIFY notify1 traptag trap -
```

このシナリオでは、デフォルトのエントリー traptag で指定されている値を使用します。

1. トラップの送信先を指定するために TARGET_ADDRESS エントリーを追加します。

```
#-------------------------------------------------------------------------------
# TARGET_ADDRESS
#    Defines a management application's address and parameters
#    to be used in sending  notifications.
# Format is:
#  targetAddrName tDomain tAddress tagList targetParams timeout retryCount storageType
#-------------------------------------------------------------------------------
TARGET_ADDRESS Target1 UDP 9.3.207.107     traptag trapparms1 - - -
```

- *targetAddrName* には、任意の名前を指定できます。 このシナリオでは Target1 が使用されています。

- *tAddress* は、グループのトラップを送信する必要がある IP アドレスです。
- *tagList* は、NOTIFY エントリーで構成された名前です。 このシナリオでは traptag です。
- *targetParams* には任意の値を指定できます。 このシナリオでは trapparms1 が使用されています。この値は TARGET_PARAMETERS エントリーで使用されます。

2. TARGET_PARAMETERS エントリーを追加します。

```
#--------------------------------------------------------------------
# TARGET_PARAMETERS
#    Defines the message processing and security parameters
#    to be used in sending notifications to a particular management target.
# Format is:
#  paramsName mpModel securityModel securityName securityLevel storageType
#--------------------------------------------------------------------
TARGET_PARAMETERS trapparms1 SNMPv3  USM     u1        AuthNoPriv -
```

- *paramsName* は、TARGET_ADDRESS エントリーの targetParams 値と同じです。この場合は、trapparms1 です。
- *mpModel* は、使用される SNMP のバージョンです。
- *securityModel* は、使用するセキュリティー・モデル (SNMPv1、SNMPv3、または USM) です。 このシナリオでは、SNMPv3 セキュリティー・フィーチャーを使用できるように、この値は USM に設定されます。
- *securityName* は、USM_USER エントリーで指定されているユーザー名です。この場合は、u1 です。
- 認証キーは使用するがプライバシー・キーは使用しないため、*securityLevel* は AuthNoPriv に設定されます。

ステップ 5. snmpd デーモンの停止および開始

/etc/snmpdv3.conf ファイルに変更を加えた後で、**snmpd** デーモンを停止してから開始します。

1. **snmpd** デーモンを停止するには、次のコマンドを入力します。

```
stopsrc -s snmpd
```

2. **snmpd** デーモンを開始するには、次のコマンドを入力します。

```
startsrc -s snmpd
```

これで、新規設定値が効力をもちます。

注: refresh -s snmpd を使用して SNMPv3 エージェントを単にリフレッシュしても、SNMPv1 で機能したようには機能しません。 /etc/snmpdv3.conf ファイルに変更を加える場合は、上述のようにデーモンを停止してから開始する必要があります。 SNMPv3 でサポートされている動的構成機能を使用してリフレッシュを行うことはできません。

ステップ 6. 構成のテスト

構成が正しいか検査するには、SNMP マネージャーで次のコマンドを実行できます。

```
clsnmp -h user1 walk mib
```

ここで、*mib* は、ユーザーがアクセスできる MIB サブツリーです。 このシナリオでは、この値は interfaces、tcp、icmp、または system とすることができます。 構成が正しい場合は、指定されたサブツリーからの情報が表示されます。

正しい出力が得られなかった場合は、この資料内の各ステップを見直し、すべての情報を正しく入力したか確認してください。

SNMPv3 のトラブルシューティング

SNMPv3 を使用している場合、以下の問題が発生する可能性があります。

- 移行の間、/etc/snmpd.conf ファイルに定義されているコミュニティーと SMUX エントリーを /etc/snmpdv3.conf ファイルに移行する必要があります。 この情報の移行については、538 ページの『SNMPv1 から SNMPv3 への移行』を参照してください。
- 要求を発行したが応答が生成されない。

  この問題で最も考えられる原因は、/etc/snmpdv3.conf ファイルまたは /etc/clsnmp.conf ファイル、あるいはその両方のファイルに構成エラーがあることです。 これらのファイルを注意深く調べ、すべての情報が正しく入力されていることを確認してください。 新規ユーザーを作成する場合の、ファイルの編集の詳細については、541 ページの『SNMPv3 でのユーザーの作成』を参照してください。
- 認証キーとプライバシー・キーの両方を使用して新規ユーザーを構成したが、このユーザーを使用したときにエラー・メッセージを受け取った。

  この問題で最も考えられる原因は、**SNMPv3** 暗号化バージョンが稼働していないことです。 次のステップを行って、稼働しているバージョンを判別してください。

  1. 「ps -e|grep snmpd」を実行します。
     - 出力が返ってこない場合は、おそらく **snmpd** デーモンを始動する必要があります。 startsrc -s snmpd を実行します。
     - 出力に snmpdv1 が含まれていれば、**SNMPv1** が稼働しています。 このバージョンが稼働しているときは、**SNMPv1** 要求を発行できます。
     - 出力に snmpdv3ne が含まれていれば、**SNMPv3** 非暗号化バージョンが稼働しています。 AIX オペレーティング・システムをインストールした後、このバージョンがデフォルトで実行されます。このバージョンではプライバシー・キーは使用できません。
     - 出力に snmpdv3e が含まれていれば、**SNMPv3** 暗号化バージョンが稼働しています。これは、別にインストールできる製品です。 **SNMPv3** 暗号化バージョンは、輸出関連法規で許可されている場合は、AIX 拡張パックに入っています。 **SNMPv3** 暗号化バージョンでは、プライバシー・キーを使用できます。
  2. 稼働しているバージョンが意図したものであるか判別します。 意図したものでなければ、次のように、**snmpv3_ssw** コマンドを使用してバージョンを変更します。
     - snmpv3_ssw -1 は **SNMPv1** に切り替えます
     - snmpv3_ssw -n は **SNMPv3** 非暗号化に切り替えます
     - snmpv3_ssw -e は **SNMPv3** 暗号化に切り替えます (インストールされている場合)
- /etc/snmpdv3.conf ファイルを変更してデーモンをリフレッシュしたあとも、変更が有効になっていない。

  /etc/snmpdv3.conf ファイルを変更したら、**SNMP** デーモンを停止して、再始動する必要があります。デーモンのリフレッシュでは動作しません。 次の手順に従ってください。

  1. stopsrc -s snmpd を実行して **SNMP** デーモンを停止します。
  2. startsrc -s snmpd を実行して **SNMP** デーモンを始動します。
- DPI2 サブエージェントが始動したが、そこから MIB 変数を照会できない。

  この問題で最も考えられる原因は、/etc/snmpdv3.conf ファイル内に public コミュニティーが構成されていないことです。 デフォルトでは、AIX に付属の DPI2 サブエージェントは、コミュニティー名 public を使用して **SNMP** エージェントに接続します。 デフォルトで、/etc/snmpdv3.conf ファイル内に public コミュニティーが構成されます。 public コミュニティーを /etc/snmpd.conf ファイルから除去した場合は、次の行をファイルに追加します。

```
VACM_GROUP group1 SNMPv1 public -
VACM_VIEW defaultView 1.3.6.1.4.1.2.2.1.1.1.0 - included -
VACM_ACCESS  group1 - - noAuthNoPriv SNMPv1  defaultView - defaultView -
COMMUNITY public    public     noAuthNoPriv 0.0.0.0     0.0.0.0         -
```

1.3.6.1.4.1.2.2.1.1.1.0 は dpiPortForTCP.0 の OID です。

- 移行前は照会できていた SMUX ピアが管理する MIB 変数を照会できない。

  SMUX エントリーが /etc/snmpdv3.conf ファイルと /etc/snmpd.peers ファイルに あることを確認してください。 新しく SMUX ピアを構成した場合は、それらがこの 2 つのファイルに入っていることも確認してください。

- 独自の MIB 変数セットを実装したが、それを他のユーザーのビューに組み込んだり、除外したりす ることができない。

  /etc/snmpdv3.conf ファイル内の VACM_VIEW エントリーの中に 、MIB 変数名ではなく、MIB 変数の OID を指定する必要があります。

- トラップの受け取りが行われない。

  /etc/snmpdv3.conf 内に、トラップのエントリーを正しく構成していることを確認してください。 さらにトラップが **SNMPv3** トラップの場合は、/etc/clsnmp.conf ファイルも構成する必要があります。 トラップの構成についての説明は、541 ページの『SNMPv3 でのユーザーの作成』を参照してください。

  また、トラップを受け取るように指定されたマシンが (/etc/snmpdv3.conf ファイル内で) トラップを listen していることを確認してください。 このプロセスは、受け取り側のマシンのコマンド・ラインで clsnmp trap を実行すると開始されます。

- なぜ DPI2 サーバーが **SNMPv3** 環境で稼働しないのか?

  SNMPv3 アーキテクチャーでは、**SNMPv3** エージェント自体が DPI2 サーバーを稼働させます。 詳しくは、530 ページの『SNMPv3 アーキテクチャー』を参照してください。

SNMPv1

以下の情報は **SNMPv1** 固有です。 **SNMPv1** を使用する場合、**snmpd** エージェントは、単純な認証方式を使用して、SNMP エージェントの管理情報ベース (MIB) 変数にアクセスできる **SNMP** マネージャー・ステーションを判別します。

この認証方式では、**SNMPv1** 用の **SNMP** アクセス・ポリシーの指定が必要です。 **SNMP** のアクセス・ポリシーにより、各 **SNMP** コミュニティーとアクセス・モードおよび MIB ビューとの関連を明確にした管理上の関係を指します。

SNMP コミュニティー とは、1 つ以上のホストから構成されるグループと、コミュニティー名を示します。 コミュニティー名は、認証のために **SNMP** マネージャーが **SNMP** 要求パケット内に組み込む必要のあるオクテットの文字列です。

アクセス・モード では、特定の **SNMP** エージェントの MIB 変数の検索および変更に対して、コミュニティー内の各ホストに許可されるアクセス権を指定します。 アクセス・モードは、なし、読み取り専用、読み取り/書き込み、書き込み専用 のいずれかでなければなりません。

MIB ビュー では、特定の **SNMP** コミュニティーがアクセスできる 1 つ以上の MIB サブツリーを定義します。 MIB ビューは、MIB ツリー全体でも、MIB ツリー全体のうちの限定した一部分でもかまいません。

SNMP エージェントは、要求を受信すると、要求側ホストの IP アドレスによってコミュニティー名を検査し、その要求側ホストがコミュニティー名で識別される **SNMP** コミュニティーのメンバーかどうかを 判別します。 要求側ホストが **SNMP** コミュニティーのメンバーである 場合は、**SNMP** エージェントは、コミュニティーに関連するアクセス・ポリシーに定義された所定の MIB 変数のための所定のアクセス権を要求側ホストが持っているかどうかを判別します。 すべての基準を満たしている場合は、**SNMP** エージェントはその要求に応じます。 基準を満たしていない場合は、**SNMP** エージェントは *authenticationFailure* トラップを生成するか、または要求側ホストに該当するエラー・メッセージを戻します。

snmpd エージェントの **SNMPv1** アクセス・ポリシーは、ユーザーが構成することができ、/etc/snmpd.conf ファイル内で指定します。 **snmpd** エージェントの **SNMP** アクセス・ポリシーを構成するには、 ファイル参照 の/etc/snmpd.conf ファイルを参照してください。

SNMP デーモンの構成

シンプル・ネットワーク管理プロトコル (SNMP) デーモンは、任意の**伝送制御プロトコル/インターネット・プロトコル (TCP/IP)** ワークステーション・ホスト上で実行可能なバックグラウンド・サーバー・プロセスです。

このデーモンは、**SNMP** エージェントとして動作し、マネージャー・アプリケーションからの **SNMP** 要求の受信、認証、処理を行います。 エージェント機能とマネージャー機能の詳細については、*Communications Programming Concepts* の Simple Network Management Protocol、How a Manager Functions、および How an Agent Functions を参照してください。

注 : SNMP デーモン、**SNMP** エージェント、およびエージェントという用語は、それぞれ同等の意味を持つものとして使用します。

snmpd デーモンを使用するには、最小構成でループバック **TCP/IP** インターフェースがアクティブでなければなりません。 **TCP/IP** を始動する前に、次のコマンドを入力してください。

```
ifconfig lo0 loopback up
```

SNMP デーモンは、**ユーザー・データグラム・プロトコル (UDP)** および**伝送制御プロトコル (TCP)** の予約済みポートにソケットをバインドしようとします。このポートは、/etc/services ファイル内で次のように定義する必要があります。

```
snmp            161/udp
snmp-trap       162/udp
smux            199/tcp
```

RFC 1157 で定められているように、snmp サービスにはポート 161 を割り当てる必要があります。 /etc/services ファイルにより、これらのサービスにはポート 161、162、および 199 が割り当てられています。 別のマシンで /etc/services ファイルのサービスがオフになっている場合には、このマシンのサーバーをサービスの範囲としている /etc/services ファイルでこれらポートを使用可能にしておかないと、**SNMP** デーモンは実行できません。

SNMP デーモンは、始動時のほか、(システム・リソース・コントローラーの制御下で **snmpd** デーモンが始動されている場合に) **refresh** コマンドが発行されたとき、または **kill -1** シグナルが発行されたときに、**SNMP** バージョンの稼働に関する構成ファイルを読み取ります。

/etc/snmpd.conf ファイル

/etc/snmpd.conf 構成ファイルでは、コミュニティー名とそれに関連するアクセス権およびビュー、トラップ通知のためのホスト、ロギング属性、**snmpd** 固有のパラメーター構成、および **SNMPv1** の **SNMP** デーモンの単一マルチプレクサー (SMUX) 構成を指定します。

詳しくは、 ファイル参照 の/etc/snmpd.conf ファイルを参照してください。

SNMP デーモンの処理

シンプル・ネットワーク管理プロトコル (SNMP) デーモンは、マネージャー・アプリケーションからの **SNMP** 要求を処理します。

エージェント機能とマネージャー機能の詳細については、*Communications Programming Concepts* の Simple Network Management Protocol (SNMP)、How a Manager Functions、および How an Agent Functions を参照してください。

SNMP メッセージ処理と認証

すべての要求、トラップ、および応答は、ASN.1 でエンコードされたメッセージのフォーマットで 送信されます。

RFC 1157 によって定義されているメッセージの構造は、次のとおりです。

Version Community PDU

ここで、*Version* は SNMP のバージョン (現在はバージョン 1)、*Community* はコミュニティー名、*PDU* は **SNMP** の要求、応答、またはトラップ・データが格納されるプロトコル・データ単位を示します。 PDU も、ASN.1 規則に従ってエンコードされます。

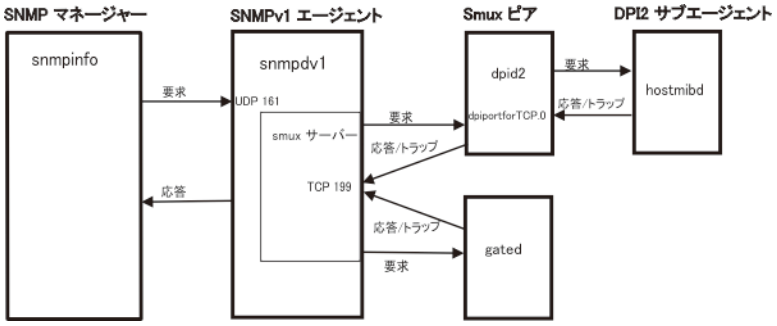


図 *28. SNMPv1* アーキテクチャーの基本部分

この図は、**SNMPv1** アーキテクチャーの例を示しています。 DPI2 サブエージェント、smux ピア、**SNMP** マネージャー、および **SNMP** エージェントが示されています。 さらに、これらの間の通信方法についても示されています。

SNMP デーモンは、すべての **SNMP** プロトコル・メッセージを**伝送制御プロトコル/インターネット・プロトコル (TCP/IP) のユーザー・データグラム・プロトコル (UDP)** によって送受信します。 要求は、予約済みポート 161 上で受け入れられます。 トラップは、 `/etc/snmpd.conf` ファイル内に指定されているホストのうち、予約済みポート 162 を listen しているホストに送信されます。

要求が受信されると、`/etc/snmpd.conf` ファイル内のコミュニティー・エントリーとビュー・エントリーで指定されている IP アドレス、コミュニティー名、許可、およびビューを含むリストと照合され、ソース IP アドレスとコミュニティー名が検査されます。 **snmpd** エージェントは、始動時と **refresh** コマンドまたは **kill -1** シグナルの発行時に、このファイルを読み取ります。 一致するエントリーが見つからないと、要求は無視されます。 一致するエントリーが見つかった場合は、`/etc/snmpd.conf` ファイル内の IP アドレス、コミュニティー、およびビュー名の関連付けに対応するコミュニティー・エントリーおよびビュー・エントリーで指定されている許可に従ってアクセスが許されます。 メッセージと PDU は共に ASN.1 規則に従ってエンコードする必要があります。

この認証方式は、完全なセキュリティーを提供することを目的としたものではありません。 **SNMP** デーモンが get 要求と get-next 要求のためにのみ使用されるなら、通常、セキュリティー上の問題は生じません。 set 要求が許可される場合は、set 特権が制限される場合があります。

詳しくは、 ファイル参照 の`/etc/snmpd.conf` ファイルを参照してください。 詳しくは、 *Communications Programming Concepts* の管理情報ベース (MIB) を参照してください。

SNMP 要求の処理

SNMP デーモンが受信できる要求 PDU には 3 つのタイプがあります。

要求タイプは RFC 1157 内に定義されています。すべての PDU のフォーマットは次のとおりです。

表 *84.* 要求 *PDU* のフォーマット

request-ID	**error-status**	**error-index**	**variable-bindings**
GET	0	0	*VarBindList*
GET-NEXT	0	0	*VarBindList*
SET	0	0	*VarBindList*

「error-status」フィールドと「error-index」フィールドは使用されないため、0 (ゼロ) に設定されていなければなりません。「variable-bindings」フィールドには、要求中の値を持つインスタンス ID の数値フォーマットの可変長リストを指定します。 「variable-bindings」フィールドには、要求中の値を持つインスタンス ID の数値フォーマットの可変長リストを指定します。「request-ID」フィールドの値が SET である場合は、「variable-bindings」フィールドは、インスタンス ID と値からなる対のリストになります。

3 つの要求タイプの詳細については、*Communications Programming Concepts* の管理情報ベース (MIB) データベースの使用を参照してください。

SNMP 応答の処理

応答 PDU のフォーマットは、要求 PDU とほぼ同じです。

表 85. 応答 *PDU* のフォーマット

request-ID	error-status	error-index	variable-bindings
GET-RESPONSE	*ErrorStatus*	*ErrorIndex*	*VarBindList*

要求が正常に処理された場合は、「error-status」フィールドと「error-index」フィールドの値は両方とも 0 (ゼロ) になり、「variable-bindings」フィールドには、インスタンス ID と値のペアの完全なリストが格納されます。

要求 PDU の「variable-bindings」フィールド内のインスタンス ID のうち、正常に処理されなかったものがある場合、SNMP エージェントは処理を停止して、処理できなかったインスタンス ID の索引を「error-index」フィールドに書き込み、エラー・コードを「error-status」フィールドに記録するとともに、部分的に完了した結果リストを「variable-bindings」フィールドにコピーします。

RFC 1157 では、「error-status」フィールドについて次の値を定義しています。

表 86. 「*error-status*」フィールドの値

値	値	説明
noError	0	処理は正常に完了しました (error-index は 0)。
tooBig	1	応答 PDU のサイズは、インストール・システムでの定義の限度を超えます (error-index は 0)。
noSuchName	2	GET 要求タイプと SET 要求タイプの関連 MIB ビュー内にインスタンス ID が存在しないか、GET-NEXT 要求の関連 MIB ビューの MIB ツリー内に後続のインスタンス ID が存在しません (error-index は非ゼロ)。
badValue	3	SET 要求の場合のみ、指定した値は、それに対応するインスタンス ID のタイプ属性の構文互換性がありません (error-index は非ゼロ)。
readOnly	4	定義されていません。
genErr	5	インストール先で定義されたエラーが発生しました (error-index は非ゼロ)。例えば、インプリメンテーションの限度を超える値を割り当てようとしました。

SNMP トラップの処理

トラップ PDU のフォーマットは、次の表の RFC 1157 で定義されています。

表 87. トラップ *PDU* のフォーマット

enterprise	agent-address	generic-trap	specific-trap	time-stamp	variable-bindings
Object ID	*Integer*	*Integer*	*Integer*	*TimeTicks*	*VarBindList*

各フィールドは次のように使用されます。

項目	説明
enterprise	エージェントの提供元ベンダーに割り当てられているオブジェクト ID。 これは **sysObjectID** 変数の値であり、**SNMP** エージェントの実装者ごとに固有の値です。 今回インストールしたエージェントについて割り当てられている値は、**1.3.6.1.4.1.2.3.1.2.1.1.3** または **risc6000snmpd.3** です。
agent-address	トラップを生成したオブジェクトの IP アドレス。
generic-trap	以下の整数。 **0** *coldStart* **1** *warmStart* **2** *linkDown* **3** *linkUp* **4** *authenticationFailure* **5** *egpNeighborLoss* **6** *enterpriseSpecific*
specific-trap	使用しません。将来の開発用に予約済みです。
time-stamp	エージェントが最後に再初期化されてから、トラップを生成したイベントが発生するまでの経過時間 (100 分の 1 秒単位)。
variable-bindings	追加情報。内容は *generic-trap* のタイプによって異なります。

以下の generic-trap 値は、所定のシステム・イベントが検出されたことを示します。

項目	説明
coldStart	エージェントを再初期化中です。 構成データか MIB 変数値のいずれかまたはその両方が変更された可能性があります。 コンピューターを再始動します。
warmStart	エージェントを再初期化中ですが、構成データまたは MIB 変数は変更されていません。 今回実装した **SNMP** エージェントでは、`/etc/snmpd.conf` ファイルの再読み取りが行われたときに *warmStart* トラップが生成されます。 `/etc/snmpd.conf` ファイル内の構成情報は、エージェント構成用の情報であり、SNMP マネージャー・データベースには副次作用を与えません。 測定エポックを再開しないでください。
linkDown	エージェントが、既知の通信インターフェースが 使用不可になっていることを検出しました。
linkUp	エージェントが、既知の通信インターフェースが 使用可能になっていることを検出しました。
authenticationFailure	認証できないメッセージを受信しました。
egpNeighborLoss	**外部ゲートウェイ・プロトコル (EGP)** 隣接が失われました。 この値が生成されるのは、**EGP** を使用して **gated** デーモンを実行しているホスト上でエージェントが実行されている場合のみです。
enterpriseSpecific	使用されません。将来の使用のために予約済みです。

linkDown トラップと *linkUp* トラップでは、variable-bindings リスト内にインスタンス ID/値の対が 1 つ格納されます。 インスタンス ID は、使用不可または使用可能になったアダプターの **ifIndex** を識別し、値は **ifIndex** の値です。 *egpNeighborLoss* のトラップでも、失われた隣接に関する *egpNeighAddr* のインスタンス ID と値から構成されるバインディングが格納されます。

EGP ファミリーの MIB 変数に関する SNMP デーモンのサポート

外部ゲートウェイ・プロトコル (EGP) が使用可能な状態でエージェント・ホストで **gated** デーモンが実行されている場合は、**gated** デーモンがサポートする **EGP** グループ内には、**snmpd** エージェントからアクセスできる管理情報ベース (MIB) 変数がいくつかあります。

次の **EGP** の MIB 変数は、1 つの固有のインスタンスを持ちます。

項目	説明
egpInMsgs	エラーなしで受信した **EGP** メッセージの数。
egpInErrors	エラー状態で受信した **EGP** メッセージの数。
egpOutMsgs	エージェント・ホストで実行されている **gated** デーモンが送信した **EGP** メッセージの総数。
egpOutErrors	リソースが限られていたためにエージェント・ホストの **gated** デーモンが送信できなかった **EGP** メッセージの数。
egpAs	エージェント・ホストの **gated** デーモンの自律システム番号。

次の **EGP** の MIB 変数は、エージェント・ホストの **gated** デーモンが獲得した **EGP** のピアまたは隣接ごとに 1 つのインスタンスを持ちます。

項目	説明
egpNeighState	この **EGP** ピアの状態。 **1** idle **2** acquisition **3** down **4** up **5** cease
egpNeighAddr	この **EGP** ピアの IP アドレス。
egpNeighAs	この **EGP** ピアの自律システム番号。 ゼロ (0) は、当該ピアの自律システム番号がまだ認識されていないことを示します。
egpInNeighMsgs	この **EGP** ピアからエラーなしで受信した **EGP** メッセージの数。
egpNeighInErrs	この **EGP** ピアからエラー状態で受信した **EGP** メッセージの数。
egpNeighOutMsgs	この **EGP** ピアに対してローカルで生成された **EGP** メッセージの数。
egpNeighOutErrs	ローカルで生成された **EGP** メッセージのうち、リソースが限られていたためにこの **EGP** ピアに送信されなかった **EGP** メッセージの数。
egpNeighInErrMsgs	この **EGP** ピアから受信した **EGP** 定義エラー・メッセージの数。
egpNeighOutErrMsgs	この **EGP** ピアに送信された **EGP** 定義エラー・メッセージの数。

項目	説明
egpNeighStateUp	この **EGP** ピアによって **EGP** が UP 状態へ移行した回数。
egpNeighStateDowns	この **EGP** ピアによって **EGP** が UP 状態から他の何らかの状態へ移行した回数。
egpNeighIntervalHello	**EGP** Hello コマンドの 100 分の 1 秒単位の再送インターバル。
egpNeighIntervalPoll	**EGP** poll コマンドの 100 分の 1 秒単位の再送インターバル。
egpNeighMode	この **EGP** ピアのポーリング・モード。 モードは、能動 (1) または受動 (2) のいずれかです。
egpNeighEventTrigger	この制御変数がトリガーとなって、この **EGP** ピアに対してオペレーターが実行する開始イベントと停止イベントが始動されます。 この MIB 変数は、開始 (1) または停止 (2) に設定できます。

gated デーモンが実行されていない場合や、**gated** デーモンが実行されていても、**snmpd** エージェントと通信するように構成されていない場合、または **gated** デーモンが **EGP** 用に構成されていない場合は、これらの値に対する get 要求や set 要求で *noSuchName* エラー応答コードが戻されます。

gated デーモン構成ファイル `/etc/gated.conf` には、次のステートメントが入っている必要があります。

```
snmp     yes;
```

gated デーモンは、**snmpd** デーモンのシンプル・ネットワーク管理プロトコル (SNMP) 単一回線マルチプレクサー (SMUX) プロトコル・ピア、つまり proxy エージェントとして機能するように内部で構成されます。 **gated** デーモンは、始動時に *ipRouteTable* MIB 変数ツリーを **snmpd** エージェントに登録します。 **gated** デーモンが **EGP** 用に構成されている場合、**gated** デーモンは **EGP** の MIB 変数ツリーも登録します。 この登録が完了すると、SNMP マネージャーは **snmpd** エージェントに対して、このエージェント・ホストの **gated** デーモンがサポートする *ipRouteTable* **EGP** MIB 変数に関する要求を正常に発行できるようになります。 **gated** デーモンが稼働中であるときは、すべての MIB 経路指定情報は **gated** デーモンを使用して取得されます。 この場合、*ipRouteTable* に対する set 要求は受け入れられません。

gated デーモンと **snmpd** デーモンとの SMUX 通信は、伝送制御プロトコル (TCP) の予約済みポート 199 を介して行われます。 **gated** デーモンが終了すると、**snmpd** は **gated** デーモンが以前に登録したツリーをすぐに登録解除します。 **gated** デーモンが **snmpd** デーモンより先に始動された場合は、SMUX 関連付けが確立されるまで、**gated** デーモンが定期的に **snmpd** デーモンを検査します。

gated デーモン・クライアントとの SMUX 関連付けを認識して許可するように **snmpd** エージェントを構成するには、ユーザーは `/etc/snmpd.conf` ファイルに SMUX エントリーを追加する必要があります。 この **gated** デーモンに関する SMUX エントリーで指定するクライアント・オブジェクトの ID およびパスワードは、`/etc/snmpd.peers` ファイルで指定したものと一致していなければなりません。

snmpd エージェントは、以下の読み取り/書き込み MIB I 変数および MIB II 変数に対する set 要求をサポートします。

sysContact
このエージェント・ホストの担当者を示すテキスト識別コード。 この情報には、「Bob Smith, 555-5555, ext 5」というような、この担当者の名前と連絡先が示されています。 値は 256 文字に制限されています。 set 要求で、この MIB 変数の文字列が 256 文字を超えている場合は、**snmpd** エージェントがエラー *badValue* を戻し、set 操作は実行されません。 *sysContact* の初期値は、`/etc.snmp.conf` で定義されます。 何も定義されていない場合、この値は null 文字列になります。

インスタンス	値	アクション
0	「文字列」	MIB 変数は「文字列」に設定されます。

sysName
このエージェント・ホストのホスト名。 これは、通常、ノードの完全修飾ドメイン名です。 値は 256 文字に制限されています。 set 要求で、この MIB 変数の文字列が 256 文字を超えている場合は、**snmpd** エージェントがエラー *badValue* を戻し、set 操作は実行されません。

インスタンス	値	アクション
0	「文字列」	MIB 変数は「文字列」に設定されます。

sysLocation
この **snmpd** エージェントが存在するマシンの物理ロケーションを示すテキスト文字列。例えば、「Austin site, building 802, lab 3C-23」となります。 値は 256 文字に制限されています。 set 要求で、この MIB 変数の文字列が 256 文字を超えている場合は、**snmpd** エージェントがエラー *badValue* を戻し、set 操作は実行されません。 *sysLocation* の初期値は、`/etc/snmp.conf` で定義されます。 何も定義されていない場合、この値は null 文字列になります。

インスタンス	値	アクション
0	「文字列」	MIB 変数は「文字列」に設定されます。

ifAdminStatus
エージェント・ホスト上のインターフェース・アダプターについて 設定したい状態。 サポートされる状態は、up と down です。 この状態を testing に設定できますが、この設定は、インターフェースの 操作状態に関しては無効です。

インスタンス	値	アクション
f	1	**ifIndex f** のインターフェース・アダプターが 使用可能にされます。

注 : *ifAdminStatus* の値を up または down に設定することは可能ですが、インターフェースの 操作状態は、実際には変更できません。 このような場合、*ifAdminStatus* の get 要求では *up* 状態が示され、一方、そのインターフェースの *ifOperStatus* では、*down* 状態が示される場合があります。 このような状況が発生した場合は、ネットワーク管理者は再度 set 要求を発行して *ifAdminStatus* を UP に設定し、操作の変更を再試行します。

atPhysAddress
エージェント・ホスト上のアドレス・テーブル・バインディングのハードウェア・アドレス部分 (アドレス解決プロトコル・テーブル内のエントリー)。 これは、*ipNetToMediaPhysAddress* と同じ MIB 変数です。

インスタンス	値	アクション
f.1.n.n.n.n	hh:hh:hh:hh:hh:hh	**ifIndex f** のインターフェースの場合、IP アドレス n.n.n.n に関する既存の ARP テーブル・バインディングがバインディング (n.n.n.n, hh:hh:hh:hh:hh:hh) に置き換えられます。 バインディングが存在しない場合は、新しいバインディングが追加されます。 hh:hh:hh:hh:hh:hh は、12 桁の 16 進数によるハードウェア・アドレスです。

atN0etAddress
atPhysAddress で指定されているハードウェア・アドレスまたは物理アドレスに対応する IP アドレス。これは、*ipNetToMediaNetAddress* と同じ MIB 変数です。

インスタンス	値	アクション
f.1.n.n.n.n	m.m.m.m	**ifIndex f** のインターフェースの場合、IP アドレス n.n.n.n に関する既存の ARP テーブル・エントリーが IP アドレス m.m.m.m に置き換えられます。

ipForwarding
このエージェント・ホストがデータグラムを転送中かどうかを示します。

表 *88. ipforwarding*

インスタンス	値	アクション
0	1	エージェント・ホストに複数のアクティブ・インターフェースがある場合、**TCP/IP** のカーネルは、パケットを転送するように構成されています。 エージェント・ホストにアクティブ・インターフェースが 1 つのみある場合は、set 要求は失敗します。
0	2	エージェント・ホスト上の **TCP/IP** のカーネルは、パケットを転送しないように構成されています。

ipDefaultTTL
エージェントのホストが発信したデータグラムの IP ヘッダーに挿入されているデフォルト存続時間 (TTL) の値。

インスタンス	値	アクション
0	n	IP プロトコル・サポート機能が使用するデフォルトの存続時間の値が、整数 n に設定されます。

ipRouteDest
経路テーブル内にある経路の宛先 IP アドレス。

インスタンス	値	アクション
n.n.n.n	m.m.m.m	経路 n.n.n.n の宛先経路が IP アドレス m.m.m.m に設定されます。

ipRouteNextHop
エージェントのホストから宛先 IP アドレスに到達するために使用できるゲートウェイ (ルーティング・テーブル内のエントリー)。

インスタンス	値	アクション
n.n.n.n	m.m.m.m	ゲートウェイ m.m.m.m を使用してネットワーク n.n.n.n に到達するためのルーティング・テーブル・エントリーがルーティング・テーブルに追加されます。IP アドレス n.n.n.n のホスト部は、ネットワーク・アドレスを示す 0 でなければなりません。

ipRouteType
エージェントのホスト上のルーティング・テーブル・エントリーの状態 (エントリーを削除するときに使用します)。

インスタンス	値	アクション
h.h.h.h	1	ホスト IP アドレス h.h.h.h への経路が削除されます。
n.n.n.n	2	ホスト IP アドレス n.n.n.n への経路が削除されます。

ipNetToMediaPhysAddress
エージェントのホスト上のアドレス・テーブル・バインディングのハードウェア・アドレス部分 (ARP テーブル内のエントリー)。 これは、*atPhysAddress* と同じ MIB 変数です。

インスタンス	値	アクション
f.1.n.n.n.n	hh:hh:hh:hh:hh:hh	**ifIndex f** のインターフェースの場合、IP アドレス n.n.n.n に関する既存の ARP テーブル・バインディングがバインディング (n.n.n.n, hh:hh:hh:hh:hh:hh) に置き換えられます。 バインディングが存在しない場合は、新しいバインディングが追加されます。 hh:hh:hh:hh:hh:hh は、12 桁の 16 進数によるハードウェア・アドレスです。

ipNetToMediaNetAddress
ipNetToMediaPhysAddress で指定されているハードウェア・アドレスまたは物理アドレスに対応する IP アドレス。 これは、*atNetAddress* と同じ MIB 変数です。

インスタンス	値	アクション
f.1.n.n.n.n	m.m.m.m	**ifIndex f** のインターフェースの 場合、IP アドレス n.n.n.n に関する既存の ARP テーブル・エントリーが IP アドレス m.m.m.m に置き換えられます。

ipNetToMediaType
IP アドレスから物理アドレスへのマッピングのタイプ。

インスタンス	値	アクション
f.1.n.n.n.n	1	**ifIndex f** のインターフェースの 場合、IP アドレスから物理アドレスへの既存の ARP バインディングに対して、マッピング・タイプが 1 (その他) に設定されます。

インスタンス	値	アクション
f.1.n.n.n.n	2	**ifIndex f** のインターフェースの 場合、IP アドレスから物理アドレスへの既存の ARP バインディングに対して、マッピング・タイプが 2 (無効) に設定されます。 副次作用として、**ipNetMediaTable** 内の対応するエントリーが無効になります。つまり、インターフェースは、この **ipNetToMediaTable** エントリーとの関連を解除されます。
f.1.n.n.n.n	3	**ifIndex f** のインターフェースの 場合、IP アドレスから物理アドレスへの既存の ARP バインディングに対して、マッピング・タイプが 3 (動的) に設定されます。
f.1.n.n.n.n	4	**ifIndex f** のインターフェースの 場合、IP アドレスから物理アドレスへの既存の ARP バインディングに対して、マッピング・タイプが 4 (静的) に設定されます。

snmpEnableAuthenTraps
snmpd エージェントが *authenticationFailure* トラップを生成するように構成されているかどうかを示します。

インスタンス	値	アクション
0	1	**snmpd** エージェントは、認証エラー・トラップを生成します。
0	2	**snmpd** エージェントは、認証エラー・トラップを生成しません。

smuxPstatus
SMUX プロトコル・ピアの状況 (SMUX ピアを削除するときに使用します)。

インスタンス	値	アクション
n	1	**snmpd** エージェントは何も実行しません。
n	2	**snmpd** エージェントは、SMUX ピア n との通信を停止します。

smuxTstatus
SMUX の MIB ツリーの状況 (MIB ツリーのマウントを削除するときに使用します)。

インスタンス	値	アクション
l.m.m.m._ _ _ .p	1	**snmpd** エージェントは何も実行しません。
l.m.m.m._ _ _ .p	2	MIB ツリー m.m.m... の SMUX マウントをアンマウントします。*l* は MIB ツリー・インスタンスの長さで、*p* は smuxTpriority です。

以下の変数は、RFC 1229 で定義されている設定可能な変数です。 これらの変数は、 **snmpd** デーモンによって設定できます。 基本のデバイスによっては、このような変数を設定できない場合があります。 どの変数がサポートされていて、どの変数がサポートされていないのかを、それぞれのデバイスごとに調べてください。

ifExtnsPromiscuous
特定のデバイスの混在モードの状況。 これによって、特定のデバイスの混在モードを使用可能または使用不可にします。 **snmpd** のアクションは最後に実行され、これで混在モードが終了します。 **snmpd** にオフにするように指示すると、マシン上の他のアプリケーションとは無関係に混在モードは完全にオフになります。

インスタンス	値	アクション
n	1	デバイス n の混在モードをオンにします。
n	2	デバイス n の混在モードをオフにします。

ifExtnsTestType
テストの初期化変数。 この変数を設定すると、適切なテストが当該デバイスについて実行されます。 オブジェクト ID がこの変数の値です。 具体的な値は、デバイス・タイプと実行するテストによって異なります。 現在のところ、**snmpd** が実行すべきであると認識する定義済みテストは testFullDuplexLoopBack テストだけです。

インスタンス	値	アクション
n	oid	oid で指定したテストを開始します。

ifExtnsRcvAddrStatus
アドレス状況変数。 この変数を指定すると、指定したアドレスが適切な継続期間で存続するようになります。 **snmpd** で設定が許されるのは一時アドレスだけです。なぜなら、このデーモンはデバイスのオブジェクト・データ・マネージャー (ODM) レコードを設定できず、マルチキャスト・アドレスまたは ブロードキャスト・アドレスだけしか設定できないからです。

インスタンス	値	アクション
n.m.m.m.m.m.m	1	一時アドレスまたは永久アドレス以外のアドレスとして、アドレスを追加します。
n.m.m.m.m.m.m	2	アドレスを使用されないように除去します。
n.m.m.m.m.m.m	3	一時アドレスとしてアドレスを追加します。
n.m.m.m.m.m.m	4	永久アドレスとしてアドレスを追加します。

以下に示す変数は、RFC 1231 で定義されている設定可能な変数です。 これらの変数は、 **snmpd** デーモンによって設定できます。 基本のデバイスによっては、このような変数を設定できない場合があります。 どの変数がサポートされているのかを、それぞれのデバイスごとに調べる必要があります。

dot5Commands
トークンリング・デバイスで実行するコマンド。

インスタンス	値	アクション
n	1	何もしません。 返されます。
n	2	トークンリング・デバイスにオープンするように指示します。
n	3	トークンリングにリセットするように指示します。
n	4	トークンリング・デバイスにクローズするように指示します。

dot5RingSpeed
現在のリング速度または通信容量。

インスタンス	値	アクション
n	1	未知の速度
n	2	1 メガビットのリング速度。
n	3	4 メガビットのリング速度。

インスタンス	値	アクション
n	4	16 メガビットのリング速度

dot5ActMonParticipate
このオブジェクトは、そのデバイスがアクティブ・モニター選択プロセスに参加するかどうかを指定します。

インスタンス	値	アクション
n	1	参加
n	2	不参加

dot5Functional
トークンリング・デバイスが受信するフレームの送信元アドレスを指定するための機能マスク。

インスタンス	値	アクション
n	m.m.m.m.m.m	機能マスクが設定されます。

以下の複合タイマー操作変数は、RFC では読み取り専用として定義されていますが、これらを読み取り/書き込み可能にすることが推奨されます。 RFC を閲覧して、これらの相互作用を完全に理解してください。要求側では、**snmpd** によってこの変数を設定できますが、デバイス側では設定できない場合があります。詳しくは、デバイス・ドライバーの資料を参照してください。

- dot5TimerReturnRepeat
- dot5TimerHolding
- dot5TimerQueuePDU
- dot5TimerValidTransmit
- dot5TimerNoToken
- dot5TimerActiveMon
- dot5TimerStandbyMon
- dot5TimerErrorReport
- dot5TimerBeaconTransmit
- dot5TimerBeaconReceive.

SNMP デーモンでは以下の変数を設定できます。 このデーモンは、FDDI ステーション・マネージメント (SMT) 7.2 プロトコル規格を使用して情報を取得します。これはマイクロコード・レベルで決定されます。FDDI の資料を参照してマイクロコードを調べ、SMT 7.2 マイクロコードが使用されていることを確認してください。

fddimibSMTUserData
32 バイトのユーザー情報を保持する変数。

インスタンス	値	アクション
n	文字列	32 バイトのユーザー情報を格納します。

fddimibSMTConfigPolicy
構成ポリシーの状況。特に、保持ポリシーの使用状況。

インスタンス	値	アクション
n	0	保持ポリシーを使用しません。
n	1	保持ポリシーを使用します。

fddimibSMTConnectionPolicy
FDDI ノードの接続ポリシーの状況。 設定できる具体的な値については、RFC 1512 を参照してください。

インスタンス	値	アクション
n	k	接続ポリシーを定義します。

fddimibSMTTNotify
近隣通知プロトコルで使用する秒単位のタイマー。 範囲は 2 秒から 30 秒までで、デフォルト値は 30 秒です。

インスタンス	値	アクション
n	k	タイマー値を定義します。

fddimibSMTStatRptPolicy
状況レポート・フレーム生成の状況。

インスタンス	値	アクション
n	1	実行されたイベントについてノードで状況レポート・フレームを生成することを示します。
n	2	状況レポート・フレームをノードが生成しないことを示します。

fddimibSMTTraceMaxExpiration
この変数は、トレースの最大タイマー終了値を定義します。

インスタンス	値	アクション
n	k	最大タイマー終了をミリ秒単位で定義します。

fddimibSMTStationAction
この変数は、SMT エンティティーに特定のアクションを実行させるために使用します。 この変数の詳細については RFC を参照してください。

インスタンス	値	アクション
n	k	SMT エンティティーに関するアクションを定義します。 値の範囲は、1 から 8 までです。

fddimibMACRequestedPaths
メディア・アクセス制御 (MAC) を挿入するパスを定義します。

インスタンス	値	アクション
n.n	k	MAC のために要求されたパスを定義します。

fddimibMACFrameErrorThreshold
MAC 状況レポートが生成される時点を決定するしきい値。 ここで指定した回数を超えたエラーが発生すると状況レポートが発行されます。

インスタンス	値	アクション
n.n	k	ここで指定された回数を超えたエラーが発生すると、MAC 状況レポートが発行されます。

fddimibMACMAUnitdataEnable
この変数は、RMT の **MA_UNITDATA_Enable** フラグの値を決定します。 このフラグのデフォルト値と初期値は真 (1) です。

インスタンス	値	アクション
n.n	1	MA_UNITDATA_Enable フラグを真としてマークします。
n.n	2	MA_UNITDATA_Enable フラグを偽としてマークします。

fddimibMACNotCopiedThreshold
MAC 条件レポートが生成される時点を決定するしきい値。

インスタンス	値	アクション
n.n	k	ここで指定された回数を超えたエラーが発生すると、MAC 条件レポートが発行されます。

以下の 3 つの変数は、相互に作用するタイマー変数です。 これらの変数のいずれかを変更する前に、**RFC 1512** で定義されている各変数の意味をよく理解してください。

- fddimibPATHTVXLowerBound
- fddimibPATHTMaxLowerBound
- fddimibPATHMaxTReq

fddimibPORTConnectionPolicies
指定したポートの接続ポリシーを指定します。

インスタンス	値	アクション
n.n	k	指定したポートの接続ポリシーを定義します。

fddimibPORTRequestedPaths
この変数は、許可されているパスのリストです。リストの各エレメントは、ポートの許可されているパスを定義します。 最初のオクテットは「なし」、2 番目のオクテットは「ツリー」、3 番目のオクテットは「ピア」にそれぞれ対応します。

インスタンス	値	アクション
n.n	ccc	ポートのパスを定義します。

fddimibPORTLerCutoff
リンク接続が断たれるリンク・エラー率の推定値。 値の範囲は 10**-4 から 10**-15 までであり、10 を底とする対数の絶対値 (デフォルトは 7) で報告されます。

項目	説明	
インスタンス	**値**	**アクション**
n.n	k	ポートの LerCutoff を定義します。

fddimibPORTLerAlarm
リンク接続で警告が生成されるリンク・エラー率の推定値。 値の範囲は 10**-4 から 10**-15 までであり、推定値の 10 を底とする対数の絶対値 (デフォルトは 8) として報告されます。

インスタンス	値	アクション
n.n	k	ポートの LerAlarm を定義します。

fddimibPORTAction

この変数は、ポートで特定のアクションを実行するために使用します。 この変数の詳細については RFC を参照してください。

インスタンス	値	アクション
n	k	定義したポートに関するアクションを定義します。 値の範囲は、1 から 6 までです。

注 : RFC 1213 では、*atEntry* テーブルと *ipNetToMediaEntry* テーブル内のすべての変数を読み取り/書き込み可能として定義しています。 ただし、set 要求のサポートがインプリメントされているのは、*atEntry* テーブルの *atPhysAddress* 変数および *atNetAddress* 変数と、*ipNetToMediaEntry* テーブルの *ipNetToMediaPhysAddress* 変数、*ipNetToMediaNetAddress* 変数、および *ipNetToMediaType* 変数についてのみです。 この 2 つのテーブル内のサポートされない残りの属性を指定した set 要求を受け入れるために、残りの変数、つまり *atIfIndex* と *ipNetToMediaIfIndex* に対する set 要求も受け入れられます。 エラー応答が set 要求の発信元に戻ることはありませんが、その後 get 要求を出すと、元の値が保持されていることが分かります。

ipRouteEntry テーブル内の変数については、RFC 1213 では *ipRouteProto* 以外のすべての変数を読み取り/書き込み可能として定義しています。 前述のように、set 要求のサポートは、*ipRouteDest*、*ipRouteNextHop*、および *ipRouteType* の各変数についてのみインプリメントされています。 ただし、サポートされないいくつかの経路属性を指定した set 要求を受け付けるために、*ipRouteEntry* テーブル内の残りの変数、つまり *ipRouteIfIndex*、*ipRouteMetric1*、*ipRouteMetric2*、*ipRouteMetric3*、*ipRouteMetric4*、*ipRouteMetric5*、*ipRouteAge*、および *ipRouteMask* に対する set 要求も受け入れられます。 エラー応答が set 要求の発信元に戻ることはありませんが、その後 get 要求を出すと、元の値が保持されていることが分かります。 **snmpd** デーモンは、**routed** デーモンによる経路指定の調整は行いません。 **gated** デーモンが実行されており、*ipRouteTable* を **snmpd** デーモンに登録している場合、*ipRouteTable* に対する set 要求は受け入れられません。

RFC 1229 には **snmpd** で設定可能な変数が定義されています。 実際の偏差については、以前のエントリーを参照してください。

以下の例では、**snmpinfo** コマンドを使用しています。 この例では、**snmpinfo** のデフォルト・コミュニティー名である public が、それぞれの MIB サブツリーに対する読み取り/書き込みアクセス権を持つことを前提としています。

```
snmpinfo -m set sysContact.0="Primary contact: Bob Smith, office phone: 555-5555,
beeper: 9-123-4567. Secondary contact: John Harris, phone: 555-1234."
```

このコマンドは、sysContact.0 の値を指定された文字列に設定します。 sysContact.0 のエントリーが既に存在する場合は、既存のエントリーが指定した文字列に置き換えられます。

```
snmpinfo -m set sysName.0="bears.austin.ibm.com"
```

このコマンドは、sysName.0 の値を、指定した文字列に設定します。 sysName.0 のエントリーが既に存在する場合は、既存のエントリーが指定した文字列に置き換えられます。

```
snmpinfo -m set sysLocation.0="Austin site, building 802, lab 3C-23, southeast
corner of the room."
```

このコマンドは、sysLocation.0 の値を、指定した文字列に設定します。 sysLocation.0 のエントリーが既に存在する場合は、既存のエントリーが指定した文字列に置き換えられます。

```
snmpinfo -m set ifAdminStatus.2=2
```

このコマンドは、ifIndex が 2 であるネットワーク・インターフェース ・アダプターを使用不可にします。 割り当てられている値が 1 の場合、そのインターフェース ・アダプターは使用可能になります。

```
snmpinfo -m set atPhysAddress.2.1.192.100.154.2=02:60:8c:2e:c2:00
snmpinfo -m set ipNetToMediaPhysAddress.2.1.192.100.154.2=02:60:8c:2e:c2:00
```

この 2 つのコマンドは、ARP テーブル・エントリー内の 192.100.154.2 のハードウェア・アドレスを 02:60:8c:2e:c2:00 に変更します。 この 2 つのコマンドは、同一の ARP テーブル・エントリーに作用します。 MIB 変数 *atPhysAddress* は、使用すべきでない変数であり、MIB 変数 *ipNetToMediaPhysAddress* に置き換えられます。 したがって、*atPhysAddress* と *ipNetToMediaPhysAddress* は、TCP/IP カーネルの ARP テーブル内にある同一の構造にアクセスします。

```
snmpinfo -m set atNetAddress.2.1.192.100.154.2=192.100.154.3
snmpinfo -m set ipNetToMediaNetAddress.2.1.192.100.154.2=192.100.154.3
```

これらのコマンドは、ARP テーブル・エントリー内の 192.100.154.2 の IP アドレスを 192.100.154.3 に変更します。 この 2 つのコマンドは、同一の ARP テーブル・エントリーに作用します。 MIB 変数 *atNetAddress* は使用すべきでない変数であり、MIB 変数 *ipNetToMediaNetAddress* に置き換えられます。 したがって、*atNetAddress* と *ipNetToMediaNetAddress* は、TCP/IP カーネルの ARP テーブル内にある同一の構造にアクセスします。

```
snmpinfo -m set ipForwarding.0=1
```

このコマンドは、エージェント・ホストに up 状態のインターフェースが 複数ある場合、パケットを転送できるように **TCP/IP** カーネルを設定します。 ホストにアクティブなインターフェースが 1 つしかない場合は、この set 要求が失敗し、**snmpd** エージェントからエラー *badValue* が戻されます。

```
snmpinfo -m set ipDefaultTTL=50
```

このコマンドは、デフォルトの存続時間 (TTL) を使用する IP データグラムが、廃棄されるまでに最高 50 個のゲートウェイを通過できるようにします。 各ゲートウェイがデータグラムを処理するたびに、ゲートウェイは、「time-to-live (存続時間)」フィールドから 1 を減算します。 また、各ゲートウェイは、データグラムを次の宛先に渡す前に、データグラムが処理されるまでそのゲートウェイで待機した秒数を「time-to-live (存続時間)」フィールドから減算します。

```
snmpinfo -m set ipRouteDest.192.100.154.0=192.100.154.5
```

このコマンドは、経路 192.100.154 が既に存在するものとして、192.100.154.0 に関連する経路の宛先 IP アドレスを IP アドレス 192.100.154.5 に設定します。

```
snmpinfo -m set ipRouteNextHop.192.100.154.1=129.35.38.47
```

このコマンドは、経路 192.100.154.1 が既に存在するものとして、ゲートウェイ・ホスト 129.35.38.47 を使用してホスト 192.100.154.1 への経路を設定します。

```
snmpinfo -m set ipRouteNextHop.192.100.154.0=192.100.154.7
```

このコマンドは、経路 192.100.154.0 が既に存在するものとして、ゲートウェイ・ホスト 192.100.154.7 を使用してクラス C ネットワーク 192.100.154 への経路を設定します。 アドレスのホスト部はネットワーク・アドレスを示す 0 でなければなりません。

```
snmpinfo -m set ipRouteType.192.100.154.5=2
```

このコマンドは、ホスト 192.100.154.5 への経路をすべて削除します。

```
snmpinfo -m set ipRouteDest.129.35.128.1=129.35.128.1
                ipRouteType.129.35.128.1=3
            ipRouteNextHop.129.35.128.1=129.35.128.90
```

このコマンドは、ホスト 129.35.128.90 から 129.35.128.1 への新しい経路をゲートウェイとして作成します。

```
snmpinfo -m set ipNetToMediaType.2.1.192.100.154.11=4
```

このコマンドは、192.100.154.11 の ARP テーブル・エントリーを静的に設定します。

```
snmpinfo -m set snmpEnableAuthenTraps=2
```

このコマンドは、指定したホスト上の **snmpd** エージェントが *authenticationFailure* トラップを生成しないようにします。

```
snmpinfo -m set smuxPstatus.1=2
```

このコマンドは SMUX ピア 1 を無効にします。その結果として、**snmpd** エージェントとこの SMUX ピアとの間の接続が終了します。

```
snmpinfo -m set smuxTstatus.8.1.3.6.1.2.1.4.21.0=2
```

このコマンドは、SMUX ツリー 1.3.6.1.2.1.4.21 のマウント、つまり *ipRoute* テーブルを無効にするか、または除去します。 インスタンスの最初の番号は、SMUX ツリー ID 内のレベル数を示します。 インスタンスの最後の番号は、smuxTpriority を示します。 この例では、1.3.6.1.2.1.4.21 というように SMUX ツリーの ID に 8 つのレベルがあります。 優先順位 0 は最高位の優先順位です。

```
snmpinfo -m set ifExtnsPromiscuous.1=1 ifExtnsPromiscuous.2=2
```

このコマンドは、インターフェース・テーブル内の最初のデバイスの混在モードをオンにし、インターフェース・テーブル内の 2 番目のデバイスの混在モードをオフにします。

```
snmpinfo -m set ifExtnsTestType.1=testFullDuplexLoopBack
```

このコマンドは、インターフェース 1 で testFullDuplexLoopBack テストを開始します。

```
snmpinfo -m set ifExtnsRcvAddrStatus.1.129.35.128.1.3.2=2
```

このコマンドは、インターフェース 1 の受け入れ可能アドレス・リストから物理アドレス 129.35.128.1.3.2 を除去するように指示します。

```
snmpinfo -m set dot5Commands.1=2
```

このコマンドは、最初のインターフェースをオープンするように 指示します。

```
snmpinfo -m set dot5RingSpeed.1=2
```

このコマンドは、最初のインターフェースのリング 速度を 1 メガビットに設定するように指示します。

```
snmpinfo -m set dot5ActMonParticipate.1=1
```

このコマンドは、最初のインターフェースのアクティブ ・モニター選択プロセスへの参加を指示します。

```
snmpinfo -m set dot5Functional.1=255.255.255.255.255.255
```

このコマンドは、すべてを許可するように機能アドレス・マスクを設定します。

```
snmpinfo -m set fddimibSMTUserData.1="Greg's Data"
```

このコマンドは、最初の SMT エンティティー上のユーザー・データを「Greg's Data」に設定します。

```
snmpinfo -m set fddimibMACFrameErrorThreshold.1.1=345
```

このコマンドは、最初の SMT エンティティーの 最初の MAC 上でフレーム・エラーのしきい値を 345 に設定します。

注 : 前述したすべての変数は、上に列記した変数設定方法のいずれかで設定します。

プロトコルと IP アドレスの詳細については、148 ページの『アドレス解決プロトコル (ARP)』 および 175 ページの『インターネット・アドレス』 を参照してください。

SNMP デーモンのトラブルシューティング

SNMP デーモンのトラブルシューティングに関するヒントには、終了問題、コミュニティー・エントリーでの MIB 変数のアクセス問題、noSuchName 問題、エージェントからの無応答問題、デーモンの障害などの解決があります。

デーモンの終了問題

snmpd エージェントが期待どおりに動作しない場合は、問題を判別して訂正するために、以下のセクションを参考にしてください。 なんらかのタイプのロギングを指定して **snmpd** エージェントを始動することが強く推奨されます。 また、**snmpd** デーモンを起動すると問題が発生する場合は、デーモン機能レベルと DEBUG 重大度レベルでロギングを実行するよう **syslogd** デーモンをセットアップすることが強く推奨されます。

snmpd デーモンが起動直後に終了してしまう場合について、考えられる障害の原因と解決法を以下に示します。

- **snmpd** デーモンが終了した理由は、**snmpd** ログ・ファイルまたは構成済み **syslogd** ログ・ファイルに記録されます。 ログ・ファイルを調べて、**FATAL** エラー・メッセージがないか確認してください。

  解決法: 問題を訂正し、**snmpd** デーモンを再始動してください。

- **snmpd** コマンド・ラインの使用法が正しくありません。 システム・リソース・コントローラー (SRC) を使用しないで **snmpd** コマンドを起動した場合は、正しい使用法のステートメントが画面にエコーされます。 SRC の制御下で **snmpd** デーモンを起動した場合、使用法メッセージは画面にエコーされません。 ログ・ファイルを調べて、使用法メッセージを見つけてください。

  解決法: ステートメントを正しく使用して、**snmpd** コマンドを起動してください。

- **snmpd** デーモンを起動するのは、root ユーザーでなければなりません。

  解決法: root ユーザーに切り替えて、**snmpd** デーモンを再始動してください。

- snmpd.conf ファイルのオーナーは、root ユーザーでなければなりません。 **snmpd** エージェントは、構成ファイルの所有権を検査します。 このファイルのオーナーが root ユーザーでない場合は致命的なエラーとなり、**snmpd** エージェントは終了します。

  解決法: root ユーザーであることを確認し、構成ファイルの所有権を root ユーザーに変更してから、**snmpd** デーモンを再始動してください。

- **snmpd.conf** ファイルが存在しなければなりません。 **snmpd** コマンド・ラインで構成ファイルに **-c** フラグを指定しなかった場合は、**/etc/snmpd.conf** ファイルは存在しません。 **/etc/snmpd.conf** ファイルが誤って除去されてしまった場合は、**bos.net.tcp.client** イメージを再インストールするか、あるいは **snmpd.conf** ファイルの管理ページにある定義に従って適切な構成エントリーを使用し、そのファイルを再作成してください。 **snmpd** コマンド・ラインで **-c** フラグを使用して構成ファイルを指定した場合は、そのファイルが存在すること、およびそのファイルのオーナーが root ユーザーであることを確認してください。 その構成ファイルの絶対パス名とファイル名を指定する必要があります。そうしないと、デフォルトの **/etc/snmpd.conf** ファイルが使用されます。

  解決法 : 指定した構成ファイルが存在すること、およびそのファイルのオーナーが root ユーザーであることを確認してください。 **snmpd** デーモンを再始動します。

- udp ポート 161 は、バインド済みです。 **snmpd** デーモンがまだ実行されていないことを確認してください。 **snmpd** デーモン・プロセスが実行されているかどうかを判別するには、**ps -eaf | grep snmpd** コマンドを発行します。 udp ポート 161 にバインドできるのは、1 つの **snmpd** エージェントのみです。

  解決法: 既存の **snmpd** エージェントを強制終了するか、あるいは別の **snmpd** デーモン・プロセスを始動しないでください。

デーモンの障害

refresh または **kill -1** シグナルを発行したときに **snmpd** デーモンに障害が起きた場合について、考えられる障害の原因と解決法を以下に示します。

- **snmpd** デーモンが終了した理由は、**snmpd** ログ・ファイルまたは構成済み **syslogd** ログ・ファイルに記録されます。 ログ・ファイルを調べて、FATAL エラー・メッセージがないか確認してください。

  解決法: 問題を訂正し、**snmpd** デーモンを再始動してください。

- **snmpd** デーモンの起動時には、構成ファイルの絶対パス名とファイル名を必ず指定してください。 **snmpd** デーモンは、起動時に fork してディレクトリーをルート・ディレクトリーに変更します。 構成ファイルの絶対パス名が指定されていないと、**snmpd** エージェントはリフレッシュ時にそのファイルを見つけられません。 これは致命的エラーであり、**snmpd** エージェントは終了します。

  解決法: **snmpd** 構成ファイルの絶対パス名およびファイル名を指定します。 構成ファイルのオーナーが root ユーザーであることを確認します。 **snmpd** デーモンを再始動します。

- **snmpd** 構成ファイルがまだ存在していることを確認してください。 **snmpd** エージェントの起動後に、このファイルを誤って除去してしまった可能性があります。 **snmpd** エージェントが構成ファイルをオープンできない場合、**snmpd** エージェントは終了します。

  解決法: **snmpd** 構成ファイルを再作成し、その構成ファイルのオーナーが root ユーザーであることを確認してから、**snmpd** デーモンを再始動してください。

MIB 変数のアクセス問題

管理情報ベース (MIB) 変数に **snmpd** エージェントからアクセスできない場合や、**snmpd** エージェントは動作しているのに、**snmpd** エージェントからの応答を待機している**シンプル・ネットワーク管理プロトコル (SNMP)** マネージャー・アプリケーションがタイムアウトになる 場合は、以下の処置をとってください。

- **netstat -in** コマンドで、**snmpd** エージェントが実行されているホストのネットワーク接続を検査します。 lo0、ループバック、およびデバイスが UP 状態であることを確認します。 デバイスが DOWN 状態である場合は、lo0 の左側に * (アスタリスク) が表示されます。 **snmpd** エージェントが要求を処理するためには、lo0 が up 状態でなければなりません。

  解決法 : 以下のコマンドを発行して、ループバック ・インターフェースを 始動してください。

  ```
  ifconfig lo0 inet up
  ```

- 要求の発行元であるホストへの経路を **snmpd** デーモンが持っていることを検証します。

  解決策: **snmpd** デーモンが稼働中のホスト上で、**route add** コマンドの発行元であるホストへの経路を追加してください。 詳しくは、**route** コマンドを参照してください。

- ホスト名とホスト IP アドレスの値が同じであることを確認します。

  解決法: ホストの IP アドレスと対応するように、ホスト名を再設定してください。

- ローカル・ホスト が lo0 の IP アドレスになるように定義されているかどうかを検査します。

  解決法: lo0 の IP アドレスで使用しているものと同じアドレス (通常は 127.0.0.1) になるように、ローカル・ホスト を定義します。

コミュニティー・エントリーでの MIB 変数のアクセス問題

MIB ビュー名を使用してコミュニティー・エントリーを構成ファイル内に指定したが、MIB 変数にアクセスできない場合、次のことを調べてください。

- コミュニティー・エントリーが正しく指定されていることを確認します。 ビュー名をコミュニティー・エントリー内に指定した場合、コミュニティー内のすべてのフィールドが必ず必要です。

  解決法 : コミュニティー・エントリーのすべてのフィールドを 構成ファイル内で指定します。 **snmpd** エージェントをリフレッシュして、要求を再試行してください。

- コミュニティー・エントリー内のアクセス・モードが要求タイプと対応していることを確認します。 **get** 要求または **get-next** 要求を発行している場合は、コミュニティーに読み取り専用許可または読み取り/書き込み許可があることを確認してください。 **set** 要求を発行している場合は、コミュニティーに読み取り/書き込み許可があることを確認してください。

  解決法 : 正しいアクセス・モードをコミュニティー・エントリー内に指定します。 **snmpd** エージェントをリフレッシュして、要求を再試行してください。

- 指定したビュー名用のビュー・エントリーを構成ファイル内のコミュニティー・エントリーに 指定したことを確認します。 コミュニティー・エントリーにビュー名は指定してあっても対応するビュー・エントリーがない場合は、**snmpd** エージェントはそのコミュニティーに対するアクセスを許可しません。 構成ファイル内のコミュニティー・エントリーに指定したビュー名について、ビュー・エントリーが必ず必要です。

  解決法 : コミュニティー・エントリーに指定したビュー名に対応するビュー・エントリーを指定します。 **snmpd** エージェントをリフレッシュして、要求を再試行してください。
- ビュー・エントリーの MIB サブツリーとして `iso` を指定した場合は、`iso.3` と指定したかどうかを検査します。 **snmpd** エージェントから `iso` ツリーの `org` 部分にアクセスするには、インスタンスとして `3` が必要です。

  解決法: ビュー・エントリー内で MIB サブツリーに `iso.3` を指定します。 **snmpd** エージェントをリフレッシュして、要求を再試行してください。
- コミュニティー・エントリー内の *IP* アドレス とネットワーク・マスク を検査します。 コミュニティー名で指定されるコミュニティーに SNMP 要求の発行元であるホストが組み込まれていることを確認します。

  解決策 : 構成ファイルのコミュニティー・エントリー内の *IP* アドレス・フィールドとネットワーク・マスク・フィールドを変更して、SNMP 要求の発行元であるホストを組み込んでください。

エージェントからの無応答問題

コミュニティー内の *IP* アドレス が 0.0.0.0 と指定されているが、**snmpd** エージェントから応答がない場合は、次の処置をとってください。

- コミュニティー・エントリー内のネットワーク・マスク・フィールドを検査します。 このコミュニティー名に対して一般アクセスを行うには、ネットワーク・マスク が **0.0.0.0** でなければなりません。 ネットワーク・マスク として 255.255.255.255 を指定した場合は、**snmpd** エージェントは指定されたコミュニティー名を伴う要求を一切許可しないように構成されます。

  解決法 : コミュニティー・エントリー内のネットワーク・マスク に 0.0.0.0 を指定します。 **snmpd** エージェントをリフレッシュして、要求を再試行してください。
- コミュニティー・エントリー内のアクセス・モードが要求タイプと対応していることを確認します。 **get** 要求または **get-next** 要求を発行している場合は、コミュニティーに読み取り専用許可または読み取り/書き込み許可があることを確認してください。 **set** 要求を発行している場合は、コミュニティーに読み取り/書き込み許可があることを確認してください。

  解決法 : 正しいアクセス・モードをコミュニティー・エントリー内に指定します。 **snmpd** エージェントをリフレッシュして、要求を再試行してください。

noSuchName 問題

snmpd エージェントでサポートされている MIB 変数を設定しようとしたときに、noSuchName エラー・メッセージが戻された場合は、次のような理由が考えられます。

発行した set 要求には、書き込みアクセス権を持つ有効なコミュニティーのコミュニティー 名が含まれていませんでした。 **SNMP** プロトコルでは、適切なアクセス権を持たないコミュニティーを指定して set 要求を発行すると、noSuchName エラー・メッセージが戻されます。

解決法 : 書き込み特権を持つとともに、その set 要求の発行元ホストを含んでいるコミュニティーのコミュニティー名を指定して、set 要求を発行してください。

ネットワーク・ファイルシステム

ネットワーク・ファイルシステム (NFS) は、ネットワーク上でファイルを格納する機構です。 これは分散ファイルシステムの 1 つです。このファイルシステムにより、ユーザーはリモート・コンピューター上のファイルやディレクトリーにアクセスし、それらのファイルやディレクトリーをローカル・ファイルやローカル・ディレクトリーと同様に扱うことができます。

例えば、ユーザーは、オペレーティング・システムのコマンドを使用して、リモート・ファイルおよびリモート・ディレクトリーの作成、除去、読み取り、書き込み、ファイル属性の設定を行うことができます。

NFS ソフトウェア・パッケージには、NFS、ネットワーク情報サービス (NIS)、およびその他のサービスとデーモンが組み込まれています。 NFS と NIS は 1 つのパッケージとしてインストールされますが、相互に独立しており、個別に構成し管理します。

AIX 5.3 以降は、NFS バージョン 2、3、および 4 のプロトコルをサポートします。 NFS バージョン 4 は、NFS の最新定義のバージョンであり、RFC 3530 によって記述されています。 AIX による NFS バージョン 4 サポートの詳細については後述します。 NFS クライアントは、デフォルトで NFS バージョン 3 プロトコルを使用します。

NFS サービス

NFS はクライアントとサーバーの 関係を通じてサービスを提供します。

ファイルシステム すなわちディレクトリー およびその他のリソースをリモート・アクセスに対して使用可能にするコンピューターを、サーバー と呼びます。 ファイルシステムを 使用可能にする操作をエクスポート と呼びます。 サーバーのリソースを使用するコンピューターとそのプロセスは、クライアント と見なされます。 サーバーがエクスポートするファイルシステムをマウントしたクライアントは 、個々のサーバー・ファイルにアクセスできます (エクスポート・ディレクトリーへのアクセスは、特定のクライアントに限定されることがあります)。

NFS が提供する主なサービスは次のとおりです。

表 *89. NFS* サービス

サービス	説明
Mount service	サーバー上の **/usr/sbin/rpc.mountd** デーモンと、クライアント上の **/usr/sbin/mount** コマンドからマウントします。 このサービスは、NFS バージョン 2 およびバージョン 3 でのみ使用可能です。
Remote File access	サーバー上の **/usr/sbin/nfsd** デーモンと、クライアント上の **/usr/sbin/biod** デーモンからアクセスします。
Remote execution service	サーバー上の **/usr/sbin/rpc.rexd** デーモンと、クライアント上の **/usr/bin/on** コマンドから実行します。
Remote System Statistics service	サーバー上の **/usr/sbin/rpc.rstatd** デーモンと、クライアント上の **/usr/bin/rup** コマンドからコンパイルします。
Remote User Listing service	サーバー上の **/usr/lib/netsvc/rusers/rpc.rusersd** デーモンと、クライアント上の **/usr/bin/rusers** コマンドからリストします。
Boot Parameters service	サーバー上の **/usr/sbin/rpc.bootparamd** デーモンから Sun OS ディスクレス・クライアントに始動パラメーターを提供します。
Remote Wall service	サーバー上の **/usr/lib/netsvc/rwall/rpc.rwalld** デーモンと、クライアント上の **/usr/sbin/rwall** コマンドから保護します。
Spray service	サーバー上の **/usr/lib/netsvc/spray/rpc.sprayd** デーモンと、クライアント上の **/usr/sbin/spray** コマンドにより、リモート・プロシージャー・コール (RPC) パケットのストリームを一方向に送信します。
PC authentication service	サーバー上の **/usr/sbin/rpc.pcnfsd** デーモンにより、PC-NFS にユーザー認証サービスを提供します。
Enhanced security service	クライアントとサーバーの 両方で、拡張セキュリティー・サービス (Kerberos 5 など) へのアクセスを提供します。**/usr/sbin/gssd** デーモンは、ネットワーク認証サービスで提供されるセキュリティー・サービスへのアクセスを NFS に提供します。 ネットワーク認証サービスおよび暗号ライブラリー・ファイル・セット (krb5.client.rte、**krb5.server.rte**、および **modcrypt.base**) がインストールされている必要があります。 これらのファイル・セットは、AIX Expansion Pack からインストールできます。

表 89. NFS サービス (続き)	
サービス	**説明**
Identity translation service	セキュリティー・プリンシパル、NFS バージョン 4 ID ストリング、および対応する数値システム ID 間の変換を実行します。 さらに、外部 NFS バージョン 4 ドメインの ID 情報のマッピングが提供されます。 これらのサービスは、**/usr/sbin/nfsrgyd** デーモンによって提供されます。

注 : 1 台のコンピューターを同時に NFS サーバーと NFS クライアントにすることができます。

NFS バージョン 2 および 3 のサーバーはステートレス です。つまり、サーバーはクライアントのトランザクション情報を保存しません。 1 つの NFS トランザクションが 1 つの完全なファイル操作に対応します。 NFS では、あとで NFS を使用するために必要な情報をクライアント側が記憶しなければなりません。

NFS バージョン 4 サーバーは、ファイル・オープン操作およびファイル・ロック操作が NFS バージョン 4 プロトコルで定義されているため、ステートフルです。

NFS アクセス制御リストのサポート

AIX NFS バージョン 4 インプリメンテーションは、2 つの ACL タイプ (NFS4 と AIXC) をサポートします。

アクセス検査の信頼できるソースは、NFS サーバーによってエクスポートされる基礎となるファイルシステムにあります。 このファイルシステムは、ファイルのアクセス制御 (ACL またはアクセス権ビット)、呼び出し元の証明書、および適用される可能性のあるその他のローカル・システムの制限を考慮します。 アプリケーションおよびユーザーは、UNIX モード・ビットまたは ACL だけを調べて、最終的にアクセスを予測できると見なすべきではありません。

aclget、**aclput**、および **acledit** コマンドは、クライアントで NFS または AIX ACL を操作するために使用できます。 詳しくは、セキュリティーの『アクセス制御リスト』を参照してください。

NFS RBAC

NFS は、Role Based Access Control (RBAC) をサポートします。 NFS のクライアントとサーバーのコマンドは RBAC 対応です。

これにより、管理者がコマンドの RBAC ロールをユーザーに割り当てると、非 root ユーザーは NFS コマンドを実行できます。 NFS コマンドに関連付けられた許可文字列および特権のリストを表示するには、システム上の /etc/security/privcmds ファイルを参照してください。

NFS4 ACL

NFS4 ACL は、NFS バージョン 4 プロトコルで定義されている ACL です。

NFS4 ACL はプラットフォーム独立であるため、他のベンダーのクライアントまたはサーバーでサポート 可能です。 NFS4 ACL をサポートする場合、NFS バージョン 4 のクライアントおよびサーバーは不要です。

AIX サーバーでは、基礎となる物理ファイルシステム・インスタンスが NFS4 ACL をサポートする場合は、AIX NFS4 サーバーはそのファイルシステム・インスタンスに対して NFS4 ACL をサポートします。 AIX 上の物理ファイルシステム・タイプのほとんどは、NFS4 ACL をサポートしません。 これらのファイルシステム・タイプには、CFS、UDF、JFS、および拡張属性バージョン 1 の JFS2 などが含まれますが、これだけに限定されているわけではありません。拡張属性バージョン 2 の JFS2 のすべてのインスタンスは NFS4 ACL をサポートします。

サーバー上のエクスポートされた NFS バージョン 4 のファイルシステム・インスタンスが NFS4 ACL をサポートする場合は、NFS バージョン 4 のクライアント・ファイルシステムは NFS4 ACL の読み取りおよび書き込みを行うことができます。

AIX ACL

AIXC ACL は、AIX サーバー所有のアクセス制御リストです。

これは NFS バージョン 4 プロトコルで定義されたものではないため、AIX サーバーおよびクライアントでのみ認識されます。

NFS バージョン 4 サーバーでは、基礎となるファイルシステム・インスタンスが AIXC ACL をサポートする場合は AIXC ACL がサポートされます。 JFS および JFS2 のすべてのインスタンスは AIXC ACL をサポートします。

NFS バージョン 4 クライアントには、AIX ACL のサポートを使用可能または使用不可にするマウント・オプションがあります。 デフォルトでは、AIXC ACL はサポートされません。 NFS バージョン 4 クライアント・ファイルシステム上のユーザーは、次の場合に AIXC ACL の読み取りおよび書き込みを行うことができます。つまり、クライアントとサーバーの 情報が AIX を実行している場合、サーバー上の基礎となる物理ファイルシステム・インスタンスが AIXC ACL をサポートする場合、および AIXC ACL が使用可能になっているファイルシステム・インスタンスを AIX クライアントがマウントする 場合です。 NFS バージョン 4 での AIXC ACL サポートは、AIX NFS バージョン 2 および NFS バージョン 3 インプリメンテーションでの AIXC ACL サポートと類似しています。

拡張属性バージョン 2 の JFS2 ファイルシステムのすべてのインスタンスは 、AIXC ACL と NFS4 ACL の両方をサポートします。 このタイプのファイルシステム内のファイルには、モード・ビットのみ (ACL なし)、NFS4 ACL、または AIXC ACL があります。 しかし、ファイルが NFS4 ACL と AIXC ACL を同時に持つことはできません。

aclgettypes コマンドを使用して、ファイルシステム・インスタンス上で読み取りおよび書き込みが可能な ACL タイプを判別できます。 このコマンドを NFS バージョン 4 サーバー上の物理ファイルシステムに対してローカルに実行する場合と、NFS バージョン 4 クライアント上の同じファイルシステムに対して実行する場合で、出力が異なる場合があります。 例えば、NFS バージョン 4 サーバー上の NFS バージョン 4 ファイルシステム・インスタンスは NFS4 ACL と AIXC ACL をサポートする場合がありますが、クライアントは NFS4 ACL だけを送信および受信するよう構成されています。 この場合は、**aclgettypes** コマンドを NFS バージョン 4 クライアント・ファイルシステムから 実行すると、NFS4 だけが戻されます。 また、クライアント上のユーザーが AIXC ACL を要求するとエラーが戻されます。

キャッシュ・ファイルシステムのサポート

キャッシュ・ファイルシステム (CacheFS) は、汎用のファイルシステム ・キャッシュ機構で、サーバーとネットワークの負荷を低減することにより、NFS サーバーのパフォーマンスと 拡張機能を改善します。

CacheFS は階層化ファイルシステムとして 設計され、ファイル相互間のキャッシュ機能を提供します。 NFS 環境では、CacheFS によりサーバーに対するクライアント数の比率が増加し、サーバーとネットワークの負荷が減少し、Point-to-Point Protocol (PPP) などの低速リンク上のクライアントのパフォーマンスが向上します。

キャッシュはクライアント・マシン上に作成されるため、キャッシュにマウントするよう指定されたファイルシステムは、ネットワークを介さずに、ローカルにアクセスできます。 ファイルは、それらに対する最初のアクセス要求が出された時点でキャッシュに書き込まれます。 キャッシュへの書き込みは、ユーザーがファイル (または複数のファイル) へのアクセスを要求するまで行われません。 最初のファイル要求は遅く感じられますが、それ以降の同じファイルへのアクセスは速くなります。

注 :

1. / (root) ファイルシステムまたは /usr ファイルシステムをキャッシュに 書き込むことはできません。
2. マウントできるのは共用ファイルシステムだけです。 (**exportfs** コマンドを参照してください。)
3. ローカルのジャーナル・ファイルシステム (JFS) のディスク・ファイルシステムをキャッシュに 書き込んでも、パフォーマンスには関係ありません。
4. 次の表のタスクを実行するためには、root 権限または system 権限を持っている必要があります。

表 *90. CacheFS* タスク

タスク	**SMIT 高速パス**	**コマンドまたはファイル**
キャッシュの設定	cachefs_admin_create	**cfsadmin -c** *MountDirectoryName*[1]

表 90. CacheFS タスク (続き)		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
ファイルのマウントの指定	cachefs_mount	**mount** -F cachefs -o backfstype=*FileSysType*, cachedir=*CacheDirectory*[,*options*] BackFileSystem *MountDirectoryName*[2] or edit /etc/**filesystems**.
キャッシュの変更	cachefs_admin_change	キャッシュを除去してから、適切な **mount** コマンド・オプションを使用して再作成する。
キャッシュ情報の表示	cachefs_admin_change	**cfsadmin -l** *MountDirectoryName.*
キャッシュの除去	cachefs_admin_remove	1. ファイルシステムをアンマウントする : **umount** *MountDirectoryName* 2. キャッシュ ID を判別する: **cfsadmin -l** *MountDirectoryName* 3. ファイルシステムを削除する: **cfsadmin -d** *CacheID CacheDirectory*
ファイルシステムの保全性の検査	cachefs_admin_check	**fsck_cachefs***CacheDirectory*[3]

注 :

1. キャッシュの作成後は、キャッシュ・ディレクトリー (cachedir) 内での操作は行わないでください。CacheFS ソフトウェア内部の矛盾の原因となります。
2. マウントするファイルの指定に **mount** コマンド・オプションを使用する場合は、システムを再始動するたびにこのコマンドを再発行する必要があります。
3. 修復を行わずにファイルシステムを検査するには、**fsck_cachefs** コマンドの **-m** オプションまたは **-o** オプションを使用してください。
4. システムを以前のバージョンの AIX から AIX バージョン 6.1 以降に移行した後、旧バージョンの AIX 内に作成されている古いキャッシュ・ファイルシステムを削除して、再作成する必要があります。

NFS マップ・ファイル・サポート

NFS ではマップ・ファイルをサポートしているため、クライアント上のプログラムは、ファイルがメモリー上に置かれているかのように、ファイルをアクセスできます。

shmat サブルーチンを使用すると、ユーザーはファイルの領域をそのアドレス・スペースにマップできます。 プログラムがこのメモリー領域に対して読み取りまたは書き込みを行うと、サーバーからファイルがメモリーに読み込まれるか、サーバー上で必要に応じてファイルが更新されます。

NFS によるファイルのマッピングには、次の 3 つの制限事項があります。

- ファイルはクライアント間では情報を十分に共用できません。
- あるクライアント上でマップ・ファイルを使用してファイルを変更しても、その変更内容は別のクライアントでは認識されません。
- ファイルの領域のロックまたはアンロックを行っても、クライアント間でデータを効果的に調整できません。

異なるクライアント上のプログラム間でデータを共有するために NFS ファイルを使用する場合は、レコード・ロック機能と正規の read および write サブルーチンを使用してください。

同じクライアント上の複数のプログラムは、マップ・ファイルを使用して効果的にデータを共用できます。通知レコード・ロックにより、クライアント上のファイルに対する更新を調整できます。 複数のクライア

ントがマップ・ファイルを使用してデータを共用できるのは、静的データベースの場合のように、データが変更されない場合だけです。

NFS プロキシー・サービス

AIX は、ネットワーク・ファイルシステム (NFS) のプロキシー・サービスをサポートします。 AIX サーバーは、ローカルでアクセス可能なファイルシステムとプロキシー・エクスポートを並行してエクスポートできます。 エクスポートされたプロキシー表示は、NFS クライアントでマウントできます。

AIX NFS プロキシー・サービスは、アクセスされたデータのディスク・キャッシュを使用して、後続の同じような要求をローカルで処理します。これにより、バックエンド・サーバーへのネットワーク・トラフィックが削減されます。 プロキシー・サービスにより、低速なネットワークまたは信頼性が低いネットワークまで NFS データ・アクセスが拡張されることでパフォーマンスが向上し、データが常駐する 1 次サーバーへのネットワーク・トラフィックが削減される可能性があります。 プロキシー・サービスは、可用性とコンテンツ管理要件に応じて、データをコピーすることなく NFS アクセスをネットワーク・エッジまで拡張するためのソリューションを提供します。 AIX NFS プロキシー・サービスを構成するには、**mknfsproxy** コマンドを使用できます。

プロキシー・キャッシングは、NFS v3 と v4 の両方のプロトコルで使用できます。 プロキシーと接続されたクライアントとの間のプロトコルは、プロキシーとバックエンド・サーバーとの間で NFS v4 プロトコルが使用される場合は NFS v3 または NFS v4 のいずれでも構いません。 ただし、NFS v3 プロトコルを使用する場合、プロキシーと接続されたクライアントとの間のプロトコルは NFS v3 でなければなりません。 バイト範囲の通知ロックに加えて、データの読み取りおよび書き込みがサポートされています。

プロキシー・サーバーとその接続されたクライアントとの間では、krb5、krb5i、および krb5p のセキュリティー・メソッドを使用できます。 これらのメソッドは、プロキシー・サーバーとメイン・サーバーとの間でも使用できます。 プロキシーを介してチケット転送テクノロジーを使用すると、クライアント上で認証を行ってメイン・サーバーに認証させることができます。 このテクノロジーの利点を生かすには、Kerberos 認証を実行する際に **-f** オプションを指定して **kinit** コマンドを使用します。 プロキシーとバックエンド・サーバー間で **auth_sys** セキュリティーを使用する場合、バックエンド・サーバーにアクセスすると、プロキシー・サーバーは Kerberos クライアント・アクセスを **auth_sys** 属性にマップします。 最良の結果を得るには、プロキシー・サーバーとバックエンド・サーバーが同じユーザーとグループの ID 定義を共有します。

NFS プロキシー・サービスには、以下の制限が適用されます。

- プロキシー・サービスには、TCP を使用して接続されたクライアントが必要です。
- プロキシー・サービスでは、**mount** コマンドおよび **unmount** コマンドを使用しないで NFS v3 クライアントが NFS v4 のエクスポートされたネームスペースをブラウズする方法を提供します。したがって、プロキシー・ファイルシステムを構成する際に `mfsid` オプションを指定して **mknfsproxy** コマンドを使用する必要があります。
- プロキシー・サービスで使用するキャッシュ・ファイルシステムは、拡張 JFS ファイルシステム (JFS2) でなければなりません。
- プロキシー・サービスは、バックエンド NFS サーバーにマウントされた AIX クライアントの最上位でキャッシュ・ファイルシステムを実行します。 AIX NFS クライアントで可能なコンカレント I/O (CIO) 機能により、キャッシュ・ファイルシステムのパフォーマンスが向上します。 下位の NFS クライアントのマウントに直接アクセスしようとすると、CIO のオープン試行と競合して失敗する可能性があります。

NFS マウントの種類

NFS マウントには、定義済み、明示、自動の 3 種類があります。

定義済み マウントは、`/etc/filesystems` ファイルで指定されます。 このファイル内の各スタンザ (またはエントリー) は、マウントの特性を定義します。 ホスト名、リモート・パス、ローカル・パスなどのデータ、およびマウント・オプションは、このスタンザ内に表示されます。 定義済みマウントは、クライアントの適切な操作に常に特定のマウントを行う必要がある場合に使用されます。

明示 マウントは、root ユーザーのニーズに対応します。 通常、明示マウントは、臨時に計画外のマウントを行う必要がある場合に短期的に実行されます。 また、明示マウントは、NFS クライアント上で通常は使用可能でないマウントが特殊なタスクにのみ必要な場合にも使用できます。 通常、これらのマウントは、必要なすべての情報を指定した **mount** コマンドを使用して、コマンド・ラインで完全修飾されます。 明示

マウントの場合、/etc/filesystems ファイルを更新する必要はありません。 明示的にマウントされたファイルシステムは、**umount** コマンドを使用して明示的にアンマウントするか、システムを再始動するまで、マウントされたままになっています。

自動 マウントは、**automount** コマンドによって制御されます。このコマンドによって、**AutoFS** カーネル・エクステンション 機能は、指定されたディレクトリーのアクティビティーをモニターします 。 プログラムまたはユーザーが現在マウントされていないディレクトリーにアクセスしようとすると、**AutoFS** は要求を代行受信し、ファイルシステムのマウントを 調整してから、要求を処理します。

NFS エクスポートおよびマウント

NFS を管理するには、ディレクトリーのエクスポートとマウントについて 理解する必要があります。

NFS サーバーは、ファイルまたはディレクトリーをエクスポートする 必要があり、その後で NFS クライアントはそのファイルまたはディレクトリーをマウントできます。 このセクションでは、これらの概念の詳細を取り上げます。

NFS ディレクトリーのエクスポート

ディレクトリーのエクスポートは NFS サーバーで実行します。 ディレクトリーをエクスポートすると、サーバーのネーム・スペースにあるディレクトリーがクライアント ・マシンで使用可能であることを宣言することになります。

エクスポートされたディレクトリーはエクスポート と呼ばれ、エクスポートされたディレクトリーのファイルシステムにあるディレクトリーに 属するすべてのファイルが含まれています。

各エクスポートはアクセス制限も定義します。 例えば、次のような制限を定義できます。

- エクスポートされたディレクトリーにアクセスできるクライアント
- クライアントがディレクトリーにアクセスするために 使用する必要のある NFS バージョン
- クライアントによるエクスポートへのファイル書き込みの可否
- クライアントがエクスポート内のディレクトリーおよびファイルにアクセスするために 使用する必要があるセキュリティー・メソッド

許容されるエクスポート制限およびエクスポート・セマンティクスの詳細については、**exportfs** コマンドの説明、および/etc/exports ファイルの説明を参照してください。

注 : エクスポートの属性を変更した場合に変更を有効にするには、ディレクトリーを再エクスポートする必要があります。 他のファイルが変更されたり、サーバー外部で変更が行われる場合に、ディレクトリーを再エクスポートしなければならない場合があります。 例えば、アクセス・リストで指定されたクライアント名が /etc/netgroup ファイルで定義された netgroup の場合、クライアント・グループの定義を変更すると、変更を有効にするにはアクセス・リスト内で netgroup を使用するすべてのエクスポートを再エクスポートする必要があります。

同様に、クライアントの IP アドレスが変更される場合は、アクセス・リスト内でそのクライアントを指定するすべてのエクスポートを再エクスポートする必要があります。 NFS サーバーは、各エクスポートに対するクライアント・アクセス権限のキャッシュを保守するためです。 キャッシュは、アンエクスポートまたは再エクスポートが行われるたびにフラッシュされます。 エクスポートのアクセス権限が変更される場合、特にクライアントの IP アドレスが変更される場合やアクセス・リストからクライアントが除去される場合、クライアントのアクセスがキャッシュに正しく反映されるように、アンエクスポートまたは再エクスポートを実行する必要があります。 NFS サーバーは **rpc.mountd** デーモンを呼び出して各クライアントのアクセス権限を取得するため、サーバーが NFS バージョン 4 アクセス用のファイルシステムだけをエクスポートする場合でも、**rpc.mountd** デーモンがサーバー上で実行されていなければなりません。

NFS ディレクトリーのマウント

NFS クライアントは、NFS サーバーによってエクスポートされたディレクトリーをマウントできます。 ディレクトリーをマウントすると、NFS サーバーにあるファイルをクライアントが 使用できるようになります。

クライアントは、ファイルがサーバーによってエクスポートされており、クライアントがエクスポートのファイルへのアクセスをエクスポート制限で許可されている場合に、サーバー上のファイルにアクセスできます。 クライアントがネーム・スペース内のマウント・ポイントにサーバーのエクスポートを正常にマ

ウントすると、そのエクスポート用のサーバーのファイルはクライアントのネーム・スペースに存在するようになり、ローカル・ファイルシステム上のファイルとして表示されます。

例えば、サーバー diamond 上の /tmp ディレクトリーをエクスポートして、そのディレクトリーをクライアント clip に /mnt としてマウントするとします。 サーバーで、次のコマンドを入力します。

```
exportfs -i -o access=clip /tmp
```

/tmp ディレクトリーがクライアントで 使用可能になります。

クライアントで、次のコマンドを入力します。

```
mount diamond:/tmp /mnt
```

サーバーの /tmp ディレクトリー内のディレクトリーおよびファイルが、クライアントの /mnt ディレクトリーに表示されるようになります。

注 :

1. NFS バージョン 2 および 3 と NFS バージョン 4 では、マウントの扱われ方が異なります。 NFS バージョン 2 および 3 では、サーバーはマウントに使用できるディレクトリーをエクスポートしていました。 次いで、NFS バージョン 2 または 3 クライアントは、アクセス先の各エクスポートを明示的にマウントする必要がありました。

   NFS バージョン 4 では、サーバーは NFS アクセス用にエクスポートするサーバー・ディレクトリーまたはファイルシステムごとに、エクスポート制御を指定します。 サーバーは、エクスポートされたすべてのデータの単一ディレクトリー・ツリーをこれらのエクスポート制御からレンダリングし、エクスポートされたディレクトリー間のギャップを埋めます。 このツリーは疑似ファイルシステムと呼ばれ、NFS バージョン 4 サーバーの疑似ルートから始まります。 NFS バージョン 4 疑似ファイルシステム・モデルは、NFS バージョン 4 クライアントが、サーバーによってエクスポートされたすべてのデータにアクセスするために、サーバーの疑似ルートの単一マウントを実行できるようにします (インプリメンテーションによって異なる)。 AIX NFS クライアントは、この機能をサポートします。 クライアントで表示される実際の内容は、サーバーのエクスポート制御によって異なります。
2. NFS バージョン 4 では、ファイルからファイルへのマウントは許可されていません。

NFS のマウント

クライアントは最初にサーバーがエクスポートしたディレクトリーをマウントして、サーバー上のファイルにアクセスします。 クライアントはディレクトリーをマウントするときに、そのディレクトリーのコピーを作成しません。 代わりに、マウント・プロセスは一連のリモート・プロシージャー・コールを使用して、クライアントがサーバー上のディレクトリーに透過的にアクセスできるようにします。

次に、マウント・プロセスについて説明します。

1. サーバーが始動すると、/etc/rc.nfs スクリプトは **exportfs** コマンド (このコマンドはサーバーの /etc/exports ファイルを読み取ります) を実行し、どのディレクトリーをエクスポートし、どのアクセス制限が必要かをカーネルに指示します。
2. 次に、/etc/rc.nfs スクリプトが、**rpc.mountd** デーモンと複数の **nfsd** デーモンを始動します。
3. 次に、/etc/rc.nfs スクリプトは **mount** コマンドを実行し、このコマンドが /etc/filesystems ファイル内からファイルシステムのリストを 読み取ります。
4. **mount** コマンドは、クライアントに必要な情報をエクスポートする 1 つ以上のサーバーを探し、そのサーバーとの間で通信を設定します。

   このプロセスをバインディング と呼びます。
5. 次に、**mount** コマンドは 1 つ以上のサーバーに対して、クライアントがその /etc/filesystems ファイル内のディレクトリーにアクセス可能にするように要求します。
6. サーバー・デーモンはクライアントのマウント要求を受信して、それを認可または拒否します。

   要求されたディレクトリーがそのクライアントで 使用できれば、サーバー・デーモンはクライアントのカーネルにファイル・ハンドル と呼ばれる ID を送信します。
7. クライアントのカーネルは、一定の情報をマウント・レコード に記録して、ファイル・ハンドルとマウント・ポイント (ディレクトリー) を連結します。

NFS バージョン 4 のマウント処理では、**rpc.mountd** デーモンとのクライアント通信は行われません。 コア NFS バージョン 4 プロトコルの操作を使用して、クライアント・サイドのマウント操作にサービスが提供されます。 NFS バージョン 4 サーバー・インプリメンテーションでは、NFS バージョン 4 アクセスの処理の一部として **rpc.mountd** デーモンのサポートが利用されます。

/etc/exports ファイル

/etc/exports ファイルは、サーバーがそのクライアントにエクスポートするすべてのディレクトリーを示します。

このファイルの各行がそれぞれ 1 つのディレクトリーを指定します。 1 つのディレクトリーを /etc/exports ファイルで 2 回指定できます (NFS バージョン 2 または NFS バージョン 3 用に 1 回と、NFS バージョン 4 用に 1 回)。サーバーは、NFS サーバーが始動されるたびに、リストされたディレクトリーを自動的にエクスポートします。 クライアントはこれらのエクスポート・ディレクトリーをマウントすることができます。 /etc/exports ファイルの行の構文は次のとおりです。

```
directory     -option[,option]
```

directory はディレクトリーの絶対パス名です。 オプションには、**ro** のような単純なフラグ、またはホスト名のリストを指定できます。 /etc/exports ファイルが存在しない場合、/etc/rc.nfs スクリプトは **nfsd** デーモンまたは **rpc.mountd** デーモンを始動しません。

/etc/exports ファイルのエントリーの例を次に示します。

```
/usr/games    -ro,access=ballet:jazz:tap
/home         -root=ballet,access=ballet
/var/tmp
/usr/lib      -access=clients
/accounts/database -vers=4,sec=krb5,access=accmachines,root=accmachine1
/tmp          -vers=3,ro
/tmp          -vers=4,sec=krb5,access=accmachines,root=accmachine1
```

この最初のエントリーは、システム ballet、jazz、および tap が /usr/games ディレクトリーをマウントできることを示しています。 これらのシステムはディレクトリーからデータを 読み取ってプログラムを実行できますが、ディレクトリーに書き込むことはできません。

この例の 2 番目のエントリーは、システム ballet が /home ディレクトリーをマウントでき、ディレクトリーへの root アクセスが許可されることを示しています。

この例の 3 番目のエントリーは、すべてのクライアントが /var/tmp ディレクトリーをマウントできることを示しています。 (アクセス・リストが指定されていないので注意してください。)

この例の 4 番目のエントリーは、ネットグループ clients で指定されるアクセス・リストを示しています。 つまり、ネットグループ clients に属すると指定されたマシンは、このサーバーから /usr/lib ディレクトリーをマウントすることができます。 (ネットグループ とは、セキュリティーまたはネットワーク編成の目的でネットワーク・リソースへのアクセスを許可されたネットワーク全体のグループのことです。ネットグループの制御は、NIS を使用して行います。

5 番目のエントリーでは、NFS バージョン 4 プロトコルと Kerberos 5 認証を使用してディレクトリーにアクセスする accmachines netgroup 内のクライアントだけが、ディレクトリー /accounts/database へのアクセスを許可されます。 root アクセスは、accmachine1 からのみ許可できます。

6 番目と 7 番目のエントリーでは、異なるバージョンとオプションを使用して /tmp ディレクトリーへのエクスポートが行われます。 同じディレクトリー用に異なる NFS の 2 つのエントリーが /etc/exports ファイルに存在する場合、 **exportfs** コマンドは両方をエクスポートします。 1 つのディレクトリーで NFS バージョン 4 と NFS バージョン 5 に同じオプションが指定されている場合は、 /etc/exports ファイル内に -vers=3:4 を指定するエントリーが必要です。

/etc/xtab ファイル

/etc/xtab ファイルのフォーマットは /etc/exports ファイルと類似していて、現在エクスポートされているディレクトリーが表示されます。

exportfs コマンドを実行するたびに、/etc/xtab ファイルが変更されます。 これにより、/etc/exports ファイルを変更しなくても、ディレクトリーを一時的にエクスポートできます。 一時的にエクスポートされたディレクトリーをアンエクスポートすると、そのディレクトリーは /etc/xtab ファイルから除去されます。

注: /etc/xtab ファイルは自動的に更新されるため、編集しないでください。

/etc/nfs/hostkey ファイル

このファイルは、Kerberos ホスト・プリンシパルと keytab ファイルのロケーションを指定するために NFS サーバーによって使用されます。

このファイルを構成および管理する方法について詳しくは、**nfshostkey** コマンドを参照してください。

/etc/nfs/local_domain ファイル

このファイルには、システムのローカル NFS ドメインが含まれています。

同じ NFS ローカル・ドメインを共有するシステムは、同じユーザー・レジストリーおよびグループ・レジストリーを共有することが暗黙に指定されます。 このファイルを構成および管理する方法について詳しくは、**chnfsdom** コマンドを参照してください。

/etc/nfs/realm.map ファイル

このファイルは、*name@kerberos-realm* という形式の着信 Kerberos プリンシパルを *name@nfs-domain* という形式にマップするために、NFS レジストリー・デーモンによって使用されます。

次いで、*name@nfs-domain* がローカル UNIX 証明書に解決されます。 このファイルを使用すると、Kerberos プリンシパルをサーバーのユーザー・レジストリーに簡単にマップできます。 これは、さまざまな Kerberos レルム内のクライアントがサーバーにアクセスする 一方で、ユーザー・ネームスペースがグローバルである場合に適切です。 このファイルには、次の形式の行が含まれている必要があります。

```
realm1 nfs-domain
realm2 nfs-domain
```

これは、サーバーがサポートするすべての Kerberos レルムに適用されます。 Kerberos レルム名がサーバーの NFS ドメインと常に同じである場合、このファイルは不要です。 *userA@kerberos-realm* を *userB@nfs-domain* にマップするより一般的な機能が必要な場合は、Enterprise Identity Mapping (EIM) サービスを使用してください。 追加情報については、592 ページの『ID マッピング』を参照してください。

このファイルのエントリーを追加、編集、または除去するには、**chnfsrtd** コマンドを使用します。

/etc/nfs/princmap ファイル

プリンシパルがサーバーの 完全修飾ドメイン名でない場合に、このファイルがホスト名を Kerberos プリンシパルにマップします。

これは、次の形式の任意の数の行で構成されます。

```
<host part of principal> alias1 alias2 ...
```

このファイルのエントリーを追加、編集、または除去するには、**nfshostmap** コマンドを使用します。

/etc/nfs/security_default ファイル

/etc/nfs/security_default ファイルには、NFS クライアントで使用されるセキュリティー・フレーバーのリストが含まれています (セキュリティーは使用される順序になっています)。

このファイルを管理するには、**chnfssec** コマンドを使用します。

リモート・プロシージャー・コール・プロトコル

NFS は、さまざまなマシン・タイプ、オペレーティング・システム、およびネットワーク体系にインプリメントされています。 NFS はこの独立性を得るために、**リモート・プロシージャー・コール (RPC)** プロトコルを使用します。

RPC はプロシージャーのライブラリーです。 これらのプロシージャーを使用すると、あるプロセス (クライアント・プロセス) が別のプロセス (サーバー・プロセス) にプロシージャー・コールを実行するよう指図できます。このプロシージャー・コールは、そのクライアント・プロセスがアドレス・スペースで実行したかのように行われます。 クライアントとサーバーは 2 つの別個のプロセスなので、同じ物理システム上に存在する必要はありません (存在しても構いません)。

NFS は、サーバーがクライアントからの特定のタイプのコールを処理する一連の **RPC** コールとしてインプリメントされます。 クライアントはクライアント・プロセスが実行するファイルシステム 操作に基づいてそれらのコールを行います。 この意味では、NFS は RPC アプリケーションです。

サーバー・プロセスとクライアント・プロセスはまったく異なるアーキテクチャーを持つ 2 つの異なる物理システム上に存在できるため、**RPC** は、2 つのシステムのデータの表現方法が同じでない場合に対処する必要があります。 この理由により、**RPC** は**外部データ表現 (XDR)** プロトコルによって定義されているデータ・タイプを使用します。

外部データ表示プロトコル

外部データ表示 (XDR) プロトコルは、さまざまなデータ・タイプの標準的な表現に関する仕様です。

標準データ・タイプ表現を使用することにより、まったく異なるアーキテクチャーを持つマシンを送信元とするデータの場合でも、プログラムはデータを確実に正しく解釈することができます。

実際には、ほとんどのプログラムは **XDR** を内部では使用しません。 代わりに、プログラムを実行しているコンピューターのアーキテクチャーに固有のデータ・タイプ表現を使用します。 プログラムは別のプログラムと通信する必要があれば、データを **XDR** フォーマットに変換してから送信します。 また、データを受信すると、そのデータを **XDR** フォーマットから独自のデータ・タイプ表現に変換します。

portmap デーモン

portmap デーモンは、クライアントがプログラム 番号とバージョン番号の組み合わせをサーバーのポート番号にマップする際に使用されます。

各 RPC アプリケーションには、プログラム番号とバージョン番号が対応付けられています。 これらの番号は、あるシステム上のサーバー・アプリケーションと通信するために使用されます。 クライアントはサーバーにサービスを要求するときに、サーバーが要求を受け入れるポート番号を認識しておく必要があります。 このポート番号は、サービスで使用している**ユーザー・データグラム・プロトコル (UDP)** または**伝送制御プロトコル (TCP)** に対応付けられています。 クライアントはプログラム番号、バージョン番号、およびサービスが存在するシステム名またはホスト名を認識します。 また、クライアントではプログラム番号とバージョン番号の対をサーバー・アプリケーションのポート番号にマップする方法が必要となります。 このマッピングは **portmap** デーモンを使用して実行されます。

portmap デーモンは NFS アプリケーションと同じシステム上で動作します。 サーバーはその始動時に **portmap** デーモンに登録されます。 この登録の関数として、サーバーはそのプログラム番号、バージョン番号、および **UDP** または **TCP** のポート番号を提供します。 **portmap** デーモンはサーバー・アプリケーションのテーブルを保持します。 サーバーへの要求を試行する際に、クライアントはそのサーバーがどのポートを使用しているのかを調べるために、まず **portmap** デーモンにアクセスします。 **portmap** デーモンはクライアントが要求しているサーバーのポート番号を戻してクライアントに応答します。 クライアントはポート番号を受信すると、そのあとのすべての要求をサーバー・アプリケーションに対して直接行うことができます。

NFS アプリケーションおよび制御

NFS および NIS のデーモンは、システム・リソース・コントローラー (SRC) によって制御されます。

すなわち、NFS および NIS のデーモンの始動、停止、およびその状況の検査には、**startsrc**、**stopsrc**、**lssrc** などの SRC コマンドを使用する必要があります。

一部の NFS デーモンは、SRC によっては制御されません。すなわち **rpc.rexd**、**rpc.rusersd**、**rpc.rwalld**、および **rpc.rsprayd** は、SRC によっては制御されません。 これらのデーモンは **inetd** デーモンにより始動され停止されます。

次の表は SRC によって制御されるデーモンとそのサブシステム 名を示しています。

表 *91.* デーモンとそのサブシステム

ファイル・パス	サブシステム 名	グループ名
/usr/sbin/nfsd	**nfsd**	nfs
/usr/sbin/biod	**biod**	nfs
/usr/sbin/rpc.lockd	**rpc.lockd**	nfs
/usr/sbin/rpc.statd	**rpc.statd**	nfs
/usr/sbin/rpc.mountd	**rpc.mountd**	nfs
/usr/sbin/nfsrgyd	**nfsrgyd**	nfs
/usr/sbin/gssd	**gssd**	nfs
/usr/lib/netsvc/yp/ypserv	**ypserv**	yp
/usr/lib/netsvc/yp/ypbind	**ypbind**	yp
/usr/lib/netsvc/rpc.yppasswdd	**yppasswdd**	yp
/usr/lib/netsvc/rpc.ypupdated	**ypupdated**	yp
/usr/sbin/keyserv	**keyserv**	keyserv
/usr/sbin/portmap	**portmap**	portmap

関連情報
システム・リソース・コントローラーの概要

biod スレッドおよび nfsd デーモンの数の変更
chnfs コマンドを使用して、システムで実行される **biod** または **nfsd** デーモンの最大数を変更できます。

例えば、**nfsd** デーモンの最大数を 1000 に設定するには、次のコマンドを実行します。

```
chnfs -n 1000
```

注 : このコマンドは、現在実行中のデーモンを停止し、SRC 構成情報を更新してから、デーモンを再始動します。 その結果、NFS サービスは一時的に使用不可になります。

biod スレッドの最大数は、biods=*n* マウント・オプションを使用して、マウントごとに指定できます。

注 : クライアントを処理する際に **nfsd** デーモンの数が不足すると、非べき等の動作エラーがクライアントに戻されます。 例えば、クライアントがディレクトリーを 除去する場合、サーバー上のディレクトリーが除去されても、ENOENT エラーが戻されます。

SRC によって制御されるデーモンのコマンド・ライン引数の変更
NFS および NIS の多くのデーモンには、そのデーモンの始動時に指定可能なコマンド・ライン引数が用意されています。これらのデーモンはコマンド・ラインから直接始動されないため、デーモンが正常に始動されるように SRC データベースを更新する必要があります。

そのためには **chssys** コマンドを使用します。 **chssys** コマンドの形式は次のとおりです。

```
chssys -s Daemon -a 'NewParameter'
```

例えば、次のとおりです。

```
chssys -s nfsd -a '10'
```

このコマンドは、デーモンの開始時にコマンド・ラインが nfsd 10 になるように **nfsd** サブシステムを変更します。 **chssys** コマンドによる変更内容は、サブシステムを停止して再始動するまで有効になりません。

NFS デーモンの始動

NFS サーバー上のファイルのファイル・サイズの限度は、**nfsd** の始動時にプロセス環境で定義されます。

特定の値を使用するには、/etc/rc.nfs ファイルを編集してください。 **nfsd** デーモンに対して **startsrc** コマンドを実行する前に、設定したい限度を指定した **ulimit** コマンドを使用します。

NFS デーモンは個別に始動することも、一度にすべてを始動することもできます。 NFS デーモンを個別に始動するには、以下のコマンドを実行します。

```
startsrc -s Daemon
```

この場合、*Daemon* は SRC によって制御される任意のデーモンです。 例えば、**nfsd** デーモンを始動するには、以下のコマンドを実行します。

```
startsrc -s nfsd
```

すべての NFS デーモンを始動するには、以下のコマンドを実行します。

```
startsrc -g nfs
```

注 : /etc/exports ファイルがなければ、**nfsd** デーモンと **rpc.mountd** デーモンは始動されません。 コマンド touch /etc/exports を実行すると、空の /etc/exports ファイルを作成できます。 これにより、**nfsd** デーモンと **rpc.mountd** デーモンを始動することができますが、ファイルシステムはエクスポートされません。

NFS デーモンの停止

NFS デーモンは個別に停止することも、一度にすべてを停止することもできます。

NFS デーモンを個別に停止するには、以下のコマンドを実行します。

```
stopsrc -s Daemon
```

この場合、*Daemon* は SRC によって制御される任意のデーモンです。 例えば、**rpc.lockd** デーモンを停止するには、以下のコマンドを実行します。

```
stopsrc -s rpc.lockd
```

一度にすべての NFS デーモンを停止するには、以下のコマンドを実行します。

```
stopsrc -g nfs
```

NFS デーモンの現行状況の表示

NFS デーモンの現行状況を、個別にあるいは一度にすべてを表示させることができます。

NFS デーモンの現行状況を個別に取得するには、以下のコマンドを実行します。

```
lssrc -s Daemon
```

この場合、*Daemon* は SRC によって制御される任意のデーモンです。 例えば、**rpc.lockd** デーモンの現行状況を取得するには、以下のコマンドを実行します。

```
lssrc -s rpc.lockd
```

すべての NFS デーモンの現行状況を一度に取得するには、以下のコマンドを実行します。

```
lssrc -a
```

NFS バージョン 4 のサポート

AIX 5.3 から、NFS バージョン 4 プロトコル機能のサポートが組み込まれるようになりました。

RFC 3530 で説明されているとおり、プロトコルの必須機能がサポートされていますが、次の例外があります。

- RPCSEC-GSS RPC 認証では、LIPKEY および SPKM-3 セキュリティー機構はサポートされません。 Kerberos V5 機構だけがサポートされます。
- UTF-8 要件は完全にはサポートされません。 特に、ファイル名およびファイルシステム・ストリング (シンボリック・リンクの内容およびディレクトリー・エントリー名など) の伝送は、UTF-8 形式で行われるという保証はありません。 NFS 属性ストリング (所有者および所有者グループなど) の伝送は、常に UTF-8 形式で行われます。 NFS サーバーおよびクライアントは、RFC 3530 で定義されているとおりに、着信ストリング・データに対して UTF-8 検証を実行します。 この検査は **nfso** コマンドを使用して、管理上使用不可にすることができます。 非 UTF-8 構成およびデータの環境で NFS バージョン 4 を使用するには、UTF-8 検査を使用不可にしなければならない場合があります。
- ディスクレス・クライアント、NIM、および UDP は、NFS バージョン 4 ではサポートされません。

NFS バージョン 4 の次のオプショナル機能がサポートされます。

- NFS バージョン 4 ACL は、NFS クライアントとサーバーの 両方でサポートされます。 NFS クライアントは、`acledit`、`aclget`、および `aclput` ユーティリティーを 使用した、NFS バージョン 4 ACL の管理をサポートします。 NFS サーバーは、NFS バージョン 4 ACL モデルをサポートする基礎となるファイルシステムで、NFS バージョン 4 ACL の保管および検索を行うことができます。 詳しくは、569 ページの『NFS アクセス制御リストのサポート』を参照してください。
- ある NFS バージョン 4 ドメインから別のドメインにプリンシパルおよびファイル 所有権属性をマップするためのサポートが提供されています。 このサポートは、主に AIX NFS サーバーで使用されることを意図しています。 これを使用する場合は、LDAP を配置する必要があります。 NFS のマッピングは、`chnfsim` ユーティリティーを 使用して管理されます。

NFS バージョン 2 および 3 と NFS バージョン 4 で同時アクセスを使用する場合は、考慮しなければならないことがあります。NFS バージョン 3 アクセスでは、NFS バージョン 4 で認可される状態のために、エラーが発生する場合があります。 また、NFS バージョン 4 アクセス用にデータをエクスポートすると、NFS バージョン 3 のパフォーマンスが影響を受ける場合があります。

NFS サーバー猶予期間

NFS バージョン 4 (NFSv4) プロトコルは、システム管理者が NFSv4 サーバーで猶予期間を使用可能にして、特定の操作を特別に処理するための機能を提供します。

この猶予期間内、管理者はサーバー・リリースの全期間中、ロック、読み取り操作、および書き込み操作を管理できます。 ロックとその関連状態は、再利用型のロック要求を使用してクライアントによりリカバリーされます。

注 : 猶予期間内にクライアントで再利用されるすべての状態が、前のインスタンスのサーバーで保持される状態になるとは限りません。 猶予期間内に再利用される状態は、NFSv4 RFC で定義されている正しい状態になります。

AIX 5L バージョン 5.3 (5300-05 テクノロジー・レベル適用) 以降、管理者は NFSv4 サーバー上で猶予期間を使用できます。 猶予期間は、デフォルトで使用不可になっています。 サーバーで猶予期間を使用可能にするには、SMIT メニューを使用するか、**chnfs** コマンド・ライン・インターフェースを 使用します。

猶予期間が使用可能になっている場合、NFSv4 サーバーは状態情報を `/var` ファイル内のディスクに記録します。 記録された状態は、サーバーの再始動時に自動的に再利用されます。

NFS DIO および CIO サポート

AIX 5L バージョン 5.3 (5300-03 推奨メンテナンス・パッケージ適用) は、バージョン 3 と 4 のプロトコルの NFS クライアントにおいてダイレクト I/O およびコンカレント I/O をサポートします。 DIO および CIO は、クライアントにだけ関係します。

DIO および CIO を使用すると、 ファイル・ベースのストレージおよびそれに関係するバックエンド・システムの 管理の集中化という利点は維持しながら、 データベースやハイパフォーマンス・コンピューティングのアプリケーション など、データ・センター・ワークロードでより高いレベルのパフォーマンスを 得ることができ、しかも、システム CPU リソースおよびメモリー・リソースを削減できます。

I/O は順次ではないことも多く、アプリケーションは、しばしば NFS クライアントでのデータ・キャッシングの恩恵を受けないか、あるいは、 すべての拡張キャッシング行います。 こういったアプリケーションは、NFS がキャッシュを使用しないとき、 読み取り予測を行わないとき、あるいは、後書きメカニズムを使用しないときに恩恵を受けます。 さらに、データベースなどのある種のアプリケーションは、 読み取りを書き込みにシリアライズする POSIX シングル・サイトのセマンティクスには 依存しません。 こららのアプリケーションは読み取りと書き込みを並行して発行しますが、 このような操作の整合性と調整については、アプリケーションに責任があります。

NFS のダイレクト I/O

DIO は、アプリケーションが NFS クライアント・キャッシング層 (仮想メモリー・マネージャー) を通過したり データ・キャッシングに関連するオーバーヘッドを負担することなしに NFS サーバーに直接に読み取りおよび書き込みを行うことを可能にします。

DIO の場合は、アプリケーション入出力要求は、NFS サーバーに対する直接リモート・プロシージャー・コール (RPC) を使用して処理されます。 DIO は、AIX *dio* マウント・オプションを使用して設定できます。 マウント・オプションを指定せずに、`AIX O_DIRECT open()` フラグを使用して DIO per ファイルを使用可能化することもできます。

NFS ダイレクト I/O を使用するには、I/O 要求サイズ、および、サーバーとクライアントが許可している最大ワイヤー転送サイズ に従って、サーバーに対する複数の RPC が必要になることがあります。 DIO についての 詳細は、**mount** コマンドの `-o` オプションを参照してください。

NFS のコンカレント I/O

CIO では、並行して出されたアプリケーションの読み取りと 書き込みは、書き込み中の読み取り妨害なしに、あるいは、読み取り中の 書き込み妨害なしに、並行して実行されます。

複数の書き込みも並行して行われます。 POSIX のアトミシティ保証は、提供されていません。 CIO が有効になっているときは、ダイレクト I/O も暗黙に設定されています。 CIO を設定するには、AIX の *cio* マウント・オプションまたは `O_CIO open()` フラグを使用します。 CIO についての 詳細は、**mount** コマンドの `-o` オプションを参照してください。

AIX バージョン 6.1 (6100-04 テクノロジー・レベル適用) 以降では、読み取り専用ファイルが既に CIOR でオープンされているときにこれらのファイルをオープンできるようにするために、**mount** コマンド、**nfs4cl** コマンド、または **open()** サブルーチンを実行できます。 **cior** マウント・オプションおよび **O_CIOR open ()** フラグは、CIO との連動でのみ使用できます。

関連情報

mount コマンド

NFS の DIO、CIO、通常オープン、およびマップ・ファイルの相互作用

DIO および CIO で起こり得る各種のアクセス・モードでの動作は以下のようになります。

既存の DIO オープンが有効になっている場合は、

- 通常のオープンは、通常のオープンがもうないという状態になるまで、DIO をオフにする。 閉じられたために通常のオープンが 0 になったとき、 まだ未解決の DIO オープンがあれば、DIO が再使用可能化されます。
- ファイルを shmat() または mmap() でマップすると、マッピング数が 0 にドロップするまでそのファイルに対する DIO は非活動化される。0 になった時点で、まだ DIO オープンがあれば、DIO は再び使用可能になります。
- CIO 用にファイルをオープンしようとすると、EINVAL エラーとなって失敗する。

通常のオープン (CIO または DIO ではないもの) が有効になっている場合は、

- DIO オープンの試行は成功するが、DIO は通常のオープン数が 0 に ドロップするまで活動化されない。
- CIO 用のオープンは、EINVAL エラーで失敗する。

CIO オープンが有効になっている場合は、

- 通常のオープン、DIO、およびファイルをマップする試みは、どれも、EINVAL エラーで 失敗する。

CIO|CIOR オープンが有効になっている場合は、

- 通常オープン、DIO、およびファイルをマップする試みは、読み取り専用オープンおよび CIO|CIOR オープンを除き、いずれも EINVAL エラーで失敗する。

注 : DIO または CIO への切り替えがある場合は、クライアントにキャッシュされている 変更がまず最初に NFS サーバーに書き戻されてから、すべてのキャッシュ情報が 削除されます。

NFS の複製とグローバル・ネームスペース

NFS バージョン 4 (NFSv4) プロトコルは、システム管理者が データのユーザーには透過的な方法で複数のサーバーにデータを分散できる機能を提供します。

AIX 5L バージョン 5.3 (5300-03 推奨メンテナンス・パッケージ適用) から提供が開始された 2 つの機能を使用できます。 最初の機能は、参照 と呼ばれるグローバル・ネームスペース・フィーチャーです。 2 つ目の機能は、レプリカ と呼ばれるデータのコピーを検出できるロケーションの指定方法です。

参照 は、サーバーのネームスペースに作成することができる 特殊なオブジェクトで、そのネームスペースにはロケーション情報が付加されています。 サーバーは NFSv4 プロトコル・フィーチャーを使用し、ロケーション情報に指定されているサーバーにクライアントをリダイレクトします。 参照によって、複数の NFS サーバー上のデータを単一の ファイル・ネームスペース・ツリーに統合するための構築ブロックが形成されます。参照機能対応の NFSv4 クライアントは、このツリーをナビゲートできます。

レプリカ とは、ある NFS サーバーのファイルシステムのコピーで 、 他の複数の NFS サーバー (または、同じサーバー上の異なるディスクなどの 代替ロケーション) に置かれているものです。 レプリカ対応の NFSv4 クライアントは、使用中のレプリカ・ロケーションが使用不能になると、 別の使用可能なレプリカに切り替えます。 レプリカについての詳細は、584 ページの『NFS のレプリカ』を参照してください。

NFS の参照機能

次の例は、参照機能について理解していただくためのシナリオです。

次の例では、4 つのサーバーがあります。

- publications という名前のサーバーには、文書ファイルが入っている。
- projects という名前のサーバーには、ユーザー作業ディレクトリーがある。
- data という名前のサーバーには、情報データベースが入っている。
- account1 という名前のサーバーは、すべての他のファイルをエクスポートする メイン NFS サーバーであり、すべてのクライアントが認識しているサーバーである。

メイン NFS サーバー上のファイルへのアクセスをすべてのクライアントに 許可する

サーバー account1 は、/etc/exports で 以下のステートメントを使用して、ディレクトリー /work をすべてのクライアントにエクスポートします。

```
/work -vers=4
```

すべてのクライアントは、以下のコマンドを使用して、account1 サーバーから / を /mnt ディレクトリーにマウントすることに よって、/work リモート・ディレクトリーにあるファイルにアクセスできます。

```
mount -o vers=4 account1:/ /mnt
```

クライアント上のユーザーが /mnt ディレクトリーの内容をリストすると、パス /mnt/work のリモート・ディレクトリー work が表示されます。 クライアントの /mnt/work ディレクトリーの内容は、account1 サーバー の /work ディレクトリーの内容と同じです。

特定のサーバーにあるファイルへのアクセスをクライアントに 許可する

クライアント・ユーザーが publications サーバーにある /usr/doc ディレクトリーにもアクセスしたいとします。

前のリリースでは、サーバーから該当するディレクトリーをエクスポートし、 そのディレクトリーをクライアントにマウントする必要がありました。

分散ネームスペースを構築するために参照機能を使用する

システム管理者は、クライアントがデータの位置を知らないまま、他のサーバーにあるデータにアクセスできるようにサーバーを セットアップすることができます。 サーバーを参照する管理者だけが、そのデータ のありかを知っていればよいのです。 参照サーバーは、参照機能を 使用して、クライアント を /usr/doc ディレクトリーのロケーションに リダイレクトすることができます。 サーバー publications で は、エクスポート・ファイルに次のステートメントを追加することによって /usr/doc ディレクトリーをエクスポートできます。

```
/usr/doc -vers=4
```

これによって、このディレクトリーが NFSv4 クライアントに使用可能になります。

これで、account1 サーバーは、参照機能を使用し、エクスポート・ファイルに 次のステートメントを追加することによって、これらのディレクトリー をクライアントに使用可能にすることができます。

```
/usr/doc -vers=4,refer=/usr/doc@publications
```

ここで、ディレクトリーをエクスポートします。 この時点で、account1 サーバーの / ディレクトリーから /mnt ディレクトリーをマウントしたクライアント は、/mnt ディレクトリーをリストしてみると、ディレクトリー usr に対する アクセス権をもっています。 クライアントは、他のサーバーにマウントを行う必要はありません。 クライアント・ユーザーは、そこにあるファイルが account1 サーバーの提供によるものではないことを知る必要さえありません。 例えば、サーバー data 上の /databases/db のディレクトリーと、サーバー projects 上の /home/accts ディレクトリーを data および projects サーバーからエクスポートし、それらのディレクトリーの参照を account1 に 作成すれば、それらのディレクトリーを account1 を介して使用可能にすることができます。

クライアント・ユーザーはデータの実際のロケーションを知らないので、 管理者は、単にサーバーのエクスポート・ファイルで参照ステートメントを変更するだけで クライアントをあるサーバーから 別のサーバーにリダイレクトすることができます。 データの ロケーション指定によって参照機能が 参照するデータの配置と正確性については、管理者に責任があります。

管理者は、2 番目のサーバーで要求が最初のサーバーに参照し直されるような循環型参照を作ら ないようにする必要があります。 上記の例では、管理者が account1 サーバー上の /usr/doc を 参照する参照を publications サーバーの /usr/doc に作成して いたとすると、結果的に、望ましくない循環参照ができあがります。

参照は、exportfs を使用して作成されますが、 これはデータのエクスポートとは異なります。 参照のために指定されるロケーションは、NFSv4 エクスポート ・ファイルシステムの root ディレクトリーに対応する必要があります。 参照は、エクスポート済みのネームスペース内、またはアンエクスポートのネームスペース内に 作成できます。 上記の例では、/usr/doc 参照は、/usr がエクスポートされていなくても account1 サーバーに作成することができます。 これは、参照を NFSv4 疑似スペースに置きます。 account1 が /usr をエクスポートしていた場合でも、参照エクスポートを行うことができます。これは、doc という名前の ディレクトリーのエクスポートが、そのディレクトリーが同一ファイルシステム にある場合は失敗するのと対照的です。 どちらの場合も、参照エクスポートは、 ファイルまたはディレクトリーが /usr/doc に存在していると、失敗します。 サーバーの NFSv4 疑似スペース内、またはエクスポート・ファイルシステム 内 に作成できる参照の数には制限がありません。

参照は、データを何もエクスポートせず、NFSv4 プロトコルに対して意味を 持つだけなので、NFSv4 でのみ使用可能です。 vers=4 オプションなしで参照をエクスポートすると、失敗します。 この例ではロケーションはただ 1 つだけ指定していますが、最高 8 個のロケーションを 指定できます。

参照を作成すると、ディレクトリー・パラメーターによって指定される ロケーションに、特殊な参照オブジェクトが作成されます。 オブジェクトに対する クライアント・アクセスは、クライアントによるそのオブジェクトの親ディレクトリーへの アクセスによって判別されるので、ほとんどの他のエクスポート・オプションには 意味がなく、指定してもかまいませんが、無視されます。 唯一の例外は、exname オプションで、これには予期される動作があります。 例えば、サーバーが参照 /n4root/special/users -

vers=4,exname=/exported/users,refer=/restricted/users@secrethost を作成すると、そのサーバーから / をマウントするクライアントは、パス /mnt/exported/users を見ることになり、これはクライアントを secrethost の /restricted/users ディレクトリーにリダイレクトします。 エクスポートするサーバー上では、参照オブジェクトは実際には /n4root/special/users のローカル・ネームスペースに作成され、 したがって、エクスポートが完了しても、そこにはファイルもディレクトリーも 存在することはできません。 参照のロケーション情報を保持するために、サーバーに 特別なオブジェクトが作成されます。 参照に到るまでのパスのディレクトリーは、存在していなければ作成されます。 参照がアンエクスポートである場合は、参照情報はオブジェクトから除去されますが、 オブジェクト自体は除去されません。 NFSv4 サーバーでは、不整合 または孤立 参照オブジェクトが結果としてできた場合、 クライアントはそれらにアクセスすることはできません。 そのオブジェクトに アクセスしようとすると、クライアントにはアクセス・エラーが戻されます。 このオブジェクトは、必要ならば **rm** を使用して除去できます。 参照は、新規の参照情報を付けて再エクスポートすることができます。 ただし、これを頻繁に行うことはお勧めできません。 その参照をアクセスするクライアントがロケーション 情報の変更に気付くのに時間がかかる恐れがあるためです。 サーバーは、ディレクトリー内の 情報が変更したことを示すために、参照の親ディレクトリーに連絡します。 この連絡は、そのディレクトリーについて キャッシュに入れておいた情報 (およびディレクトリー内の参照) が変り、 再取り込みが必要になったことをクライアントに気付かせるのに役立ちますが、 クライアントが気付くのにどれほどの時間がかかるかは保証されません。

ファイルシステム・ロケーション・リストで指定されたロケーションの順序を変更するための refer オプションの使い方については、『scatter オプションを使用したファイルシステム・ロケーション・リストの再配列』を参照してください。

NFS のレプリカ
NFSv4 の管理者は、複製によって、複数の NFSv4 サーバー上にデータのコピーを置き、NFSv4 クライアントにそのレプリカの在所 を通知することができます。

基本データ・サーバーがクライアントにアクセス 不能になった場合は、クライアントは、レプリカ・サーバーの 1 つを使用して、 複製されたファイルシステムで 操作を継続することができます。 レプリカ・ファイルシステムは、基本サーバーにあるデータの正確なコピーで あると想定されます。 最高 8 つまでのレプリカ・ロケーションをセットアップできます。 AIX サーバーは、レプリカ・ファイルシステムを 基本ファイルシステムから 作成する方法やデータに矛盾がないようにする方法は指定しません。 レプリカを読み取り/書き込み用として指定する場合は、レプリカのデータが 基本ファイルシステムと 矛盾がないように保持する必要があります。

レプリカは、別のサーバーのディレクトリー (複数の場合もある) のコピー を持つサーバーです。 クライアントは、基本サーバーが使用不可になると、 レプリカ・ロケーションから同じファイルにアクセスすることができます。 次のシナリオで例を示します。

account1 サーバーにある /data ディレクトリー内のファイル が、サーバー inreserve にある /backup/data ディレクトリーでも 使用可能である場合、NFSv4 クライアントにそのことをエクスポート上の レプリカ・ロケーションの指定によって知らせることができます。 管理者は、次のようなステートメントを 追加することにより、/data ディレクトリーをエクスポートし、 レプリカ・コピーのロケーションを指定することができます。

```
/data -vers=4,replicas=/data@account1:/backup/data@inreserve
```

account1 サーバーが使用不可になったとき、account1 サーバーの /data ディレクトリーの下にあるファイルを使用している クライアント・ユーザーは、inreserve サーバーの /backup/data ディレクトリーにあるファイルの使用を開始することができます。 このとき、このクライアントは、別のサーバーに切り替わったことは認識しません。

ファイルシステム・ロケーション・リストで指定されたロケーションの順序を変更するための replicas オプションの使い方については、『scatter オプションを使用したファイルシステム・ロケーション・リストの再配列』を参照してください。

レプリカの指定を可能にするための *NFS* 構成要件
管理者は、ルート・レプリカの使用可能化、使用不可化、または指定を行える必要があります。

ルート・レプリカの使用可能化、使用不可化、および指定を行うには、次のコマンドを使用します。

```
chnfs -R {on|off|host[+host]}
```

レプリカを指定するには、サーバーは、揮発性 NFSv4 ファイル・ハンドルを 発行するために **chnfs -R** (**chnfs -R** on) で構成される 必要があります。 ファイル・ハンドルは、NFS サーバーがサーバー 上のファイルまたは ディレクトリーを識別するためにクライアントに発行する ID です。 デフォルトでは、サーバーは持続ファイル・ハンドルを発行します。 ファイル・ハンドル・タイプを切り替えると、その切り替えが行われたときに サーバーをアクティブに 使用していた NFSv4 クライアントでアプリケーションのエラー が起きることがあります。 **chnfs -R** によってファイル・ハンドル・モードを 変更するためには、どのファイルシステムも NFSv4 アクセス用にエクスポートできません。 ファイル・ハンドルのファイル属性指定は、新規提供の NFS サーバーで行うか、または NFS アクティビティーを 最小化または停止できるときに行う必要があります。 モードを変更するときにサーバーにアクティブに 接続していたクライアントは、それらのクライアントの NFSv4 マウントをアンマウントし、再マウントする 必要がある場合があります。 このアクションを最小化するために、 クライアント・マウント数を少なくして、NFSv4 サーバーの エクスポート・ファイル・スペースのトップレベル・ディレクトリー をマウントするだけに削減することができます。

NFSv4 クライアントは、異なるエクスポート・アクセス属性を持つレプリカに フェイルオーバーすることはできません。 管理者は、すべてのレプリカが 同じエクスポート・アクセス制御と同じアクセス・モード (読み取り専用または 読み取り/書き込み) で指定されるようにする必要があります。 エクスポートされる GPFS では例外がありえますが、複製データは読み取り専用にエクスポートされるものと予期されます。 また、すべてのレプリカ・ロケーションで、データ内容を保守するのも管理者の 仕事です。 ディレクトリー・ツリーとすべてのデータ内容は同一に保持されなければなりません。 データ内容を更新するときは、データを使用することになるアプリケーション 間で最も互換性のある方法で行うべきです。

レプリカに関しては、exname エクスポート・オプションを使用すること によって、サーバーのローカル・ファイルシステム・ネームスペースの詳細を NFSv4 クライアントには非表示にすることができます。 詳しくは、exportfs コマンドおよび /etc/exports ファイルを参照してください。

General Parallel File System (GPFS) などのエクスポート・クラスター・ファイルシステムで replicas オプションを使用して、同じ GPFS ビューが表示される複数の NFS サーバー・ノードを指定できます。 この構成では、読み取り/書き込みアクセス用のデータのエクスポートが有効になります。 ただし、読み取り/書き込みレプリカでは、書き込み操作の進行中に レプリカのフェイルオーバーが 発生すると、その書き込みを行っていた アプリケーションに回復不能なエラーが起きることがあります。 同様に、フェイルオーバー中に実行していた **mkdir** または 排他的ファイル作成操作は EXISTS エラーになることがあります。

複製のエクスポートでは、ファイルシステム 全体をエクスポートする必要があります。 すなわち、エクスポートされるディレクトリーは、ローカル・ファイルシステムの ルートでなければなりません。 複製されるファイルシステムをエクスポートする サーバーは、そのサーバー自体をそのエクスポートのロケーションの 1 つとして指定する必要があります。 複数のインターフェースを 持つサーバーの 場合は、その中にサーバーの基本ホスト名を入れる必要があります。 複製ファイルシステムをエクスポートするサーバーがそのサーバー自体を エクスポートのロケーションの 1 つとして指定しないと、そのエクスポートを行う サーバーは、最初のレプリカ・ロケーションとして自動的にレプリカ・ロケーション・リストに追加され ます。 レプリカ・リスト内でのレプリカ・ロケーションの順序が、フェイルオーバー時に クライアントが使用するロケーションの優先順序を指定します。 例えば、serverA で ユーザーが /webpages をエクスポートしたいとします。このとき serverB の /backup/webpages ディレクトリーに /webpages のレプリカを置くとすれば、/etc/exports ファイルの次のエントリーに よって、/webpages が serverA からエクスポートされ、 さらに、serverB の /backup/webpages にそのファイルシステム のコピーがあることがクライアントに通知されます。

```
/webpages -vers=4,ro,replicas=/webpages@serverA:
/backup/webpages@serverB
```

serverA の /webpages も、serverB の /backup/webpages も、そのファイルシステムのルート・ディレクトリーである と見なされます。 serverA は、エクスポートでリストされていなければ、 最初のレプリカ・ロケーションとして自動的に追加されます。 これは、エクスポートされる データにとって、そのデータをエクスポートしているサーバーが 優先されるサーバーだと考えられるためです。

レプリカは、NFSv4 プロトコルでのみ使用されます。 上記のエクスポートを NFSv3 (vers=3:4) と指定することもできますが、その複製情報は NFSv3 クライアントには使用可能になりません。 ただし、NFSv3 を

使用するクライアントは、serverA の /webpages の 情報にはアクセスできます。しかし、serverA が使用不可になっても、 そのレプリカにフェイルオーバーはしません。

複数ロケーションでの *NFS* クライアント側のサポート

クライアントは、現行サーバーから複製データにアクセスできない事態が起きると、 次に最も優先順位の高いサーバーからデータをアクセスしようとします。

クライアントは、レプリカ・リストに指定されている順序がレプリカの 優先順位であると解釈します。

クライアント管理者は、**nfs4cl** コマンドの **prefer** サブコマンドを使用して、レプリカの優先順位をオーバーライドすることができます。 **nfs4cl** コマンドは、すべてのファイルシステム 情報をクライアントに 表示したり、あるいは、ファイルシステムのファイルシステム ・オプションを 変更し、現在の NFSv4 統計と属性を表示または変更します。

NFS レプリカと参照に対する共通した考慮事項

同一のデータ (ファイルシステム) に到る異なる 2 つのパスがあった場合は、 クライアントは、2 つ目のパスをファイルへのシンボリック・リンクであると して扱います。

例えば、server A が以下のようにエクスポートしたとします。

```
/tmp/a              -vers=4,replicas=/tmp/a@B:/tmp/a@A
/tmp/b              -vers=4,refer=/tmp/a/b@B
```

また、server B が以下のようにエクスポートしました。

```
/tmp/a              -vers=4
/tmp/a/b            -vers=4
```

この例では、クライアントは、コマンド mount -o vers=4 A:/ /mnt を使用して server A の / を /mnt にマウントします。 クライアント・ユーザーは、cd /mnt/tmp/a/b または cd /mnt/tmp/b で、server B 上の /tmp/a/b にアクセスします。 まずユーザーがディレクトリーを cd /mnt/tmp/a/b に変更すると、パス /mnt/tmp/b は /mnt/tmp/a/b へのシンボリック・リンクとして機能します。 このシナリオでは、ユーザーが /mnt/tmp/b 内に存在し、コマンド /bin/pwd を使用する場合、/bin/pwd > は /mnt/tmp/a/b を戻します。

注 : 上記のようなエクスポート指定はお勧めできません。 管理者は、エクスポートされたデータへのパスとして考えられるネームスペース・パスが 1 つだけであるように エクスポート指定をセットアップしてください。

参照のターゲット・データが実際に複製されていれば、参照に複数のロケーションを リストすることができます。 クライアントは、使用可能なサーバーにある参照 ターゲットを検出するためにのみ参照ロケーションを使用します。 クライアントは、参照ターゲットへのアクセスをいったん確立すると、その検出された データのための新しいロケーション情報を獲得します。

クライアントは、参照ロケーション情報についての変更を即時には検出しないことがあるので、 参照ロケーションを頻繁に除去したり変更したりすることはお勧めできません。 参照ロケーションのターゲットを再配置するときは、エクスポートの参照指定 内のロケーション情報を変更するとともに、その新しいロケーションを転送するように推奨します。 古いロケーションにあったデータは数時間または数日間保持して、クライアントが新しい ロケーションを知り、使用できるようになるまでの時間的な余裕を残してください。

複製も参照も、64 ビット・カーネルを実行しているサーバーでのみ実行できます。 クライアントは、32 ビット・カーネルでも 64 ビット・カーネルでも実行できます。

レプリカを読み取り/書き込み用として指定する場合は、レプリカのデータを 基本ファイルセットと矛盾がないように保持する必要があります。

NFS クライアントのフェイルオーバー動作

フェイルオーバー とは、クライアントが、現在通信している サーバーがアクセス不可になったと判断した後、レプリカ・ロケーションを 別のレプリカ・ロケーションへ切り替えることです。

以下のチューナブル・オプションは、NFS クライアントのフェイルオーバー 動作に影響します。

NFS マウント・オプション *timeo*
このマウント・オプションは、TCP/IP 層がタイムアウト応答で戻るまでの 待機時間を指定します。

NFS マウント・オプション *retrans*
このマウント・オプションは、NFS RPC 層が RPC タイムアウト・エラー (ETIMEDOUT) を戻すまでにクライアント 要求を何回試行するかの 回数を指定します。

nfso オプション *nfs_v4_fail_over_timeout*
この nfso オプションを使用すると、クライアントがレプリカの フェイルオーバーを行うまでの最小待機時間を指定できます。 このオプションは、NFS クライアント に対するグローバルな指定であり、マウントごとのデフォルトの動作をオーバーライドします。 デフォルトでは、nfs_v4_fail_over_timeout はアクティブではありません。 その値は 0 になっています。

nfs_v4_fail_over_timeout が非アクティブのときは、 フェイルオーバーしきい値はマウントの timeo オプションの値の 2 倍に設定されています。 この時間間隔の間に RPC コールが成功しないと、クライアントは別の使用可能なレプリカを 見るけるためにフェイルオーバー処理を開始します。 ただし、実際にクライアントが 待機する時間は、retrans オプションによって変わります。 retrans が 2 よりも大きいと、クライアントは retrans 値 に timeo 値を乗じた値 (retrans×timeo) によって決まる RPC タイムアウト を受け取るまでおそらく待機します。 したがって、timeo と retrans のオプションによって、NFS マウント・ベースで フェイルオーバーの動作を制御するため調整することができます。 また、これらのオプションを **nfs4cl** コマンドによって 細分レベルをもっと上げて設定することもできます。

nfso nfs_v4_fail_over_timeout を非ゼロ値に設定する場合は、その値は、クライアントが使用不可サーバーでレプリカのフェイルオーバーを 考慮するまでの待機時間を秒数で示します。 timeo と retrans の オプションが nfso の設定を超える RPC タイムアウト動作になる場合は、 フェイルオーバー処理は、その RPC タイムアウトが生成されるまでは始まりません。

retrans、timeo、および nfs_v4_fail_over_timeout のオプションの詳細については、**mount**、**nfs4cl**、および **nfso** のコマンドの NFS 固有オプションを参照してください。

サーバーが使用不可になるという事態でレプリカのフェイルオーバーが起きる他に、 クライアントが 1 つのレプリカ・ロケーションから別のレプリカ・ロケーションに自発的に 切り替えを行う場合があります。 1 つは、管理者が **nfs4cl** コマンド を使用して優先レプリカを設定する場合です。 この場合、そのサーバーが 現在クライアントが使用しているサーバーでなければ、 クライアントは優先サーバーへの切り替えを開始します。 また、関連データに最新のアクティビティーがあった 場合、クライアントは、およそ 30 分の時間間隔で NFS サーバーから レプリカ・ロケーション情報の再取り込みを行います。 ロケーションの順序が変更になっている場合は、 そのサーバーが現在クライアントが使用しているサーバーでなくて、 かつ、**nfs4cl** コマンドで管理者がレプリカ優先を設定していないならば、 クライアントは最初のロケーションに切り替えようとします。

ソフト *NFS* マウントに対するフェイルオーバー動作
NFS のデフォルトのマウント・モデルはハード・マウントで、レプリカのフェイルオーバー 動作がハード・マウントに適用されます。 NFS のソフト・マウントが使用される場合は、フェイルオーバー 動作は異なります。

ソフト・マウントの設定によって、レプリカ・フェイルオーバー のために設定された待機時間より前に RPC タイムアウトが起きた場合は、 そのタイムアウトの結果、呼び出し側アプリケーションが ETIMEDOUT エラーになります。 複製データがある場合のソフト・マウントの使用はお勧めできません。 ソフト・マウントを使用し、nfso nfs_v4_fail_over_timeout 値が 設定されている場合は、その nfso 設定を超えるように retrans と timeo のマウント・オプションを設定するようにお勧めします。 このような設定によって、複製データを使用するアプリケーションに ETIMEDOUT が戻るのを回避します。

scatter オプションを使用したファイルシステム・ロケーション・リストの再配列
exportfs コマンドの scatter オプションを使用すると、**exportfs** コマンドの refer オプションまたは replicas オプションのいずれかを使用してファイルシステム・ロケーション・リストに指定されたロケーションの順序を変更できます。

このオプションを使用すると、サーバー・ロケーションのさまざまな組み合わせを生成できるので、異なるサーバーを好みの順序に配列した複数のリストを作成できます。 その結果、異なるクライアントに異なるサーバー・ロケーション・リストを指定することができます。 このような再配列を行うと、異なるクラ

イアントのロケーション・リスト内で異なるサーバーが最初のサーバーになるため、ロード・バランシングに役立ちます。 また、サーバーがダウンした場合、サーバー・ロケーション・リスト内の次のロケーションにあるサーバーも異なるので、フェイルオーバー負荷が複数のサーバーに分散されます。 scatter オプションは、NFS バージョン 4 プロトコルによるアクセス用にエクスポートされたディレクトリーにのみ適用されます。

scatter オプションの値は次のとおりです。

- **full** - すべてのサーバーが、代替ロケーションの組み合わせを形成するように再配列されます。 組み合わせの総数は、12 またはサーバー数のいずれか大きい方に制限されます。
- **partial** - 生成されるすべての組み合わせで、最初のサーバー・ロケーションはサーバー・リスト内の最初のサーバーに固定されます。 それ以外のロケーションは、full オプションで再配列される場合と同様にリストされます。
- **none** - ファイルシステム・ロケーション・リストの再配列は行われません。 これが scatter オプションのデフォルト値です。 この値は、ロケーション・リストの以前の再配列を無効にする場合に使用します。

注 : exportfs コマンドを実行するときに noauto フラグを指定しない場合、ロケーション・リストにはレプリカ・ロケーションの 1 つとして 1 次ホスト名が入れられます。

ホスト s1、s2、および s3 で /common/documents ディレクトリーの参照機能を指定してから full オプションを使用してそれらを再配列するには、/etc/exports ファイルに次の行を追加してから、/common/documents ディレクトリーをエクスポートします。

```
/common/documents -ver=4, refer=/common/documents@s1:/common/document@s2a:/common/
documents@s3,scatter=full
```

ホスト s1、s2、s3、および s4 で /common/documents ディレクトリーのレプリカを 指定してからそれらを部分的に (すべての組み合わせで s1 が最初のフェイルオーバー・サーバーになるように) 再配列するには、/etc/exports ファイルに次の行を追加してから、/common/documents ディレクトリーをエクスポートします。

```
/common/documents -vers=4, replicas=/common/documents@s1:/common/documents@s2:/common/
   documents@s3:/common/documents@s4,scatter=partial
```

NFS サーバーとクライアント間の委任

委任 とは、ある種の責任をクライアントに委任するサーバーの機能です。

AIX 5L バージョン 5.3 (5300-03 推奨メンテナンス・パッケージ適用) から、委任機能を使用できます。 サーバーがファイルについての委任をクライアントに付与すると、 そのクライアントは、そのファイルを他のクライアントと 共有することに関して ある種のセマンティクスを保証されます。 ファイルがオープンしていれば、サーバーは そのクライアントにファイルの読み取り委任を提供できます。 クライアントに読み取り委任が 付与されると、その他のクライアントはその 委任の間はそのファイルへの 書き込みができないようになります。 クライアントに書き込み委任が 付与されると、その他のクライアントはそのファイルへの 読み取りおよび書き込みアクセスができないことがそのクライアントに約束されます。 AIX サーバーは、読み取り委任のみを付与します。 AIX サーバーは、64 ビットの AIX カーネルでのみ委任をサポートします。 AIX クライアントは、 読み取りと書き込みの両方の委任をサポートします。

サーバーがクライアントに 委任を付与するためには、 クライアントは、最初にサーバーにコールバック・アドレスを提供する必要があります。 委任がリコールされると、サーバーは、リコール要求をこのアドレスに 送信します。 デフォルトによって、クライアントは、サーバーとの正常な通信に 使用されている IP アドレスを示します。 複数のネットワーク・インターフェースを 持つクライアントの場合は、特定のアドレスを /etc/nfs/nfs4_callback.conf ファイルで指定できます。 このファイルでのエントリーのフォーマットは、以下のとおりです。

server-host client-ip-address

ここで、*server-host* は NFSv4 サーバーの名前またはアドレス、*client-ip-address* は サーバー・コールバック情報を提供するときに使用されるクライアント・アドレスです。 *server-host* 名が IPv4 アドレス 0.0.0.0 または IPv6 アドレス 0::0 である場合は、指定された *client-ip-address* がそのファイルにリストされていないすべての サーバーに使用されます。 このファイルが存在しないか、または サーバーのためのエントリー

(またはデフォルト・エントリー) が見つからない 場合は、クライアントは、サーバーへの既存の接続に基づいてアドレスを 選択します。

委任は、サーバーによってリコールされることがあります。 付与されている委任と矛盾するようなアクセスを他のクライアントが要求した場合は、 サーバーは初期のクライアントに通知するとともにその委任をリコールすることができます。 そのためには、サーバーとクライアントとの 間にコールバック・パスがあることが必要です。 このコールバック・パスがないと、委任は付与できません。 ファイル委任が 付与されていると、他の NFSv4 クライアントからのアクセス、NFS バージョン 2 および 3 のクライアントからのアクセス、およびファイル・サーバーでのそのファイルへの ローカル・アクセスは、委任のリコールとなることがあります。 GPFS を NFSv4 でエクスポートする場合、ネットワーク内の GPFS ノードからアクセスすると、委任のリコールの原因となることがあります。

委任において重要なことは、委任によって、クライアントはローカル・サービス 操作 (OPEN、 CLOSE、 LOCK、 LOCKU、 READ および WRITE など) をサーバーとの 即時相互作用なしに行えることです。

サーバー委任は、デフォルトによって使用可能化されます。 サーバー委任は、`nfso -o server_delegation=0` コマンドで使用不可に することができます。 管理者は、**exportfs** `deleg=yes | no` オプションを使用して、 ファイルシステム 単位ごとに委任付与を可能にしたり、不可にすることができ、これは **nfso** の設定をオーバーライドします。

クライアント委任は、`nfso -o client_delegation=0` コマンドによって 使用不可にすることができます。 クライアント 委任は、クライアントでのマウントが行われる前に設定する必要があります。

管理者は、多数のクライアントが多数の共通ファイルに書き込みを行うファイルシステムを エクスポートする場合は、そのファイルシステムに 対する委任を使用不可にすることができます。

クライアントに連絡できない (例えば、ネットワークまたはクライアント に障害が起きている) と、他のクライアントはデータのアクセスが 遅延することがあります。

Kerberos 保護のコールバック・パスに対する汎用ホスト・プリンシパルの確立

IBM ネットワーク認証サービス (Kerberos) に対して、コールバック・パスをセットアップできます。

委任を受け取るクライアントは、それ自体のホスト・プリンシパルを持つフル・クライアントでなければなりません。 ただし、すべてのクライアントの汎用ホスト・プリンシパルを確立してコールバックに使用することができます。

すべてのクライアントの汎用ホスト・プリンシパルを確立してコールバックに使用する手順は、次のとおりです。

1. ホスト・プリンシパルの 作成に使用する方法と同じ方法でサービス・プリンシパル (例えば、`nfs/client`) を作成するには、セキュリティーの Kerberos プリンシパルの作成を参照してください。
2. サービス・プリンシパルの keytab エントリーを作成します。
   例えば、slapd_krb5.keytab という keytab を作成するには、以下のように実行します。

```
kadmin.local: ktadd -k /etc/security/slapd_krb5.keytab ldap/plankton.austin.ibm.com
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/security/slapd_krb5.keytab.
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/security/slapd_krb5.keytab.
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab
WRFILE:/etc/security/slapd_krb5.keytab.
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/security/
slapd_krb5.keytab.
kadmin.local:
```

3. この keytab を使用するすべてのクライアントに配布します。
4. **nfshostkey** コマンドを使用してクライアントを構成します。

このプロセスは Kerberos と使用するサーバーを構成するプロセスと同一ですが、この汎用プリンシパルはサーバーには使用できません。各サーバーには、`nfs/`*`hostname`* 形式のサーバー自体のプリンシパルが必要です。

STNFS 短期ネットワーク・ファイルシステム

短期ネットワーク・ファイルシステム (STNFS) ファイルシステムは、ネットワーク・ファイルシステム (NFS) によってバックアップされるファイルシステムです。 STNFS ファイルシステムによって、ファイルに対するローカル変更が可能になります。 変更はサーバー上に保管されません。

注:

1
多くの STNFS クライアントが 1 つのサーバーの同じファイルシステム ・イメージを共有できますが、すべての変更は、その変更を行うクライアントによってのみ表示されます。

2
クライアントによって行われたすべての変更は、ファイルシステムのアンマウント 時またはクライアントのリブート時に失われます。

3
システム・メモリーが所定のしきい値を下回った場合は、STNFS からの書き込み操作は失敗します。このしきい値は、STNFS の内部にあり、外部から構成することはできません。

NFS 短期ファイルシステムのマウント

mount コマンドは、短期ベースで NFS ファイルシステムをマウントするために 使用します。例えば、次のコマンドを入力します。

```
mount -v stnfs -o options server:/remote-path /local-path
```

使用可能なオプションは次のとおりです。

vers=3
NFS バージョン 3 を使用してサーバーと通信します。

vers=4
NFS バージョン 4 を使用してサーバーと通信します。

rsize=size
読み取りサイズのバイトを設定します。

proto=udp
UDP を使用して NFS サーバーと通信します。

proto=tcp
TCP を使用して NFS サーバーと通信します。

hard
NFS ハード・マウントを使用します。

soft
NFS ソフト・マウントを使用します。

sec
指定されたセキュリティー区分を使用します。

デフォルト・オプションは、次のとおりです。

vers=3
rsize=32768
proto=tcp
hard
sec=sys

NFS の構成に関するチェックリスト

NFS ソフトウェアをシステムにインストールした 後、NFS の構成を開始できます。 次のステップに従って、NFS を構成してください。

以下のタイプのセキュリティーを 使用するように NFS を構成するには、まず C (CLiC) カーネル・ライブラリー内の CryptoLite をインストールする必要があります。

- krb5
- krb5i
- krb5p

各ステップについて詳しく説明します。

1. ネットワーク内でサーバーとなるシステムと、クライアントとなるシステムを 決定します (1 つのシステムをサーバーとクライアントの 両方として構成することができます)。
2. 使用する NFS のバージョンを決定します。
3. RPCSEC-GSS セキュリティーを使用するかどうかを決定します。 使用する場合は、594 ページの『RPCSEC-GSS 用のネットワークのセットアップ』の考慮事項を参照してください。
4. システムごとに (クライアントでもサーバーでも)、591 ページの『システム始動時の NFS デーモンの始動』 の手順に従って操作します。
5. NFS サーバーごとに、591 ページの『NFS サーバーの構成』 の手順に従って操作します。
6. NFS クライアントごとに、592 ページの『NFS クライアントの構成』 の手順に従って操作します。
7. ネットワーク上のパーソナル・コンピューターで NFS サーバーにアクセスしたい (ファイルシステムをマウントするだけでなく) 場合は、607 ページの『PC-NFS』 の手順に従って PC-NFS を構成します。
8. NFS バージョン 4 を使用することを計画している場合は、580 ページの『NFS バージョン 4 のサポート』の考慮事項を参照してください。

システム始動時の NFS デーモンの始動

デフォルトでは、NFS デーモンはインストール時には始動されません。

インストール時にはすべてのファイルがシステム 上に配置されますが、NFS をアクティブにする 手順は実行されません。 システムの始動時に NFS デーモンを始動するには、次の方法があります。

- SMIT 高速パス `smit mknfs`
- **mknfs** コマンド。

これらのすべての方法は、システムを始動するたびに `/etc/rc.nfs` スクリプトが実行されるように、`inittab` ファイルにエントリーを配置します。 そしてこのスクリプトが、特定のシステムに必要なすべての NFS デーモンを始動します。

NFS サーバーの構成

NFS サーバーを構成するには、以下の手順を使用します。

NFS サーバーを構成するには、次の操作を行います。

1. `/etc/exports` ファイルを作成します。 575 ページの『/etc/exports ファイル』を参照してください。
2. Kerberos を使用している場合は、NFS サーバーを Kerberos クライアントとしてセットアップします。 594 ページの『RPCSEC-GSS 用のネットワークのセットアップ』を参照してください。
3. NFS バージョン 4 を使用している場合は、**chnfsdom** コマンドを使用して NFS バージョン 4 ドメインを確立します。

   最初は、サーバーのインターネット・ドメインをファイルに指定できます。 ただし、サーバーのインターネット・ドメインとは異なる NFS バージョン 4 ドメインを定義することもできます。 詳しくは、**nfsrgyd** コマンドの説明を参照し、NFS レジストリー・デーモンに関する記述を調べてください。
4. Kerberos を使用した NFS バージョン 4 を使用する場合は、`/etc/nfs/realm.map` ファイルを作成しなければならいない場合があります。 576 ページの『/etc/nfs/realm.map ファイル』を参照してください。
5. サーバーで Kerberos 認証を使用する場合は、サーバーで拡張セキュリティーを使用可能にする必要があります。

   拡張セキュリティーは、SMIT を使用して、または **chnfs -S -B** コマンドで使用可能にできます。

NFS クライアントの構成

NFS クライアントを構成するには、以下の手順を使用します。

1. 579 ページの『NFS デーモンの始動』 の手順に従って NFS を始動します。
2. **mkdir** コマンドを使用して、ローカル・マウント・ポイントを確立します。

   NFS がマウントを正常に完了するためには、NFS マウントのマウント・ポイント (プレースホルダー) として機能するディレクトリーが存在する必要があります。 このディレクトリーは空でなければなりません。 このマウント・ポイントは他のディレクトリーとまったく同じように作成でき、特殊な属性は不要です。

   注: ファイルシステムをマウントするためには、すべての NFS マウントについてそのためのマウント・ポイントがシステム上に既に存在している必要がありますが、例外が 1 つあります。 **automount** デーモンを使用する場合は、マウント・ポイントを作成する必要はありません。
3. Kerberos を使用する場合は、次のステップに従ってください。

   a) NFS クライアントを Kerberos レルムに構成します。

   これは、**config.krb5** コマンドで行います。 構成の詳細については、「*IBM Network Authentication Service Administrator's and User's Guide*」を参照してください。

   b) Kerberos マウントを介してファイルにアクセスするクライアント上のすべてのユーザーの Kerberos プリンシパルを作成します。

   これは、**kadmin** コマンドで行います。 Kerberos プリンシパルの作成方法については、「*Network Authentication Service Administrator's and User's Guide*」を参照してください。

   c) クライアント・マシンの Kerberos プリンシパルの確立はオプションです。

   プリンシパルを持たないクライアントはスリム・クライアント と呼ばれ、プリンシパルを持つクライアントはフル・クライアント と呼ばれます。 スリム・クライアントは、状態管理で使用される NFS バージョン 4 のクライアントからサーバーへの 特定のコンテキスト管理操作を実行する場合に、弱い NFS RPC セキュリティーを使用します。 フル・クライアントは、構成にもよりますが、Kerberos ベースの強い RPC セキュリティーを使用できます。 スリム・クライアント構成は、管理オーバーヘッドが少なく、多くの環境にとって十分の構成です。 高水準のセキュリティーを必要とする展開では、フル・クライアント構成を実行できます。
4. NFS バージョン 4 を使用している場合は、**chnfsdom** コマンドを使用して NFS バージョン 4 ドメインを確立する必要もあります。

   最初は、クライアントのインターネット・ドメインをファイルに指定できます。 ただし、クライアントのインターネット・ドメインとは異なる NFS バージョン 4 ドメインを定義することもできます。 この説明については、NFS レジストリー・デーモン **nfsrgyd** の資料を参照してください。
5. クライアントで Kerberos 認証を使用する場合は、クライアントで拡張セキュリティーを使用可能にする必要があります。

   拡張セキュリティーを使用可能にするには、SMIT または **chnfs -S -B** コマンドを使用します。 **chnfs** の詳細については、**chnfs** コマンドのリファレンス・ページを参照してください。
6. 601 ページの『定義済み NFS マウントの確立』 の手順に従って定義済みマウントを確立し、マウントします。

ID マッピング

ID マッピングは、外部ユーザーおよびグループをローカル・ユーザーおよびグループに変換する手段を、ローカル NFS サーバーおよびクライアントに提供します。

AIX は、LDAP を ベースにした EIM テクノロジーを使用して、ID マッピングを実行します。 すべての NFS ID マッピングは LDAP サーバーに保管されます。

EIM クライアントをセットアップするには、`bos.eim.rte` および `ldap.client` ファイルセットがインストールされていなければなりません。 EIM サーバーには、`ldap.server` ファイルセットも必要です。 適切なファイルセットのインストール後に、`/usr/sbin/chnfsim` を使用して EIM が構成されます。 最小セットアップ・オプションは、次のとおりです。

```
/usr/sbin/chnfsim -c -a -t [type] -h [EIM server] -e [LDAP/EIM domain] -f [LDAP suffix] -w [administrator password]
```

これは、EIM クライアントとサーバーの 両方を構成して、ID マッピングに特定の EIM サーバーを使用するようにします。 コマンドに指定したホスト名がローカル・ホスト名の場合は、LDAP サーバーもセットアップされます。

構成ステップの完了後に、EIM 管理者は LDAP サーバーに NFS ID マッピング・データを取り込むことができます。 個別のユーザーまたはグループ (John Doe など) は、マッピング ID として認識されます。 そのユーザーの NFS 所有者ストリング (*johndoe@austin.ibm.com*) は、ID マッピングとして認識されます。 LDAP サーバーにこのデータを入力するには、次のコマンドを実行します。

```
/usr/sbin/chnfsim -a -u -i "John Doe" -n johndoe -d austin.ibm.com
```

マッピング ID はユーザーまたはグループの記述名で、ID マッピングは *name@domain* NFS 所有者ストリングです。 レルムからドメインへのマッピングも LDAP サーバーに保管されます。 Kerberos レルム `kerb.austin.ibm.com` が NFS ドメイン `austin.ibm.com` にマップするよう入力するには、次のコマンドを実行します。

```
/usr/sbin/chnfsim -a -r kerb.austin.ibm.com -d austin.ibm.com
```

EIM のマッピング・データを使用するように NFS を構成するには、NFS レジストリー・デーモンを再始動する必要があります。 NFS レジストリー・デーモンは開始時に EIM サーバーが使用可能かどうかをチェックし、使用可能であれば、すべてのマッピング機能は EIM を経由し、すべてのローカル・マッピングは使用されなくなります。

EIM については、セキュリティー の『エンタープライズ識別マッピング』を参照してください。

NFS ファイルシステムのエクスポート

以下の手順を使用して、NFS ファイルシステムをエクスポートできます。

- SMIT を使用して NFS ファイルシステムをエクスポートするには、次のようにします。
  1. コマンド `lssrc -g nfs` を入力して、NFS が既に稼働中であることを検証します。 出力は **nfsd** および **rpc.mountd** デーモンがアクティブであることを示していなければなりません。 アクティブでない場合は、579 ページの『NFS デーモンの始動』 の手順に従って NFS を始動してください。
  2. コマンド・ラインから次のようにタイプ入力し、Enter キーを押します。

     ```
     smit mknfsexp
     ```

  3. 「PATHNAME of directory to export (エクスポートするディレクトリーのパス名)」、「MODE to export directory (エクスポート・ディレクトリーのモード)」、および「EXPORT directory now, system restart or both (ディレクトリーをエクスポートする時期)」の各フィールドに、該当する値を指定します。
  4. 希望するオプションの特性をさらに指定するか、残りのフィールドをそのままにしてデフォルト 値を受け入れます。
  5. 変更が終了すると、SMIT により `/etc/exports` ファイルが更新されます。 `/etc/exports` ファイルが存在しない場合は、新しく作成されます。
  6. エクスポートしたいディレクトリーごとに、ステップ 3 から 5 までを繰り返します。
- テキスト・エディターを使用して NFS ファイルシステムをエクスポートするには、次の操作を行います。
  1. 任意のテキスト・エディターを使用して `/etc/exports` ファイルをオープンします。
  2. ディレクトリーの絶対パス名を使用して、エクスポートしたいディレクトリーごとに 1 つのエントリーを作成します。 エクスポートしたいディレクトリーを、それぞれ左端から表示します。 ディレクトリーの中にはエクスポート済みの他のディレクトリーを入れないでください。 `/etc/exports` ファイル内のエントリーの構文の詳細については、ファイル参照 の `/etc/exports` ファイルの説明を参照してください。
  3. `/etc/exports` ファイルを保存してクローズします。
  4. NFS が稼働中である場合は、以下のコマンドをタイプ入力して Enter キーを押します。

     ```
     /usr/sbin/exportfs -a
     ```

-a オプションは **exportfs** コマンドに対して、/etc/exports ファイル内のすべての情報をカーネルに送信するように指示します。 NFS が動作していない場合は、579 ページの『NFS デーモンの始動』 の手順に従って NFS を始動してください。

- NFS ファイルシステムを一時的に (/etc/exports ファイルを変更せずに) エクスポートするには、以下のコマンドをタイプ入力して Enter キーを押します。

```
exportfs -i /dirname
```

この場合、*dirname* はエクスポートしたいファイルシステムの 名前です。 **exportfs -i** コマンドを実行すると、指定したディレクトリーは /etc/exports ファイル内で検査されず、すべてのオプションがコマンド・ラインから直接取り出されます。

AIX NFS バージョン 4 サポートでは、NFS サーバーによってクライアントにレンダリングされる 代替ネーム・スペースを管理者が作成および制御することができます。 これは、**exname** エクスポート・オプションを使用して行います。 このサポートは、サーバーのローカル・ファイルシステム ・ネーム・スペースの詳細を NFS クライアントから隠すために使用することもできます。

RPCSEC-GSS 用のネットワークのセットアップ

このシナリオでセットアップするネットワークには、5 つのサーバーが組み込まれ、RPCSEC-GSS に構成されます。

このネットワーク上の 5 つのサーバーは、以下のとおりです。

- kdc.austin.ibm.com
- alpha.austin.ibm.com
- beta.austin.ibm.com
- gamma.austin.ibm.com
- zeta.austin.ibm.com

システム kdc.austin.ibm.com は鍵配布センター (KDC) サーバーとして構成され、Kerberos レルム AUSTIN.IBM.COM が作成されます。このレルムでは、kdc.austin.ibm.com と zeta.austin.ibm.com 以外のすべてのシステムは、RPCSEC-GSS でエクスポートされるファイルシステムを提供する NFS サーバーになります。

システム alpha.austin.ibm.com および beta.austin.ibm.com 間には追加リンクがあり、そのリンクの両端でこれらのシステムは互いに対して fast_alpha.test.austin.com および fast_beta.test.austin.ibm.com として表示されます。 このため、追加の構成ステップが必要になります。

さらに、このネットワークには、いずれかのシステムで 構成されている以下のユーザーが存在します。

- adam
- brian
- charlie
- dave
- eric

注 : 次に示すセットアップは、単なる例であり、すべての環境に適切であるとは限りません。 新規 Kerberos レルムをセットアップする 前に、「Administrator's and User's Guide for the Network Authentication Service」を参照してください。

注 : Kerberos を使用する場合は、ネットワーク全体でシステム時刻が妥当な範囲でほぼ同じである必要があります。 この手順を開始する前に、ネットワーク全体で時刻を自動的に同期化するメカニズム (AIX **timed** デーモンまたは NTP セットアップなど) をセットアップする必要があります。

1. KDC サーバーをセットアップします。

注 : KDC サーバーは、他の目的には使用しないのが理想的です。KDC で暗号漏えいが発生すると、すべての Kerberos プリンシパルで暗号漏えいが発生します。

このシナリオでは、kdc.austin.ibm.com が鍵配布センターとして構成されます。 次に示すものは、**des3** 用の構成です。 パフォーマンス上の理由で **des** を選択する場合は、下記の **kadmin** で、addprinc および ktadd 呼び出しに引数 -e des-cbc-crc:normal を追加してください。

aes 暗号化を使用してネットワークを構成する場合は、**kadmin** コマンドで addprinc および ktadd 呼び出しに -e aes256-cts:normal 引数を追加してください。

a) krb5.server.rte ファイルセットを kdc.austin.ibm.com にインストールします。

b) KDC サーバーをセットアップします。 このシナリオでは、次のコマンドを使用します。

```
config.krb5 -S -d austin.ibm.com -r AUSTIN.IBM.COM
```

このコマンドの実行後に、マスター・データベース・パスワードおよび管理プリンシパルのパスワードを入力するようにシステムに求められます。

c) /usr/krb5/sbin/kadmin.local コマンドを KDC サーバーで実行して、ユーザーおよびホストごとにプリンシパルを作成します。 この例では、関連ユーザーの UNIX ユーザー名と一致する Kerberos プリンシパルが作成されます。 プリンシパル名は NFS によってユーザー名にマップされ、プリンシパルに関連した UNIX 証明書が判別されます。 プリンシパルとユーザー名の間でより一般的なマッピングを使用する方法については、592 ページの『ID マッピング』を参照してください。 このネットワークでは、次のプリンシパルを作成します。

- adam
- brian
- charlie
- dave
- eric
- nfs/alpha.austin.ibm.com
- nfs/beta.austin.ibm.com
- nfs/gamma.austin.ibm.com

注 : 選択するユーザー・プリンシパル名は、システムが構成するユーザー・レジストリー (/etc/passwd、**LDAP**、**NIS** など) 内の対応するユーザー名と一致する必要があります。 NFS はプリンシパル名をユーザー名として使用し、ローカル・システム上のユーザーおよびグループの ID を取得します。 名前が一致しない場合は、そのアクセスは匿名アクセスとして扱われます。

KDC の構成が完了しました。

2. 次に、**config.krb5** コマンドを使用して、各 NFS クライアントおよびサーバーを Kerberos クライアントとして構成します。

これを行う方法は、KDC の構成方法によって異なります。 このシナリオでは、各 NFS システムで次のコマンドを実行します。

```
config.krb5 -C -d austin.ibm.com -r AUSTIN.IBM.COM -c kdc.austin.ibm.com -s
kdc.austin.ibm.com
```

これで、任意の構成済みシステムで任意のユーザー・プリンシパルとして **kinit** を実行できるようになりました。 例えば、ユーザー adam として **kinit** を実行するには、次のコマンドを実行します。

```
/usr/krb5/bin/kinit adam
```

adam の Kerberos パスワード (AIX パスワードではない) を指定する必要があります。

この例では、**kinit** を使用してユーザーを認証します。 システム・ログイン時に Kerberos 認証を使用するように AIX を構成することができます。 詳しくは、セキュリティーの Kerberos を使用した AIX への認証を参照してください。

3. これで、各 NFS サーバーを適切な keytab エントリーで構成できるようになります。

   このシナリオでは、一例として keytab エントリーを alpha.austin.ibm.com 用に構成しました。beta.austin.ibm.com および gamma.austin.ibm.com の場合も、プロセスはまったく同じです。

   a) alpha.austin.ibm.com から、**kadmin** コマンドを実行します。 その後、次のコマンドを実行します。

   ```
   ktadd nfs/alpha.austin.ibm.com
   ```

   これで keytab ファイルが作成されます。

   b) 次に、**nfshostkey** コマンドで作成したばかりの keytab ファイルを使用するよう、**gssd** デーモンをセットアップします。

   このシナリオでは、次のコマンドを実行します。

   ```
   nfshostkey -p nfs/alpha.austin.ibm.com -f /etc/krb5/krb5.keytab
   ```

   c) 次のコマンドを実行して、**gssd** デーモンを自動的に開始するようセットアップします。

   ```
   chnfs -S -B
   ```

   このセットアップの手順を、システムごとに繰り返します。

4. この時点で NFS サーバーは機能しますが、すべてのユーザーは nobody として表示されます。 すべてのユーザーがすべてのサーバー上に存在し、同じ UID および GID を持つことが推奨されています。存在しないユーザーは、エクスポートされたディレクトリーに nobody としてのみアクセスします。 ユーザー名を正しくマップするには、NFS レジストリー・デーモンを構成する必要があります。

   a) **chnfsdom** コマンドを使用して、ドメインをセットアップします。 このシナリオでは、すべての NFS サーバーに対して次のコマンドを実行し、austin.ibm.com をドメインとしてセットアップします。

   ```
   chnfsdom austin.ibm.com
   ```

   b) /etc/nfs/realm.map ファイルをセットアップします。このファイルには 1 行が含まれていて、レルム名の後ろにローカル・ドメインが続きます。

   この例のネットワークの場合、これら 2 つのファイルはすべての NFS サーバーで次のようになります。

   ```
   realm.map AUSTIN.IBM.COM         austin.ibm.com
   ```

   このファイルのレルム・エントリーでは、大/小文字の区別はありませんので、このエントリーは技術的には必要ではありません。

   c) zeta.austin.ibm.com (NFS サーバーではない) について、chnfs -S -B コマンドを使用して **gssd** デーモンを開始します。 Kerberos クライアント操作を試行する前に、ユーザーは **kinit** を使用して有効な証明書を取得する必要があります。

5. このシナリオでは、alpha.austin.ibm.com と beta.austin.ibm.com の間に、高速ネットワーク・リンク構成があります。 このリンクの両端では、beta.austin.ibm.com に対して alpha.austin.ibm.com が fast_alpha.test.austin.ibm.com と表示され、alpha.autsin.ibm.com に対して beta.austin.ibm.com が fast_beta.test.austin.ibm.com と表示されます。 nfs/fast_alpha.test.austin.ibm.com と nfs/fast_beta.test.austin.ibm.com はどちらも有効なプリンシパルではないため、このリンクをマウントに使用することはできません。

   この問題を訂正するには、**nfshostmap** コマンドを使用します。このコマンドは、この状態を処理するプリンシパルをマップします。

   a) alpha.austin.ibm.com で、次のコマンドを実行します。

   ```
   nfshostmap -a beta.austin.ibm.com fast_beta.test.austin.ibm.com
   ```

これにより、fast_beta.test.austin.ibm.com のプリンシパルが beta.austin.ibm.com 用であることが alpha.austin.ibm.com に通知されます。

b) beta で、次のコマンドを実行します。

```
nfshostmap -a alpha.austin.ibm.com fast_alpha.test.austin.ibm.com
```

サーバーは複数のホスト・プリンシパルを持つ場合があります。 fast_alpha の IP アドレスが 10.0.0.1、fast_beta の IP アドレスが 10.0.0.2 の場合、次の手順を実行して複数のホスト・プリンシパルを追加します。

a) nfs/fast_alpha.test.austin.ibm.com および nfs/fast_beta.test.austin.ibm.com プリンシパルを該当する Keytab ファイルに追加します。

b) 次のように、alpha サーバーで **nfshostkey** コマンドを実行します。

```
nfshostkey -a -p nfs/fast_alpha.test.austin.ibm.com -i 10.0.0.1
```

c) 次のように、beta サーバーで **nfshostkey** コマンドを実行します。

```
nfshostkey -a -p nfs/fast_beta.test.austin.ibm.com -i 10.0.0.2
```

NFS ファイルシステムのアンエクスポート

以下の手順を使用して、NFS ディレクトリーをアンエクスポートできます。

- SMIT を使用して NFS ディレクトリーをアンエクスポートするには、次の操作を行います。

  a) コマンド・プロンプトから次のようにタイプ入力して Enter キーを押します。

  ```
  smit rmnfsexp
  ```

  b)「PATHNAME of exported directory to be removed (除去されるエクスポート・ディレクトリーのパス名)」フィールドに該当するパス名を入力します。

  これでディレクトリーが /etc/exports ファイルから除去され、アンエクスポートされます。

  NFS バージョン 4 を使用してディレクトリーをクライアントにエクスポートしてある 場合は、アンエクスポートはサーバーでのファイル状態が原因で失敗する場合があります。 ファイル状態は、エクスポートされたディレクトリー内のファイルがクライアントによって開かれていることを意味します。 アプリケーションがそのデータの 使用を中止する処置をとるか、またはデータを強制的にアンエクスポートする (**exportfs -F**) ことができます。この場合は、そのデータを使用中のアプリケーションで障害が発生することがあります。

- テキスト・エディターを使用して NFS ディレクトリーをアンエクスポートするには、次の操作を行います。

  a) 任意のテキスト・エディターを使用して /etc/exports ファイルをオープンします。

  b) アンエクスポートしたいディレクトリーのエントリーを 探して、その行を削除します。

  c) /etc/exports ファイルを保存してクローズします。

  d) NFS が現在稼働中の場合は、次のように入力します。

  ```
  exportfs -u dirname
  ```

  この場合、*dirname* は /etc/exports ファイルから上記の手順で削除したディレクトリーの絶対パス名です。 NFS V4 クライアントがアクセスしているためにアンエクスポートが 失敗する場合は、-F オプションを追加してディレクトリーを強制的にアンエクスポートできます。

エクスポート済みファイルシステムの変更

以下の手順を使用する、エクスポート済み NFS ファイルシステムの 変更

- SMIT を使用してエクスポート済みの NFS ファイルシステムを 変更するには、次のようにします。

  1. ファイルシステムをアンエクスポートするには 、次のように入力します。

```
exportfs -u /dirname
```

この場合、*dirname* は変更したいファイルシステムの 名前です。

2. 次のように入力します。

```
smit chnfsexp
```

3.「PATHNAME of exported directory (エクスポート・ディレクトリーのパス名)」フィールドに該当するパス名を入力します。
4. 必要な変更を行います。
5. SMIT を終了します。
6. 次のように入力して、ファイルシステムを 再度エクスポートします。

```
exportfs /dirname
```

この場合、*dirname* は上記の手順で変更したファイルシステムの 名前です。

- テキスト・エディターを使用して、エクスポート済みの NFS ファイルシステムを 変更するには、次のようにします。

1. ファイルシステムをアンエクスポートするには 、次のように入力します。

```
exportfs -u /dirname
```

この場合、*dirname* は変更したいファイルシステムの 名前です。

2. 任意のテキスト・エディターを使用して /etc/exports ファイルをオープンします。
3. 必要な変更を行います。
4. /etc/exports ファイルを保存してクローズします。
5. 次のように入力して、ファイルシステムを 再度エクスポートします。

```
exportfs /dirname
```

この場合、*dirname* は上記の手順で変更したファイルシステムの 名前です。

エクスポート済みファイルシステムへの root ユーザー・アクセス

ファイルシステムがエクスポートされると 、デフォルトでは、そのエクスポート済みファイルシステムへの root ユーザーによる root アクセスは認可されなくなります。

あるホスト上の root ユーザーが NFS に対して特定のファイルへのアクセスを要求すると、このリクエスターのユーザー ID が NFS によってユーザー nobody のユーザー ID にマップされます (nobody は /etc/password ファイルにデフォルトで入っているユーザー名の 1 つです)。 ユーザー nobody のアクセス権は、特定のファイルについて一般 (*others*) に与えられるアクセス権と同じです。 例えば、*others* にファイルに対する実行許可だけが与えられている場合は、ユーザー nobody はファイルの実行だけしかできません。

root ユーザーがエクスポート済みのファイルシステムにアクセスできるようにするには、597 ページの『エクスポート済みファイルシステムの 変更』の手順に従って操作します。 SMIT メソッドを使用する場合は、「ホストでの root アクセスの許可」フィールドで、root アクセス権を与えたいホストの名前を指定します。 テキスト・エディターを使用してファイルを編集する場合は、ファイルシステム・エントリーに修飾子 -root=hostname を追加します。 例えば、次のとおりです。

```
/usr/tps -root=hermes
```

このエントリーは、ホスト hermes 上の root ユーザーが root 権限を使用して /usr/tps ディレクトリーにアクセスできることを示しています。

NFS ファイルシステムの明示的なマウント

NFS ディレクトリーを明示的にマウントするには、次の手順で行います。

1. NFS サーバーがディレクトリーをエクスポートしたことを 検証します。

   ```
   showmount -e ServerName
   ```

   この場合、*ServerName* は NFS サーバーの名前です。 このコマンドにより、NFS サーバーから現在エクスポートされているディレクトリーの名前が表示されます。 マウントしたいディレクトリーが表示されていない場合は、サーバーからそのディレクトリーをエクスポートします。

   注 : showmount コマンドは、NFS バージョン 4 ファイルシステムとしてのみエクスポートされたファイルシステムでは 機能しません。 NFS バージョン 4 の場合、クライアントはサーバー用のルート・ファイルシステムをマウントして、エクスポートされたディレクトリー構造をトラバースできます。 エクスポートされた個々のファイルシステムを 明示的にマウントしなくても、クライアントはファイルシステムにアクセスできます。

2. **mkdir** コマンドを使用して、ローカル・マウント・ポイントを確立します。

   NFS がマウントを正常に完了するためには、NFS マウントのマウント・ポイント (プレースホルダー) として機能する NULL (空の) ディレクトリーが存在していなければなりません。 このマウント・ポイントは他のディレクトリーとまったく同じように作成でき、特殊な属性は不要です。

3. 次のように入力します。

   ```
   mount ServerName:/remote/directory /local/directory
   ```

   この場合、*ServerName* は NFS サーバーの名前、`/remote/`*directory* はマウントしたい NFS サーバー上のディレクトリー、`/local/`*directory* は NFS クライアント上のマウント・ポイントです。

4. クライアント・マシン上で次の SMIT 高速パスを入力します。

   ```
   smit mknfsmnt
   ```

5. 次のフィールドのうち、ネットワーク構成で該当するフィールドを変更します。

   ネットワーク構成によっては、この画面上のすべてのエントリーに入力する必要はありません。

   注 : SMIT インターフェースを 使用している場合は、フィールドごとにタブ・キーを押して正しい値に変更しますが、ステップ 7 を完了するまで、Enter キーを押さないでください。

   - PATHNAME of mount point (マウント・ポイントのパス名)
   - PATHNAME of remote directory (リモート・ディレクトリーのパス名)
   - HOST where remote directory resides (リモート・ディレクトリーが存在するホスト)
   - MOUNT now, add entry to `/etc/filesystems` or both? (マウントする時期 (即時、`/etc/filesystems` にのみ)
   - `/etc/filesystems` entry will mount the directory on system RESTART (`/etc/filesystems` のエントリーはシステム 再起動時にディレクトリーをマウントする)
   - MODE for this NFS file system (この NFS ファイルシステムのモード)

6. 使用する NFS の構成に応じて、残りのエントリーのデフォルト値を変更するかそのまま使用します。
7. この画面での変更がすべて終了すると、SMIT により NFS ファイルシステムがマウントされます。
8. 「**Command: (コマンド)**」フィールドに「OK」状況が表示されたら、SMIT を終了します。

これで、NFS ファイルシステムが 使用できます。

自動マウント・サブシステム

自動マウント・サブシステムでは、root ユーザーが初期マウント・ポイントを指定すると、root 以外のユーザーがリモート・ファイルシステムをマウントできます。

この情報は、/etc/auto_master ファイルで指定します。 これらのマウント・ポイント (キーと呼ばれる) には対応するマップがあり、これによってマウントされるリモート・ファイルシステムが決定されます。 /etc/auto_master ファイルの形式は次のとおりです。

```
/key    map
```

注 : /etc/auto_master ファイルは、**automount** コマンドの最初の実行時に読み取られます。また、このファイルに対する変更は、**automount** コマンドを実行するまで有効になりません。

最も一般的なマップは、直接マップ、間接マップ、およびホスト・マップです。

直接マップ

直接マップでは、/etc/auto_master ファイルに特殊キー (/-) が必要です。

このマップは次の形式のファイルになります。

```
/directkey    [-options]    server:/dir
```

ユーザーが */directkey* ディレクトリーにアクセスすると、**automount** デーモンは *server*:/*dir* を */directkey* にマウントします。

間接マップ

間接マップは、マウント・ポイントでマウントされるリモート・ファイルシステムを判別する別形式のマップです。

間接マップの形式は次のとおりです。

```
indirectkey    [-options]    server:/dir
```

ユーザーが /key/*indirectkey* ディレクトリーにアクセスすると、**automount** デーモンは *server*:/*dir* を /key/*indirectkey* にマウントします。

ホスト・マップ

ホスト・マップでは、/etc/auto_master ファイルに特殊マップ (-hosts) が必要です。

automount デーモンは、/etc/hosts ファイルにリストされているサーバーごとに、/key ディレクトリーにサブディレクトリーを作成します。 ユーザーが /key/server ディレクトリーにアクセスすると、**automount** デーモンはサーバーのエクスポートされたディレクトリーを /key/server ディレクトリーにマウントします。

AutoFS を使用したファイルシステムの自動マウント

AutoFS は、**automount** コマンドを使用して、自動マウント構成情報を **AutoFS** カーネル・エクステンション機能に伝搬し、**automountd** デーモンを始動します。

この構成の伝搬を介して、拡張機能は、ファイルシステム内のファイルまたはディレクトリーがオープンされると、そのファイルシステムを自動的かつ透過的にマウントします。 拡張機能はマウントおよびアンマウント要求について **automountd** デーモンに通知するだけであり、**automountd** デーモンが要求されたサービスを実際に行います。

automountd デーモン内では名前とロケーションのバインディングは動的なので、**automountd** デーモンが使用する (NIS) マップの更新はユーザーから見て透過的に実行されます。 また、ファイルとディレクトリーへの参照がハードコーディングされているアプリケーションに、共用ファイルシステムを事前にマウントする必要はありませんし、特定のアプリケーションにマウントする必要があるホストのレコードを保守する必要もありません。

AutoFS は必要に応じてファイルシステムをマウントできます。 この方法でディレクトリーをマウントすると、すべてのファイルシステムを常にマウントしておく必要がなくなり、使用中のファイルシステムだけをマウントすればよくなります。

例えば、NFS ディレクトリーを自動的にマウントするには、次のようにします。

1. 次のように入力して、NFS サーバーがディレクトリーをエクスポートしたことを検証します。

   ```
   showmount -e ServerName
   ```

   この場合、*ServerName* は NFS サーバーの名前です。 このコマンドにより、NFS サーバーから現在エクスポートされているディレクトリーの名前が表示されます。
2. **AutoFS** マスター・ファイルおよびマップ・ファイルを作成します。 **AutoFS** は、これらのマップ・ファイル内で指定したディレクトリーをマウントまたはアンマウントします。
   例えば、**AutoFS** で、必要に応じて /local/dir1 および /local/dir2 ディレクトリーを、**serve1** サーバーからそれぞれ /remote/dir1 および /remote/dir2 ディレクトリーにマウントするとします。 auto_master ファイル・エントリーは、次のようになります。

   ```
   /remote        /tmp/mount.map
   ```

   /tmp/mount.map ファイル・エントリーは、次のようになります。

   ```
   dir1    -rw    serve1:/local/dir1
   dir2    -rw    serve1:/local/dir2
   ```
3. **AutoFS** カーネル・エクステンション機能がロードされ、**automountd** デーモンが実行中であることを確認します。 これは以下の 2 つの方法で実行することができます。

   a) **automount** コマンドの使用:「/usr/sbin/automount -v」を発行します。

   b) **SRC** の使用:「lssrc -s automountd」を実行します。 **automountd** サブシステムが 実行中でない場合には、「startsrc -s automountd」コマンドを実行します。

   注 : automountd デーモンを **startsrc** コマンドで開始すると、auto_master ファイルに対して行ったすべての変更が無視されます。
4. **automount** デーモンを停止するには、stopsrc -s automountd コマンドを実行します。

   何らかの理由で、**automountd** デーモンが **SRC** を使用しないで始動されている場合、以下のコマンドを実行します。

   ```
   kill automountd_PID
   ```

   ここで *automountd_PID* は **automountd** デーモンのプロセス ID です。 (**ps -e** コマンドを実行すると、**automountd** デーモンのプロセス ID が表示されます。) **kill** コマンドは SIGTERM シグナルを **automountd** デーモンに送信します。

定義済み NFS マウントの確立

以下のいずれかの手順を使用して、定義済みの NFS マウントを確立することができます。

注 : システムの始動時にマウントされる定義済みマウントを確立するには、/etc/filesystems ファイル内で **bg** (バックグラウンド) および **intr** (割り込み可能) オプションを定義します。 割り込み不可でフォアグラウンドで動作するマウントは、クライアント・システムの始動時にネットワークまたはサーバーが停止状態になると、クライアントを停止させることがあります。 クライアントがネットワークまたはサーバーにアクセスできない場合は、システムを保守モードで再始動して、該当するマウント要求を変更する必要があります。

- SMIT を使用して定義済みマウントを確立するには、次の操作を行います。

  a) 次のように入力します。

  ```
  smit mknfsmnt
  ```

  b) 事前に定義するマウントごとに、この画面上で値を指定します。 それぞれの必須フィールド (左マージンにアスタリスク (*) が付けられているフィールド) ごとに値を指定してください。 これら以外のフィールドについても、値を指定するか、それぞれのデフォルト値を受け入れます。

この方法によって、/etc/filesystems ファイル内に必要なマウントのエントリーが作成され、マウントが試行されます。

- /etc/filesystems ファイルを編集して NFS デフォルト・マウントを確立するには、次の操作を行います。

  a) テキスト・エディターを使用して /etc/filesystems ファイルを開きます。

  b) システムの始動時にマウントする各リモート・ファイルシステムのエントリーをそれぞれ追加します。 例えば、次のとおりです。

```
/home/jdoe:
dev = /home/jdoe
mount = false
vfs = nfs
nodename = mach2
options = ro,soft
type = nfs_mount
```

このスタンザは、システムに対して /home/jdoe リモート・ディレクトリーを同じ名前のリモート・マウント・ポイント経由でマウントするように指示します。 このファイルシステムは読み取り専用 (ro) としてマウントされます。 soft としてもマウントされるため、サーバーが応答しなければ、エラーが戻されます。 *type* パラメーターを nfs_mount として指定すると、**mount -t nfs_mount** コマンドを発行したとき、システムは /home/jdoe ファイルを (type = nfs_mount グループで指定された他のファイルシステムとともに) マウントしようと試みます。

次のサンプル・スタンザは、システムに対してシステムの始動時に /usr/games ファイルシステムをマウントすることを指示しています。 マウントに失敗すると、システムはバックグラウンド内でマウントを続けようとします。

```
/usr/games:
dev = /usr/games
mount = true
vfs = nfs
nodename = gameserver
options = ro,soft,bg
type = nfs_mount
```

NFS マウントに関連するスタンザには、次のパラメーターが必要です。

表 *92. NFS* マウントに関連するスタンザの必須パラメーター

項目	**説明**
dev= *filesystem_name*	マウントしようとするリモート・ファイルシステムのパス名を指定します。
mount=[true|false]	true オプションを指定すると、システムのブート時に NFS ファイルシステムがマウントされます。 false オプションを指定すると、システムのブート時に NFS ファイルシステムはマウントされません。
nodename= *hostname*	リモート・ファイルシステムが存在するホスト・マシンを指定します。
vfs=nfs	マウントしようとする仮想ファイルシステムが NFS ファイルシステムであることを指定します。

NFS マウントに関連するスタンザでは、次のパラメーターはオプションです。

項目	説明
type= *type_name*	マウントしようとするファイルシステムを、*type_name* マウント・グループの一部として定義します。 このパラメーターは、指定したファイルシステムのグループを 同時にマウントする **mount -t** コマンドとともに使用します。
options= *options*	次の *options* パラメーターの 1 つ以上を指定します。 **biods=*N*** 使用する **biod** デーモンの最大数を指定します。 デフォルトは、NFS バージョン 2 の場合は 7、NFS バージョン 3 および 4 の場合は 4 です。 **bg** 1 回目のマウント試行が失敗した場合に、バックグラウンド内で再度マウントを試行することを指定します。 **fg** 1 回目のマウント試行が失敗した場合に、フォアグラウンド内で再度マウントを試行することを指定します。 **noacl** NFS ジャーナル・ファイルシステムが提供するアクセス制御リスト (ACL) サポートを、このマウントに限り使用不可にします。 2 つのシステム間で使用される場合、NFS はアクセス制御リストをサポートします。 ファイルシステムをマウントするときに noacl オプションを使用すると、NFS は ACL を使用しません。 noacl オプションを選択すると、システム上の NFS クライアントが ACL をサポートしない NFS サーバーからマウントされる場合と同様になります。 ACL の詳細については、569 ページの『NFS アクセス制御リストのサポート』を参照してください。 **retry=*n*** マウント試行回数を設定します。 **rsize=*n*** 読み取りバッファーのサイズを、*n* で指定されたバイト数に設定します。 **wsize=*n*** 書き込みバッファーのサイズを、*n* で指定されたバイト数に設定します。 **timeo=*n*** NFS のタイムアウトを、*n* で指定された値 (10 分の 1 秒単位) に設定します。この変数を使用して、サーバーの負荷によって応答時間が不足する状況がネットワーク内で発生しないようにします。 **retrans=*n*** NFS の再送信回数を、*n* で指定された回数に設定します。 **port=*n*** サーバー・ポートを、*n* で指定された番号に設定します。 **soft** サーバーが応答しない場合にエラーを戻します。

項目	説明
	hard サーバーが応答するまで要求を試行し続けます。 **注 :** hard マウントを指定すると、応答を待つ間にプロセスが停止する場合があります。 プロセスに割り込んでキーボードから終了できるようにするには、マウント変数で intr 変数を使用します。 **intr** ハード・マウント時にキーボード割り込みを可能にします。 **ro** read-only 変数を設定します。 **rw** read-write 変数を設定します。 この変数とともに hard 変数を使用すると、soft マウントが read-write として試行された場合に、アプリケーションと矛盾するようなエラー条件を回避します。 ハード・マウントとソフト・マウントの問題については、613 ページの『NFS のトラブルシューティング』を参照してください。 **secure** NFS トランザクションにより安全なプロトコルを使用するように指定します。 **sec** **sec** オプションは、NFS マウント用のセキュリティー・フレーバー・リストを指定します。 使用可能なフレーバーは、**des**、**unix**、**sys**、**krb5**、**krb5i**、および **krb5p** です。 このオプションは、AIX 5.3 以降のみに適用されます。 **actimeo=*n*** 正規ファイルとディレクトリーのフラッシュ時間を *n* 秒だけ延長します。 **注 :** 属性キャッシュはクライアント上でファイル属性を保持します。 ファイルの属性には、削除されるまでの時間が割り当てられます。 フラッシュ時間が経過する前にファイルが変更されると、フラッシュ時間は前回の変更以降の経過時間分だけ延長されます (最近変更されたファイルはすぐに再変更される可能性があると想定しています)。 正規ファイルとディレクトリーにフラッシュ時間の最小延長と最大延長があります。 **vers** NFS バージョンを指定します。 デフォルトは、クライアントとサーバーの間で使用される NFS プロトコルのバージョンのうち、両方のシステムで使用可能な最も高いバージョンです。 NFS サーバーが NFS バージョン 3 をサポートしていない場合は、NFS マウントは NFS バージョン 2 を使用します。 **vers** オプションを使用して、NFS バージョンを選択します。 NFS マウントは、指定しない限りデフォルトで NFS バージョン 4 を使用しません。

項目	説明
	acregmin=*n* ファイルが変更されたあとで、少なくとも *n* 秒間、キャッシュ済み属性を保持します。 **acregmax=*n*** ファイルが変更されたあとで、最大 *n* 秒間、キャッシュ済み属性を保持します。 **acdirmin=*n*** ディレクトリーが更新されたあとで、少なくとも *n* 秒間、キャッシュ済み属性を保持します。 **acdirmax=*n*** ディレクトリーが更新されたあとで、最大 *n* 秒間、キャッシュ済み属性を保持します。 **cio** このファイルシステムを 並行リーダー/ライター用にマウントすることを指定します。 このファイルシステム 内のファイルに対する I/O は、ファイルが open() システム・コールで O_CIO と指定されてオープンしているかのように動作します。 このオプションを使用すると、CIO 以外の方式でのアクセスはできなくなります。 cio オプションを指定してマウントされたファイルシステムでキャッシュ I/O を使用することはできません。 すなわち、mmap() および shmat() などのマッピング・コマンドを、cio オプションで マウントされたファイルシステムのファイルに 使用すると EINVAL で失敗します。 この副次作用の 1 つとして、cio でマウントされたファイルシステムからはバイナリーを 実行できません。 ローダーが mmap() を使用することがあるからです。 **dio** ファイルシステムでの I/O が、すべてのファイルが open() システム・コールで O_DIRECT と指定されてオープンしているかのように動作することを指定します。 **注:** -odio または -ocio フラグを使用すると、 ある種のワークロードのパフォーマンスを高めることに役立ちますが、これらのフラグの使用によってそのファイルシステムにはファイル・キャッシング が使えなくなることをユーザーは知っている必要があります。 このようなファイルシステムでは 先読みが使用不可であるため、大規模な順次読み取りではパフォーマンスが低下する恐れがあります。

項目	説明

	maxpout=*n*
	このファイルシステムのファイルに関してスレッドがスリープに 入るべきページアウト・レベルを指定します。 **maxpout** を指定したときは、**minpout** も指定する必要があります。 この値は負数ではなく、かつ、**minpout** よりも大きくなければなりません。 デフォルトは、カーネルの **maxpout** レベルです。
	minpout=*n*
	このファイルシステムのファイルに関してスレッドが作動可能になるページアウト・レベルを指定します。 **minpout** を指定したときは、**maxpout** も指定する必要があります。 この値は負数であってはなりません。 デフォルトは、カーネルの **minpout** レベルです。
	rbr
	読み取り時 release-behind 機能を使用します。 このファイルシステム 内でファイルの順次読み取りが検出されると、そのファイルが 使用する実メモリー・ページは、ページが内部バッファーにコピーされた後、 リリースされます。
	
	注 : 次のオプションを設定しない場合、カーネルはそれぞれのデフォルト値に自動的に設定します。
	`fg`
	`retry=10000`
	`rsize=8192`
	`wsize=8192`
	`timeo=7`
	`retrans=5`
	`port=NFS_PORT`
	`hard`
	`secure=off`
	`acregmin=3`
	`acregmax=60`
	`acdirmin=30`
	`acdirmax=60`

c) システムの始動時に自動的にマウントしないディレクトリー・エントリーを除去します。

d) ファイルを保存してクローズします。

e) **mount -a** コマンドを実行して、`/etc/filesystems` ファイル内で指定されたディレクトリーをすべてマウントします。

明示的または自動的にマウントされたファイルシステムのアンマウント

明示的または自動的にマウントされた NFS ディレクトリーをアンマウントするには、以下の手順を使用します。

明示的または自動的にマウントされた NFS ディレクトリーをアンマウントするには、次のように入力します。

```
umount /directory/to/unmount
```

定義済み NFS マウントの除去

以下のいずれかの手順を使用して、定義済みの NFS マウントを除去することができます。

- SMIT を使用して定義済みの NFS マウントを除去するには、次の操作を行います。
  1. 次のように入力します。

```
smit rmnfsmnt
```

- /etc/filesystems ファイルを編集して定義済みの NFS マウントを除去するには、次の操作を行います。
  1. コマンド umount /directory/to/unmount を入力します。
  2. エディターを使用して /etc/filesystems ファイルをオープンします。
  3. アンマウントしたディレクトリーのエントリーを探して、それを削除します。
  4. ファイルを保存してクローズします。

PC-NFS

PC-NFS は、パーソナル・コンピューター用のプログラムで、ネットワーク・ファイルシステム (NFS) サーバーでエクスポートされるファイルシステムをパーソナル ・コンピューターでマウントできるようにします。

また、パーソナル・コンピューターから NFS サーバーにネットワーク・アドレスとホスト名を要求することができます。 さらに、NFS サーバーが **rpc.pcnfsd** デーモンを実行している場合、パーソナル・コンピューターから認証サービスと印刷スプール・サービスにアクセスできます。

次のシステム上で **rpc.pcnfsd** デーモンを構成することができます。

- ユーザー認証サービスを実行するシステム
- 印刷スプール機能のあるシステム
- すべてのネットワーク情報サービス (NIS) マスター・サーバーとスレーブ・サーバー

注 : 通常、NIS ネットワークは PC-NFS が NIS サーバーをデフォルト・サーバーとして選択できるように構成されているため、すべてのサーバーが **rpc.pcnfsd** デーモンを実行していることが重要です。 このデーモンをすべての NIS サーバー上で実行できない場合、または特定のサーバーへの要求を制限したい場合は、各パーソナル・コンピューター上で autoexec.bat ファイルに **net pcnfsd** コマンドを追加すると、特定の NIS サーバーを使用させることができます。

関連情報
Network Information Service (NIS)

PC-NFS 認証サービス

デフォルトでは、PC-NFS はそれ自体を NFS サーバーに対して nobody ユーザーとして表します。 nobody 特権を使用すると、すべてのパーソナル・コンピューターのユーザー・ファイルは nobody が所有するファイルとして表現されるので、異なるパーソナル・コンピューターのユーザーを区別できません。

rpc.pcnfsd デーモンの認証機能を使用すると、個々のユーザーを識別して異なる特権を与えることによって、システム・リソースとセキュリティーをモニターすることができます。

rpc.pcnfsd デーモンの実行中に、PC-NFS ユーザーはパーソナル・コンピューターから **net name** コマンドを入力して、このオペレーティング・システムにログインする 場合と同じ方法で PC-NFS にログインできます。 ユーザーの名前とパスワードは **rpc.pcnfsd** デーモンによって検査されます。 この認証プロシージャーではサーバーのセキュリティーは 向上しませんが、NFS を介して使用可能なファイルへのアクセスをより厳密に制御できるようになります。

PC-NFS の印刷スプール・サービス

rpc.pcnfsd デーモンの印刷スプール・サービスは、PC-NFS を実行するパーソナル・コンピューターが、直接接続されていないプリンターを使用して印刷できるようにします。

特に PC-NFS は、パーソナル・コンピューターのプリンター用のファイルを NFS サーバー上のファイルにリダイレクトします。 このファイルは NFS サーバー上のスプール・ディレクトリーに入っています。 ファイルがリダイレクトされると、**rpc.pcnfsd** デーモンはサーバーの印刷機能を呼び出します。 (スプール・ディレクトリーは PC-NFS クライアントがマウントできるように、エクスポート済みのファイルシステム内にある必要があります。) PC-NFS は **rpc.pcnfsd** デーモンによるファイル印刷を要求するときに、次の情報を提供します。

- 印刷するファイルの名前
- クライアント上のユーザーのログイン ID

- 使用するプリンターの名前

rpc.pcnfsd デーモンの構成

最高のパフォーマンスを得るには、以下の手順を使用して **rpc.pcnfsd** デーモンを構成します。

rpc.pcnfsd デーモンを構成するには、次の操作を行います。

1. 使用しているパーソナル・コンピューター上に PC-NFS プログラムをインストールします。
2. NFS サーバー上でスプール・ディレクトリーのロケーションを選択します。
   デフォルトのスプール・ディレクトリーは /var/tmp です。 スプール・ディレクトリーには最低 100K バイトのフリー・スペースが必要です。
3. スプール・ディレクトリーをエクスポートします。 エクスポートされたディレクトリーには、ネットワーク内でアクセス問題が生じるようなアクセス制限を加えないでください。
   この手順の詳細については、593 ページの『NFS ファイルシステムのエクスポート』を参照してください。
4. 608 ページの『rpc.pcnfsd デーモンの始動』 の手順に従って、**rpc.pcnfsd** デーモンを開始します。
5. 609 ページの『rpc.pcnfsd デーモンがアクセス可能かどうかの検査』 の手順に従って、**rpc.pcnfsd** デーモンがアクセス可能かどうかを検査します。

注 : プリンター・リダイレクトの要求が原因で PC-NFS スプール・ディレクトリー内に長さがゼロのファイル・リストが残ることがあるため、スプール・ディレクトリーを定期的に調べて、そのようなエントリーを消去してください。

rpc.pcnfsd デーモンの始動

デフォルトのスプール・ディレクトリーを使用して **rpc.pcnfsd** デーモンを始動するには、以下の手順を使用します。

1. テキスト・エディターを使用して、/etc/inetd.conf ファイル内の次のエントリーをアンコメントします。

   ```
   pcnfsd sunrpc_udp udp wait root /usr/sbin/rpc.pcnfsd pcnfsd 150001 1
   ```

2. ファイルを保存し、テキスト・エディターを終了します。

デフォルトではないディレクトリーを 使用して **rpc.pcnfsd** デーモンを始動するには、次の操作を行います。

1. テキスト・エディターを使用して、/etc/rc.nfs ファイルに次のエントリーを追加します。

   ```
   if [ -f /usr/sbin/rpc.pcnfsd ] ; then
   /usr/sbin/rpc.pcnfsd -s spooldir ; echo ' rpc.pcnfsd¥c'
   fi
   ```

   この場合、*spooldir* はスプール・ディレクトリーの絶対パス名を指定します。
2. ファイルを保存し、テキスト・エディターを終了します。
3. テキスト・エディターを使用して、/etc/inetd.conf ファイル内の次のエントリーをコメントにします。

   ```
   #pcnfsd sunrpc_udp udp wait root /usr/sbin/rpc.pcnfsd pcnfsd 150001 1
   ```

   この行の先頭にポンド記号 (#) を付けると、**inetd** デーモンによる (デフォルトのスプール・ディレクトリーを使用した) **rpc.pcnfsd** デーモンの始動が行われなくなります。
4. コマンド・ラインに次のように入力して、**rpc.pcnfsd** デーモンの印刷スプーラーを始動します。

   ```
   /usr/sbin/rpc.pcnfsd -s spooldir
   ```

   この場合、*spooldir* はスプール・ディレクトリーの絶対パス名を指定します。

inetd 構成データベースの更新の詳細については、408 ページの『inetd デーモンの構成』 を参照してください。

注 : rpc.pcnfsd デーモンが使用するデフォルト・ディレクトリーを、inetd.conf ファイルで変更することはできません。

rpc.pcnfsd デーモンがアクセス可能かどうかの検査

rpc.pcnfsd デーモンがアクセス可能かどうかを判別するには、以下の手順を使用します。

rpc.pcnfsd デーモンがアクセス可能かどうかを検査するには、次のように入力します。

```
rpcinfo -u host 150001
```

この場合、*host* は **rpc.pcnfsd** を構成しているシステムのホスト名を指定し、15001 は **rpc.pcnfsd** デーモンの RPC プログラム番号です。 このコマンドを入力すると、プログラムが作動可能で待ち状態にあることを示すメッセージが表示されます。

LDAP 自動マウント・マップ

LDAP サーバーから自動マウントのマップを取得するように自動マウント・サブシステムを 構成することができます。

LDAP での自動マウント・マップを管理するには、次の行を /etc/irs.conf ファイルに追加します。

```
automount nis_ldap
```

LDAP での自動マウント・マップを管理するためには、該当する LDIF ファイルを作成する必要があります。 **nistoldif** コマンドを使用して、ローカル自動マウント・マップ・ファイルを LDIF フォーマットに変換することができます。 例えば、LDAP サーバーの名前が ldapserver の場合、その基本サフィックスは dc=suffix で、/etc/auto_home map ファイルには次の行が入ります。

```
user1         server1:/home/user1
user2         server1:/home/user2
user3         server1:/home/user3
```

次のコマンドを使用して /etc/auto_home マップ・ファイル用の LDIF ファイルを作成し、それを LDAP サーバーに追加してください。

```
nistoldif -d dc=suffix -sa -f /etc/auto_home > /tmp/auto_home.ldif
ldapadd -D cn=admin -w passwd -h ldapserver -f /tmp/auto_home.ldif
```

LDAP サーバーの既存の自動マウント・エントリーを編集または除去するためには、LDIF ファイルを手動で作成する必要があります。 例えば user2 のホーム・ディレクトリーが server2 上にある場合、次の LDIF を作成する必要があります。

```
# cat /tmp/ch_user2.ldif
dn: automountKey=user2,automountMapName=auto_home,dc=suffix
changetype: modify
replace: automountInformation
automountInformation: server2:/home/user2
```

上記の LDIF を作成した後、次のコマンドを実行します。

```
ldapmodify -D cn=admin -w passwd -h ldapserver -f /tmp/ch_user2.ldif
```

ユーザーを除去する場合も、LDIF ファイルを作成する必要があります。 例えば user3 を除去するには、次の LDIF を作成します。

```
# cat /tmp/rm_user3.ldif
dn: automountKey=user3,automountMapName=auto_home,dc=suffix
changetype: delete
```

上記の LDIF を作成した後、次のコマンドを実行します。

```
ldapmodify -D cn=admin -w passwd -h ldapserver -f /tmp/rm_user3.ldif
```

WebNFS

オペレーティング・システムは WebNFS 用の NFS サーバー機能を提供します。

Oracle によって定義された WebNFS は、NFS プロトコルの簡易拡張機能であり、インターネット・ファイアウォールを通してサーバーおよびクライアントに容易にアクセスできるようにします。

WebNFS 拡張のある Web ブラウザーでは、NFS URL を使用して、サーバーから直接データにアクセスできます。 NFS URL の一例を次に示します。

```
nfs://www.YourCompany.com/
```

WebNFS は既存の web ベースのプロトコルと連携して、クライアントにデータを提供します。

WebNFS も NFS サーバーの拡張容易性を利用します。

ネットワーク・ロック・マネージャー

ネットワーク・ロック・マネージャーはネットワーク・ファイルシステム (NFS) と相互に動作する機能で、ネットワークを介して System V 形式の通知ファイルとレコード・ロックを提供します。

ネットワーク・ロック・マネージャー (**rpc.lockd**) とネットワーク状況モニター (**rpc.statd**) はネットワーク・サービス・デーモンです。 **rpc.statd** デーモンはユーザー・レベルのプロセスですが、**rpc.lockd** デーモンは (NFS サーバーのように) カーネル・スレッドのセットとしてインプリメントされます。 カーネルが基本的なネットワーク・サービスを提供するためには、どちらのデーモンも不可欠です。

注 :

1. 必須ロックまたは強制ロックは NFS ではサポートされません。
2. ネットワーク・ロック・マネージャーは、NFS バージョン 2 およびバージョン 3 に固有の機能です。

ネットワーク・ロック・マネージャーのアーキテクチャー

ネットワーク・ロック・マネージャーには、サーバー機能とクライアント機能の両方が組み込まれています。

クライアント機能はアプリケーションからの要求を処理し、要求をサーバー側のネットワーク・ロック・マネージャーに送信します。 サーバー機能はクライアントからのロック要求を受け入れ、該当するロック・コールをサーバー側で生成します。 サーバーはそのあとでクライアントのロック要求に応答します。

ステートレスの NFS と異なり、ネットワーク・ロック・マネージャーには暗黙の状態があります。 言い換えると、ネットワーク・ロック・マネージャーはクライアントが現在ロック中であるかどうかを記憶している必要があるということです。 ネットワーク状況モニター **rpc.statd** は、ネットワーク・ロック・マネージャーがネットワーク上の他のマシンの状況をモニターできるように、簡易プロトコルをインプリメントします。 正確な状況情報があれば、ネットワーク・ロック・マネージャーはステートレス NFS 環境内で整合性のある状態を維持することができます。

ネットワーク・ファイルのロック処理

アプリケーションがローカル・ファイルのロックを取得したい場合、`lockf`、`fcntl`、または `flock` サブルーチンを使用してカーネルにロック要求を送信します。

カーネルはそのロック要求を処理します。 ただし、NFS クライアント上のアプリケーションがリモート・ファイルに関するロック要求を行うと、ネットワーク・ロック・マネージャー・クライアントはその要求を処理するためにサーバーに対してリモート・プロシージャー・コール (RPC) を生成します。

クライアントは、最初のリモート・ロック要求を受信すると、サーバー内での守備範囲をクライアントの **rpc.statd** デーモンに登録します。 サーバー側のネットワーク・ロック・マネージャーの場合も同じです。 クライアントから最初の要求を受信すると、クライアント内での守備範囲をローカル・ネットワーク状況モニターに登録します。

クラッシュ回復プロセス

それぞれのマシン上の **rpc.statd** デーモンは、他のあらゆるマシン上の **rpc.statd** デーモンに対して、各自のアクティビティーを 通知します。 **rpc.statd** デーモンは、別のマシンがクラッシュまたはリカバリーしたという通知を受信すると、それを同じマシン上の **rpc.lockd** デーモンに通知します。

サーバーがクラッシュした場合、ロックされたファイルがあるクライアントはそのロックを 回復しなければなりません。 クライアントがクラッシュした 場合は、そのクライアントのサーバーは 回復中にクライアント・ロックを保持する必要があります。 また、NFS 全体の透過性を保存するために、アプリケーションそのものが介入しなくてもクラッシュ回復を行う必要があります。

クラッシュを回復させる手順は簡単です。 クライアントの障害が検出されると、サーバーはクライアント・アプリケーションが必要に応じて再びロックを要求するものと想定して、障害を起こしたクライアントのロックを解除します。 サーバーのクラッシュと回復が検出されると、クライアントのロック・マネージャーは、前にサーバーから認められたすべてのロック要求を再送信します。 サーバーはこの再送信された情報を使用して、猶予期間中にロック状態を再構成します (デフォルトでは、猶予期間は 45 秒間で、この期間内にサーバーはクライアントがロックを 再使用できるようにします)。

rpc.statd デーモンは、/var/statmon/sm および /var/statmon/sm.bak に保存されているホスト名を使用して、マシンに回復操作が必要になったときに通知すべきホストを把握します。

ネットワーク・ロック・マネージャーの始動

デフォルトでは、/etc/rc.nfs スクリプトは他の NFS デーモンとともに **rpc.lockd** および **rpc.statd** デーモンを始動します。

NFS が既に実行されている場合は、579 ページの『NFS デーモンの現行状況の表示』 の手順に従って、クライアント上で **rpc.lockd** および **rpc.statd** デーモンが動作しているかどうかを検査します。 この 2 つのデーモンの状況はアクティブ である必要があります。 **rpc.lockd** デーモンと **rpc.statd** デーモンがアクティブでない、つまり動作していない場合は、次の操作を行います。

1. テキスト・エディターを使用して、/etc/rc.nfs ファイルをオープンします。
2. 次の行を検索します。

```
if [ -x /usr/sbin/rpc.statd ]; then
       startsrc -s rpc.statd
fi
if [ -x /usr/sbin/rpc.lockd ]; then
       startsrc -s rpc.lockd
fi
```

3. これらの行の先頭にポンド記号 (#) が付いている場合は、ポンド記号を削除し、ファイルを保存して終了します。 次に、579 ページの『NFS デーモンの始動』 の手順に従って、**rpc.statd** デーモンと **rpc.lockd** デーモンを始動します。

   注 : 順序が重要です。 必ず最初に **statd** デーモンを始動してください。
4. NFS が稼働中で、/etc/rc.nfs ファイル内のエントリーが正しい場合は、579 ページの『NFS デーモンの停止』 および 579 ページの『NFS デーモンの始動』 の手順に従って、**rpc.statd** デーモンと **rpc.lockd** デーモンを停止し再始動します。

   注 : 順序が重要です。 必ず最初に **statd** デーモンを始動してください。

上記の操作を行っても **rpc.statd** デーモンと **rpc.lockd** デーモンが動作しない場合は、611 ページの『ネットワーク・ロック・マネージャーのトラブルシューティング 』 を参照してください。

ネットワーク・ロック・マネージャーのトラブルシューティング

ネットワーク・ロック・マネージャーの問題が発生した場合は、以下のヒントを使用して解決できる可能性があります。

クライアント上で次のようなメッセージが表示される場合があります。

```
clnttcp_create: RPC: Remote System error - Connection refused
rpc.statd:cannot talk to statd at {server}
```

そのとき、マシンは、回復手順が必要であることを通知する必要があるマシンが別に存在すると見なします。 あるマシンが再始動した場合、または **rpc.lockd** デーモンと **rpc.statd** デーモンが停止して再始動した場合は、マシン名が /var/statmon/sm から /var/statmon/sm.bak に移動され、さらに **rpc.statd** デーモンが /var/statmon/sm.bak 内の各エントリーに対応する各マシンに対し、リカバリー手順が必要であることを通知しようと試みます。

rpc.statd デーモンが当該マシンに通知できた場合は、/var/statmon/sm.bak 内のそのマシンのエントリーが削除されます。 **rpc.statd** デーモンがそのマシンに通知できなかった場合は、デーモンは一定のインターバルで試行を繰り返します。 マシンが応答に失敗するたびに、timeout により上記のメッセージが生成されます。 このデーモンはロックの完全性を確保するために試行を続けますが、ロックのパフォーマンスには逆効果になる場合があります。 ターゲット・マシンが応答しないだけなのか、半永久的に稼働停止になっているのかに応じて、処理を行う必要があります。 このメッセージが表示されないようにするには、次の操作を行います。

1. 579 ページの『NFS デーモンの現行状況の表示』 の手順に従って、サーバー上で **statd** デーモンと **lockd** デーモンが稼働中であることを検証します。 (この 2 つのデーモンの状況はアクティブ である必要があります。)
2. この 2 つのデーモンが動作していない場合は、579 ページの『NFS デーモンの始動』 の手順に従って、サーバー上で **rpc.statd** デーモンと **rpc.lockd** デーモンを始動します。

   注 : 順序が重要です。 必ず最初に **statd** デーモンを始動してください。

   デーモンを再始動したら、猶予期間があるので注意してください。 この期間に、**lockd** デーモンは、以前にサーバーとのロックを保持していた他のクライアントからのレクラメーション 要求を許可するため、デーモンを再始動した直後は新しいロックを取得できません。

以下に、上記のメッセージが表示されないようにする別の方法を示します。

1. 579 ページの『NFS デーモンの停止』 の手順に従って、クライアント上で **rpc.statd** デーモンと **rpc.lockd** デーモンを停止します。
2. クライアント上で次のように入力して、/var/statmon/sm.bak ファイルからターゲット・マシンのエントリーを削除します。

   ```
   rm /var/statmon/sm.bak/TargetMachineName
   ```

   このアクションを実行すると、ロック・リカバリーに関与しなければならない可能性があることをターゲット・マシンが認識しないようになります。 そのため、このアクションは、影響を受けたマシンとのネットワーク・ロックに関与している実行中のアプリケーションがターゲット・マシン上にないことが判別可能な場合にのみ使用するようにする必要があります。
3. 579 ページの『NFS デーモンの始動』の手順に従って、クライアント上で **rpc.statd** デーモンと **rpc.lockd** デーモンを始動します。

クライアントからのロックを取得できない場合は、次の操作を行います。

1. **ping** コマンドを使用して、クライアントとサーバーが相互に連絡して認識できることを検証します。 マシンが両方とも動作し、ネットワークがそのままであれば、各マシンについて /var/statmon/hosts ファイルにリストされたホスト名を調べます。 マシンが認識されるには、ホスト名はサーバーとクライアント間で正確に一致している必要があります。 ネーム・サーバーをホスト名解決に使用している場合は、ホスト情報が /var/statmon/hosts ファイル内の情報と正確に同じであることを確認してください。
2. 579 ページの『NFS デーモンの現行状況の表示』 の手順に従って、クライアントとサーバーの 両方で **rpc.lockd** デーモンと **rpc.statd** デーモンが 稼働中であることを検証します。 この 2 つのデーモンの状況はアクティブ である必要があります。
3. これらのデーモンが非アクティブの場合は、579 ページの『NFS デーモンの始動』の手順に従って、**rpc.statd** デーモンと **rpc.lockd** デーモンを始動します。
4. 状況がアクティブである場合は、クライアントとサーバーの 両方で、2 つのデーモンをリセットする必要がある場合があります。 そのためには、ロックを要求しているアプリケーションをすべて停止します。

5. 次に、579 ページの『NFS デーモンの停止』 の手順に従って、クライアントとサーバーの 両方で **rpc.statd** デーモンと **rpc.lockd** デーモンを停止します。
6. ここで 579 ページの『NFS デーモンの始動』 の手順に従って、**rpc.statd** デーモンと **rpc.lockd** デーモンを、最初にサーバー上で、次にクライアント 上で再始動します。

   注 : 順序が重要です。 必ず最初に **statd** デーモンを始動してください。

この手順でロックの問題を解決できない場合は、次の操作を行うことによって、デバッグ・モードで **lockd** デーモンを実行します。

1. 579 ページの『NFS デーモンの停止』の手順に従って、クライアントとサーバーの 両方で **rpc.statd** デーモンと **rpc.lockd** デーモンを停止します。
2. 579 ページの『NFS デーモンの始動』 の手順に従って、クライアント 上とサーバー上で **rpc.statd** デーモンを始動します。
3. 次のように入力して、クライアント 上とサーバー上で **rpc.lockd** デーモンを始動します。

   ```
   /usr/sbin/rpc.lockd -d1
   ```

   -d1 フラグを指定して呼び出すと、**lockd** デーモンは診断メッセージを syslog に出力します。 最初に猶予期間に関するメッセージがいくつか表示されますが、タイムアウトを待ってください。 猶予期間がサーバーとクライアントの 両方でタイムアウトになると、ロック問題が発生しているアプリケーションを実行し、ロック要求がクライアントとサーバー 間で送信されているかどうかを検査します。

NFS クライアントが NFS サーバーとの通信に使用する IP ポート番号の範囲を制限するには、`/var/statmon/environment` ファイルに NFS_PORT_RANGE 変数を設定します。

NFS ポートの範囲
NFS_PORT_RANGE 環境変数を使用して、クライアントがサーバーに 対して行うネットワーク呼び出しのソース・ポートを制限できます。

この環境変数を使用する場合は、`/etc/environment` ファイルに追加する必要があります。この環境変数の形式は次のとおりです。

```
NFS_PORT_RANGE=udp[4000-5000]:tcp[7000-8000]
```

この例では、クライアントによって送信される UDP パケットのソース・ポートの範囲は 4000 から 5000、TCP 接続のソース・ポートの範囲は 7000 から 8000 です。ポートの再使用の問題を回避するために、この範囲内で指定されたポート番号を、`/etc/services` ファイル内でいずれのネットワーク・ファイル・システム (NFS) デーモン用の固定ポート番号としても使用してはなりません。

NFS セキュリティー

NFS セキュリティーに関する情報については、複数の場所から参照できます。

DES セキュリティーについては、セキュリティーの『ネットワーク・ファイル・システムのセキュリティー』のトピックに詳しい説明があります。Kerberos セキュリティーの詳細については、594 ページの『RPCSEC-GSS 用のネットワークのセットアップ』を参照してください。

NFS のトラブルシューティング

他のネットワーク・サービスの場合と同様に、ネットワーク・ファイルシステム (NFS) を使用するマシン上でも問題が発生することがあります。 それらの問題に対処するトラブルシューティングでは、NFS の問題を突き止める方法を理解し、NFS 関連のエラー・メッセージを認識して適切な解決法を選択する必要があります。

NFS の問題を突き止める場合、サーバー、クライアント、ネットワークの 3 つの主な障害点を分離し、どれが機能していないかを判別します。

注 : ファイル・ロックの問題については、611 ページの『ネットワーク・ロック・マネージャーのトラブルシューティング』を参照してください。

ハード・マウントおよびソフト・マウントされたファイルの問題

ネットワークまたはサーバーに問題が発生した場合、ハード・マウントされたリモート・ファイルにアクセスするプログラムと、ソフト・マウントされたリモート・ファイルをアクセスするプログラムでは、障害の状態が異なります。

サーバーがハード・マウント要求に応答できない場合は、NFS により次のメッセージが出力されます。

```
NFS server hostname not responding, still trying
```

ハード・マウントされたリモート・ファイルシステムでは、クライアントはマウント要求が成功するまで再試行するため、プログラムはサーバーが応答するまで停止します。 ハード・マウントを実行するときには、サーバーが応答しない場合にクライアントがバックグラウンドで再試行するように、**mount** コマンドと一緒に **-bg** フラグを使用してください。

サーバーがソフト・マウント要求に応答できない場合は、NFS により次のメッセージが出力されます。

```
Connection timed out
```

ソフト・マウントされたリモート・ファイルシステムは、しばらく試行しても成功しない場合はエラーを戻します。 多くのプログラムではファイルシステム操作の戻り条件を検査しないため、ソフト・マウントされたファイルにアクセスしてもこのエラー・メッセージは表示されません。 ただし、コンソールにはこの NFS エラー・メッセージが出力されます。

NFS の問題の識別

NFS の問題が発生した場合は、以下の手順に従ってください。

クライアントに NFS の問題が発生した場合は、次の操作を行います。

1. ネットワーク接続が正しいことを検証します。
2. 579 ページの『NFS デーモンの現行状況の表示』の手順に従って、クライアント上で **inetd**、**portmap**、および **biod** デーモンが稼働中であることを検証します。
3. マウントしようとするファイルシステムに有効なマウント・ポイントが存在することを検証します。
   詳しくは、592 ページの『NFS クライアントの構成』を参照してください。
4. クライアントのシェル・プロンプトから次のコマンドを実行することによって、サーバーが始動し、稼働中であることを検証します。

   ```
   /usr/bin/rpcinfo -p server_name
   ```

   サーバーが始動している場合は、次のようなプログラム、バージョン、プロトコル、ポートの各番号のリストが出力されます。

   ```
   program    vers  proto      port
   100000        2      tcp       111    portmapper
   100000        2      udp       111    portmapper
   100005        1      udp      1025    mountd
   100001        1      udp      1030    rstatd
   100001        2      udp      1030    rstatd
   100001        3      udp      1030    rstatd
   100002        1      udp      1036    rusersd
   100002        2      udp      1036    rusersd
   100008        1      udp      1040    walld
   100012        1      udp      1043    sprayd
   100005        1      tcp       694    mountd
   100003        2      udp      2049    nfs
   100024        1      udp       713    status
   100024        1      tcp       715    status
   100021        1      tcp       716    nlockmgr
   100021        1      udp       718    nlockmgr
   100021        3      tcp       721    nlockmgr
   100021        3      udp       723    nlockmgr
   100020        1      udp       726    llockmgr
   100020        1      tcp       728    llockmgr
   100021        2      tcp       731    nlockmgr
   ```

   このような応答が戻されない場合は、サーバーのコンソールからサーバーにログインし、579 ページの『NFS デーモンの現行状況の表示』の指示に従って、**inetd** デーモンの状況を検査してください。

5. クライアントのシェル・プロンプトで次のコマンドを入力して、**mountd**、**portmap**、および **nfsd** デーモンが稼働中であることを検証します。

```
/usr/bin/rpcinfo  -u  server_name  mount
/usr/bin/rpcinfo  -u  server_name  portmap
/usr/bin/rpcinfo  -u  server_name  nfs
```

サーバー上でデーモンが動作している場合は、次の応答が戻されます。

```
program  100005  version  1  ready  and  waiting
program  100000  version  2  ready  and  waiting
program  100003  version  2  ready  and  waiting
```

上記の例に示されているように、プログラム番号はそれぞれコマンドに対応しています。 このような応答が戻されない場合は、サーバーのコンソールからサーバーにログインし、579 ページの『NFS デーモンの現行状況の表示』 の指示に従って、デーモンの状況を検査してください。

6. サーバー上の /etc/exports ファイルに、クライアントがマウントしようとしているファイルシステムの名前がリストされていること、また、そのファイルシステムがエクスポートされていることを 検証します。 そのためには次のコマンドを入力します。

```
showmount -e server_name
```

このコマンドでは、*server_name* によって現在エクスポートされているファイルシステムがすべて 表示されます。

7. NFS バージョン 4 の場合、NFSv4 ドメインが正しく設定されていることを確認する。
8. NFS バージョン 4 の場合、**nfsrgyd** デーモンが実行中であることを確認する。
9. 拡張セキュリティーを使用している場合は、620 ページの『RPCSEC-GSS の問題判別』を参照する。

非同期書き込みエラー
NFS によってマウントされたファイルシステムのファイルにアプリケーション ・プログラムがデータを書き込む場合、書き込み操作は **biod** デーモンによって非同期処理されるようにスケジュールされます。

データが実際にディスクに書き込まれるのと同時に NFS サーバー上でエラーが発生した場合は、NFS クライアントにエラーが戻され、**biod** デーモンは NFS データ構造に内部的にエラーを保存します。 この格納されたエラーは、あとでアプリケーション・プログラムが fsync 関数または close 関数を再度コールするときに、このアプリケーション・プログラムに戻されます。 したがって、このようなエラーの場合、プログラムがファイルをクローズするまでアプリケーションには 書き込みエラーが通知されません。 このイベントの典型的な例として、サーバー上のファイルシステムがいっぱいになっているために、クライアントによる書き込みの試行が失敗する場合があります。

nfs_server エラー・メッセージ
送信バッファーが小さすぎると、エラー・メッセージが戻されます。

ネットワーク上で送信バッファーが不足すると、次のエラー・メッセージが表示されることがあります。

```
nfs_server:  bad  sendreply
```

送信バッファーを増やすには、システム管理インターフェース ・ツール (SMIT) の高速パス smit commodev を使用してください。次にアダプターのタイプを 選択して、送信バッファーの数を増やします。

mount エラー・メッセージ
リモート・マウント・プロセスは、いくつかの原因で失敗することがあります。 ここでは、マウント障害に関連するエラー・メッセージについて説明します。

mount: ... already mounted
マウントしようとしているファイルシステムは既にマウントされています。

mount: ... not found in /etc/filesystems
指定されたファイルシステム 名またはディレクトリー名に一致するエントリーがありません。

ディレクトリー名またはファイルシステム 名のいずれかを指定して **mount** コマンドを入力した場合は、ファイルシステムまたはディレクトリーのフィールドが 引数に一致するエントリーが /etc/

filesystems ファイル内で検索されます。 例えば、**mount** コマンドで次のようなエントリーが見つかったとします。

```
/dancer.src:
          dev=/usr/src
          nodename  =  d61server
          type  =  nfs
          mount  =  false
```

この場合は、コマンド・ラインから次のコマンドを入力した場合と同様にマウントが実行されます。

```
/usr/sbin/mount -n dancer -o rw,hard /usr/src /dancer.src
```

... not in hosts database
ネットワーク情報サービス (NIS) を使用していないネットワークでは、このメッセージは **mount** コマンドで指定されたホストが /etc/hosts ファイルに含まれていないことを示します。 NIS を実行しているネットワークでは、このメッセージは NIS が /etc/hosts データベース内でこのホスト名を見つけられなかったか、マシン上の NIS **ypbind** デーモンが動作していないことを示します。 /etc/resolv.conf ファイルがあり、ホスト名を解決するのにネーム・サーバーを使用している場合は、**named** データベースに問題がある可能性があります。 619 ページの『NFS サーバー上のホスト名レゾリューション』を参照してください。

mount コマンドのスペルと構文を検査してください。 コマンドが正しく、ネットワークで NIS を実行していないときに、このホスト名についてこのメッセージだけが表示された場合は、 /etc/hosts ファイル内のエントリーを検査してください。

ネットワークで NIS を実行している場合は、コマンド・ラインに次のように入力して、**ypbind** デーモンが動作しているかどうかを確認します。

```
ps  -ef
```

ypbind デーモンがリストに表示されます。 **rlogin** コマンドを使用して別のマシンにリモート・ログインするか、**rcp** コマンドを使用して別のマシンに何かをリモート・コピーします。 どちらも失敗した場合は、**ypbind** デーモンが機能していないか、停止している可能性があります。

このホスト名についてのみこのメッセージが表示される場合は、NIS サーバー上の当該の /etc/hosts エントリーを確認してください。

mount: ... server not responding: port mapper failure - RPC timed out
マウント元のサーバーが停止状態になっているか、そのポートマッパーが機能していないか停止しています。 **inetd**、**portmap**、および **ypbind** デーモンを活動化するために、サーバーの再始動を試みてください。

サーバーが UP 状態であるにもかかわらず **rlogin** コマンドでそのサーバーにリモート・ログインできない場合は、別のマシンへのリモート・ログインを試行してネットワーク接続を調べてください。 また、サーバーのネットワーク接続も調べてください。

mount: ... server not responding: program not registered
これは **mount** コマンドがポートマッパーに達しているのに、NFS マウント・デーモン **rpc.mountd** が登録されなかったことを意味します。

mount: access denied ...
サーバーからマウントしようとしているファイルシステムのエクスポート・リストに、使用しているマシンが含まれていません。

コマンド・ラインから次のコマンドを実行すると、サーバーのエクスポート済みファイルシステムのリストを表示することができます。

```
showmount  -e  hostname
```

必要なファイルシステムがリストに 含まれていない場合や、マシン名またはネットグループ名がファイルシステムのユーザー・リストに含まれていない場合は、サーバーにログインして、/etc/exports 内に正しいファイルシステム・エントリーがあるかどうかを検査してください。 ファイルシステム 名

が /etc/exports ファイルには記述されていても **showmount** コマンドからの出力には含まれていない場合は、**mountd** デーモンに障害があります。 デーモンがファイル内のその行を構文解析できないか、ディレクトリーが見つからないか、ディレクトリー名がローカルにマウントされたディレクトリーではありません。 /etc/exports ファイルを調べても問題がなく、かつネットワークが NIS を実行している場合は、サーバー上の **ypbind** デーモンを調べてください。 このデーモンが機能していないか、停止している可能性があります。

mount: ...: Permission denied
このメッセージは、サーバー上で認証の一部が失敗したことを示す総称メッセージです。 上記の例のように、ユーザーがエクスポート・リストに含まれていないために、サーバーがユーザーのマシン上の **ypbind** デーモンを認識できなかったか、あるいはユーザーが提示した ID がサーバーに受け入れられなかった 可能性があります。

サーバー上の /etc/exports ファイルと、可能であれば **ypbind** デーモンを調べてください。 このケースでは、単に **hostname** コマンドでホスト名を変更して **mount** コマンドを再試行するだけで構いません。

mount: ...: Not a directory
リモート・パスまたはローカル・パスのどちらかがディレクトリーではありません。 コマンドのスペルを検査してから、両方のディレクトリーで実行してください。

mount: ...: You are not allowed
mount コマンドはマシン上のすべてのユーザーのファイルシステムに影響を与えるため、このコマンドを実行するためには、root 権限があるユーザーか、システム・グループのメンバーである 必要があります。 NFS のマウントとアンマウントを実行できるのは、root ユーザーとシステム・グループのメンバーだけです。

関連情報
Network Information Service (NIS)

NFS の低速アクセスの原因
リモート・ファイルへのアクセスが異常に遅く思われる場合は、デーモンの消失、**tty** 回線の不良、またはそれに類似する原因でアクセスが阻止されていないかどうかを確認してください。

ネットワーク接続
ネットワーク接続に関する情報を収集するには、**nfsstat** コマンドを使用します。

nfsstat コマンドにより、パケットが脱落しているかどうかを判別できます。 **nfsstat -c** コマンドと **nfsstat -s** コマンドを使用して、クライアントまたはサーバーが大型ブロックを再伝送しているかどうかを判別します。 パケットが脱落しているか、サーバーが使用中の場合は、常に再伝送の可能性があります。 5% の再送率は高いと見なされます。

通信アダプターの送信キュー・パラメーターを変更すると、再送信の可能性を低くすることができます。これらのパラメーターを変更するには、SMIT メニューを使用できます。 詳しくは、オペレーティング・システムおよびデバイスの 管理の使用可能なシステム 管理インターフェース を参照してください。

NFS サーバーには次の値を使用してください。

注 :

1. 引き続き再伝送がある場合は、上記の値を NFS クライアントに適用してください。
2. ネットワーク上のすべてのノードで同じ MTU サイズを使用する必要があります。

表 *93.* 通信アダプター最大伝送単位 *(MTU)* と送信キューのサイズ

アダプター	MTU	送信キュー
トークンリング		

表 93. 通信アダプター最大伝送単位 (MTU) と送信キューのサイズ (続き)

アダプター	MTU	送信キュー
4 Mb	1500 3900	50 40 (**nfsstat** コマンドがタイムアウトになると増加)
16 Mb	1500 8500	40 (**nfsstat** コマンドがタイムアウトになると増加) 40 (**nfsstat** コマンドがタイムアウトになると増加)
イーサネット	1500	40 (**nfsstat** コマンドがタイムアウトになると増加)

各トークンリング速度の最大伝送単位 (MTU) サイズを大きくすると、プロセッサーの使用量が減少し、読み取り/書き込み操作が著しく改善されます。

MTU サイズの設定

MTU サイズを設定するには、SMIT 高速パス `smit chif` を使用してください。

該当するアダプターを選択して、「Maximum IP Packet Size (最大 IP パケット・サイズ)」フィールドに MTU 値を入力します。

ifconfig コマンドを使用して MTU サイズを設定できます (また、MTU サイズを 8500 に設定するには、このコマンドを使用する必要 があります。) **ifconfig** コマンドのフォーマットは次のとおりです。

```
ifconfig trn NodeName up mtu MTUSize
```

この場合、`trn` はアダプター名、例えば `tr0` です。

また、**ifconfig** コマンドと SMIT を併用して MTU サイズを設定する方法もあります。

1. 上記の例のように、トークンリングの **ifconfig** コマンドを `/etc/rc.bsdnet` ファイルに追加します。
2. `smit setbootup_option` 高速パスを入力します。 「**Use BSD Style (BSD スタイルの使用)**」フィールドを **yes** に切り替えます。

送信キューのサイズ

通信アダプターの送信キューのサイズは SMIT で設定します。

`smit chgtok` 高速パスを入力し、該当するアダプターを選択して、「Transmit (送信)」フィールドにキューのサイズを入力します。

プログラムの停止

ファイル関連の作業中にプログラムが停止する場合、NFS サーバーが停止している可能性があります。

この場合、次のエラー・メッセージが表示されることがあります。

```
NFS server hostname not responding, still trying
```

NFS サーバー (*hostname*) がダウンしています。 これは NFS サーバー、ネットワーク接続、または NIS サーバーに問題があることを示します。

マシンが完全に停止している場合は、ファイルシステムのマウント元のサーバーを検査します。 1 つ以上のサーバーがダウンしていても問題はありません。 サーバーが再び始動すると、プログラムは自動的に続行されます。 ファイルは破棄されません。

ソフト・マウントされたサーバーが機能していなくても、他の作業には影響しません。 ソフト・マウントされたリモート・ファイルへのアクセスの試行中にタイムアウトになったプログラムは失敗して errno メッセージが発行されますが、他のファイルシステムへのアクセスは引き続き可能です。

すべてのサーバーが動作している場合は、同じサーバーを使用している他のマシンに問題が発生しているかどうかを調べます。 複数のマシンでサービス関連の問題が起こっていた場合は、サーバー上の **nfsd** デーモンに問題があるということです。 この場合は、サーバーにログインして **ps** コマンドを実行し、**nfsd** デーモンが動作していて CPU 時間を累積しているかどうかを調べます。 問題がある場合は、**nfsd** デーモンを停止してから再始動できます。 この方法で問題が解決されなかった場合は、サーバーを再始動する必要があります。

他のシステムが始動し、動作しているように見える場合は、ネットワーク接続とサーバーの接続を検査してください。

許可方式と認証方式

マウントが正常に確立されたあとで、リモート・ファイルまたはディレクトリーの読み取り、書き込み、または作成に問題が発生することがあります。 通常、このような障害は、許可または認証に問題があるために起こります。

許可と認証の問題の原因は、NIS を使用しているかどうか、セキュア・マウントを指定しているかどうかによって異なります。

最も単純な問題は、非セキュア・マウントを指定していて、NIS を使用していない場合に発生します。 このケースでは、ユーザー ID (UID) とグループ ID (GID) はもっぱらサーバーの /etc/passwd ファイルとクライアントの /etc/group ファイルからマップされているにすぎません。 この方式では、B という名前のユーザーがクライアントとサーバーの 両方で B として識別されるためには、ユーザー B は /etc/passwd ファイル内でも同じ UID 番号を与えられている必要があります。 次の例は、このような場合の問題の原因を示しています。

```
User  B  is  uid  200  on  client  foo.
User  B  is  uid  250  on  server  bar.
User  G  is  uid  200  on  server  bar.
```

/home/bar ディレクトリーはサーバー bar からクライアント foo にマウントされています。 ユーザー B がクライアント foo 上で /home/bar リモート・ファイルシステムのファイルを 編集している場合は、このユーザーがファイルを保管するときに混乱が生じます。

サーバー bar は、これらのファイルは user G に属するものと見なします。なぜなら、bar では UID 200 は G に対応しているからです。 ここで、B が **rlogin** コマンドを使用して bar に直接ログインすると、B はリモートでマウントされたファイルシステム 上で作業していたときに作成したファイルにアクセスできない可能性があります。 しかし、G にはこれが行えます。なぜなら、マシンによる権限の裁定は、名前ではなく、UID によって行われるからです。

このような問題の唯一の恒久的な解決策は、2 つのマシン上で整合性のある UID を再度割り当てることです。 例えば、B にサーバー bar 上で UID 200 を与えるか、またはクライアント foo 上で UID 250 を与えます。 このようにした場合は、B が所有するファイルに対して **chown** コマンドを実行して、それらが該当するマシン上の新しい ID と一致するようにする必要があります。

ネットワーク内のすべてのマシンで UID および GID のマッピングにおける整合性を維持する上で何かと問題があるため、このタイプの問題を回避するための適切なマッピングの実行を目的として、しばしば NIS が使用されます。

NFS サーバー上のホスト名レゾリューション

NFS サーバーはマウント要求を処理するときに、要求を出したクライアントの名前を探します。 サーバーはクライアントの IP アドレスを取り出し、そのアドレスに一致する対応ホスト名を探します。

ホスト名が見つかると、サーバーはエクスポート・リスト内で要求されたディレクトリーを探し、そのディレクトリーのアクセス・リストにクライアントの名前があるかどうかを調べます。 クライアントのエントリーがあり、そのエントリーがネーム・レゾリューションのために戻されたエントリーと正確に一致する場合は、マウント認証のその部分が認められたことになります。

サーバーは、IP アドレスからホスト名へのネーム・レゾリューションを実行できない場合にはマウント要求を拒否します。 サーバーは、マウント要求を出したクライアントの IP アドレスと一致するものを見つけなければなりません。 すべてのクライアントにアクセス権を与えてディレクトリーがエクスポートされた場合でも、サーバーは名前の逆照合を実行してマウント要求を許可できなければなりません。

また、サーバーは、クライアントの正しい名前を検索できなければなりません。 例えば、/etc/exports ファイル内に次のようなエントリーがあるとします。

```
/tmp      -access=silly:funny
```

この場合、/etc/hosts ファイルには次のような対応するエントリーがあります。

```
150.102.23.21        silly.domain.name.com
150.102.23.52        funny.domain.name.com
```

名前が正確に対応していないので注意してください。 サーバーがホスト silly および funny について IP アドレスからホスト名への一致を照合する場合、文字列名はエクスポートのアクセス・リスト内のエントリーと正確には一致しません。 このタイプのネーム・レゾリューションに伴う問題は通常、ネーム・レゾリューションに **named** デーモンを使用する場合に発生します。 大部分の **named** デーモン・データベースには、ユーザーがホストを参照するときに完全名を入力しなくてもすむように、完全ドメイン名に別名があります。 別名の「ホスト名から IP アドレスへの対応エントリー」があっても、逆照合がない場合があります。 一般に、名前の逆照合 (IP アドレスからホスト名へ) 用のデータベースには、そのホストの IP アドレスと (別名ではなく) 完全ドメイン名が入ったエントリーがあります。 さらに短い別名を使用してエクスポート・エントリーを作成すると、クライアントがマウントしようとするときに問題が発生する場合があります。

NFS 構造内のグループ数に関する制限

NFS バージョン 2 または 3 を使用するシステムでは、複雑な問題が生じないようにするために、ユーザーは 16 個を超えるグループのメンバーになることはできません。

グループは、**groups** コマンドにより定義されます。 ユーザーが 17 以上のグループのメンバーで、17 番目 (またはそれ以降) のグループが所有するファイルにアクセスしようとすると、システムはファイルの読み取りまたはコピーを許可しません。 ユーザーがそれらのファイルにアクセスできるようにするには、グループの順序を変更します。

上記の情報は、デフォルト動作について説明しています。 詳しくは、**mount** コマンドの **maxgroups** パラメーターを参照してください。

旧バージョンの *NFS* を持つ *NFS* サーバー

NFS バージョン 3 クライアントは、NFS バージョン 4 サーバーにマウントできません。

バージョン 3 より前の NFS サーバーからバージョン 3 の NFS クライアントへファイルシステムをマウントする場合、マウントを実行するクライアント上のユーザーが 8 を超えるグループのメンバーになっていると、問題が発生します。 一部のサーバーはこのような状況に正しく対処できず、マウント要求を拒否します。 この問題の解決策は、ユーザーのグループ・メンバーシップを 8 より小さい数に変更してからマウントを再試行することです。 次のエラー・メッセージは、このグループ問題の特徴を示しています。

```
RPC:    Authentication  error;  why=Invalid  client  credential
```

RPCSEC-GSS の問題判別

RPCSEC-GSS で問題がある場合は、以下の解決策を検討してください。

- クライアントで **klist** コマンドを使用して、有効な最新の証明書があることを確認します。
- クライアント、サーバー、および KDC のクロックが同期していることを確認します。 Kerberos レルム全体で整合性のある時間を確保するために、NTP または同等のセットアップを使用することをお勧めします。
- サーバーに有効な keytab ファイルおよびホスト・プリンシパルがあることを確認します。 次のコマンドが失敗する場合は、サーバーは機能しません。

  ```
  kinit -kt 'tail -n 1 /etc/nfs/hostkey' 'head -n 1 /etc/nfs/hostkey'
  ```

- 次のコマンドで、**gssd** デーモンが実行中であり、クライアントおよびサーバー上で反応することを確認します。

```
rpcinfo -u localhost 400234
```

  gssd デーモンが応答しない場合は、RPCSEC-GSS は失敗します。この問題を訂正するには、**gssd** デーモンを停止してから再始動します。
- 保全性またはプライバシーで書き込みエラーが発生する場合は、カーネル・モジュールを使用していることを確認してください。 保全性およびプライバシーは、カーネル・モジュールを使用しない限りサポートされません。 (カーネル・モジュールは Kerberos カーネル・モジュール (/usr/lib/drivers/nfs.ext) です。これは、拡張パックから modcrypt.base ファイルセットとともにインストールされます。)
- 特定のユーザーが、アクセスできるはずのデータへのアクセスを拒否される場合は、KDC の関連したプリンシパルがユーザーの AIX アカウント名と正しく同期していることを確認してください。
- システム・ログを活動化します。 ほとんどの RPCSEC-GSS エラーがログに記録されます。 エラーには 2 つの部分があります。最初の部分は GSS エラー・コード (詳細については RFC 2744 を参照) で、2 番目の部分は Kerberos エラー・コードです。

  注: システム・ログを活動化すると、システム・パフォーマンスに影響が及ぶ場合があります。したがって、問題判別の完了後にログを非活動化する必要があります。

  共通エラー・コードとその解決策のいくつかを次に示します。

  KRB5_CC_NOTFOUND
  有効な Kerberos 証明書を検出できませんでした。 **kinit** コマンドで、この問題を修正できる場合があります。

  KRB5_KDC_UNREACH
  KDC が到達不能です。 KDC が起動していることと、クライアントまたはサーバーと KDC の間にネットワーク上の問題がないことを確認してください。

  KRB5_KT_NOTFOUND
  サーバー・プリンシパル用の keytab エントリーを検出できませんでした。 **nfshostkey -l** コマンドを使用して、正しいプリンシパル (nfs/<完全修飾ドメイン名> でなければならない) および keytab ファイルを使用していることを確認してください。 **klist -ke** を使用して、適切なエントリー用のサーバー keytab ファイルを確認してください。

  KRB5KRB_AP_ERR_TKT_NYV
  クロックの問題を示している場合がほとんどです。

  KRB5KRB_AP_WRONG_PRINC および KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN
  これら両方のエラーは、クライアントがクライアント用に使用しているプリンシパルが、サーバーのホスト・プリンシパルと一致しないことを示しています。

  KRB5KRB_AP_WRONG_PRINC
  クライアントがサーバー・ホスト名を nfs/<完全修飾ドメイン名> という形式の既存のプリンシパルに正常に解決できたが、サーバーのホスト・プリンシパルがこのプリンシパルと一致しないことを示しています。

  KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN
  クライアントがサーバーのホスト名を既存のプリンシパルに解決できなかったことを示しています。 **nfshostkey -l** コマンドを使用して、サーバーに正しいプリンシパルがあることを確認します。 サーバーに正しいプリンシパルがある場合は、おそらくクライアントのホスト・マッピング・テーブルを更新する必要があります。詳しくは、**nfshostmap** コマンドを参照してください。

EIM の問題判別

EIM をトラブルシューティングする場合は、以下のヒントを参照してください。

EIM で問題がある場合は、以下のことを検討してください。

- **nfsrgyd** または **chnfsim** コマンドで EIM LDAP サーバーに接続できない場合は、次のコマンドを入力して ibmslapd プロセスが EIM LDAP サーバーで実行されていることを確認してください。

```
ps -ef | grep ibmslapd
```

ibmslapd プロセスが実行されていない場合は、次のコマンドを入力してこのプロセスを活動化します。

```
/usr/sbin/ibmslapd
```

- **nfsrgyd** または **chnfsim** コマンドで EIM LDAP サーバーに接続できるが、ID マッピング操作を実行できない場合は、ibmslapd プロセスが構成専用モードで実行されていないことを確認してください。

  この状況は、ibmslapd サーバーの始動時に ldapdb2 データベースが実行されていない場合に発生することがあります。 次のステップに従ってください。

  a) EIM LDAP サーバーに root ユーザーとしてログインします。

  b) /var/ldap/ibmslapd.log ファイルを表示します。 ibmslapd プロセスの最後の開始時刻を確認します。 また、サーバーが ldapdb2 データベースに接続できなかったので、サーバーが構成専用モードで始動されたかどうかを確認します。

  サーバーが ldapdb2 データベースに接続できなかった場合は、データベースを開始する必要があります。 次のステップに従って、ldapdb2 データベースを開始します。

  a) EIM LDAP サーバーに root ユーザーとしてログインします。

  b) 次のコマンドを入力して、ibmslapd プロセスがアクティブになっているかどうかを確認します。

  ```
  ps -ef | grep ibmslapd
  ```

  アクティブになっている場合は、次のコマンドを実行して使用不可にします。

  ```
  kill ibmslapd pid
  ```

  ここで、*pid* は、**ps -ef** コマンドで戻されるプロセス ID です。

  c) ibmslapd プロセスを使用不可にした後で、ldapdb2 データベースを開始します。

  a. EIM LDAP サーバーに ldapdb2 ユーザーとしてログインします。

  b. db2start と入力します。

  d) ldapdb2 データベースの開始後に、ibmslapd プロセスを活動化します。

  a. EIM LDAP サーバーに root ユーザーとしてログインします。

  b. ibmslapd と入力します。

NFS カーネル・エクステンションがロードされていない場合に発生する問題

一部の NFS コマンドは、NFS カーネル・エクステンション機能をロードしないと正常に実行されません。 このような従属性のあるコマンドには、**nfsstat**、**exportfs**、**mountd**、**nfsd**、**biod** などがあります。

NFS をシステムにインストールすると、カーネル・エクステンション機能は /usr/lib/drivers/nfs.ext ファイルに入れられます。 インストールされたファイルは、システムの構成時に NFS カーネル・エクステンション機能としてロードされます。 このカーネル・エクステンション機能が実行するスクリプトは、 /etc/rc.net ファイルをロードします。 このスクリプトでは、他にも処理が行われますが、その 1 つは NFS カーネル・エクステンション機能をロードすることです。 NFS カーネル・エクステンション機能をロードする前に、**伝送制御プロトコル/インターネット・プロトコル (TCP/IP)** カーネル・エクステンションと **nfs_kdes_null.ext** ファイルをロードしなければならないことに注意してください。

注 : システムを最初に始動するときに NFS カーネル・エクステンション機能をカーネルにロードするには、**gfsinstall** コマンドを使用します。 このコマンドはシステムを始動するたびに何度でも実行でき、問題を起こすこともありません。 現在、システムは /etc/rc.net ファイルと /etc/rc.nfs ファイルの両方で使用される **gfsinstall** コマンドを備えています。 これらのコールのどちらも除去する必要はありません。

Kerberos サポートがインストールされていない場合に発生する問題
Kerberos サポートがインストールされていないと、**gssd** デーモンは開始しません。

krb5.client.rte および modcrypt.base ファイルセットがインストールされていることを確認します。 どちらもインストールされていない場合は、**gssd** デーモンは実行されません。

レジストリー・デーモンが実行されていない場合に確認する項目
NFS バージョン 4 ドメインが構成されていない場合は、**nfsrgyd** デーモンは実行されません。

NFS バージョン 4 ドメインを構成する方法の詳細については、576 ページの『/etc/nfs/local_domain ファイル』を参照してください。

NFS ファイル
ここでは、NFS ファイルとその説明を参照できます。

項目	説明
bootparams	ディスクレス・クライアントがブートに使用できるクライアントを表示します。
exports	NFS クライアントにエクスポートできるディレクトリーを 表示します。
filesystems	システムの 再始動時にマウントされるすべてのファイルシステムを 表示します。
hostkey	Kerberos ホスト・プリンシパルと keytab ファイルのロケーションを指定します。
local_domain	システムのローカル NFS ドメインが入っています。
networks	インターネット・ネットワーク上のネットワークに関する情報が入っています。
pcnfsd.conf	**rpc.pcnfsd** デーモンの構成オプションを提供します。
princmap	プリンシパルがサーバーの 完全修飾ドメイン名でない場合は、ホスト名を Kerberos プリンシパルにマップします。
realm.map	着信 Kerberos プリンシパルをマップするために NFS レジストリー・デーモンによって使用されます。
rpc	リモート・プロシージャー・コール (RPC) プログラムのデータベース情報が入っています。
security_default	NFS セキュリティー・デフォルトが入っています。
xtab	現在エクスポートされているディレクトリーを表示します。

NFS コマンド
ここでは、NFS コマンドとその説明を参照できます。

項目	説明
chnfs	指定した数の **biod** デーモンと **nfsd** デーモンを開始するようにシステム 構成を変更します。
chnfsdom	ローカル NFS ドメインを変更します。
chnfsim	NFS 外部 ID マッピングを変更します。
chnfssec	NFS クライアントによって 使用されるデフォルトのセキュリティー・フレーバーを変更します。
chnfsrtd	ローカル NFS のレルムからドメインへのマッピングを 変更します。
mknfs	NFS デーモンを実行するためにシステムを構成します。

項目	説明
nfso	ネットワーク・ファイルシステム (NFS) のネットワーク変数を構成します。
automount	NFS ファイルシステムを 自動的にマウントします。
chnfsexp	NFS クライアントにディレクトリーをエクスポートするときに 使用する属性を変更します。
chnfsmnt	NFS サーバーからディレクトリーをマウントするときに 使用する属性を変更します。
exportfs	ディレクトリーを NFS クライアントにエクスポートおよびアンエクスポートします。
lsnfsexp	ネットワーク・ファイルシステム (NFS) を用いてエクスポートされたディレクトリーの特性を表示します。
lsnfsmnt	マウントできる NFS ファイルシステムの 特性を表示します。
mknfsexp	ディレクトリーを NFS クライアントにエクスポートします。
mknfsmnt	NFS サーバーからディレクトリーをマウントします。
nfshostkey	NFS サーバー用のホスト・キーを構成します。
nfs4cl	クライアントが NFS バージョン 4 を使用してアクセスするファイルシステムについての情報を表示します。
nfs4smctl	NFS バージョン 4 状態の取り消しを管理します。
rmnfs	システム構成を変更して、実行中の NFS デーモンを停止させます。
rmnfsexp	NFS クライアントのディレクトリーをアンエクスポートします 。
rmnfsmnt	NFS のマウントを除去します。

NFS デーモン

ここでは、NFS デーモンとその説明を参照できます。

ロック・デーモン

項目	説明
lockd	RPC パッケージを使用してロック要求を処理します。
statd	NFS 上でロック・サービスのためのクラッシュ機能およびリカバリー機能を提供します。

ネットワーク・サービスのデーモンとユーティリティー

項目	説明
biod	ファイルに関するクライアント要求を処理します。
mountd	ファイルシステムのマウントを 求めるクライアントからの要求に応じます。
nfsrgyd	セキュリティー・プリンシパル、NFS バージョン 4 ID ストリング、および対応する数値システム ID 間の変換を実行します。 さらに、外部 NFS バージョン 4 ドメインの ID 情報のマッピングが提供されます。
nfsd	クライアントからのファイルシステム 操作を要求するデーモンを開始します。
nfsstat	特定のマシンの呼び出しを受信する機能に関する情報を表示します。
on	リモート・システム上でコマンドを実行します。
pcnfsd	PC-NFS (パーソナル・コンピューター・ネットワーク・ファイルシステム) クライアントからのサービス要求を処理します。
portmap	RPC プログラム番号をインターネットのポート 番号に変換します。
rexd	リモート・マシン用のプログラムを実行します。

項目	説明
rpcgen	RPC プロトコルをインプリメントするための C コードを生成します。
rpcinfo	リモート・プロシージャー・コール (RPC) サーバーの状況を報告します。
rstatd	カーネルから入手したパフォーマンス統計情報を戻します。
rup	ローカル・ネットワーク上のリモート・ホストの状況を表示します。
rusers	リモート・マシンにログオンしたユーザーのリストを報告します。
rusersd	**rusers** コマンドからの照会に応答します。
rwall	ネットワーク上の全ユーザーにメッセージを送信します。
rwalld	**rwall** コマンドからの要求を処理します。
showmount	リモートからファイルシステムをマウントしているすべてのクライアントのリストを 表示します。
spray	指定された数のパケットをホストに送信し、パフォーマンス統計情報を報告します。
sprayd	**spray** コマンドで送信されたパケットを受信します。

セキュア・ネットワークのデーモンとユーティリティー

項目	説明
chkey	暗号鍵を変更します。
gssd	NFS が、ネットワーク認証サービスで提供されるセキュリティー・サービスにアクセスできるようにします。
keyenvoy	ユーザー・プロセスと key server デーモンの媒介として機能します。
keylogin	ユーザーの秘密鍵を復号して格納します。
keyserv	公開鍵と秘密鍵を格納します。
mkkeyserv	**keyserv** デーモンを開始し、`/etc/rc.nfs` ファイル内の該当エントリーのコメント文字を削除します。
newkey	`publickey` ファイル内に新しいキーを作成します。
rmkeyserv	**keyserv** デーモンを停止し、`/etc/rc.nfs` ファイルの **keyserv** デーモンのエントリーにコメントを付けます。
ypupdated	Network Information Service (NIS) マップ内の情報を更新します。

NFS セキュリティーに関する追加情報については、セキュリティーのネットワーク・ファイルシステムのセキュリティーを参照してください。

Sun ディスクレス・クライアントのサポート

項目	説明
bootparamd	ブート用の情報をディスクレス・クライアントに提供します。

NFS サブルーチン

ここでは、NFS サブルーチンについて説明します。

項目	説明
`cbc_crypt`、`des_setparity`、または `ecb_crypt`	データ暗号化規格 (DES) ルーチンをインプリメントします。

SMB プロトコル

Server Message Block (SMB) プロトコルは、ネットワーク上のファイル、ディレクトリー、プリンター、シリアル・ポート、および他のリソースに対する共用アクセスに使用されるクライアント/サーバー間通信プロトコルです。 また、このプロトコルは、認証済みプロセス間通信 (IPC) メカニズムも提供します。

Server Message Block (SMB) ファイルシステム

Server Message Block file (SMBFS) では、SMB プロトコル・バージョン 1.0 を使用することによって、SMB サーバー上の共用リソースを AIX オペレーティング・システムのローカルのファイルシステムとしてアクセスできます。

このファイルシステムでは、ユーザーはファイルとディレクトリーの作成、削除、読み取り、書き込み、およびアクセス時間の変更を行うことができます。ファイルとディレクトリーのオーナーまたはアクセス・モードは変更できません。

SMBFS は SMB サーバー上のファイルにアクセスするために使用できます。 SMB サーバーは、Samba を実行するサーバー、あるいは Windows XP、Windows NT、Windows 2000 のサーバーまたはワークステーションです。これらのサーバー・タイプは、それぞれ、ディレクトリーを共用リソースとしてエクスポートできます。 そのあと、この共用リソースは SMBFS を使用して AIX システムにマウントできます。

SMB ファイルシステムのインストール

AIX システムに SMBFS をインストールするには、`bos.cifs_fs` パッケージをインストールします。

`bos.cifs_fs` パッケージがインストールされるときに、デバイス `nsmb0` が作成されます。 このデバイスによって、**mount** コマンドを使用して SMB サーバーとクライアント 間の接続を SMB プロトコル・バージョン 1.0 で確立できます。

SMB ファイルシステムのマウント

AIX の **mount** コマンドを使用して、SMBFS をマウントすることができます。 例:

```
mount -v cifs -n pezman/user1/pass1 -o uid=201,fmode=750 /home /mnt
```

-o フラグを使用して、マウント・オプションを指定できます。 コマンド・ライン・オプションを区切るには、コンマとスペースではなく、コンマだけを使用します。 ファイルシステムのオプションは、次のとおりです。

項目	説明
fmode	ファイルまたはディレクトリーを 8 進モードに設定します。 デフォルト値は 755 です。
uid	マウント中に UID をファイルに割り当てます。 デフォルトは **root** です。
gid	マウント中に GID をファイルに割り当てます。 デフォルトは **system** です。
wrkgrp	SMB サーバーが属するワークグループ。
op	オポチュニスティック・ロックを使用する場合は、値を 1 に設定します。 オポチュニスティック・ロックを使用しない場合は、値を 0 に設定します。
opfs	ロック・キャッシュ・ファイルを保管するために使用するキャッシュ・ファイルシステムの名前。
opsz	オポチュニスティック・ロックに使用する個々のキャッシュ・ファイルのサイズ。
opfssz	オポチュニスティック・ロックで使用されるキャッシュ・ファイルシステムのサイズ 。

SMIT ユーティリティー `smit cifs_fs` を使用して、ファイルシステムをマウントすることもできます。このユーティリティーは、必要な情報をすべて収集したあと **mount** コマンドを実行します。

SMBFS をマウントするには、認証のためのユーザー名とパスワードをサーバーに提供する必要があります。 このユーザー名とパスワードは、サーバー上で行うすべての必要なファイル操作に使用されます。

SMIT のパネルの「**Password (パスワード)**」フィールドは、必須であるとはマークされていません。 パスワードのフィールドに入力されていないと、cifscred ファイルで一致するユーザーまたはサーバーの信用証明情報がないか検索されます。 一致する情報がある場合は、cifscred ファイルから保管されたパスワードが使用されます。一致する情報がない場合は、標準的な AIX パスワード・プロンプトを使用してパスワードの入力を求められます。 この方法では、パスワードを表示させずに、パスワードを入力できます。

注 : SMBFS のマウントに使用するパスワードの長さは最大 14 文字とし、パスワードには特殊文字を含むことができます。

read などのファイルシステムのコマンドが SMBFS マウント・ポイント内のファイルに対して呼び出されるたびに、要求がサーバーに送信されてファイルが読み取られます。ユーザー名とパスワードはこの要求の一部として送信され、サーバー上でそのファイルの読み取り操作を行う許可がユーザーにあるかどうかサーバーが決定できるようにします。 したがって、ファイルの操作が許可できるものであるかどうかについての最終的な権限は、サーバーにあります。

ただし、**mount** コマンドの fmode オプションは、サーバーに照会される前にクライアント・システム上の root ユーザーがサーバー上のファイルへのアクセスを制御できる方法を提供します。 ユーザーが fmode オプションを指定しない場合、デフォルトは 755 です。 次の表に、write 要求を使用するときの fmode オプションの影響を示します。

表 *94.* 与えられた許可に基づく、ユーザーのアクセスの許可または拒否の *5* つのケース

ケース番号	サーバーに対し認証されているユーザー	クライアント・サイドで書き込みアクセス権限を 待つユーザー	マウント・オーナー、グループ、およびモード	サーバー上のオーナー、グループ、およびモード	アクセスの許可
ケース 1	user1	user2	user1、staff rwxr-xr-x	user1、staff rwxrwxr-x	いいえ
ケース 2	user1	root	user1、staff rwxr-xr-x	user2、staff rwxr-xr-x	いいえ
ケース 3	user1	user1	user1、staff rwxr-xr-x	user2、staff rwxrwxr-x	はい
ケース 4	user1	user1	user1、staff rwxr-xr-x	root、system rwx------	いいえ
ケース 5	user1	user1	user1、staff rwxr-xr-x	root、system rwxrwxrwx	はい

ケース 1 では、クライアントのマウントでのオーナー、グループ、およびモードが user2 に対して書き込みアクセスを許可していないため、アクセスが拒否されました。

ケース 2 では、root はクライアント・サイドのすべてに対してアクセス権を持っていますが、サーバーが認証したユーザー user1 は、サーバー上のファイルに対するアクセス権を持っていないため、アクセスが拒否されました。

ケース 3 では、user1 はマウントのオーナーであり、user1 は サーバー上のグループ staff のメンバーであってサーバー上のファイルに対するアクセス権があるため、アクセスが認可されました。

ケース 4 では、user1 はマウントのオーナーですが、ファイルはサーバー上の root によって所有されていて、グループその他からのアクセスが許可されていないため、アクセスは拒否されました。

ケース 5 では、user1 はマウントのオーナーであり、user1 は他の許可を通して サーバー上のファイルに対するアクセス権があるため、アクセスが認可されました。

注:

1. AIX SMBFS クライアントは、SMBv1 のみサポートします。
2. マウントされたファイルシステムでは、ファイルのサイズが 4 GB+4096 バイト以下の場合、1 つのファイルから別のファイルへのコピー操作が正常に行われます。 このサイズを超えるファイルの場合は、警告メッセージが表示され、元のファイルのうち 4 GB+4096 バイトが宛先にコピーされます。
3. マウントされたファイルシステムでは、以下の文字をファイル名に使用することはできません: 円記号キー {¥}、スラッシュ・キー {/}、コロン {:}、アスタリスク {*}、疑問符 {?}、不等号キー (小なり) {< }、不等号キー (大なり) {>}、垂直棒キー { | }。

保管されたパスワード

SMBFS は `server/user/password` の信用証明情報を `/etc/cifs_fs/cifscred` ファイルに保管することで、SMBFS のマウント時にパスワードを自動的に検索できます。

このファイルで信用証明情報の追加、変更、および除去を行うには、`/usr/sbin` ファイル内の **mkcifscred**、**chcifscred**、および **rmcifscred** のコマンドを使用します。 このファイルに追加されたパスワードは暗号化されます。 パスワードを指定しないでマウントを試行すると、`cifscred` ファイルから一致する信用証明情報が検索されます。 一致する情報がある場合は、`cifscred` ファイルから保管されたパスワードが使用されます。一致する情報がない場合は、標準的な AIX パスワード・プロンプトを使用してパスワードの入力を求められます。

保管されたパスワードのサポートには、以下の制限があります。

- 保管されたパスワードの検索を正しく行うには、サーバー命名規則が一貫していなければなりません。 例えば、ホスト名または完全修飾ドメイン名 (FQDN) ではなく、IP アドレスを使用して信用証明情報が追加された場合、パスワードは IP アドレスでマウントする場合のみ検索されます。
- 保管されたパスワードの検索方法では、プレーン・テキスト・パスワードの認証はサポートされていません。 サーバーでプレーン・テキスト・パスワードが必要な場合は、認証が失敗します。

/etc/filesystems のサポート

SMBFS は `/etc/filesystems` をサポートしており、システム起動時の自動マウントが可能です。

`/etc/filesystems` をサポートすることで、マウント時に保管サーバー、ユーザー、パスワード、およびオプションのデータにアクセスすることもできます。 `/etc/filesystems` の cifs スタンザを追加、変更、除去、およびリストするには、それぞれ **mkcifsmnt**、**chcifsmnt**、**rmcifsmnt**、および **lscifsmnt** の各コマンド (`/usr/sbin` 内に配置) を使用します。 証明書は `cifscred` ファイル内に保管する必要があります。

SMBFS のトラブルシューティング

mount コマンドまたは `smit cifs_fs` 高速パスがエラーを戻した場合、以下のトラブルシューティング手順を考慮してください。

- ユーザー名とパスワードが正しいことを確認します。 ユーザー名とパスワードは、サーバー上の共用へのアクセスを許可するために必要です。
- サーバー名が正しいことを確認します。 サーバー名が正しければ、サーバーがクライアントと同じサブネットの一部でない場合に備えて、完全修飾のホスト名を使用します。 サーバーの IP アドレスの使用を試みることもできます。
- **lsdev -L|grep nsmb** コマンドがデバイス名を戻すことを確認します。 `nsmb` デバイスが使用不可であれば、AIX クライアントは SMB サーバーへの接続を確立できません。
- 共用名が正しいことを確認します。 サーバー上に共用がないか、指定されたユーザー名とパスワードでアクセスできない場合、SMB サーバーは接続要求を拒否します。
- イベント ID 525 を使用して、SMBFS 用のシステム・トレース・データを収集します。
- サーバーで NTLM、LM、またはプレーン・テキストのパスワードの受け入れが構成されていることを確認します。 これらのタイプのパスワードの暗号化だけが SMBFS でサポートされています。

- ドメインに対する認証が必要な場合は、そのドメイン名を **wrkgrp** オプションで指定する必要があります。 このオプションを指定しないと、認証はサーバーによりローカルで処理されます。

Server Message Block (SMB) クライアント・ファイルシステム

SMB クライアント・ファイルシステムは SMB プロトコルのバージョン 2.1 とバージョン 3.0.2 に基づいています。 SMB クライアント・ファイルシステムを 使用すれば、SMB サーバー上のファイルにアクセスできます。

SMB サーバーは、Windows Server 2012 オペレーティング・システムまたは Windows Server 2016 オペレーティング・システムが稼働するサーバーです。 どちらのサーバー・オペレーティング・システム・タイプでも、ディレクトリーを共用リソース としてエクスポートできます。 その後で、この共用リソースは、SMB クライアント・ファイルシステムを 使用して AIX 論理区画にマウントできます。 SMB クライアント・ファイルシステムを 使用すれば、AIX 論理区画上のローカル・ファイルシステムとして SMB サーバー上の共用リソースにアクセスできます。 SMB クライアント・ファイルシステムは、SMB サーバー上においてファイルやディレクトリーの作成、削除、読み取り、および書き込みを行う場合に使用したり、そのファイルやディレクトリーへのアクセス 期間を変更する場合にも使用したりできます。 ただし、このようなファイルやディレクトリーの所有者やアクセス権は変更できません。

SMB クライアント・ファイルシステムでは、以下の SMB プロトコル 3.0.2 機能を使用できます。

SMB 3.0.2 セキュア・ダイアレクト・ネゴシエーション

SMB プロトコル・バージョン 3.0.2 を使用すれば、SMB サーバーからの共用リソースを AIX 仮想ファイルシステム (VFS) 内にマウントできます。

SMB 3.0.2 ダイアレクトサーバーは、セキュリティー・リスクを回避するためにセキュア・ダイアレクト・ネゴシエーションを提供します。 SMB 3.0.2 ダイアレクトがネゴシエーションされるとき、SMB クライアントは、ネゴシエーション情報を検証するために、署名済みの必須要求を送信する必要があります。

SMB 3.0.2 署名

SMB プロトコル 3.0.2 では、署名に最新の暗号化アルゴリズムが使用されます。 Advanced Encryption Standard (AES) Cipher-based Message Authentication Code (CMAC) (AES-128-CMAC) は、発信メッセージに署名すること、および着信メッセージを検証することで、SMB クライアントと SMB サーバーの間で交換されるメッセージの整合性を保証するためのものです。

SMB 3.0.2 暗号化

SMB 暗号化により、SMB データをエンドツーエンドで暗号化できるようになり、非トラステッド・ネットワークで盗聴されないようにデータが保護されます。 SMB 暗号化は、共用リソースごとに構成することも、ファイル・サーバー全体で構成することもできます。 また、SMB 暗号化は、データが非トラステッド・ネットワークをトラバースするさまざまなシナリオに対して有効にできます。

- 629 ページの『SMB クライアント・ファイルシステムのインストール』
- 630 ページの『SMB クライアント・ファイルシステムをローカル・マウント・ポイントとしてマウント』
- 631 ページの『SMB クライアント・ファイルシステムの Kerberos 認証』
- 632 ページの『保管されたパスワード』
- 633 ページの『/etc/filesystems ファイル・サポート』

SMB クライアント・ファイルシステムのインストール

AIX オペレーティング・システムでの SMB クライアント・ファイルシステムは、SMB プロトコルのバージョン 2.1 またはバージョン 3.0.2 を使用してユーザー認証セッションを開始するために、Kerberos ベースの GSSAPI を必要とします。 AIX オペレーティング・システムでは、この GSSAPI は、IBM Network Authentication Service (NAS) バージョン 1.16.1.0 以降のファイルセットでユーザー・スペース・ライブラリーによって提供されます。 SMB バージョン 3.0.2 は、署名と暗号化のための鍵の生成に AIX OpenSSL ライブラリーを使用します。 そのため、1.0.2.2002 以降のバージョンの `openssl.base` ファイルセットをインストールする必要があります。 このファイルセットは AIX Expansion Pack に含まれています。

AIX 論理区画に SMB クライアント・ファイルシステムをインストールするには、以下の手順を実行します。

1. Go to the AIX Web ダウンロード・パック・プログラム Web ページにアクセスし、自分の IBM ID とパスワードを使用してサインインします。
2. 「**SMB CLIENT バージョン 3.0.2 (SMB CLIENT Version 3.0.2)**」オプションを選択して、「**ダウンロード**」をクリックします。

   自分の IBM 資格情報に、SMB クライアント・ファイルシステム・パッケージをダウンロードする資格が必要です。 さもないと、そのパッケージはダウンロードできません。
3. **installp** コマンドを使用して `smbc.rte` パッケージをインストールします。

   `smbc.rte` パッケージがインストールされると、デバイス `nsmbc0` が作成されます。 このデバイスで **mount** コマンドを実行すれば、SMB クライアント・プロトコルのバージョン 2.1 またはバージョン 3.0.2 を使用して SMB サーバーと SMB クライアントのファイルシステム 間で接続を確立できます。

SMB クライアント・ファイルシステムをローカル・マウント・ポイントとしてマウント

SMB クライアント・ファイルシステムをマウントするには、以下のコマンドを使用します。

```
mount -v smbc -n windows_server/Kerberos_username/password_for_Kerberos_user    ¥
-o wrkgrp=workgroup,[[port=139|445],[signing=required|enabled],[pver=2.1|3.0.2|auto],  ¥
[encryption=desired|required|disabled],[secure_negotiate=desired|required|disabled]   ¥
share_point_to_mount_created_on_windows local_mount_point
```

例えば、次のとおりです。

```
mount -v smbc -n llm140.xyz.com/cec102usr1/Passw0rd     ¥
-o "wrkgrp=SMB_302.test,port=445,signing=required,encryption=required,    ¥
secure_negotiate=desired,pver=auto" /some_share /mnt
```

mount コマンドの **-o** フラグで以下のパラメーターを指定できます。 これらのパラメーターはコンマでのみ区切る必要があります。 コンマの前後にスペースを入れないでください。

fmode
ファイルまたはディレクトリーを、アクセス権用の 8 進モードに設定します。 デフォルト値は 755 です。

uid
マウント操作中にユーザー ID をファイルに割り当てます。 デフォルト値は `root` です。

gid
マウント操作中にグループ ID をファイルに割り当てます。 デフォルト値は `system` です。

wrkgrp
SMB サーバーが属しているワークグループを指定します。 このパラメーターは、SMB クライアント・ファイルシステムをマウントするために必須です。

ポート
ポート番号を指定します。 有効な値は 445 と 139 です。 デフォルト値は 445 です。 ポート 139 は、指定したサーバー・アドレスが IPv4 フォーマットの場合にのみサポートされます。

pver
これは、SMB サーバーとの通信に使用される SMB プロトコルのバージョンを指定します。 有効な値は 2.1、3.0.2、および `auto` です。 値に `auto` を指定した場合は、指定した SMB サーバーに基づいて SMB プロトコルのバージョン 2.1 またはバージョン 3.0.2 が使用されます。

signing
これは、SMB クライアント・ファイルシステムで 通信用にデジタル署名が必要かどうかを指定します。 有効な値は `enabled` と `required` です。 **signing** パラメーターが `enabled` に設定されている場合、SMB サーバー・ファイルシステムで 通信用にデジタル署名が必要となる場合を除き、SMB クライアント・ファイルシステムはデータ・パケットにデジタル署名しません。 **signing** パラメーターが `required` に設定されている場合、SMB クライアント・ファイルシステムは 通信用のデータ・パケットにデジタル署名する必要があります。 **mount** コマンドを使用して `signing` パラメーターの値を指

定しない場合は、smbctune コマンドを使用して設定されたカーネル・チューナブル・パラメーター値からのデフォルト値が使用されます。

secure_negotiate
これは、SMB クライアント・ファイルシステムで暗号化が必要であるかどうかを指定します。 有効な値は、desired、required、および disabled です。 このパラメーターを mount コマンドで指定しない場合は、smbctune コマンドを使用して設定されたカーネル・チューナブル・パラメーター値からのデフォルト値が使用されます。

encryption
これは、SMB クライアント・ファイルシステムで暗号化が必要であるかどうかを指定します。 有効な値は、desired、required、および disabled です。 このパラメーターを mount コマンドで指定しない場合は、smbctune コマンドを使用して設定されたカーネル・チューナブル・パラメーター値からのデフォルト値が使用されます。

SMB クライアント・ファイルシステムの Kerberos 認証

SMB クライアント・ファイルシステムをマウントするには、Kerberos ユーザー名と Kerberos パスワードを指定して SMB サーバーに対して認証を行う必要があります。 このユーザー名とパスワードは、SMB サーバー上で行うすべての必要なファイル操作に使用されます。 パスワードを指定しないと、標準 AIX パスワード・プロンプトを使用してパスワードが要求されます。

注 : SMB クライアント・ファイルシステムのマウントに使用されるパスワードの長さは最大で 255 文字にすることができます。 このパスワードには特殊文字を含めることができます。

read コマンドなどのファイルシステム・コマンドを SMB マウント・ポイントでファイルに対して実行すると、そのファイルを読み取るための要求が SMB サーバーに送信されます。 また、認証されたセッション ID もこの読み取り要求の一部として送信されます。 SMB サーバーは、このセッション ID を使用して、ユーザーがサーバーに対して認証されているかどうかを判別し、ファイルに対して読み取り操作を実行します。 このようにして、SMB サーバーはファイルに対するアクセスを許可し、ファイルに対して操作を実行できるかどうかを制御します。

SMB クライアント・ファイルシステム 上の root ユーザーは、SMB サーバーの照会前に **mount** コマンドの **fmode** オプションを使用して SMB サーバーでのファイルに対するアクセスを制御します。 **fmode** オプションに対して値を指定しないと、**fmode** オプションではデフォルト 値 755 が使用されます。 各種操作で **fmode** オプションがどのように作用するのかについて以下の表示で説明します。

表 *95. SMB* サーバー上のファイルまたはディレクトリーの指定アクセス権に基づいてユーザーがアクセスを許可/拒否されるケース

ケース番号	SMB サーバーに対して認証されたユーザー	書き込みアクセスを要求するクライアント・システム上のユーザー	マウント・オーナー、グループ、アクセス・モード	SMB サーバーでのファイル/ディレクトリー・オーナー、グループ、SMB サーバーでのアクセス・モード	アクセス権
ケース 1	user1	user2	user1、staff、rwxr-xr-x	user1、staff、rwxrwxr-x	×
ケース 2	user1	root	user1、staff、rwxr-xr-x	user2、staff、rwxr-xr-x	×
ケース 3	user1	user1	user1、staff、rwxr-xr-x	user2、staff、rwxrwxr-x	はい
ケース 4	user1	user1	user1、staff、rwxr-xr-x	root、system、rwx------	×

表 95. SMB サーバー上のファイルまたはディレクトリーの指定アクセス権に基づいてユーザーがアクセスを許可/拒否されるケース (続き)

ケース番号	SMB サーバーに対して認証されたユーザー	書き込みアクセスを要求するクライアント・システム上のユーザー	マウント・オーナー、グループ、アクセス・モード	SMB サーバーでのファイル/ディレクトリー・オーナー、グループ、SMB サーバーでのアクセス・モード	アクセス権
ケース 5	user1	user1	user1、staff、rwxr-xr-x	root、system、rwxrwxrwx	はい

ケース 1 では、SMB クライアント上のマウント・ポイントにおけるオーナー、グループ、およびモードが user2 に書き込み権限を提供していなかったため、ファイルまたはディレクトリーへのアクセスが user2 に対して拒否されます。

ケース 2 では、root ユーザーが SMB クライアントですべてのアクセス権限を持っているとしても、SMB サーバー認証ユーザー user1 が SMB サーバー上のファイルに対するアクセス権限を持っていないため、ファイルまたはディレクトリーへのアクセスが root ユーザーに対して拒否されます。

ケース 3 では、user1 はマウント操作時にマウント・オーナーであり、SMB サーバー上のグループ staff のメンバーであり、そのサーバー上のファイルに対するアクセス権限を持っていたため、ファイルまたはディレクトリーにアクセスできます。

ケース 4 では、user1 はマウント操作時にオーナーであったとしても、ファイルは SMB サーバー上の root ユーザーによって所有されていて、グループ・メンバーおよび他のユーザーがアクセス権を持っていないため、ファイルまたはディレクトリーへのアクセスは user1 に対して拒否されます。

ケース 5 では、指定のアクセス・モードにより、すべてのグループ・メンバーと他のユーザーに対するすべてのアクセス権が指定されるため、user1 はファイルまたはディレクトリーにアクセスできます。

注 : マウントされたファイルシステムでは、円記号キー (¥)、スラッシュ・キー (/)、コロン (:)、アスタリスク (*)、疑問符 (?)、不等号キー (小なり) (<)、不等号キー (大なり) (>)、垂直棒キー (|) はファイルの名前で使用できません。

保管されたパスワード

SMB クライアント・ファイルシステムでは、SMB クライアント・ファイルシステムをマウントするときにパスワードを自動的に取得できるように、サーバー名、ユーザー名、およびパスワードの資格情報を /etc/smbcred ファイルに保管できます。 /usr/sbin/ ディレクトリーにある **lssmbcred** コマンド、**mksmbcred** コマンド、**chsmbcred** コマンド、および **rmsmbcred** コマンドを使用して /etc/smbcred ファイル内の資格情報を表示、追加、および削除できます。 /etc/smbcred ファイルに追加されるパスワードは暗号化されます。 パスワードを指定せずに SMB クライアント・ファイルシステムをマウントするときは、一致する資格情報が /etc/smbcred ファイルで検索されます。 一致する資格情報が見つかると、/etc/smbcred ファイルにある保管済みパスワードが使用されます。 一致する資格情報が見つからない場合は、標準 AIX パスワード・プロンプトを使用してパスワードが要求されます。

保管済みパスワードに関する以下の制限事項を考慮してください。

- 保管済みパスワードを取得するには、サーバー命名規則が一貫していなければなりません。 例えば、ホスト名や完全修飾ドメイン名 (FQDN) ではなく IP アドレスを使用して資格情報が追加されている場合は、IP アドレスを使用して SMB クライアント・ファイルシステムをマウントするときに限りパスワードを取得できます。
- smbc.rte ファイルセットをアンインストールする 前に /etc/filesystems ファイルから資格情報エントリーを削除する必要があります。

/etc/filesystems ファイル・サポート

SMB クライアント・ファイル・システムは、システム始動操作時にファイルシステムの 自動マウント操作を許可するために /etc/filesystems ファイルをサポートします。 /etc/filesystems ファイルにより、ファイルシステムのマウント時に保管済みのサーバー名、ユーザー名、パスワード、および構成データにアクセスすることもできます。 SMB クライアント・ファイルシステム・スタンザを /etc/filesystems ファイルに手動で追加するときは、SMB クライアント・ファイルシステム 資格情報を /etc/smbcred ファイルに保管する必要があります。

例:

```
$cat /etc/filesystems
.....................
.....................
.....................

/mnt1:
dev = /fvt_share
vfs = smbc
mount = true
options = "wrkgrp=SMB_21.FVT"
nodename = <servername>/<username>

/mnt:
dev = /fvt_share
vfs = smbc
mount = true
options = "wrkgrp=SMB_21.FVT,signing=required"
nodename = <servername>/<username>
```

非同期通信

AIX には、次のカテゴリーの非同期デバイス・ドライバー (tty デバイス・ドライバーとも呼ばれる) があります。

- システム・プレーナー上のシリアル・ポート用のドライバー
- アダプターによってシステムに 接続されたシリアル・ポート用のドライバー
- 疑似 tty ドライバー

最初のカテゴリーのドライバーは、PCI アダプターです。これらのアダプターには、2 ポート、8 ポート、および 128 ポートのアダプターがあります。

2 番目のカテゴリーの 8 ポートおよび 128 ポートの PCI アダプターは、Intel 8086 プロセッサーを使用してホスト CPU からほとんどの文字処理をオフロードするため、インテリジェント・アダプターと 呼ばれます。 これらのアダプターは、ハードウェア割り込み主導ではなく 20 ms ポーリング・プログラム主導で、大部分のシリアル・デバイスおよびアプリケーションに 適したパフォーマンス特性を提供します。 多くのデバイスをシステムに 追加してもシステム・ワークロードはほとんど増えないため、これらのアダプターは、ハードウェア割り込みを使用する場合と比較してかなり多くのシリアル・デバイスをサポートできます。 また、これらのアダプターは特許を得たソフトウェア・パフォーマンス機能拡張を使用しているため、データを大きいブロックで移動する限り、ネイティブのシステム ・ポートと比較して大量のデータをより速くかつ効率的に送受信できます。 詳しくは、/usr/include/sys/pse/README.pse ファイルの wantio の説明を参照してください。

注 : 内蔵 POWER5 システム・ポート はシリアル・ポートと類似していますが、システム・ポートは特別にサポートされた機能にのみ使用可能です。 詳しくは、640 ページの『システム・ポートとシリアル・ポートの異なる機能』 を参照してください。

ただし、一部のデバイスおよびアプリケーションでは、単一文字処理に非常に短い待ち時間が期待されており、またその必要があるため、これらのインテリジェント・アダプターに 接続する場合にタイミングの問題が発生することがあります。 文字待ち時間 (文字エコー) は単一文字をシリアル・ポートで受信し、その文字をアプリケーションに配送してから同じシリアル・ポートにエコーするのにかかる時間と定義できます。

割り込み主導ポートはシステムで最高の優先割り込み (INTCLASS0) を使用するため、アイドル・システムでの待ち時間の値の範囲は 0.10 から 0.20 ms になります。 8 ポート PCI アダプターでは、待ち時間の平均

値は 10 から 12 ms になりますが、20 ms ポーリング・プログラムのために個々の時間には 10 ms の誤差があります。 128 ポート PCI アダプターには同じ 20 ms ポーリング・プログラムがあり、リモート・アクセス・ノード (RAN) へのポーリング通信リンクで通信します。 RAN では、ポーリング・ドライバーによるシリアル・ポートの制御が可能です。 これらのポートの待ち時間の値は平均 30 ms ですが、60 ms を超える場合もあります。

8 ポート PCI および 128 ポート PCI アダプターの待ち時間の値は、「イベント遅延」(EDELAY) パラメーターを使用して、特殊なアプリケーション用に調整できます。 単一文字を受信する場合の反応を最大にするには、EDELAY パラメーターの値を低くします。 これによって、単一文字をシリアル・ポートからアプリケーションに渡すために必要な時間は最小限に抑えられますが、複数の文字がまとめて受信されると、スループットおよび全体的なシステム・パフォーマンスが低下する場合があります。

2 ポート PCI EIA-32 アダプターは、非同期のシリアル通信アダプターで、Exar 17D152 Universal PC Dual UART に基づいています。 この 2 ポート・アダプターは、2 本の DB-9 コネクターをサポートし、 モデムや tty 端末など非同期の EIA-32 デバイスへの接続性を提供します。

IBM eServer™ p5 プラットフォームでは、AIX に使用できるネイティブ・システム・ポートはありません。仮想端末インターフェースは、 ハイパーバイザーによって FSP に置かれている物理シリアル・ポートを サポートするように拡張されていますが、このインターフェースは、特定の シリアル・デバイスのセットのみをサポートするので、汎用の物理シリアル・ポートの 代替としては適切ではありません。 2 ポート・アダプターの動作は、 ネイティブ・システム・ポートにやや似ています。 このアダプター・デバイス・ドライバーは、割り込み駆動であり、 プログラマブル伝送をサポートし、FIFO トリガー・レベルを受信します。 これは PCI アダプターです。したがって、デバイス・ドライバーは EEH、 ホット・プラグ、および VPD 照会をサポートします。 この 2 ポート・アダプターは、ブート、インストール、および KDB サポートなど、仮想端末が使用されている間は、 ネイティブ・システム・ポート・フィーチャーをサポートしません。

疑似 tty ドライバーは、**rlogin** または **telnet** コマンドを使用してネットワークを介してシステムにアクセスする場合、またはグラフィック・モニターのウィンドウ操作システムを使用してシステムにアクセスする場合に使用されます。 疑似 tty ドライバーは、文字ベースのレガシー・アプリケーション (**vi** テキスト・エディターなど) をシリアル以外の通信メディアで実行する手段を提供します。 疑似 tty ドライバーは、対称ではないことに注意してください。 スレーブ側では、以前のアプリケーションとの POSIX 規格準拠インターフェースが 提供されます。 マスター側は、**rlogin** または **telnet** デーモンまたは X Window システムなどのエンティティーによって 制御され、疑似 tty ドライバーにシリアル端末デバイスのエミュレーションを提供します。 AIX は、大量の疑似 tty デバイスを効率的にサポートできます。

非 POSIX 回線速度

POSIX およびそれ以降の UNIX 規格 (X/OPEN など) で指定されるシリアル・デバイスとのインターフェースは、`/usr/include/termios.h` に定義されている `termios` データ構造に依存しています。 しかし、このデータ構造を使用して 38,400 ビット/秒を上回る回線速度を指定することはできません。 現在使用されているほとんどのシリアル・ハードウェアは、最大で 230,000 bps の速度をサポートできます。 AIX で、このような高速の回線速度を使用するには、SMIT を使用してポートを構成するときに、希望の速度を指定する必要があります。指定した回線速度をシリアル・ポート・ハードウェア (UART) がサポートできる場合に、ポートを構成できます。

`termio` または `termios` 構造を使用した取得属性 ioctl は、回線速度を 50 bps と報告します。 ポートでは変更されない限り非 POSIX 回線速度が使用されるため、回線速度を本当に変更する意図がないのであれば、`termio` および `termios` 構造を使用する設定属性 ioctl で、CBAUD フラグを変更しないでください。 要求した回線速度をシリアル・ポート・ハードウェア (UART) がサポートできない場合は、ポートの構成は失敗し、エラーが戻されます。

注 : 8 ポートおよび 128 ポート PCI アダプターは、非 POSIX 回線速度 (115,200 および 230,000 bps のみ) をサポートします。 128 ポート PCI アダプターには、さらに 2.5 Mbps の総帯域幅 (8 ワイヤー・ケーブル) という制約がありますが、これはそれぞれが 230,000 bps を維持して転送を行う 11 個のデバイスによって完全に消費されます。 この制約はアダプターを RAN に接続している回線に対するものであるため、単一アダプターは 22 個のデバイスが存在する場合に完全に消費されます。

非同期通信アダプター

非同期通信製品には、低コスト、マルチユーザー、中から高性能の端末およびデバイス通信という利点があります。

AIX では、多くのユーザーがシステム・リソースおよびアプリケーションにアクセスできます。 各ユーザーは、端末セッションによって接続する必要があります。 シリアル・ポートによるローカル接続またはリモート接続が可能です。

各システム装置では、少なくとも 1 つの標準シリアル・ポートが使用できます (一部のシステムには 3 つのシリアル・ポートがあります)。 これらのポートは、非同期通信およびデバイス接続をサポートできます。

非同期ポートには、EIA 232、EIA 422、または RS-423 規格に適合する次のような非同期周辺装置を接続できます。

- 非同期モデム
- バーコード・スキャナー
- グラフィックおよび文字プリンター
- キーボードおよびディスプレイ 端末
- パーソナル・コンピューター
- プロッターおよびプリンター
- POS 端末
- センサーおよび制御デバイス
- テキスト・スキャナー
- タイム・クロック

非同期通信オプション

PCI バスを使用するアダプターを持つシステム装置に拡張非同期機能を追加できます。

選択できる非同期接続のタイプは、いくつかの要因によって決まります。 次の表に、これらの製品の要約を示します。

表 *96.* 非同期通信プロダクト

非同期接続	POWER® プロセッサー・ベース	Itanium ベース	バス・タイプ	フィーチャー・コードまたはマシン・タイプ (モデル)	ポートごとの最大データ速度 (Kbps)	主要な機能
標準シリアル・ポート	X	X	システム・プレーナー	該当なし	universal asynchronous receiver and transmitter (UART) のボー・レート生成プログラム・クロック速度に基づいて選択可能。	標準機構
232 RAN	X	X		8130	57.6	リモート機能
拡張 232 RAN	X	X		8137	230	リモート機能
16 ポート RAN EIA 422	X	X		8138	230	リモート機能
128 ポート・コントローラー	X			8128	230	効率的な高デバイス・カウント

表 96. 非同期通信プロダクト (続き)						
非同期接続	**POWER® プロセッサー・ベース**	**Itanium ベース**	**バス・タイプ**	**フィーチャー・コードまたはマシン・タイプ (モデル)**	**ポートごとの最大データ速度 (Kbps)**	**主要な機能**
128 ポート・コントローラー	**X**			2933	230	効率的な高デバイス・カウント
128 ポート・コントローラー	**X**	**X**	PCI	2944	230	効率的な高デバイス・カウント

注 : ラック・マウント RAN FC は 8136 です。

注 : ポートごとの最大データ速度は回線帯域幅によって制限されます (標準 RAN の場合は 1.2 Mbps、拡張 RAN の場合は 2.4 Mbps)。

この表の最初の機能は、すべてのシステム装置に標準のプレーナー接続シリアル・ポートを表しています。 次の機能はアダプターです。 128 ポート非同期サブシステムには、このサブシステムに接続するリモート非同期ノード (RAN) も含まれます。

プレーナー接続非同期ポート
ほとんどのシステム装置モデルには、2 つの内蔵 (標準) EIA 232 非同期シリアル・ポートがあります。 EIA 232 非同期シリアル・デバイスは、9 ピン D シェル・コネクターを持つ標準シリアル・ケーブルを使用して標準シリアル・ポートに直接接続できます。

一部のマルチプロセッサー・システムには、リモート・サービス・センターとの通信に使用される 3 番目のシリアル・ポートがあります。

注 : Itanium ベース・システムには、1 つまたは 2 つの内蔵シリアル・ポートがあります。 初期ワークステーション・モデルには 1 つのポートがありますが、初期サーバー・クラス・モデルには 2 つのポートがあります。

アダプター接続非同期ポート
各アダプターにはバス・スロットが必要です。また、各アダプターは、必要なバス・タイプをサポートするシステムでのみ使用できます。

128 ポート、ISA 8 ポート・アダプター、および PCI 8 ポート・アダプターはインテリジェント・アダプターで、メイン・システム・プロセッサーに大幅なオフロードを提供します。

EIA 232 は最も一般的な通信規格ですが、EIA 422A (ケーブル距離が長い場合に使用される) もサポートされます。 EIA 422A インプリメンテーションには、デバイス状況の検出機能または RS 232 モデム制御シグナルは含まれていません。

注 : Itanium ベースのプラットフォームは、8 および 128 ポート PCI アダプターのみをサポートします。

ノード接続非同期ポート
128 ポート・アダプター (Micro Channel、ISA、または PCI バスで使用可能) には、1 つから 8 つのリモート非同期ノード (RAN) を接続できます。

各 RAN にはデバイス接続用の 16 の非同期ポートがあり、それぞれが別個の電源を持っています。 最大 4 つの RAN を、128 ポート・アダプター・カードの 2 つの接続からデイジー・チェーン接続できます。 RAN は、16 EIA 232 デバイスまたは 16 EIA 422 デバイスをサポートできます。 128 ポート・コントローラーは、特定の CPU 使用レベルで可能な非同期セッションの数を増やすインテリジェント・アダプターです。

128 ポート機能の追加特性を次に示します。

- フル・パフォーマンス・レーティングを維持できる RAN の設置範囲は、8 ワイヤー・シールド配線を使用してシステム・プロセッサーから 300 メートルまでです。
- RAN とシステム・プロセッサー間のデータ速度を落とせば、距離を 1200 メートルまで延長できます。

- 同期 EIA 232 および EIA 422 モデムを使用して、RAN をシステム・プロセッサーに対してリモートに設置できます。 4 つの RAN のデイジー・チェーンでは、それぞれ、チェーン内のある地点で 1 組みのモデムだけを使用できます。
- システム・パフォーマンスは、システム・プロセッサーから tty 文字処理をオフロードすることによって向上します。

製品選択の考慮事項

適切な非同期製品は、個々の状態によって異なります。

次の質問は、インストールする必要がある製品を選択するのに役立ちます。

拡張性

どれほどの数の非同期ポートが必要ですか?
将来どれほどの数のポートが必要になりますか?

トポロジー

デバイスを別の建物またはリモート・ロケーションに置きますか?
システム/ネットワーク管理はどこで行いますか?
HACMP クラスターはありますか?
必要な配線のタイプまたは既に使用されている配線のタイプは何ですか?

パフォーマンス

アプリケーションは CPU 集中ですか?
どのようなタイプのデバイスを接続しますか?
デバイスの総計の相対非同期帯域幅要求はどれほどですか?

表 *97.* 相対デバイス帯域幅要求

低要求	**中要求**	**高要求**
ASCII 端末、POS 端末、非同期モデム	プリンター、低速 FAX/モデム、バーコード・スキャナー	シリアル X 端末、高速 FAX/モデム、高速プリンター、ファイル転送アプリケーション

デバイス・インターフェース 要件

どの非同期インターフェースが 必要ですか (EIA 232、EIA 422A、EIA 423 など)?
デバイスまたはアプリケーションで EIA 232 の全インターフェースが 必要ですか?

セキュリティー

システム・アシュアランス・カーネル (SAK) は必要ですか? (プレーナー接続ポートのみ)

次の表は、詳細な製品特性を示しています。

表 *98.* 非同期接続製品の特性

特性	**ネイティブ・シリアル・ポート**	**2 ポート PCI**	**8 ポート PCI**	**128 ポート PCI (RAN あり)**
アダプターごとの 非同期ポート数	該当なし	2	8	128
アダプターの 最大数	該当なし	制限なし	20	20
非同期ポートの最大数	2 または 3	2	160	2560
RAN ごとの非同期ポート数	該当なし	該当なし	該当なし	16
RAN の最大数	該当なし	該当なし	該当なし	160

表 98. 非同期接続製品の特性 (続き)

特性	ネイティブ・シリアル・ポート	2 ポート PCI	8 ポート PCI	128 ポート PCI (RAN あり)
最大スピード (KBit/秒)	UART のボー・レート生成プログラム・クロック速度に基づいて選択可能。	230	230	230
接続方式	プレーナー	direct	direct	ノード
サポートされる非同期エレクトリカル・インターフェース	EIA 232	EIA 232	EIA 232 EIA 422A	EIA 232 EIA 422
標準コネクター	DB9	DB9	DB25M	RJ-45 (10 ピンまたは 8 ピン)
DB25 ケーブル・オプション	該当なし	該当なし	該当なし	RJ-45-DB25
ラック・マウント・オプション	該当なし	該当なし	該当なし	はい
電源装置	該当なし	該当なし	該当なし	外付け
サポートされるシグナル (EIA 232)	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI

非同期通信アダプター・アプリケーション
各製品オファリングは、製品の長所を引き出す代表的なシナリオによって表現されています。 このトピック内のアダプターはそれぞれの 仕様と一緒にリストされているため、各シナリオを選択することができます。

項目	説明
2 ポート PCI バス EIA 232	• PCI スロットが使用可能。 • アダプター 当たり最高 2 ポート。 • すべて EIA 232 のポートが必要。 • 230 Kbps の非同期速度。
8 ポート PCI バス EIA 232/EIA 422	• PCI スロットが使用可能。 • 拡張がほとんどまたはまったくなしで 8 ポート未満。 • すべて EIA 232、すべて EIA 422、または EIA 232 と EIA 422 混用のポートが必要。 • メイン CPU から文字割り込みおよび端末 I/O 処理をオフロードする。 • 230 Kbps の非同期速度。 • 高速 (33.6 Kbps) モデム用の最大パフォーマンス (データ圧縮あり)。

項目	説明
128 ポート・アダプター (PCI)	• Micro Channel、ISA、または PCI バス・スロットが使用可能。(Micro Channel または ISA の詳細については、732 ページの『128 ポート・アダプター (マイクロチャネル、ISA)』を参照。) • 16 ポートは、追加スロットなしで 128 ポートまで拡張可能。 • 最も遠い端末を 230 Kbps の最大データ速度でシステムから約 90 メートル (300 フィート) の位置に配置可能。 • 端末の計画: 構内の近い位置、構内の遠い位置、リモート。 • 低プロセッサー要求で高い非同期スループットが必要。 • 端末接続プリンター機能が必要。 • 光ファイバーまたは同期モデムによるリモート接続が必要。

非同期ソリューションのシナリオ
以下の顧客シナリオは、8 ポート PCI と 128 ポート非同期コントローラーで解決されています。

項目	説明
不動産事務所	• 単純さとコストが最優先。 • オペレーティング・システムおよびサーバー。 • データベースにアクセスするサーバーに 6 から 10 のデバイスを接続。 • 非同期通信に 1 つのスロットが使用可能。 • デバイスとサーバーの距離は 61 メートル (200 フィート) 未満。 **解決策:** 8 ポート PCI。
小売 POS	• 店舗ごとのコストが最優先。 • オペレーティング・システムおよびサーバー。 • 20 以上の ASCII 端末: レジなど。 • 非同期通信に 1 つのスロットが使用可能。 • 端末の追加に備えて将来の拡張が計画されている。 **解決策:** 2 つの RAN のある 128 ポート非同期コントローラー。 将来 RAN を追加して拡張する。

トポロジーの考慮事項

非同期アダプターのファミリーは、距離トポロジーが関係する幅広い選択肢を提供します。

一般にプレーナー接続アダプターおよび直接接続アダプターからの最大ケーブル長は、指定した最大データ速度で作動するポートと非同期デバイスの間の距離です。 128 ポート・アダプターは、アダプター・カードからアダプターに接続されたデイジー・チェーン RAN まで測定されます。 128 ポートでは、EIA 422 同期モデムを使用して RAN をアダプターに接続することによって、効率的に距離の制限をなくすことができます。

正しく配線することが極めて重要であり、配線は環境ごとに固有です。

シリアル通信

ここでは、非同期通信規格、ハードウェア、用語、および概念について説明します。

シリアル・ポートは、非同期デバイスをコンピューターに物理的に接続するために使用します。 シリアル・ポートはシステム装置の背面にあり、内蔵されているか、あるいは 2、8、16、および 128 ポート非同期通信アダプターなどのマルチポート・アダプターを使用します。

注 : POWER5 内蔵システム・ポートは、汎用の全機能搭載型シリアル・ポートではありません。 詳しくは、640 ページの『システム・ポートとシリアル・ポートの異なる機能』を参照してください。

シリアル・ポートの機能を理解するには、まずパラレル通信について考察する必要があります。 標準パラレル・ポートは、8 ピン (ワイヤー) を使用して、単一文字を構成するデータ・ビットを同時に伝送します。 次の図は、文字 a のパラレル伝送を示しています。

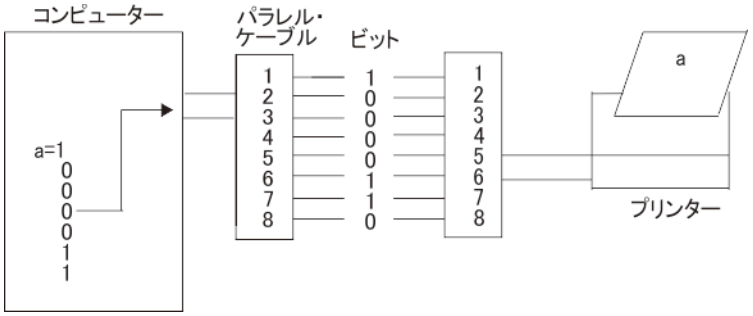


図 *29.* パラレル通信ポート

シリアル・ポートでは、同じデータ文字をデバイスに送信するには単一ピン (ワイヤー) だけが必要です。 これを実現するために、データはパラレル形式 (コンピューターで送信される) から順次形式 (ビットは次々と連続して編成される) に変換されます。 次にデータは、最下位ビット (またはゼロ・ビット) が最初に送信される形でデバイスに伝送されます。 データがリモート・デバイスで受信されると、パラレル形式に変換されます。 次の図は、文字 a のシリアル伝送を示しています。

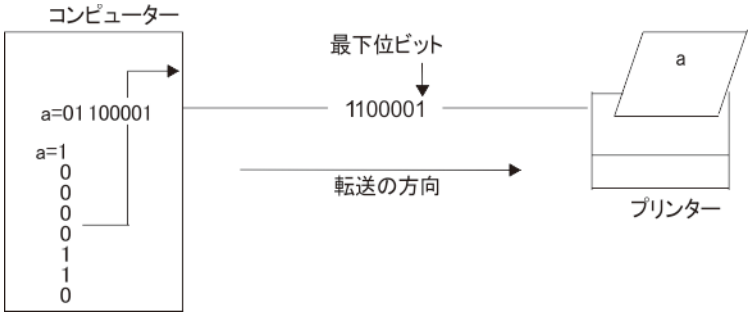


図 *30.* シリアル通信ポート

単一文字のシリアル伝送は単純で分かりやすいものですが、次の図に示すとおり、大量の文字が連続して伝送されると、複雑な状況が発生します。 受信システムは、1 つの文字がどこで終了し、次の文字がどこから始まるかを認識できません。 この問題を解決するには、通信リンクの両端で同期化するかタイミングを合わせる必要があります。

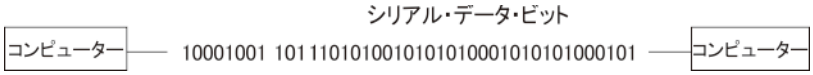


図 *31.* シリアル伝送

システム・ポートとシリアル・ポートの異なる機能

内蔵 POWER5 システム・ポート はシリアル・ポートと類似していますが、システム・ポートは特別にサポートされた機能にのみ使用可能です。

システム・ポートは、ハードウェア管理コンソール (HMC) ポートが HMC に接続されている場合は使用不可です。 HMC ポートまたはシステム・ポートのいずれかを使用できますが、両方は使用できません。

HMC が接続されていない場合でも、内蔵システム・ポートはシリアル接続 TTY コンソール機能に限定されます。 これらのシステム・ポートは、承認済みのコール・ホーム・モデム、非同期端末装置、および特定の UPS にのみ正しく機能します。 その他のシリアル・デバイスを接続するには (HACMP のシステム間の接続を含む)、PCI スロットにシリアル・ポート・アダプターが必要です。

同期
同期は、シリアル伝送のタイミングを合わせて送信されるデータを正しく識別するためのプロセスです。

2 つの最も一般的なモードは、同期と非同期です。

同期伝送
同期 という語は、連続的で整合したタイミングでデータ・ブロックを転送することを表現するために使用されます。

このタイプの接続は、大量のデータをある場所から別の場所へ即座に転送する必要がある場合に使用されます。 同期接続のスピードは、データを個々の文字ではなく大きなブロックで転送することによって達成されます。

データ・ブロックは、グループ化されて規則的にインターバルがあけられ、syn または同期アイドル文字と呼ばれる特殊文字が先頭に置かれます。 次の図を参照してください。

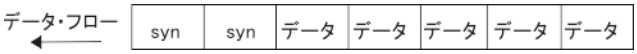


図 *32.* 同期伝送

syn 文字がリモート・デバイスで受信されると、デコードされ、接続を同期化するために使用されます。 接続が正しく同期化されると、データ伝送を開始できます。

このタイプの接続の例として、大容量テキスト文書の伝送があります。 文書が同期回線で転送される前に、文書は文章のブロックまたは段落のブロックに分割されます。 次に、ブロックは通信リンクでリモート・サイトに送信されます。 他の送信モードでは、テキストは、文章または段落内の単語を構成する長い文字ストリング (または文字) に編成されます。 これらの文字は通信リンクで一度に 1 つずつ送信され、リモート・ロケーションで再アセンブルされます。

同期接続に必要なタイミングは、通信リンクに置かれているデバイスから取得されます。 同期リンク上のすべてのデバイスは、同じ刻時に設定しておく必要があります。

同期通信に固有の特性のリストを次に示します。

- 伝送される文字の間にギャップはない。
- タイミングは、接続の両端にあるモデムまたは他のデバイスによって提供される。
- 特殊 syn 文字が、伝送されるデータの先頭に置かれる。
- syn 文字が、タイミング上の理由でデータ・ブロックの間に使用される。

非同期伝送
非同期 という用語は、送信データが各文字の始まりと終わりを指定するスタート・ビットおよびストップ・ビットでエンコードされるプロセスを表現するために使用されます。

非同期伝送の例を次の図に示します。

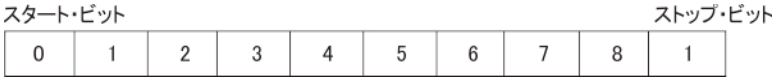


図 *33.* 非同期伝送

これらの追加ビットは、文字が完全に送信または受信されたときを示すことによって接続のタイミングまたは同期を提供します。したがって、各文字のタイミングはスタート・ビットで始まり、ストップ・ビットで終わります。

文字伝送の間にギャップが現れると、非同期回線はマーク状態にあると考えられます。 マークは、次の図に示すとおり、回線が活動化していない期間に送信される 2 進数の 1 (または負電圧) です。

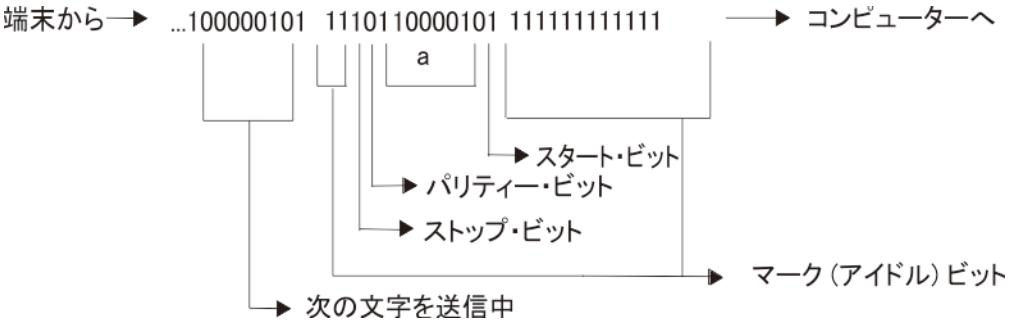


図 *34.* データ・ストリーム内のマーク (アイドル) ビット

マーク状態が正電圧 (2 進数の 0) によって中断されると、受信システムはデータ文字がその後に続くことを認識します。 データ文字の前に置かれるスタート・ビットが常にスペース・ビット (2 進数の 0) で、文字の終わりを示すストップ・ビットが常にマーク・ビット (2 進数の 1) であるのは、このような理由によります。

非同期通信に固有の特性のリストを次に示します。

- 各文字の先頭にスタート・ビットが置かれ、1 つ以上のストップ・ビットが後ろに続く。
- 文字の間にギャップまたはスペースが存在する場合がある。

シリアル通信パラメーター
シリアル通信時に使用されるパラメーターには、1 文字あたりのビット数、ビット/秒 (bps)、ボー・レート、パリティー、スタート・ビット、ストップ・ビット、およびマーク・ビットがあります。

1 文字あたりのビット数
1 文字あたりのビット数 (bpc) は、シリアル通信中に単一データ文字を表すために使用されるビット数を示します。

この数値は、文字に組み込まれるパリティー、ストップ、またはスタート・ビットの合計を反映しているわけではありません。 bpc の 2 つの可能な設定値は 7 と 8 です。

1 文字あたりのビット数の設定に 7 を使用すると、標準 ASCII 文字セットの最初の 128 文字 (0 から 127) だけを送信できます。 これらの各文字は、7 つのデータ・ビットで表現されます。 ASCII 拡張文字セット (128 から 255) を送信するには、1 文字あたりのビット数の設定に 8 を使用する必要があります。 これらの各文字は、8 つのデータ・ビットでのみ表現できます。

ビット/秒 *(bps)*
ビット/秒統計の説明を示します。

ビット/秒 (bps) は、通信回線で 1 秒あたりに伝送されるデータ・ビットの数 (2 進数の 1 と 0) です。

ボー・レート
ボー・レートは、シリアル通信シグナルが 1 秒あたりに状態を変更する回数です。状態は、電圧レベル、周波数、または周波数位相角のいずれかです。

シグナルがデータ・ビットごとに 1 回変更する場合は、1 bps は 1 ボーと等しくなります。 例えば、300 ボー・モデムは、1 秒で 300 回状態を変更します。

パリティー・ビット
パリティー・ビットは、スタート・ビットおよびストップ・ビットとは異なり、オプションのパラメーターで、伝送されるデータ文字がリモート・デバイスで正しく受信されているかどうかを判別するために、シリアル通信で使用されます。

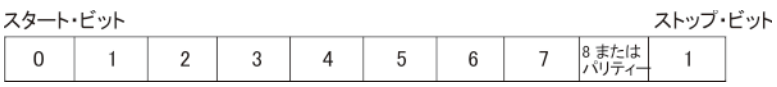


図 *35.* パリティー

パリティー・ビットには、次の 5 つの値のいずれかを指定できます。

項目	説明
none	伝送されるデータ文字のパリティー・ビットをローカル・システムが作成してはならないことを指定します。 また、リモート・ホストから受信されるデータのパリティー・ビットをローカル・システムがチェックしないことも指示します。
even	単一文字で使用される 2 進数の 1 の総数が偶数になることを指定します。 偶数にならない場合は、2 進数の 1 の総数が偶数になるよう、パリティー・ビットは 1 になる必要があります。 例えば、文字 a (2 進数の 1100001) を偶数パリティーで伝送する場合は、送信システムは 2 進数の 1 の数を合計し (この場合は 3)、2 進数の 1 が偶数になるようパリティー・ビットを 1 にします。 文字 A (2 進数の 1000001) が同じ環境で伝送される場合は、2 進数の 1 の総数が偶数になるようパリティー・ビットは 0 になります。
odd	偶数パリティーと同じガイドラインで作動しますが、2 進数の 1 の総数が奇数になります。
space	パリティー・ビットが常に 2 進ゼロになることを指定します。 スペース・パリティーはビット埋め込みとも呼ばれます。この語は、スペース・パリティーが、8 ビット・データしか受け入れられないデバイスに伝送される 7 ビット・データの充てん文字として使用されることに由来します。 このようなデバイスは、スペース・パリティー・ビットを、伝送される文字の追加データと見なします。
mark	スペース・パリティーと同じガイドラインで作動しますが、パリティー・ビットは常に 2 進数の 1 になります。マーク・パリティー・ビットは充てん文字としての機能しか果しません。

スタート、ストップ、およびマーク・ビット

スタート・ビットおよびストップ・ビットは、伝送されるデータ文字のタイミングまたは同期をとる手段として非同期通信で使用されます。

これらのビットを使用しないと、送信および受信システムは 1 つの文字がどこで終わり、次の文字がどこから始まるかを認識できません。

伝送中にデータ文字を分離するために使用される別のビットは、マーク (またはアイドル) RS ビットです。このビット (2 進数の 1) は、通信回線がアイドル状態になっていて、文字が送信または受信されていない場合に伝送されます。

システムがスタート・ビット (2 進数の 0) を受信した場合、これは、スタート・ビットのあとに固定数文字ビット (**bits per character** パラメーターで決定される)、およびパリティー・ビット (**parity** パラメーターで決定される) さえもが続いていると認識されます。 次に、システムはストップ・ビット (2 進数の 1) を受信します。 以下の例では、**parity** ビットが存在し、**bits per character** は 7 となっています。

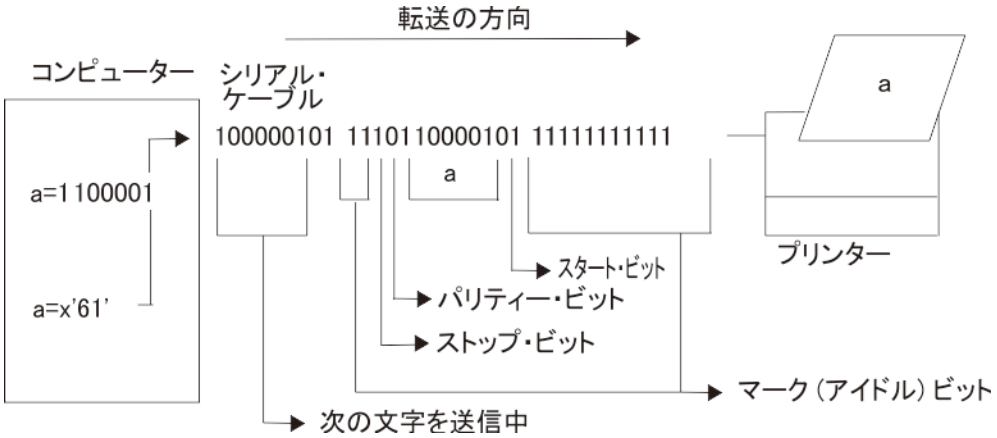


図 *36.* スタート、ストップ、およびマーク・ビット

EIA 232D 規格
EIA 232D 規格は、コンピューターとモデムの間の接続を規定する目的で 1969 年に策定されました。

これは頭字語で、正式には次のようになります。

Electronics Industry Association (EIA) accepted standard, ID number 232 revision D

EIA 232D は、2 つのデバイス間の物理的および電気的な接続の特性を規定しています。 シリアル通信に必要な各ピンまたはワイヤーには名前および省略語が割り当てられています。

表 *99. EIA 232D* 接続

シグナル	装置タイプ	シンボル	ピン
送信データ	DCE	TxD	2
受信データ	DTE	RxD	3
送信要求	DCE	RTS	4
送信可	DTE	CTS	5
データ・セット・レディー	DTE	DSR	6
信号用接地		SG	7
キャリア検出	DTE	CD	8
データ端末レディー	DCE	DTR	20
リング・インディケーター	DTE	RI	22

EIA 232D では、出力にピン 2 (TxD) を使用するデバイス (コンピューターや端末など) には、データ端末装置 (DTE) という名前が付けられます。 入力にピン 2 (TxD) を使用するデバイス (モデムなど) には、データ通信機器 (DCE) という名前が付けられます。

EIA 232D は、コネクターも指定します。 通常、DTE デバイスはオス・コネクターですが、DCE デバイスはメス・コネクターです。 この規格は、製造メーカーによって常に守られるわけではないため、ユーザーはケーブル接続を行う前にデバイスの資料を検討する必要があります。

非同期通信方式
ここでは、単方向と両方向 (半二重と全二重を含む) の 2 つの形式の非同期通信について説明します。

シンプレックスつまり単方向通信は、2 つのデバイス間の最も単純な接続形式です。 この通信モードでは、データは単方向でのみ送信されるため、接続する必要があるのは TxD (または RxD) と SG などの 2 回線だけです。

両方向通信には、半二重と全二重という 2 つの形式があります。 半二重モードの接続では、データが両方向に送信されます (同時ではありません)。 半二重の例として、CB ラジオがあります。これを使用すると、両方向通信は可能ですが一度に 1 人しか話せません。

全二重 (または二重モード) では、データ通信は両方向に同時に行われます。 全二重の例として、双方が同時に会話しているときの電話での会話があります。

フロー制御
シリアル・デバイスでは、システムによって送信されるデータ量を制限するために、ある種のデータ・フロー制御が必要です。

プリンターやモデムなどのシリアル・デバイスは、接続先のコンピューターと同じほど迅速または効率的にデータを処理しません。

フロー制御 という語は、シリアル・デバイスに送信されるデータ量をシリアル・デバイスが制御する方法を表現するために使用されます。

RTS/CTS ハードウェア・フロー

送信要求/送信可 (RTS/CTS) は、フロー制御ではなく、ペーシングまたはハードウェア・ハンドシェークと呼ばれることがあります。

ハードウェア・ハンドシェークという語は、データ伝送制御の手段としてケーブル接続および電圧が使用されることに由来します。 制御文字をデータ・ストリームで送信する XON/XOFF とは異なり、RTS/CTS はデバイス配線の専用ピンまたはワイヤーで正電圧および負電圧を使用します。

正電圧はデータ伝送が許可されることを意味するのに対し、負電圧はデータ伝送を中断すべきことを示します。

DTR/DSR ハードウェア・フロー

ハードウェア・フロー制御の別の形式であるデータ端末レディー (DTR) は、通常はプリンターなどのデバイスによって生成され、システムと通信可能であることを示します。 このシグナルは、データ・フローを制御するために、システムによって生成されるデータ・セット・レディー (DSR) と併せて使用されます。

正電圧はデータ伝送が許可されることを意味するのに対し、負電圧はデータ伝送を中断すべきことを示します。

XON/XOFF ソフトウェア・フロー

送信オン/送信オフ (XON/XOFF) フロー制御には、データ・ストリームでデータ伝送制御文字 (TxD および RxD) を送信することが関係しています。 このような理由で、これはソフトウェア・フロー制御と呼ばれます。

データがモデムに送信されると、データはバッファーに入れられます。 バッファーが最大容量に達する直前に、モデムは XOFF 文字をシステムに送信し、システムはデータの送信を停止します。 モデムのバッファーがほとんど空になってデータを受け入れる準備ができると、モデムは XON 文字をシステムに送信し、データが送信されるようにします。

RTS/CTS ハードウェア・ハンドシェーク用のポートの構成

サーバーに接続された 9600 以上の速度で作動しているモデムでは、XON/XOFF フロー制御ではなく、RTS/CTS ハードウェア・ハンドシェークを使用するようお勧めします。

これにより、リソースが限られているシステムでバッファー・オーバーランを回避できます。 RTS は、どの tty ポートでもデフォルト値ではないため、システム管理者が正しく設定する必要があります。

前提条件

RTS/CTS をサポートするには、少なくとも 5 ワイヤー・ケーブルを使用する必要があります。

ポートで RTS/CTS を使用可能にするには、次のステップを実行してください。

1. `smit tty` 高速パスを使用します。
2. 「**Change/Show Characteristics of a TTY (TTY の特性の変更/表示)**」を選択します。
3. RTS/CTS を使用可能にする tty を選択します。
4. 「FLOW CONTROL to be used (使用されるフロー制御)」フィールドを **rts** に設定します。
5. 「**Do (実行)**」を選択します。
6. SMIT を終了します。

TTY 端末デバイス

tty 端末デバイスは、文字単位で入出力を行うキャラクター型デバイスです。

端末デバイスに読み書きを行うプログラムと端末デバイスとの間の通信は、tty インターフェースによって制御されます。 以下は、tty デバイスの例です。

- モデム
- ASCII 端末
- システム・コンソール (LFT)
- AIXwindows の **aixterm**

SMIT ツール、またはデバイス固有のコマンドを使用することで、システム上のその他のデバイスと同様、tty デバイスの追加、削除、リスト、および変更を行うことができます。

各種のディスプレイおよび端末の TERM 値
端末機能についての情報は、terminfo データベースに保管されます。

TERM 環境変数の値は、terminfo データベース内の特定の端末記述を識別します。 これにより、現在の tty デバイスと効率的に通信するためにプログラムが必要とする情報がすべて提供されます。

表 *100.* 各種の端末の *TERM* 値

ディスプレイ /端末	値
3161 ASCII 端末	ibm3161
3163 ASCII 端末	ibm3161
DEC VT100 (端末)	vt100
DECVT220	vt220
カートリッジ付き 3151 ASCII ディスプレイ ・ステーションまたはカートリッジ付き 3161 ASCII ディスプレイ ・ステーション	ibm3161-C
3162 ASCII ディスプレイ ・ステーション	ibm3161
カートリッジ付き 3162 ASCII ディスプレイ ・ステーション	ibm3162
6091 ディスプレイ	lft
AIXwindows	aixterm

terminfo データベースのエントリーについては、terminfo ファイル・フォーマットを参照してください。 termcap エントリーを terminfo エントリーに変換するには、**captoinfo** コマンドを参照してください。 (termcap ファイルには、旧バークレー・システムの端末記述が保管されています。)

TTY 特性
伝送制御手順 は、コンピューターと非同期デバイスが通信するための、ハードウェアに依存しないユーザー・インターフェースを 提供します。

例えば、ユーザーは、特定の文字シーケンスを入力することによって、1 つの行を消去したり、現在実行中のプロセスに割り込むことができます。 また、**chdev** コマンド、システム管理インターフェース・ツール (SMIT)、または **stty** コマンドを使用して、その文字シーケンスの意味を定義したり、通信速度などその他の端末特性を設定することもできます。

接続済み TTY デバイスの属性要件
ホストと接続済み tty デバイスが正しく通信するためには、以下の要件を満たす必要があります。

- 通信ケーブルが正しく配線されている。
- 接続済み tty デバイスとホストの間の通信値 (回線速度、文字サイズ、パリティー、ストップ・ビット、およびインターフェース) が一致している。

TTY デバイスの管理
ここでは、デバイス管理タスクおよびその関連した SMIT 高速パスとコマンドを示します。

表 *101. TTY* デバイスの管理タスク

タスク	SMIT 高速パス	コマンドまたはファイル
定義済み TTY デバイスのリスト	smit lsdtty	**lsdev -C -c tty -H**
TTY の追加	smit mktty	**mkdev -t tty**[1,2]

表 101. TTY デバイスの管理タスク (続き)		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
TTY の別のポートへの移動 [3]	`smit movtty`	**chdev -l** *Name* **-p** *ParentName* **-w** *ConnectionLocation*[2,4]
TTY の特性の変更/表示	`smit chtty`	**lsattr -l***Name* **-E** (to show); **chdev -l** *Name* (to change)[4,5]
TTY の除去 [3]	`smit rmtty`	**rmdev -l** *Name*
定義済み TTY の構成 (使用可能にする)	`smit mktty`	**mkdev -l** *Name*

注 :

1. 新規 tty デバイスを指定するのに、他のフラグを使用することができます。 次の例は、speed 属性を「19200」に設定し、他の属性を foo ファイルから取り出した値に設定して、8 ポート非同期通信アダプター sa3 上のポート 0 に接続された RS-232 tty デバイスを定義および構成しています。

   ```
   mkdev -t tty -s rs232 -p sa3 -w 0 -a speed=19200 -f foo
   ```

2. **mkdev** コマンドおよび **chdev** コマンドは、SMIT で使用できないオプションをサポートします。
3. このタスクを実行する前に、tty を使用不可にします。 **pdisable** コマンドを参照してください。
4. コマンド・ラインから tty に関する指定特性を変更する場合、フラグを使用します。
5. Posix ボー・レートを「List (リスト)」機能から選択するか、または非 Posix ボー・レートを直接フィールドに入力できます。 選択したボー・レートをモデムのハードウェアがサポートできない場合は、システムによってエラー・メッセージが表示されます。

コマンド・ラインから tty を追加または変更する場合は、次のリストを参照し、設定したい特性に対応する *Attribute* 名を調べて、それを **-a** *Attribute*=*Value* フラグに指定してください。 例えば、tty デバイスのボー・レートを設定するには、`-a speed=Value` と指定します。

表 102. TTY 属性	
特性	**属性名**
Enable LOGIN (LOGIN を可能にする)	login
BAUD rate speed (ボー・レート速度)	speed
PARITY (パリティー)	parity
BITS per character (1 文字当たりのビット数)	bpc
Number of STOP BITS (ストップ・ビット数)	stops
TIME before advancing to next port setting (次のポート設定までの待機時間)	timeout
XON-XOFF handshaking (XON-XOFF 初期接続手順)	xon
TERMINAL type (ターミナル・タイプ)	term
FLOW CONTROL to be used (使用するフロー制御)	flow_disp
OPEN DISCIPLINE to be used (オープン時に使用する回線規約)	open_disp
STTY attributes for RUN time (実行時用の STTY 属性)	runmodes

表 *102. TTY* 属性 *(*続き*)*	
特性	**属性名**
STTY attributes for LOGIN (ログイン時用の STTY 属性)	logmodes
RUN shell activity manager (シェル・アクティビティー・マネージャーの実行)	shell
LOGGER name (ロガー名)	logger
STATUS of device at BOOT time (ブート時のデバイスの状況)	autoconfig
TRANSMIT buffer count (送信バッファー数)	tbc
RECEIVE trigger level (受信トリガー・レベル)	rtrig
STREAMS modules to be pushed at open time (オープン時にプッシュされる STREAMS モジュール数)	modules
INPUT map file (入力マップ・ファイル)	imap
OUTPUT map file (出力マップ・ファイル)	omap
CODESET map file (コード・セット・マップ・ファイル)	csmap
INTERRUPT character (INTERRUPT 文字)	intr
QUIT character (QUIT 文字)	quit
ERASE character (ERASE 文字)	erase
KILL character (KILL 文字)	kill
END OF FILE character (END OF FILE 文字)	eof
END OF LINE character (END OF LINE 文字)	eol
2nd END OF LINE character (2 番目の END OF LINE 文字)	eol2
DELAY SUSPEND PROCESS character (DELAY SUSPEND PROCESS 文字)	dsusp
SUSPEND PROCESS character (SUSPEND PROCESS 文字)	susp
LITERAL NEXT character (LITERAL NEXT 文字)	lnext
START character (START 文字)	start
STOP character (STOP 文字)	stop
WORD ERASE character (WORD ERASE 文字)	werase
REPRINT LINE character (REPRINT LINE 文字)	reprint
DISCARD character (DISCARD 文字)	discard

TTY のトラブルシューティング

一般的な TTY のトラブルシューティング・シナリオは、次のように複数あります。

一般的な TTY のトラブルシューティング・シナリオには、`Respawning Too Rapidly` エラー、ハングした TTY ポート、および共通エラー・ロギング・ファイル、コマンド、エラー報告メッセージがあります。

Respawning Too Rapidly エラー

システムは、短時間に特定の tty 用に生成された **getty** プロセスの数を記録します。 所定の時間枠で生成された **getty** プロセスの数が 5 を超える場合は、コンソールに「Respawning Too Rapidly」エラーが表示され、システムはポートを使用不可にします。

tty は、約 19 分間またはシステム管理者がポートを再び使用可能にするまで使用不可になります。 その 19 分が経過した時点で、システムは自動的にポートを使用可能にします。その結果、新しい **getty** プロセスが作成されます。

考えられる原因は、以下のとおりです。

- モデム構成が正しくありません。
- ポートは定義されて使用可能になっていますが、ケーブルやデバイスがそのポートに接続されていません。
- 配線が正しくないか、確実に接続されていません。
- 通信回線にノイズがあります。
- /etc/environment または /etc/inittab ファイルが破壊されたか、変更されています。
- tty 構成が破壊されています。
- ハードウェアに欠陥があります。

以下のリカバリー手順から、状況に合った手順を使用してください。

- モデム構成が正しくありません。

  モデムのキャリア検出が強制的に high になっていない ことを確認します。

  注 : 以下は、Hayes 互換モデムに適用されます。

  1. モデムに接続し、アクティブ・プロファイルを調べます。
  2. モデムのキャリア検出は、**&C0** (強制的に high) ではなく、**&C1** に設定します。 キャリア属性の設定と変更を行うには、次の AT モデム・コマンドを使用します。

     ```
     AT&C1
     AT&W
     ```

     注 :

     a. 662 ページの『cu コマンドを使用した AT コマンドの送信』 を参照してください。

     b. 詳しくは、使用するモデムの資料を参照。
- tty を使用不可にするか、tty 定義を除去するか、ポートにデバイスを接続します。
  - tty 定義を使用不可にするには、次の **chdev** コマンドを使用します。

    ```
    chdev -l ttyName -a Login=disable
    ```

    このコマンドを実行すると、システムの再始動後も tty は使用可能になりません。
  - tty 定義を除去するには、次のようにします。

    1. tty ポートを使用不可にして、**pdisable** コマンドを使用します。

       ```
       pdisable ttyName
       ```

    2. システムから tty 定義を除去します。 詳しくは、646 ページの『TTY デバイスの管理』 を参照してください。
- ケーブルが不良ではないか、またはしっかり接続されているかどうか検査します。
  1. 配線を検査します。 緩んでいる接続を確実に接続し直し、破損しているコネクターや不適切なコネクターを交換します。
  2. 疑わしいケーブルが IBM シリアル・ケーブル P/N 6323741 であるか、それと同じ規格に従ったケーブルかどうかを検査します。 破損しているケーブルや不適切なケーブルは交換します。

- 通信回線のノイズを除去します。
  1. ケーブルの長さとインピーダンスが正しいかどうかを検査します。
  2. 比較的長いケーブルの必要個所に toroid リングがあるかどうかを確認します。
  3. ケーブルの敷設状況を検査します。蛍光灯やモーターの近くにはケーブルを敷設しないでください。
- `/etc/environment` または `/etc/inittab` ファイルが破壊されたり変更されていないかどうかを検査します。
  1. 可能な場合は、上記のファイルを正しいコピーと照合します。
  2. 上記のファイルをバックアップ用にコピーし、必要に応じて変更します。
  3. `/etc/environment` ファイルから、次の行以外 の行をすべて除去します。
     - ブランク行
     - コメント行
     - *variable=value*
  4. `/etc/inittab` ファイル内の tty デバイス行を調べます。 tty が off に設定されている場合は、tty ポートが使用されていない可能性があります。 tty ポートが使用されていない場合は、tty 定義を除去するか、デバイスをそのポートに接続してください。
- 破壊されている tty 構成を除去します。
  1. tty 定義を除去します。 詳しくは、646 ページの『TTY デバイスの管理』を参照してください。
  2. tty 定義を除去する前にそのハードコピー・レコードが必要な場合は、イメージ・キー (F8 または Esc +8) を押してください。 これにより、現在の画面イメージが取り込まれ、`$HOME` ディレクトリーの `smit.log` ファイルにコピーされます。
  3. tty 定義を読み取ります。 646 ページの『TTY デバイスの管理』 の中の TTY の追加の説明を参照してください。
- 欠陥のあるハードウェアを検出します。
  1. **diag** コマンドを使用して診断を実行します。
  2. ハードウェアに問題がある場合は、各問題の解決手順に従います。

エラー・ログ情報と *TTY* ログ *ID*

TTY に関連したコマンドとロギング・ファイルは、以下のとおりです。

コマンド: **errclear**

このコマンドは、エラー・ログからエントリーを削除します。 `errclear 0` でログ全体を削除できます。また、エラー ID 番号、クラス、タイプなどを指定してエントリーを除去することもできます。

コマンド: **errpt**

このコマンドは、システム・エラー・ログのエントリーからエラー・レポートを生成します。 このコマンドで最もよく使用されるフォーマットは、`errpt -a | pg` です。このフォーマットでは、最新のエラーで始まる詳細レポートが生成されます。

ファイル: `/var/adm/ras/errlog`

このファイルは、システムが検出したエラーと障害のインスタンスを保管します。 `errlog` ファイルは、かなり長くなる傾向があります。 定期的にクリアしないと、ハード・ディスク上の多くのスペースが占有されてしまうことがあります。 このファイルを消去するには、前述の **errclear** コマンドを使用してください。

ファイル: `/usr/include/sys/errids.h`

`errids.h` ヘッダー・ファイルは、エラー ID とエラー・ラベルを相関させます。

TTY に関連した一般的なエラー報告メッセージは、以下のとおりです。

表 *103. TTY* エラー・メッセージ

メッセージ	説明	コメント
Core Dump	ソフトウェア・プログラムが異常終了しました。	このエラーは、ソフトウェア・プログラムが異常終了し、それが原因でコア・ダンプが発生した場合にログに記録されます。 ユーザーがアプリケーションを正しく終了できないか、ユーザーがアプリケーション内で作業しているときにシステムがシャットダウンしたか、ユーザーの端末がロックしてアプリケーションが停止した可能性があります。
Errlog On	Errdaemon がオンになりました。	エラー・ロギングが開始されると、**エラー**・デーモンによってこのエラーが記録されます。 システムは、シャットダウン時にエラー・ロギングを自動的にオフにします。
Lion Box Died	64 ポート集線装置との通信が断たれました。	集線装置との通信が断たれると、64 ポート集線装置ドライバーによってこのエラーが記録されます。 このエラーを受信した場合は、日時スタンプを検査して、ユーザーがこのメッセージの原因かどうか調べてください。 このエラーが連続する場合は、64 ポート・アダプターかその 関連ハードウェアに問題がある可能性があります。
Lion Buffero	バッファーがオーバーランしました (64 ポート集線装置の場合)。	64 ポート集線装置内のハードウェア・バッファーがオーバーランすると、このエラーが発生します。 デバイスと配線が許す限り、送信要求 (RTS) 初期接続手順をポートとデバイスに追加してみてください。 また、ボー・レートも下げてみてください。
Lion Chunknumc	チャンク数が正しくありません (64 ポート・コントローラーの場合)。	チャンク内の文字数の値がバッファー内の実際の値と一致しないと、このエラーが発生します。 ハードウェアに問題があるかもしれないので、デバイスを診断してみてください。
Lion Hrdwre	64 ポート・コントローラーのメモリーにアクセスできません。	64 ポート・コントローラーのメモリーにアクセスできないと、64 ポート集線装置ドライバーによってこのエラーが記録されます。
Lion Mem ADAP	メモリーの割り当てができません (ADAP 構造の場合)。	adap 構造用の `malloc` ルーチンが失敗すると、64 ポート集線装置ドライバーによってこのエラーが記録されます。

表 *103. TTY* エラー・メッセージ *(*続き*)*

メッセージ	説明	コメント
Lion Mem List	メモリーの割り当てができません (TTYP_T リストの場合)。	*ttyp_t* リスト構造用の `malloc` ルーチンが失敗すると、64 ポート集線装置ドライバーによってこのエラーが記録されます。
Lion Pin ADAP	メモリーのピン割り当てができません (ADAP 構造の場合)。	adap 構造用の `pin` ルーチンが失敗すると、64 ポート集線装置ドライバーによってこのエラーが記録されます。
SRC	ソフトウェア・プログラム・エラーが発生しました。	ある異常条件が発生すると、システム・リソース・コントローラー (SRC) のデーモンによってこのエラーが記録されます。 異常条件は、サブシステム 障害、通信障害、その他の障害の 3 領域に分類されます。
Lion Unkchunk	64 ポート集線装置から未知のエラー・コードが返りました。	エラー・コード: 受信されたチャンク内の文字数
TTY Badinput	配線が正しくないか、確実に接続されていません。	システムが処理できるよりも高速でポートが入力を生成しており、入力の一部が廃棄されています。 通常、1 つ以上の RS-232 シグナルが短期間にその状況を頻繁に繰り返し変更したために、システムが割り込みハンドラーの中で多くの時間を費やすことになり、そのために、正しくない入力が発生します。 シグナル・エラーは、通常、コネクターが確実に接続されていないか、壊れている (ケーブルが不良である、アース接続されていない、非シールド・ケーブルである) ことが原因で発生します。 あるいは、「ノイズが多い」通信リンクによっても発生します。

表 *103. TTY* エラー・メッセージ *(*続き*)*

メッセージ	説明	コメント
TTY Overrun	入力時に受信装置がオーバーランしました。	ほとんどの TTY ポートは 16 文字の入力 FIFO を持ち、デフォルトの設定値では、14 文字を受信したあとで割り込みが発生するように指定されています。 このエラーは、ドライバー割り込みハンドラーが入力 FIFO をクリアして、FIFO データが失われたときに報告されます。 解決策は使用しているハードウェアによって異なります。 • 8 ポートおよび 128 ポート・アダプターの場合 フロー制御が正しく構成されているか検査します。 正しく構成されていれば、診断を実行して、適切なハードウェアと交換します。 • ネイティブ・ポートの場合 問題がアイドル中のシステムで発生したのであれば、ワークロードを別のポートに移動します。 これで問題がなくなるようであれば、システムのファームウェアをアップグレードします。 • 一般的な解決策 – このポートの「RECEIVE trigger level (受信トリガー・レベル)」パラメーターを、3 から、2 または 1 に減らします。 – このポートの回線速度を下げます。 – システムの割り込み使用不可の時間を少なくできないか、他のデバイスとプロセスを調べます。
TTY TTYHOG	TTYHOG がオーバーランしました。	このエラーは、通常、使用するフロー制御方式が、送信側と受信側の間でマッチしていないために発生します。 TTY ドライバーは、送信側に休止を何度か要請しますが、入力が停止されず、そのためにデータが廃棄されてしまいます。 送信側と受信側で構成されているフロー制御メソッドを調べて、それぞれで同じメソッドが使用されていることを確認します。

表 103. TTY エラー・メッセージ (続き)		
メッセージ	**説明**	**コメント**
TTY Parerr	入力上にパリティー/フレーム・エラーがあります。	このエラーは、非同期ポートへの着信データに文字単位のパリティー・エラーが発生したことを示します。 これは、通常、送信側と受信側の間で、回線制御のパラメーター (パリティー、回線速度、文字サイズ、またはストップ・ビット数) が マッチしていないために発生します。 回線制御のパラメーターは、通信を行う両端で同じに設定されていなければなりません。
TTY Prog PTR	ドライバー内部エラー	*t_hptr* ポインターが null の場合、tty ドライバーによってこのエラーが記録されます。

ハングした *TTY* ポートのクリア

このハングしたポートのクリア例では、ハングした tty ポートが `tty0` であると想定します。

この手順を完了するには、root 権限が必要です。

1. 次のように入力して、現在、tty がプロセスを処理しているかどうか判別します。

   ```
   ps -lt tty0
   ```

   次のような結果が戻されます。

   ```
        F S UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
   240001 S 202 22566  3608   0  60 20 781a   444 70201e44   tty0  0:00 ksh
   ```

   ここでのプロセス ID (PID) は 22566 です。 このプロセスを強制終了するには、次のように入力します。

   ```
   kill 22566
   ```

   コマンド `ps -lt tty0` を入力して、プロセスが正常にクリアされたことを確認します。 プロセスがまだ存在している場合は、次の例に示すように、kill コマンドに `-9` フラグを追加します。

   注 : slattach プロセスの強制終了には、`-9` オプションを使用しないでください。 slattach プロセスを `-9` フラグを付けて強制終了すると、SLIP ロックが `/etc/locks` ファイル内に 残る可能性があります。 slattach のあと、クリーンアップのためにこのロック・ファイルを削除してください。

   ```
   kill -9 22566
   ```

2. 次のように入力して、tty を使用しようとしているプロセスがあるか判別します。

   ```
   ps -ef | grep tty0
   ```

   注 : `ps -ef | grep tty` コマンドによって、次のような表示が戻された場合、

   ```
   root 19050      1    0    Mar 06       -  0:00 /usr/sbin/getty /dev/tty
   ```

   この tty は正しいケーブルに接続されていません ("-" は、日付 (`Mar 06`) と時刻 (`0:00`) の間に表示される)。 この状況は、システム・ログイン・プロセス (getty) がこの tty をオープンしようとして 、RS-232 シグナルがデータ・キャリア検知 (DCD、Data Carrier Detect) の状態にならなかったために オープン・プロセスがハングしていることを示します。 この問題は、配線に正しい null モデム・アダプターを 使用することで修正することができます。 getty が tty ポートをオープンできるときは、"-" の部分は tty 番

号で置き換えられます。 配線の詳細については、661 ページの『適切なケーブルを使用したモデムの接続』を参照してください。

注: 次のコマンドを使用して、tty0 上のログイン・プロセスを使用不可にできます。

```
pdisable tty0
```

プロセスが正常にクリアされても、まだ tty が応答しないようであれば、次のステップに進みます。

3. 次のコマンドを入力します。

```
fuser -k /dev/tty0
```

これは、ポート上で実行中であることが分かったプロセスをクリアして、PID を表示します。 tty がまだ使用可能でなければ、次のステップに進みます。

4. **strreset** コマンドを使用して、リモート・エンドへの接続が切れたために送達できないデータが原因で、ハングしたポートからの発信データをフラッシュします。

注: **strreset** コマンドによってハングしたポートの問題が修正された場合、リモート・エンドへの接続が切れたことにより、バッファーに入っているデータが自動的にフラッシュされるため、ポートには配線または構成の問題があります。

次のように入力して、まず、tty のメジャーおよびマイナー・デバイス番号を判別する必要があります。

```
ls -al /dev/tty0
```

次のような結果が表示されます。

```
crw-rw-rw-    1 root     system    18,   0 Nov  7  06:19 /dev/tty0
```

これは、tty0 のメジャー・デバイス番号が 18 で、マイナー・デバイス番号が 0 であることを示しています。これらの番号を、次のように、**strreset** コマンドを使用するときに指定します。

```
/usr/sbin/strreset -M 18 -m 0
```

tty がまだ使用可能でなければ、次のステップに進みます。

5. 停止した tty ポートからケーブルを切り離し、それから再接続します。 AIX は、データ・キャリア検知 (DCD、Data Carrier Detect) シグナルを使用して、ポートに接続されたデバイスが存在するかどうか判別します。

DCD をドロップし、ケーブルを切り離して再接続することによって、多くの場合、ハングしたプロセスをクリアできます。

tty が構成されているポートの場所を判別するには、次のコマンドを入力します。

```
lsdev -Cl tty0
```

次のような結果が表示されます。

```
tty0    Available   00-00-S1-00  Asynchronous Terminal
```

この出力の 3 番目の列は、tty のロケーション・コードを示しています。 この例の S1 は、シリアル・ポートがネイティブ・シリアル・ポート 1 に構成されていることを示します。 ロケーション・コードの解釈については、../devicemanagement/devloccodes.html (オペレーティング・システムおよびデバイスの管理) を参照してください。

tty がまだ使用可能でなければ、次のステップに進みます。

6. **stty-cxma** を使用して、ポートをフラッシュします。 次のように入力します。

```
/usr/lbin/tty/stty-cxma flush tty0
```

このコマンドは、8 ポートおよび 128 アダプターのポートに構成された tty 用です。 しかし、場合によっては、他の tty ポートを使用して、正常にフラッシュすることもできます。

tty がまだ使用可能でなければ、次のステップに進みます。

7. ハングしている端末のキーボードで、Ctrl キーを押したまま Q を押します。これによって **Xon** 文字が送信され、中断されていた出力が再開されます。

   tty がまだ使用可能でなければ、次のステップに進みます。

8. プログラムは、tty ポートをオープンし、属性をいくつか変更したあと、それらの属性を元の状態にリセットしないでクローズしてしまうことがあります。 これを訂正するために、tty を DEFINED 状態に下げ、次を入力して tty を使用可能にします。

   ```
   rmdev -l tty0
   ```

   このコマンドは、tty に関する情報をデータベースに残しますが、システム上で tty を使用不可にします。

   次のコマンドは、tty を再びアクティブ化します。

   ```
   mkdev -l tty0
   ```

   tty がまだ使用可能でなければ、システムがリブートできるようになるまで、デバイスを別のポートに移動して、その場所で tty を構成することを考えます。 リブートしてもポートがクリアされなければ、ほぼハードウェアの障害です。 次を入力して、ポートにハードウェア障害がないかエラー・レポートをチェックしてください。

   ```
   errpt -a | pg
   ```

これまでに挙げたコマンドの内、いくつかは機能せずに、デバイスが使用中であることを示すメソッド・エラーを戻すことがあります。 これは、プロセスが tty 上で実行中であるためです。 上述したステップではハングした tty をフリーにできなければ、最後の手段として、AIX システムをリブートして、カーネルをフラッシュしてください。そうするとプロセスがクリアされます。

モデム

モデムは、通常の電話回線を介したシリアル通信を可能にします。 モデムの概念には、規格、一般的なモデムのセットアップ、よく使用されるモデムの具体的な構成に関するヒントがあります。

モデム とは、通常の電話回線を介して、あるコンピューターを別のコンピューターに接続するためのデバイスです。 現在の電話システムでは、直接デジタル接続に必要な電圧の変化を伝えることができません。モデムは、電話回線を介して送信するためにデジタル情報を音声のトーンに変調し、受信時にそのトーンをデジタル情報に復調することによって、この制限を克服します。 モデムは一般に、基本ネットワーク・ユーティリティー (BNU) または他の形の UNIX 間コピー・プログラム (UUCP) とともに使用されます。 高速 (14,400 bps 以上) のモデムをシリアル回線インターフェース・プロトコル (SLIP) とともに使用すると、伝送制御プロトコル/インターネット・プロトコル (TCP/IP) 接続も可能になります。

モデムの速度を示すのに、bps の代わりにボー という用語がよく使用されます。 ボーとは、実際には変調率の測定単位です。 旧型モデムでは、シグナルの変換ごとに 1 ビットしかエンコードされなかったので、モデムのボー・レートとモデムの速度が同じでした。 しかし、より高速で動作するモデムでも一般に 2,400 (または 1,200) ボーで動作し、1 シグナル変換当たり複数ビットをエンコードします。 モデムの bps レートは、1 シグナル当たりのデータ・ビット数にボーを掛けて算出します (例えば、2,400 ボー x 1 シグナル変換当たり 6 ビット = 毎秒 14,400 ビット)。 ほとんどの最近のモデムはさまざまなスピード (例えば、28,800、14,400、9,600、7,800、4,800、および 2,400 bps など) で通信できます。

通信規格

従来の 300、1,200、2,400 bps という速度は、明確に定義されていました。 しかし、モデムのメーカーがより高速にするメソッドを考案し始めるにつれて、各社は、他社のモデムと互換性のない専用のメソッドを使用するようになりました。 現在、ITU-TSS (以前の国連国際電信電話諮問委員会、略語は CCITT) は、ほとんどの高速通信に関する規格を定義しています。

高速モデムであっても、コンピューター通信の他のメソッドよりははるかに低速です。 高速モデムは 28,800 bps で動作しますが、イーサネット接続は 10,000,000 bps で動作します。 データのスループットを高めるために、高速モデムには、通常、1 つ以上のデータ圧縮アルゴリズムがあります。 これらのアルゴリズムによって、高速モデムのスループットを 、57,600 bps (データ転送速度が 14,400 bps の場合) ま

たは 115,200 bps (データ転送速度が 28,800 bps の場合) の速度まで高めることができます。 これらの圧縮アルゴリズムでは、転送されるデータによってスループットが変化することに注意してください。 データがあらかじめ圧縮されている場合 (例えば、**compress** コマンドによって)、高速モデムのデータ圧縮メソッドには利点がなく、あってもごくわずかであり、データ・スループットを低下させることさえあります。 データ圧縮テクノロジーがインプリメントされているモデムを使用する場合は、コンピューターとモデム間のデータ端末装置/データ回線終端装置 (DTE/DCE) 接続のスピードは、モデム間接続における公称データ速度と等しいか、それより大きくなります。 例えば、V.42*bis* データ圧縮機能のある V.32*bis* モデムの場合、モデムのデータ転送速度 (電話回線を介してモデムが通信する速度) は 14,400 bps です。 V.42*bis* 圧縮が機能していれば、実際のデータ・スループットが 57,600 bps に達することがあります。 データ圧縮によって提供されるより大きいスループットに対応するために、コンピューターとモデム間のリンクの速度を 57,600 bps に設定してください。

ITU-TSS では、データ圧縮アルゴリズムなどの高速通信のための規格を定義しています。 ITU-TSS は、一般に V.*nn* と指定されます。この *nn* は番号です。 また、やや一般的ではない別の規格として、Microcom Networking Protocol (MNP) があります。 MNP は、バージョン (クラスと呼ばれる) 1 から 9 が使用可能です。この規格は、比較的早期に使用できるようになった高性能の高速プロトコルで、ITU-TSS 規格が登場するまでは事実上の標準のようになっていました。

全二重および半二重伝送

通信規格について学ぶ際は、半二重伝送と全二重伝送の違いについて理解することが重要です。

半二重 (HDX) 伝送では、データ・パケットは一方のシステムから送信され、もう一方のシステムで受信されます。 受信システムが確認応答を送信側に送信するまで、別のデータ・パケットを送信できません。

全二重 (FDX) 伝送では、送信システムと受信システムの両方が互いに同時に通信します。言い換えると、両方のモデムが同時にデータを送受信できます。 モデムは、別のデータ・パケットの受け取りを認知しながら、データ・パケットを受信できるという意味です。

ITU-TSS 通信規格

ここでは、ITU-TSS が定義した一般的な通信規格について説明します。

これらは通信規格の一部ですのでご注意ください。 完全なリストについては、国際電気通信連合 (ITU) のインターネット Web サイトをご覧ください。

項目	説明
V.29	半二重 9,600 bps 通信のための ITU-TSS 規格。
V.32	全二重 9,600 bps 通信のための ITU-TSS 規格。
V.32*bis*	14,400 bps 通信のための ITU-TSS 規格。 V.32*bis* は、V.32 規格の改訂版です。
V.34	33,600 bps 通信のための ITU-TSS 規格。 この規格では、MNP クラス 9 で使用していたデータ圧縮方式の代わりに、複数ビットのエンコード方式を使用することによって 33,600 bps のデータ転送速度を実現することに注意してください。この規格は、以前は V.*fast* と呼ばれていました。
V.42	非同期から同期への変換を使用した、DCE 用の ITU-TSS エラー訂正プロシージャー。
V.42*bis*	ITU-TSS のデータ圧縮規格の改訂版。

Microcom Networking Protocol

もう 1 つの事実上の業界標準として **Microcom Networking Protocol (MNP)** があります。これは、当初は Microcom, Inc. によって開発されました。

MNP は、バージョン (クラスと呼ばれる) 1 から 9 が使用可能です。この規格は、ITU-TSS 規格が登場する前に使用可能だった高性能の高速プロトコルです。 **MNP** を使用すると、送信データ・パケット内のエラーはリモート・モデムによって検出され、エラーのあるデータ・パケットの再送が要求されます。 データ・エラーを認識して迅速に訂正できるため、**MNP** は今日の最も一般的なプロトコルの 1 つとなっています。

MNP 通信規格を以下の表に示します。

項目	説明
MNP クラス 1	約 70% の効率を実現する非同期、半二重、バイト指向のデータ転送メソッド。 この規格は、最近のモデムではあまり採用されていません。
MNP クラス 2	**MNP クラス 1** の全二重版。これも最近のモデムではあまり採用されていません。
MNP クラス 3	約 108% の効率を実現する同期、ビット指向、全二重のデータ転送メソッド。 非同期接続に必要なスタート/ストップ・ビットが除去されているので、100% を超える効率が実現されます。 モデムとシステム 間の DTE/DCE は引き続き非同期です。
MNP クラス 4	パケット・サイズを変更する機構 (最適パケット・アセンブリー) と、冗長な管理オーバーヘッドを除去する手段 (データ・フェーズの最適化) を含む、**MNP クラス 3** の拡張版。 **MNP クラス 4** のモデムでは、約 120% の効率が実現されます。
MNP クラス 5	クラス 4 の機能とともにデータ圧縮機能が含まれています。 **MNP クラス 5** のモデムでは、200% の効率が実現されます。
MNP クラス 6	複数の非互換変調技法を 1 台のモデムに組み込むことができます (汎用リンク・ネゴシエーション)。 これにより、**MNP クラス 6** のモデムは、低速で通信を開始し、高速への状態遷移を交渉できます。 また、**クラス 6** には、半二重変調の使用効率を動的に割り当て全二重サービスをシミュレートする、統計二重化方式も含まれています。 **MNP クラス 5** の全機能がサポートされます。
MNP クラス 7	拡張データ圧縮機能が組み込まれています。 クラス 4 と組み合わせて 300% の効率を実現できます。
MNP クラス 8	該当しません。
MNP クラス 9	拡張データ圧縮機能を V.32 の機能と併用することにより、最高 28,800 bps のデータ転送速度を実現できます。

モデムの考慮事項
一般ユーザー向けのモデム・インターフェース 要件は異なることがあります。

このオペレーティング・システムに接続されるモデムの構成は、パーソナル・コンピューター (PC) またはワークステーションに接続されるモデムとは異なります。

サポートされるモデム
EIA 232 に準拠していて、コマンドに応答して結果を戻すことができるモデムであれば、どのモデムでもこのオペレーティング・システムに接続できます。

データ・キャリア検知の処理
サーバーは、データ・キャリア検知 (DCD、Data Carrier Detect) シグナルを使用して、モデムの状態をモニターします。

モデム・ポートの DCD シグナルが「high」の場合、サーバーはモデムが使用中であると見なします。 したがって、どのような事情で、このシグナルが強制的に「high」状態になるかを知っておくことが重要です。 DCD シグナルは、次の理由で high になります。

- 「**TTY Configuration (TTY 構成)**」パネルで、ランタイム・フィールドの stty 属性に `clocal` を使用する。
- SMIT「**TTY Configuration (TTY 構成)**」パネルの「Ignore Carrier Detect (キャリア検知を無視)」フィールドで、128 ポート・アダプターに接続されている ttys に「`enable` (使用可能)」に設定してある。
- AT コマンドまたはスイッチでモデムが DCD high を強制する。
- tty ポートがアプリケーションによって既に使用されている。

  注 : モデムが別のモデムと接続すると、モデムは DCD を発します。 ほとんどのモデムのデフォルト設定では、モデムがアイドルの場合でも、このシグナルは常に「high」に設定されています。 DCD を強制的に「high」にしないでください。

データ端末装置またはデータ回線終端装置の速度

2 つの異なるハードウェア・グループを表現するために、データ端末装置 (DTE) およびデータ通信装置 (DCE) が使用されます。

DTE という語は、ユーザー情報を表示するデバイスに主に使用されます。 これには、ユーザー用のデータを保管または生成するデバイスも含まれます。 システム装置、端末、およびプリンターはすべて DTE カテゴリーに属します。

DCE には、通信回線によってシステムにアクセスするために使用できるデバイスが含まれます。 DCE の最も一般的な形式はモデムおよびマルチプレクサーです。

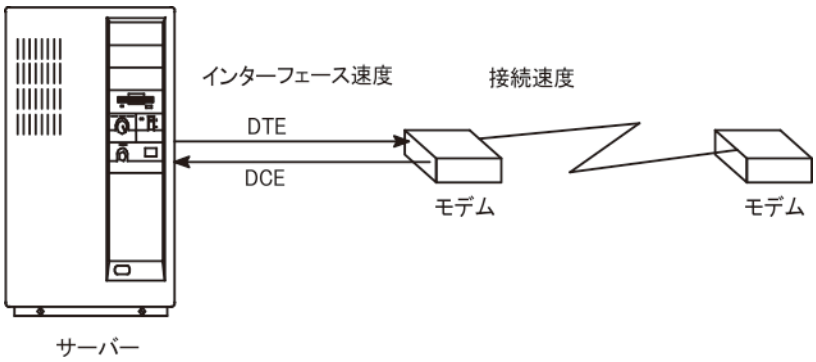


図 *37.* モデム・スピードの考慮事項

このオペレーティング・システムでのモデムが関係するシリアル通信 (上記の図) については、次の 3 つの主な考慮事項があります。

- DTE インターフェース 速度 (サーバーからモデム)。 これは、サーバーがモデムと通信する速度です。
- DCE インターフェース 速度 (モデムからサーバー)。「シリアル・ポート・インターフェース 速度」と呼ばれることもあります。 これは、モデムがサーバーと通信する速度です。
- 接続速度 (モデムからモデム)。 これは、モデムが別のモデムと通信 (対話) する速度です。

最新の高速モデムでは、DCE インターフェース 速度と接続速度を異ならせることができます。 このため、DTE 速度を単一のボー・レートにロックする一方で、モデム間の通信を正常に行えるよう、接続速度を必要に応じて上げたり下げたりできます。

最新式の高速モデムは、サーバーに送信するデータをバッファーに保持し、システムで受け入れられるようになったときにデータを送信します。 また、他のモデムに送信するデータをバッファーに保持し、相手側で受け入れられるようになったときにデータを送信します。 この種のデータ伝送では、モデムおよびサーバーがフロー制御 に参加する必要があります。

モデム制御シグナル

モデムは、呼び出しを開始および受信するためによく使用されます。 したがって、可能な限り速い速度の接続を取り決め、接続の停止後に既知の状態リセットするよう、モデムをプログラムすることが重要です。

サーバーは、データ端末レディー (DTR) シグナルをオンからオフに切り替えて、接続を終了するようモデムに指示します。 ほとんどのモデムは、オンからオフの DTR 変位が発生した場合にリセットするよう構成できます。

注 : stty ランタイム属性の hupcl フラグを使用不可にすることによって、DTR をドロップしないよう tty を構成できます。

サーバーとモデムの間の接続を完全に機能させるには、配線は次の要件を満たしている必要があります。

- 仕様を満たしている必要がある。
- 正しくシールドされている必要がある。
- RxD、TxD、RTS、CTS、SG、DCD、および DTR シグナルが提供される必要がある。

  注 : 16 ポート非同期通信アダプターは、RTS および CTS シグナルをサポートしません。 したがって、このアダプターで RTS/CTS ハードウェア・フロー制御を使用することはできません。

このアダプターでモデムを 使用してバイナリー・データを転送する場合は、誤ったデータを検出して、欠落データを再送するファイル転送プロトコル (Xmodem、zmodem、Kermit、および UUCP など) を使用する必要があります。

次に、サーバーで使用されるシグナルについて説明します。

シグナル **説明**

FG フレーム接地。 ケーブル・シールドを提供する EIA 232D 仕様のピン 1。 正しく使用すると、シグナルはケーブルの 一方の端のピン 1 に付加され、ケーブルの周囲の金属の覆いに接続されます。

TxD 送信データ。 EIA 232D 仕様のピン 2。 データはこのシグナルで送信されます。 サーバーによって制御されます。

RxD 受信データ。 EIA 232D 仕様のピン 3。 データは、モデムによって送信され、モデムによって制御される、このシグナルで受信されます。

RTS 送信要求。 EIA 232D 仕様のピン 4。 RTS/CTS フロー制御が有効になっている場合に使用されます。 このシグナルは、システムでデータを送信する準備ができると high になり、システムがモデムにデータの送信を停止させたい場合にドロップされます。

CTS 送信可。 EIA 232D 仕様のピン 5。 RTS/CTS フロー制御が有効になっている場合に使用されます。 このシグナルは、モデムでデータを送信または受信する準備ができると high になります。 モデムがサーバーにデータの送信を停止させた場合にドロップされます。 モデムによって制御されます。

DSR データ・セット・レディー。 EIA 232D 仕様のピン 6。 モデムが使用可能な状態にあることをサーバーに伝えます。 モデムによって制御されます。

SG 信号用接地。 EIA 232D 仕様のピン 7。 このシグナルは、他のシグナルに基準電圧を提供します。

DCD データ・キャリア検出。 EIA 232D 仕様のピン 8。 これは、モデムが別のモデムと接続されていることをサーバーに伝えます。 このシグナルが high になると、サーバーで実行されているプログラムがポートを開くことができるようになります。 モデムによって制御されます。

DTR データ端末レディー。 EIA 232D 仕様のピン 20。 これは、サーバーが立ち上がっていて、接続を受け入れる準備ができていることをモデムに伝えます。 このシグナルは、サーバーがモデムに別のモデムとの接続をドロップさせたい場合にドロップされます。 ポートが開かれると、high になります。 サーバーによって制御されます。

RI リング表示。 EIA 232D 仕様のピン 22。 これは、モデムが呼び出しを受信していることをサーバーに伝えます。 これはまれにしか使用されず、一般の操作では不要です。 モデムによって制御されます。

モデム配線

次の表は、モデムを任意のシリアル・コントローラーに正しく接続するために必要なケーブル情報の要約を示しています。

アダプター/コントローラー	**IBM 部分番号**
ネイティブ・シリアル (S1 または S2)	00G0943*, 6326741
2 ポート・コントローラー	00G0943*, 6326741
8 ポート・コントローラー	6323741
128 ポート・コントローラー	43G0935, 6323741

IBM 部品番号	**説明**	**長さ (フィート)**
00G0943*	シリアル・ポート・ジャンパー (ピッグテール)	.33

IBM 部品番号	説明	長さ (フィート)
6323741	非同期	10
43G0935	RJ-45 から DB25 へのコンバーター・ケーブル	2

*一部のマシン・タイプでは、この部品番号は不要です。

オペレーティング・システム上の TTY デバイスのセットアップ

SMIT を使用して、デバイス接続の TTY ポートを定義します。

ほとんどのフィールドは、汎用のデバイス・タイプ用です。 モデムに影響を及ぼす可能性のある唯一のフィールドは「Enable LOGIN (ログインを可能にする)」フィールドで、次の値が入ります。

項目	説明
DISABLE (使用不可)	ポート上で getty プロセスが実行されていません。 この設定は、ダイヤルアウトのみのモデム・ポートに使用します。
ENABLE (使用可能)	ポート上で getty プロセスが実行されています。 この設定は、ダイヤルインのモデムにのみ使用します。
SHARE (共有)	getty プロセスがポート上で実行されています。しかし getty プロセスは、手動で「使用不可」または「使用可能」に変更しなくても プログラムがこのポートのダイヤルインとダイヤルアウトをできるようにします。 この設定は、両方向のポートにのみ使用します。
DELAY (遅延)	getty は、両方向モードのポートで実行されます。しかし getty プロセスがユーザーから キー・ストロークを受け取るまではヘラルドを送信しません。

128 ポート非同期通信アダプターに特定のフィールド:

項目	説明
Force Carrier or Ignore Carrier Detect (キャリア強制またはキャリア検知の無視)	disable (使用不可)*
Perform Cooked Processing in Adapter (アダプターでの 加工処理の実行)	disable (使用不可)

注 : アスタリスク (*) で示されたこの設定は、10 ピン RJ-45 コネクターが使用されているときは「使用不可」に設定されます。 この設定は、8 ピン RJ-45 コネクターが使用されている場合は、「使用可能」に設定します。

適切なケーブルを使用したモデムの接続

モデムをセットアップするには、まず適切なケーブルでモデムを 接続します。

部品番号とその説明を次に示します。

6323741
すべての非同期デバイスの接続に使用します。他のケーブル部品とともに使用する場合もあります。

59F3740
10 ピン/25 ピン変換 D シェル・コネクター。次の図に示すように、非同期ケーブル 6323741 を固有シリアル・ポートの S1 と S2 に接続するために使用します。

図 *38. 10* ピン*/25* ピン変換コネクター

この図には 10 ピン/25 ピン変換コネクターが示されています。

ケーブルの接続例を次に示します。

1. モデムを固有のシリアル・ポート S1 に接続するには、次のケーブルを使用します。

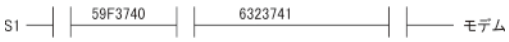


図 *39.* モデムから固有シリアル・ポートへのケーブル・アセンブリー

この図には、シリアル・ポート側の終端の 59F3740 ケーブルと、モデム側の終端の 6323741 が示されています。

2. モデムを 8 ポート非同期アダプター (EIA-232) インターフェース ・ケーブル部品に接続するには、次のケーブルを使用します。

図 *40.* モデムへの *8* ポート・インターフェースのケーブル ・アセンブリー

この図には、6323741 ケーブルでモデムに接続される 8 ポート・インターフェースが 示されています。

モデム用の TTY の追加
モデム用の TTY を追加する場合は、以下の情報を使用してください。

まず、システムの電源はオン、モデムの電源はオフになっていることを確認します。 SMIT 高速パス `smit mktty` を使用します。

モデム構成
モデムを構成する際には、ここで示されている 2 つの方法のどちらか一方だけを使用してください。

基本ネットワーク・ユーティリティー (BNU) をインストールしている場合は、662 ページの『cu コマンドを使用した AT コマンドの送信』を 参照してください。 BNU をインストールしていなければ、663 ページの『C プログラムを使用した AT コマンドの送信』を 参照してください。 BNU のインストールの詳細については、489 ページの『基本ネットワーク・ユーティリティー 』を参照してください。

cu コマンドを使用した *AT* コマンドの送信
基本ネットワーク・ユーティリティー (BNU) がインストールされている場合は、次のように **cu** コマンドを使用してモデムを構成してください。

このセクションで説明するコマンドと設定により、サーバーのシリアル・ポート上の操作に必要な基本パラメーターで Hayes 互換モデムが構成されます。

1. **/usr/lib/uucp/Devices** ファイルに次の行を追加します。 この行が既にファイルにある場合は、追加しないでください。 (# は、実際のポート番号に置き換えてください。)

```
Direct tty# - Any direct
```

2. 次を入力して、tty が使用不可であることを確認します。

```
pdisable tty#
```

3. 次のコマンドを入力します。

```
cu -ml tty#
```

「Connected」というメッセージが表示されます。

4. 次を入力して、モデムのアテンションがあることを 確認します。

```
AT
```

モデムは「OK」と応答します。 そうでない場合は、665 ページの『 モデムのトラブルシューティング 』を参照してください。

この他の AT コマンドとその説明は、667 ページの『AT コマンド』を参照してください。

5. 選択した getty オプションに応じて、次のコマンドの 1 つを入力します。 *n* は tty デバイスに置き換えてください。
   - `penable tty`*n*
   - `pshare tty`*n*

- pdelay tty*n*
- pdisplay tty*n*

これでモデムには、ほとんどのオペレーティング・システムのシリアル通信に必要な基本コマンドが構成されました。 問題があれば、**cu -dl** コマンドを呼び出して、接続の診断トレースを開始してください。

C プログラムを使用した *AT* コマンドの送信

cu コマンドを使用してモデムを構成できなかった場合、または BNU をインストールしていない場合は、次の C プログラムを実行してください。

以下のコードが入っている motalk.c というファイルを作成します。 ファイルを保存します。 プログラムのコメント内の指示に従ってプログラムをコンパイルし、実行します。

```
/*************************************************************/
/*  MoTalk - A "C" program for modem setup.                  */
/*           This program is meant as an aid only and is     */
/*           not supported by IBM.                          */
/*                  compile:  cc -o motalk motalk.c          */
/*                  Usage:  motalk /dev/tty? [speed]         */
/*************************************************************/
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
#include <termio.h>
FILE *fdr, *fdw;
int fd;
struct termio term_save, stdin_save;
void Exit(int sig)
{
   if (fdr) fclose(fdr);
   if (fdw) fclose(fdw);
   ioctl(fd, TCSETA, &term_save);
   close(fd);
   ioctl(fileno(stdin), TCSETA, &stdin_save);
   exit(sig);
}
main(int argc, char *argv[])
{
   char *b, buffer[80];
   int baud=0, num;
   struct termio term, tstdin;
   if (argc < 2 || !strcmp(argv[1], "-?"))
   {
      fprintf(stderr, "Usage: motalk /dev/tty? [speed]¥n");
      exit(1);
   }
   if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0)
   {
      perror(argv[1]);
      exit(errno);
   }
   if (argc > 2)
   {
      switch(atoi(argv[2]))
      {
         case   300: baud = B300;
                     break;
         case  1200: baud = B1200;
                     break;
         case  2400: baud = B2400;
                     break;
         case  4800: baud = B4800;
                     break;
         case  9600: baud = B9600;
                     break;
         case 19200: baud = B19200;
                     break;
         case 38400: baud = B38400;
                     break;
         default:    baud = 0;
                     fprintf(stderr, "%s: %s is an unsupported baud¥n", argv[0],argv[2]);
                     exit(1);
      }
   /* Save stdin and tty state and trap some signals */
   ioctl(fd, TCGETA, &term_save);
```

```
    ioctl(fileno(stdin), TCGETA, &stdin_save);
    signal(SIGHUP, Exit);
    signal(SIGINT, Exit);
    signal(SIGQUIT, Exit);
    signal(SIGTERM, Exit);
    /*  Set stdin to raw mode, no echo */
    ioctl(fileno(stdin), TCGETA, &tstdin);
    tstdin.c_iflag = 0;
    tstdin.c_lflag &= ~(ICANON | ECHO);
    tstdin.c_cc[VMIN] = 0;
    tstdin.c_cc[VTIME] = 0;
    ioctl(fileno(stdin), TCSETA, &tstdin);
    /*  Set tty state */
    ioctl(fd, TCGETA, &term);
    term.c_cflag |= CLOCAL|HUPCL;
    if (baud > 0)
    {
       term.c_cflag &= ~CBAUD;
       term.c_cflag |= baud;
    }
    term.c_lflag &= ~(ICANON | ECHO); /* to force raw mode */
    term.c_iflag &= ~ICRNL; /* to avoid non-needed blank lines */
    term.c_cc[VMIN] = 0;
    term.c_cc[VTIME] = 10;
    ioctl(fd, TCSETA, &term);
    fcntl(fd, F_SETFL, fcntl(fd, F_GETFL, 0) & ~O_NDELAY);
    /*  Open tty for read and write */
    if ((fdr = fopen(argv[1], "r")) == NULL )
    {
       perror(argv[1]);
       exit(errno);
    }
    if ((fdw = fopen(argv[1], "w")) == NULL )
    {
       perror(argv[1]);
       exit(errno);
    }
    /*  Talk to the modem */
    puts("Ready... ^C to exit");
    while (1)
    {
       if ((num = read(fileno(stdin), buffer, 80)) > 0)
          write(fileno(fdw), buffer, num);
       if ((num = read(fileno(fdr), buffer, 80)) > 0)
          write(fileno(stdout), buffer, num);
       Exit (0);
    }
}
```

Hayes モデムおよび Hayes 互換モデムの使用

Hayes モデムおよび Hayes 互換モデムでは、以下の手順を使用します。

1. 必要に応じて、SMIT 高速パスの `smit chtty` を使用して、tty 設定を変更します。
   例えば、「Enable LOGIN (ログインを可能にする)」フィールドの値を 「**Share (共用)**」または「**Enable (使用可能)**」に変更できます。
2. `/usr/lib/uucp/Systems` ファイルに次の行を追加します。

   ```
   hayes Nvr HAYESPROG 2400
   ```

3. `/usr/lib/uucp/Devices` ファイルに以下を追加します。

   ```
   # For programming the  hayes modem only:
   HAYESPROG tty0 - 2400 HayesProgrm2400
   #regular ACU entry:
   ACU tty0 - Any hayes
   ```

4. `/usr/lib/uucp/Dialers` ファイルに以下を追加します。

   ```
   # This Entry is used to PROGRAM the modem ONLY:
   # the next 3 lines should be made into one:
   HayesProgrm2400 =,-, "" ¥d¥dAT¥r¥c OK AT&F¥r¥c OK ATM1¥r¥c OK
   AT&D3¥r¥c OK AT&K3&C1¥r¥c OK ATL0E0Q2¥r¥c OK ATS0=1¥r¥c OK AT&W¥r¥c
   OK
   hayes     =,-,     "" ¥dAT¥r¥c OK ATDT¥T¥d¥r¥c CONNECT
   ```

5. モデムに設定情報を書き込むには、コマンド `cu -d hayes` を入力します。

   このコマンドは、**cu** コマンドを使用してモデムに設定情報を書き込みます。 別のシステムに接続されていないので、このコマンドは失敗します。 出力の中に「`sendthem AT&W`」が表示され、次に「`OK got it`」が表示されれば、モデムに設定情報が書き込まれています。

   バイナリー・ファイルを転送しないか BNU を使用する場合は、**&K3** コマンドを削除し、使用するフロー制御として XON を設定します。 ただし、(XON-XOFF handshaking ではなく) ハードウェアのフロー制御を使用する方が効率的です。 そのためには、次のステップの設定と `Dialers` エントリーを使用します。
6. モデムに設定情報を書き込んだあと、システム・デバイス・ドライバーをハードウェアのフロー制御を使用できるように構成できます。 SMIT (`smit chtty` 高速パス) を使用して、フロー制御を RTS に変更します。モデムのマニュアルで、ご使用のモデムがハードウェア・フロー制御をサポートしているかどうかを確認してください。

モデムのトラブルシューティング
コンピューターでモデムを使用した際に問題が発生した場合は、以下の点を考慮してください。

- 一部のモデムは、大文字小文字を区別します。 AT コマンドは大文字を使用してください。
- 通常の操作では、DTR がドロップされた場合 (&D3 設定)、モデムをリセットしてください。 ただし、モデムを初めてセットアップするときは、DTR がドロップされても (&D2 設定)、モデムをリセットしないことをお勧めします。 モデムによってモデム自体がリセットされると、モデムのメモリーに保管されていないプログラムされた設定は、すべて失われてしまいます。

  モデムがリセットされなければ、&C1 が設定されたときの変更も保護されています。 キャリア検知状況を変更すると、モデムによってはキャリア検知回線が切り替わる可能性があり、それによって **cu** コマンドが回線をドロップすることになります。最終的にセットアップが完了したあとで、モデムを &D3 に設定することもできます。
- この一連のトピックで示すコマンドは、ほとんどの Hayes 互換モデムに標準のものですが、ご使用のモデムの標準であるかどうかは保証できません。先へ進む前に、ご使用のモデムの資料にあるコマンドと比較してください。

モデムの問題をデバッグする便利な方法は、モデムを取り外し、同じポートに ASCII 端末を (インターポーザーまたはヌル・モデムで) 接続してモデムとして配線するという方法です。 端末をモデムと同じ回線速度、1 文字あたりのビット数、およびパリティーでセットアップします。 ポートがログインに使用可能な場合は、画面にログイン・ヘラルドが表示される必要があります。端末画面にヘラルドが表示される場合は、問題はモデムの構成にあると特定できます。

以下のヒントは、モデム接続に関連する問題を切り分ける際に役立ちます。

表 *104.* モデムの問題と解決

問題	**解決**
respawn が速すぎる	getty プログラムが **init** によって respawn されています。

表 104. モデムの問題と解決 (続き)	
問題	**解決**
コンソールまたは errpt にメッセージが表示される	**init** が 225 秒以内に 5 回以上プログラムを respawn しなければならない状況の場合は、コンソールにメッセージを表示し、しばらくプログラムを respawn しません。 解決策は、getty が動作しない理由を突き止めることです。 複数の原因が考えられます。 • モデムの設定が誤っている。通常、これはモデムまたは配線で CD を high に固定したり、「echo」または「command response」をオンにしてあることの結果です。 (ポート構成の runmodes および/または logmodes に clocal を追加することによっても CD は high と見なされます。また、128 ポートで強制されている場合もあります。) • CD シグナルが切り替わる。 getty プロセスは、CD がオンからオフの状態に切り替わるたびに停止します。 (このアクションは、いくつかの理由に起因していることが考えられます。 ケーブルが正しくシールドされていることを確認してください。 ログインとログアウトを数回連続して繰り返すことも、この原因の 1 つです。)
モデムへの接続後にログイン・プロンプトが表示されない	ポート上で getty が実行されていることを確認します。 実行されている場合は、相手側がモデムに接続した後で、モデム接続のキャリア検出シグナルが発せられることを確認します。 CD が正しくアサートされている場合は、モデムが正しいポートに接続されていることを確認します。 まだログインが表示されない場合は、インターポーザー付きの端末をモデムの代わりにケーブルに接続し、ログイン・プロンプトが表示されることを確認してください。 それでもプロンプトが表示されない場合は、文字を端末画面にエコーしてみて、ケーブルおよびハードウェアが正しく機能していることを確認してください。
リモート・モデムが接続すると即時に切断される	サーバーがモデムを listen するのと同じ速度でモデムがサーバーと対話していることを確認してください。 tty で別のボー・レートを試してみるか、モデムをプログラムして、tty ポートの速度と一致するよう DTE 速度をロックします。 モデムまたはポートがキャリア検出シグナルを high のままにしていないこと、またはポートが別のプロセスで使用されていないことを確認します。
ログイン・プロンプトが表示されず、文字化けが発生する	これは、プロトコルの相違に起因しています。 モデムと tty ポートで、同じパリティー、ボー・レート、フロー制御、および文字サイズが使用されていることを確認してください。
正常セッション後にログインできなくなることがある	切断後にモデムがリセットされていないことが原因と考えられます。 モデムのマニュアルを参照して、オンからオフへの DTR 変位後にモデムをリセットする方法を確認してください。
errpt で受信側バッファーがオーバーランする	UART チップ・バッファーがオーバーランしています。 SMIT で tty の受信トリガーの値を下げてください。 この解決策は、ネイティブの 8 または 16 ポート非同期通信アダプターでのみ有効です。 モデムおよび tty ポートが同じフロー制御を使用していることを確認してください。
errpt で ttyhog エラーが発生する	モデムおよび tty が同じフロー制御を使用していないか、フロー制御が行われていない。

ソフトウェア・サービスへのモデムに関する質問表

モデムの問題について問い合わせる前に、効率的に処理するために以下の基本情報を収集してください。

以下の情報が利用可能です。

- オペレーティング・システムのレベル。 そのレベルでのオペレーティング・システムの使用期間内。
- 今まではモデムが機能していたかどうか。
- 使用中のモデムのタイプ。 電話接続の相手側のモデムのタイプ。
- モデムを接続しているアダプター・タイプ。

- モデムを接続しているポート番号。
- モデムを接続している tty 番号。
- 使用中のケーブルのタイプ。
- ログイン設定の内容 (share、delay、enable)。
- モデムが他のモデムに接続できるかどうか。
- 他のモデムが自分のモデムに接続できるかどうか。
- SMIT、モデム、ポートの次の値。
  - XON/XOFF
  - RTS/CTS
  - BPS
- 問題を説明するときには、次の状態も説明してください。
  - ポートが断続的にロックされるかどうか。
  - ダイヤルアウトできるかどうか。 他のユーザーがダイヤルインできるかどうか。
  - その他の具体的なエラー条件。
- コンソールにエラーが発生しているかどうか。 発生している場合は、どのようなエラーか。
- エラー・レポートにエラーが記載されているかどうか (**errpt** または **errpt -a**)。
- ダイヤルアウトに使用しているコマンド。
- システムにインストールされているソフトウェア 。

AT コマンド

Hayes Smartmodem コマンド・セットには、多くの一般的なモデムで使用される AT コマンド・セットが含まれます。

この情報は、Hayes Microcomputer Products, Inc. 発行の、Hayes Smartmodem 2400 の「*Quick Reference Card*」から転載したものです。関連する AT コマンドについては、このモデムの文書を参照してください。

項目	説明
AT	コマンド接頭部 - コマンド・ラインの先頭に付けます。
<CR>	復帰 (改行) 文字 - コマンド・ラインを終了します。
A	オフフックにします。コマンド・モードのままです。
A/	直前のコマンド・ラインを繰り返します。 このコマンドには、先頭の **AT** も末尾の **<CR>**/ も付きません。
B0	1,200 bps 通信用の CCITT V.22 規格を選択します。
B1	1,200 bps 通信用の Bell 212A 規格を選択します。
D	発信モードに入り、後続の番号をダイヤルし、オンラインにしようとします。 通常、D のあとにはトーンを表す T が続きます。パルスを表す P も使用できます。
DS=*n*	ロケーション *n* に保管されている番号をダイヤルします。
E0	コマンド状態での文字エコーを使用不可にします。
E1	コマンド状態での文字エコーを使用可能にします。
H0	オンフックにします (電話を切ります)。
H1	フック・スイッチと補助リレーを操作します。
I0	プロダクト識別コードを戻します。
I1	ファームウェア ROM 上でチェックサムを実行し、チェックサムを戻します。

項目	説明
I2	ファームウェア ROM 上でチェックサムを実行し、その結果として「OK」または「ERROR」を戻します。
L0	スピーカーをオフにします。
L1	スピーカーの音量を下げます。
L2	スピーカーの音量を中程度にします。
L3	スピーカーの音量を上げます。
M0	スピーカーをオフにします。
M1	キャリアが検出されるまで、スピーカーをオンにします。
M2	スピーカーを常にオンにします。
M3	ダイヤル中以外は、キャリアが検出されるまでスピーカーをオンにします。
O0	オンライン状態に入ります。
O1	オンライン状態に入って、イコライザーのリトレーニングを開始します。
Q0	モデムは結果コードを戻します。
Q1	モデムは結果コードを戻しません。
Sr	レジスター r へのポインターを設定します。
Sr=*n*	レジスター r を値 *n* に設定します。
V0	結果コードを数値フォーマットで表示します。
V1	結果コードを詳細フォーマット (ワード) で表示します。
X0	結果コード 0 から 4 で表される機能を使用可能にします。
X1	結果コード 0 から 5、10 で表される機能を使用可能にします。
X2	結果コード 0 から 6、10 で表される機能を使用可能にします。
X3	結果コード 0 から 5、7、10 で表される機能を使用可能にします。
X4	結果コード 0 から 7、10 で表される機能を使用可能にします。
Y0	長時間の無応答時の切断を使用不可にします。
Y1	長時間の無応答時の切断を使用可能にします。
Z	モデムをリセットします。
&C0	データ・キャリアが常に存在するものと想定します。
&C1	データ・キャリアが存在するかどうかを追跡します。
&D0	DTR シグナルを無視します。
&D1	DTR がオンからオフに変わると、コマンド状態になります。
&D2	DTR がオンからオフに変わると、回線を切ってコマンド状態になります。
&D3	DTR がオンからオフに変わると、リセットします。
&F	出荷時の設定をアクティブな 構成として再呼び出しします。
&G0	ガード・トーンなし。
&G1	500 Hz のガード・トーン。
&G2	1,800 Hz のガード・トーン。
&J0	RJ-11/RJ41/RJ45S telco ジャック。

項目	説明
&J1	RJ-11/RJ-13 telco ジャック。
&P0	make/break 比が 39/61 のパルス・ダイヤル。
&P1	make/break 比が 33/67 のパルス・ダイヤル。
&Q0	非同期モードで動作します。
&Qn	同期モード *n* で動作します。
&R0	RTS に従って CTS を追跡します。
&R1	RTS を無視します。常に CTS が存在するものと想定します。
&S0	DSR シグナルが存在するものと想定します。
&S1	DSR シグナルが存在するかどうかを追跡します。
&T0	進行中のテストを終了します。
&T1	ローカル・アナログ・ループバックを開始します。
&T3	デジタル・ループバックを開始します。
&T4	リモート・モデムからのリモート・データ・リンク (RDL) 要求を受け入れます。
&T5	リモート・モデムからの RDL 要求を拒否します。
&T6	リモート・デジタル・ループバックを開始します。
&T7	自己診断テスト付きでリモート・デジタル・ループバックを開始します。
&T8	自己診断テスト付きでローカル・アナログ・ループバックを開始します。
&V	アクティブな構成、ユーザー・プロファイル、保管されている番号を表示します。
&Wn	アクティブな構成の保管可能なパラメーターをユーザー・プロファイル *n* に保管します。
&X0	モデムが送信クロック・シグナルを出力します。
&X1	データ端末装置が送信クロック・シグナルを出力します。
&X2	受信キャリアが送信クロック・シグナルを出力します。
&Yn	ユーザー・プロファイル *n* を再呼び出しします。
&Zn=*x*	電話番号 *x* をロケーション *n* に保管します。

S レジスターの要約
S レジスターおよびその範囲と説明を次の表に示します。

表 *105. S* レジスターの説明

レジスター	範囲	説明
S0	0 から 255	応答前のリングの回数を選択します。
S1	0 から 255	リング・カウント (1 回のリングごとに増分されます。)
S2	0 から 127	エスケープ・シーケンス文字 (ASCII) を定義します。
S3	0 から 127	復帰文字 (ASCII) を定義します。
S4	0 から 127	改行文字 (ASCII) を定義します。

表 *105. S* レジスターの説明 *(*続き*)*		
レジスター	**範囲**	**説明**
S5	0 から 32、127	バックスペース文字 (ASCII) を定義します。
S6	2 から 255	ブラインド・ダイヤリング前の待ち時間を秒単位で選択します。
S7	1 から 55	キャリア/ダイヤル・トーンの待ち時間を秒単位で選択します。
S8	0 から 255	コンマのポーズ期間を秒単位で選択します。
S9	1 から 255	0.1 秒単位のキャリア検知応答時間を増分します (10 = 1 秒)。
S10	1 から 255	キャリア消滅から回線切断までの 0.1 秒単位の遅延時間を増分します。
S11	50 から 255	ミリ秒単位のトーンの期間/インターバル
S12	50 から 255	0.02 秒間隔のエスケープ・シーケンスのガード時間
S13	—	予約済み
S14	—	予約済み
S15	—	予約済み
S16	—	予約済み - このレジスターの機能は、**&T** コマンドで制御されます。
S17	—	予約済み
S18	0 から 255	秒単位のテスト・タイマー期間
S19	—	予約済み
S20	—	予約済み
S21	—	予約済み
S22	—	予約済み
S23	—	予約済み
S24	—	予約済み
S25	0 から 255	0.01 秒間隔で DTR の変化の検出時間を選択します。
S26	0 から 255	0.01 秒間隔の RTS から CTS までの遅延時間
S27	—	予約済み

非同期通信アダプターの結果コード
非同期通信アダプターが戻す結果コードを番号、ワード、説明を含めて次の表に示します。

表 *106.* 非同期通信アダプター結果コード

番号	ワード	説明
0	OK	コマンドが実行されました。
1	CONNECT	0 から 300 bps で接続が確立されました。
2	RING	リング・シグナルが検出されました。
3	NO CARRIER	キャリア・シグナルが消滅したか、検出されませんでした。
4	ERROR	無効なコマンド、チェックサム、コマンド・ラインのエラー、または長すぎるコマンド・ライン
5	CONNECT 1,200	1,200 bps で接続が確立されました。
6	NO DIALTONE	ダイヤル・トーンは検出されませんでした。
7	BUSY	ビジー・シグナルが検出されました。
8	NO ANSWER	システムへダイヤルしましたが、応答がありません。
9	CONNECT 2,400	2,400 bps で接続が確立されました。

ダイヤル修飾子
以下の表では、ダイヤル修飾子とその説明を参照できます。

項目	説明
0-9 # * A-D	ダイヤル用の数字と文字。
P	パルス・ダイヤル。
T	トーン・ダイヤル。
,	次の文字の処理を遅らせます。
!	短期間、回線をオン・フックします。
@	無音の状態になるまで待ちます。
W	ダイヤル・トーンを待ちます。
;	ダイヤル後にコマンド状態に戻ります。
R	モードが反転します。
S=*n*	ロケーション *n* に保管されている番号をダイヤルします。

モデムに対する支援

モデムに問題が発生した場合は、以下のように支援を受けることができます。

- 地元の担当者がモデムの構成を支援します。
- 提供されるサポート・サービスの中から、オンサイト・アシスタンスや電話サポートなどの、さまざまなサポート・オプションを選択できます。 支援については、サービス技術員に連絡してください。

- 多くの場合に見過ごされるのは、モデム製造メーカーの支援です。 ほとんどの製造メーカーは、何らかの形で製品のオンライン・アシスタンスを提供しています。

/usr/lib/uucp/Dialers.samples ファイル・エントリー

以下のファイル・エントリー例は、保証なしで提供され、記されているモデルに対してのみ現状のまま機能するため、お客様の特定のニーズを満たしているとは限りません。

個別のニーズを満たすようにするには、多少の変更を加える必要があります。 設定の詳細な説明については、モデムのマニュアルを調べてください。

モデムをプログラムするための設定を使用するには、/usr/lib/uucp/Systems ファイルに、次のようなエントリーが必要です。

```
hayes Nvr HayesPRGM Any
```

/usr/lib/uucp/Devices ファイルには、次のようなエントリーが必要です。

```
HayesPRGM tty0 - 2400 HayesProgrm2400
```

上記の 2 つのエントリーを用意した上で、次の **cu** コマンドを使用して、モデムをプログラムします。

```
cu -d hayes

# COMPONENT_NAME: cmduucp
#
#
# (C) COPYRIGHT International Business Machines Corp. 1994
# Licensed Materials - Property of IBM
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM
# Corp.
###############################################################################
# Motorola UDS Modem
#
# Use udsmodemPROGRAM to program the modem.
# Port needs to have rts/cts set.
# Use uds or hayes dialer.
#
# The "udsmodemPROGRAM" line should be a single, continuous line
#
###############################################################################
udsmodemPROGRAM =,-, "" ¥&FQ2¥r¥c OK
ATE0Y0&C1&D2&S1%B5%E0*LC¥r¥c OKAT&K3&W¥r¥c OK

uds =,-, "" ¥dAT¥r¥c OK¥r ATDT¥T¥d¥r¥c CONNECT

###############################################################################
#
# IBM 7855 Model 10
# Use IBMProgrm to program the modem.
# This sets rts/cts flow control, turns
# off xon/xoff, and sets the DTE speed at 19,200 bps.
# The modem will connect at the appropriate speed and
# flow control with the server.
# Port needs to have rts/cts set.
#
# The "IBMProgrm" line should be a single, continuous line
#
###############################################################################
IBMProgrm =,-, "" ¥dATQ0¥r¥c OK AT&F¥r¥c OK ATM1¥r¥c OK
AT&D3¥r¥c OK AT&C1¥R2¥Q2¥M14¥r¥c OK AT&B8N1L0E0¥A0¥r¥c OK
ATS0=1¥r¥c OK ATQ1&W0&Y0¥r¥c ""

###############################################################################
# The following are used for Dialing out on a 7855
# regular ACU device. We have to turn on result
# codes (Q0) because they are turned off when we
# programmed it. (Keeps all upper case login from
# happening on dial in attempts.)
# We have to have an extra "¥" before "¥N" because
# the BNU programs strips it if it's before an "N".
###############################################################################
ibm =,-, "" ¥dATQ0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 ECL (No Compression)
```

```
ibmecl =,-, "" ¥dAT¥¥N3%C0Q0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 ECLC (Compression)
ibmeclc =,-, "" ¥dAT¥¥N3%C1Q0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 ECLC Compression with 256 byte block size
ibmeclc256 =,-, "" ¥dAT¥¥N3%C1Q0¥A3¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 1200bps
ibm_ne12 =,-, "" ¥dATQ0¥¥N0&A2%C0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 2400bps
ibm_ne24 =,-, "" ¥dATQ0¥¥N0&A3%C0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 9600bps
ibm_ne96 =,-, "" ¥dATQ0¥¥N0&A6%C0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 19200bps
ibm_ne192 =,-, "" ¥dATQ0¥¥N0%C0¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 12000bps
ibm_ne120 =,-, "" ¥dATQ0¥¥N3%C0&AL8¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 1200bps (Dial Quietly)
ibmq12 =,-, "" ¥dATQ0¥r¥c OK AT&A2M0DT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 2400bps (Dial Quietly)
ibmq24 =,-, "" ¥dATQ0¥r¥c OK AT&A3M0DT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 9600bps (Dial Quietly)
ibmq96 =,-, "" ¥dATQ0¥r¥c OK AT&A6M0DT¥T¥d¥r¥c CONNECT

# IBM 7855 No Compression 19200bps (Dial Quietly)
ibmq192 =,-, "" ¥dATQ0¥r¥c OK ATM0DT¥T¥d¥r¥c CONNECT

##############################################################################
#
# Intel 9600EX Modem
# Use IntelProgram to program the modem.
# This sets rts/cts flow control, and turns
# off xon/xoff.
# Port needs to have rts/cts set. (Use hayes dialer)
#
# The "IntelProgram" line should be a single, continuous line
#
##############################################################################
#IntelProgram =,-, "" ¥d¥dAT¥r¥c OK AT&F¥r¥c OK AT&S1M1¥r¥c OK
AT&D3¥r¥c OKAT&C1¥r¥c OK ATL0E0Y0&Y0¥X1¥r¥c OK ATS0=1¥r¥c OK
AT&W¥r¥c OK

##############################################################################
# Practical Peripherals 1440FXMT Modem
# Use PracPerProgram144 to program the modem.
# This sets rts/cts flow control, and turns
# off xon/xoff. (Use hayes dialer)
# DTE speed will be locked at connect speed when
# the modem is programmed. (Suggestion: 38400 baud)
#
# The "PracPerProgram144" line should be a single, continuous
# line
##############################################################################
PracPerProgram144 =,-, "" ¥d¥dAT¥r¥c OK AT&F¥r¥c OK ATM1¥r¥c OK
AT&D3¥r¥c OKAT&C1&K3¥r¥c OK ATQ2E1&Q9¥r¥c OK ATS0=1S9=20¥r¥c OK
AT&W¥r¥c OK

##############################################################################
# Practical Peripherals 9600 bps Modem
# Use PracPerProgram9600 to program the modem.
# This sets rts/cts flow control, and turns
# off xon/xoff. (Use hayes dialer)
#
# The "PracPerProgram144" line should be a single, continuous
# line
##############################################################################
PracPerProgram9600 =,-, "" ¥d¥dAT¥r¥c OK AT&F¥r¥c OK ATM1¥r¥c OK
AT&D3¥r¥c OKAT&C1&K3¥r¥c OK ATL0E0¥r¥c OK ATS0=1S9=20¥r¥c OK
AT&W¥r¥c OK

##############################################################################
# Practical Peripherals 2400 bps Modem
# Use PracPerProgram to program the modem
#
```

```
# The "PracPerProgram2400" line should be a single, continuous
# line
#############################################################################
PracPerProgram2400 =,-, "" ¥d¥dAT¥r¥c OK AT&F¥r¥c OK ATM1¥r¥c OK
AT&D3¥r¥c OKAT&C1¥r¥c OK ATL0E0¥r¥c OK ATS0=1S9=20¥r¥c OK AT&W¥r¥c OK

#############################################################################
# Hayes 2400 bps Modem
# Use HayesProgrm2400 to program the modem.
# (Use hayes dialer to dial)
#
# The "HayesProgrm2400" line should be a single, continuous line
#
#############################################################################
HayesProgrm2400 =,-, "" ¥d¥dAT¥r¥c OK AT&F¥r¥c OK ATM1¥r¥c OK
AT&D3¥r¥c OKAT&C1¥r¥c OK ATL0E0¥r¥c OK AT S0=1¥r¥c OK AT&W¥r¥c OK

#############################################################################
# Telebit t2000 Trailblazer Plus
# Use TelebitProgrm to program the modem
# This sets rts/cts flow control, and turns
# off xon/xoff and sets the Default DTE speed at
# 19,200 bps.
# Port needs to have rts/cts set.
# This sets modem to send PEP tones last as they can
# can confuse some other modems.
#
# The "TelebitProgram" line should be a single, continuous line
#
#############################################################################
TelebitProgram =,-, "" ¥dAT&F¥r¥c OK
ats2=255s7=60s11=50s41=2s45=255s51=254s52=2s54=3s58=2s64=1s66=1¥r¥c OK
ATs69=1s92=1s96=0s105=0s110=1s111=30s130=3s131=1F1M0Q6TV1W0X3Y0¥r¥c OK
ATE0&W¥r¥c OK
# Telebit T2000 dialers Entries:
# Forces a PEP connection:
tbfast =,-, "" ¥dATs50=255s7=60¥r¥c OK¥r ATDT¥T¥r¥c
CONNECT-¥d¥c-CONNECT

# 2400bps connection:

#tb2400 =,-, "" ¥dATs50=3¥r¥c OK¥r ATDT¥T¥r¥c CONNECT

# 2400 MNP:
tb24mnp =,-, "" ¥dAT¥r¥c OK ATS0=0S95=2S50=3S41=0¥r¥c OK
ATDT¥T¥r¥c CONNECT

# 1200bps connection:#tb1200 =,-, "" ¥dATs50=2¥r¥c OK¥r
ATDT¥T¥r¥c CONNECT

# 1200 MNP:
tb12mnp =,-, "" ¥dAT¥r¥c OK ATS0=0S95=2S50=2S41=0¥r¥c OK
ATDT¥T¥r¥c CONNECT

#############################################################################
# Telebit WorldBlazer
# WORLDBLAZERProgram sets the DTE speed at 38400, but
# you could set it higher if the DTE connection can
# handle it. We answer with PEP tones last so as not
# to confuse other modems. This turns off xon/xoff
# and turns on RTS/CTS flow control. The port should
# be locked to 38400 with these settings, and needs
# to have RTS/CTS turned on.
#
# The "WORLDBLAZERProgram" line should be a single, continuous
# line
#############################################################################
WORLDBLAZERProgram =,-, "" ¥dAT¥r¥c AT AT&F3M0¥r¥c AT
ATs51=253s92=1¥r¥c ATAT&W¥r¥c AT

#############################################################################
# ACU Dialers for various BAUD rates for the
# WorldBlazer - each sets the modem to attempt to
# connect at a specific speed and lower. The
# WBlazer will accept whatever the remote modem can
# do. You will want to use PEP for other Telebits,
# so use WBlazer38400 or WBlazer19200 for those
#############################################################################
# WBlazer =,-, "" ¥dAT¥r¥c OK ATDT¥T¥d¥r¥c CONNECT
WBlazer38400 =,-, "" ¥dATs50=255¥r¥c OK ATDT¥T¥d¥r¥c CONNECT
WBlazer19200 =,-, "" ¥dATs50=255¥r¥c OK ATDT¥T¥d¥r¥c CONNECT
# WBlazer14400 attempts to negotiate a V.42bis connection.
```

```
WBlazer14400 =,-, "" ¥dATs50=7¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# For a V.32 connection:
WBlazer9600 =,-, "" ¥dATs50=6¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# For a V.22 connection:
WBlazer2400 =,-, "" ¥dATs50=3¥r¥c OK ATDT¥T¥d¥r¥c CONNECT

# For a 1200 bps connection:
WBlazer1200 =,-, "" ¥dATs50=2¥r¥c OK ATDT¥T¥d¥r¥c CONNECT
```

128 ポート・モデム配線の考慮事項
このオペレーティング・システムでは、モデム制御アプリケーションに DSR がなくても構いません。最近のほとんどのモデムに自動応答機能が備わっているため、一般にリング・インディケーター・シグナルは不要です。

10 ピン RJ-45 プラグは、広く使用されている配線サブシステムではないため、小売マーケットで入手することは困難な場合があります。 このオペレーティング・システムの TTY サブシステムには、ALTPIN と呼ばれるオプショナル機能があります。これは、ポート用に DSR (データ・セット・レディー) の論理機能を DCD (データ・キャリア検知) と交換します。 ALTPIN を使用可能にすると、DCD が 8 ピン RJ-45 コネクターのピン 1 で使用可能になります (10 ピン・コネクターのピン 2 と同等)。

128 ポート RAN 用に 8 ワイヤー・モデム・ケーブルを作成する場合は、8 ピン RJ-45 プラグを使用して、次の表のように配線してください。

表 *107. 128* ポート・モデム配線

項目	**説明**	**モデム**
システム側コネクター 8 ピン RJ-45	**装置終了** RI	22
1	DSR	6
2	RTS	4
3 (シャーシ)	GND	SHELL
4	TxD	2
5	TxD	3
6 (シグナル)	GND	7
7	CTS	5
8	DTR	20
	CD	8

注 : DSR および CD の物理的な位置は、stty-cmxa コマンドを使用して ALTPIN パラメーターを使用可能にすると、交換される場合があります。

システム装置と接続されたモデムとの間の非同期シグナル通信は、次の表のように行われます。 ここでは、データはシステム装置からリモート・システムに送信されます。

表 *108.* 非同期シグナル通信

デバイス	**シグナル**	**オン/オフ**	**意味**
コンピューター	DTR	+	もしもし、モデムさん、別のシステムに接続する準備ができましたか?
モデム	DSR	+	はい、準備できました。 さあ、ダイヤルしてください。
モデム	DCD	+	別のシステムと通話中です。

表 *108.* 非同期シグナル通信 (続き)

デバイス	シグナル	オン/オフ	意味
コンピューター	RTS	+	了解、データをすぐに送信してもいいですか?
モデム	CTS	+	どうぞ、そうしてください。
コンピューター	TxD		データをモデムに送信中。
モデム	RxD		データを受信しました。
モデム	CTS	–	もうデータを送信しないでください。データの送信中です。
モデム	CTS	+	OK、データを受け入れる準備ができたので送ってください。
データ送信ステップが繰り返されます。(コンピューター)	DTR	–	完了しました。 さあ、回線を切ってください。
モデム	DCD	–	了解。
別のシステムからの着呼を受け取る際のRS/6000® とモデムの間のシグナル通信は次のようになります。コンピューター	DTR	+	準備完了、ダイヤルイン用にポートを「使用可能」にしました。
モデム	DSR	+	こちらも準備できましたが、呼び出しを待っています。
呼び出しが発生しました。モデム	DCD	+	呼び出しがあったので、回線をつなぎます。
モデム	CTS	+	別のボックスからデータを受け取りました。すぐにデータを送信してもいいですか?
コンピューター	RTS	+	受信の準備はできています。 さあ、送信してください。
モデム	RxD		開始します。
モデムがデータの送信を続けます。(コンピューター)	RTS	–	待ってください。 バッファーがいっぱいになりました。データを送信しないでください。
コンピューター	RTS	+	いいですよ、 データを送ってください。
モデム	DCD	–	通信が終わりました。
コンピューター	DTR	–	了解、回線を切ってください。

stty-cxma 端末オプション

stty-cxma は、PCI 2 ポート、8 ポートおよび 128 ポート・アダプターの 設定と表示を行う ユーティリティー・プログラムで、/usr/lbin/tty ディレクトリーにあります。

フォーマットは次のとおり。

```
stty-cxma [-a] [option(s)] [ttyname]
```

stty-cxma は、オプションを指定しないと、すべての特殊なドライバーの設定、モデム・シグナル、および標準入力によって参照される tty デバイスの **stty(1)** によって 表示される標準パラメーターを表示します。 コマンド・オプションによって、フロー制御の変更、透過印刷オプションの設定、モデム制御回線の強制、およびすべての tty 設定の表示を行うことができます。 認識されなかったオプションは、**stty(1)** に渡されて解釈されます。 オプションは次のとおりです。

-a
stty -a コマンドで報告されるすべての標準 tty 設定だけでなく、すべての固有のアダプター・オプションの設定を表示します。

ttyname
標準入力ではなく、指定された tty デバイスのオプションの 設定と表示を行います。 この形式は、前に **/dev/** が付いた tty パス名か、tty で始まる単純な tty 名を使用します。 このオプションは、キャリアがないときにモデム制御回線で使用できます。

次のオプションでは、直ちに実行される一時的なアクションを指定します。

break
250 ms のブレーク・シグナルを tty 回線に送出します。

flush
tty 入出力の即時フラッシュ (廃棄) を指示します。

flushin
tty 入力のみフラッシュします。

flushout
tty 出力のみフラッシュします。

次のオプションでは、デバイスがクローズするときにリセットされるアクションを 指定します。 デバイスは、次にオープンされたときには、デフォルト値を使用します。

stopout
ちょうど XOFF 文字を受信したかのように出力を停止します。

startout
ちょうど XON 文字を受信したかのように停止していた出力を再開します。

stopin
入力を停止するフロー制御を活動化します。

startin
停止された入力を再開するフロー制御を解放します。

[-]dtr [drop]
DTR モデムのハードウェア・フロー制御が選択されていなければ、DTR モデム制御回線を立ち上げます。

[-]rts [drop]
RTS モデムのハードウェア・フロー制御が選択されていなければ、RTS モデム制御回線を立ち上げます。

次のオプションは、システムがリブートされるか、オプションが変更されるまで有効です。

[-]fastcook
インテリジェント・カード上で出力の加工処理を行うことによって、ホスト CPU の使用量を削減し、ロー・モード入力のパフォーマンスを向上させます。

[-]fastbaud
ボー・レート・テーブルを変更します。これにより、サポートされるデバイスに ついて、50 ボーが 57,600 ボーに、75 ボーが 76,800 ボーに、110 ボーが 115,200 ボーに、200 ボーが 230,000 ボーになります。

[-]rtspace
RTS ハードウェア入力フロー制御を使用可能または不可にして、リモート送信を一時停止するために RTS をドロップします。

[-]ctspace
CTS ハードウェア出力フロー制御を使用可能または不可にして、CTS がドロップしたときにローカル送信を一時停止します。

[-]dsrpace
DSR ハードウェア出力フロー制御を使用可能または不可にして、DSR がドロップしたときにローカル送信を一時停止します。

[-]dcdpace
DCD ハードウェア出力フロー制御を使用可能または不可にして、DCD がドロップしたときにローカル送信を一時停止します。

[-]dtrpace
DTR ハードウェア入力フロー制御を使用可能または不可にして、リモート送信を一時停止するために DTR をドロップします。

[-]forcedcd
キャリア・センスを使用不可 [再度使用可能] にして、キャリアが存在しないときも tty をオープンして使用できるようにします。

[-]altpin
RJ-45 コネクター・ピン配列をデフォルトの 10 ピン・コネクター値または 8 ピン・コネクター値にマップします。 このパラメーターが **enabled** の場合、10 ピン RJ-45 コネクターではなく 8 ピン RJ-45 コネクターを 使用しているときに、DCD が使用可能になるように、DSR および DCD の位置が切り替えられます。 (デフォルトは **disable** です。)

設定可能な値:

enabled (8 ピン・コネクター値を指定)

disable (10 ピン・コネクター値を指定)

startc *c*
XON フロー制御文字を設定します。 文字は、10 進数、8 進数、または 16 進数で指定します。 8 進数であることは前に 0 を付けて示し、16 進数であることは前に 0x を付けて示します。 例えば、標準の XON 文字 CTRL-Q は、17 (10 進数)、021 (8 進数)、または 0x11 (16 進数) で入力できます。

stopc*c*
XOFF フロー制御文字を設定します。 文字は、10 進数、8 進数、または 16 進数で指定します (8 進数または 16 進数のフォーマットは、**startc** を参照してください)。

astartc*c*
補助 XON フロー制御文字を設定します。 文字は、10 進数、8 進数、または 16 進数で指定します (8 進数または 16 進数のフォーマットは、**startc** を参照してください)。

astopc*c*
補助 XOFF フロー制御文字を設定します。 文字は、10 進数、8 進数、または 16 進数で指定します (8 進数または 16 進数のフォーマットは、**startc** を参照してください)。

[-]aixon
XON および XOFF に 2 つの固有文字が使用されるように、補助フロー制御を使用可能にします。 両方の XOFF 文字を受信した場合、送信は、両方の XON 文字を受信するまで再開されません。

[-]2200flow
ポート上に 2200 スタイルのフロー制御を使用します。 2200 端末装置は接続されたプリンターをサポートし、端末 XON (0xF8)、プリンター XON (0xF9)、端末 XOFF (0xFA)、および プリンター XOFF (0xFB) の 4 つのフロー制御文字を使用します。

[-]2200print
これらのフロー制御文字の解釈方法を決定します。 2200print が設定されると、端末装置と透過プリンターについて、他に依存しないフロー制御が実行されます。 そうでなければ、端末とプリンターのフロー制御は、互いに論理的に結び付けられます。 いずれかの XOFF 文字を受信した場合、すべての出力は、対応する XON 文字を受信するまで一時停止されます。

maxcps*n*
透過プリンターに出力するときの、1 秒あたりの文字数 (字/秒 (cps)) の最大値を設定します。 速度は、平均の印刷速度よりわずかに低い値に設定します。 数値が低すぎると、プリンターの速度が遅くなります。 数値が高すぎると、プリンターはフロー制御を使用し、ユーザーの入力時間が少なくなります。 デフォルトは 100 cps (字/秒) です。

maxchar*n*
ドライバーが出力キューに置く、透過印刷文字の最大数を設定します。 この数を少なくすると、システムのオーバーヘッドが増加します。この数を増やすと、透過プリンターが使用中であるときに、オペレーターのキー・ストロークのエコー時間が遅くなります。 デフォルトは 50 文字です。

bufsize*n*
透過プリンターの入力バッファー・サイズについて、ドライバーの見積サイズを設定します。 非アクティブ期間のあと、ドライバーは、maxcps 速度になるまで、できるだけ多くの文字を一気に透過プリンターに送信します。 デフォルトは 100 文字です。

onstr*s*
透過印刷をオンにする端末エスケープ・シーケンスを設定します。 文字列は、標準 ASCII 印字文字および非印字文字から構成します。 制御文字 (非印字文字) は、8 進数で入力する必要があります。3 桁の 8 進数の前に円記号を指定します。 例えば、エスケープ文字 33 (8 進数) は、¥033 と入力します。 透過印刷が文字列 <Esc>[5i (ANSI 標準) によってオンにされる場合、¥033[5i と入力します。

offstr*s*
透過印刷をオフにする端末エスケープ・シーケンスを設定します。 文字列のフォーマットは、**onstr** *s* を参照してください。

term*t*
透過印刷のオン/オフ文字列に、内部デフォルト・テーブルにある値を設定します。 内部デフォルトは、adm31、ansi、dg200、dg210、hz1500、mc5、microterm、multiterm、pcterm、tvi、vp-a2、vp-60、 vt52、vt100、vt220、wyse30、wyse50、wyse60、または wyse75 の端末装置に使用されます。 ターミナル・タイプが内部デフォルト・テーブルにない場合、ditty はターミナル・タイプの terminfo エントリーを読み取り、透過印刷のオン/オフ文字列を、terminfo エントリーにある mc5/mc4 属性で指定された値に設定します。

非同期 Point-to-Point Protocol (PPP) サブシステム

非同期 PPP サブシステムは SLIP に代わる機能を提供します。

PPP は、2 地点間メディアを介してマルチプロトコル・データグラムを移送するための標準メソッドを提供します。 **PPP** は、次の主要な 3 つの層からなっています。

1. マルチプロトコル・データグラムをカプセル化するためのメソッド。 **PPP** は、TCP/IP ネットワーク層プロトコルをサポートします。
2. データ・リンクの接続の確立、構成、テストを行うための**リンク制御プロトコル (LCP)**。 **PPP** は、ストリーム・カーネル・エクステンション機能によってこれをインプリメントします。
3. さまざまなネットワーク層プロトコルの確立と構成を行うための**ネットワーク制御プロトコル (NCP)** ファミリー。 **PPP** は、TCP/IP 接続の折衝のための **Internet Protocol Control Protocol (IPCP/IPv6CP)** をサポートします。

この **PPP** のインプリメンテーションは、次の Request For Comments (RFC) をサポートしています。

- RFC 1661、*Point-to-Point Protocol*、*LCP*
- RFC 1332、*PPP Internet Protocol Control Protocol (IPCP) (PPP* インターネット・プロトコル制御プロトコル *(IPCP))*
- RFC 1662、*PPP in HDLC-like Framing (HDLC* に似たフレームの *PPP)*
- RFC 1334、*PPP Authentication Protocols (PPP* 認証プロトコル*)*

- RFC 1990、*PPP Multilink (PPP 多重リンク)*
- RFC 2472、*IP Version 6 over PPP (PPP における IP バージョン 6)*

PPP は、クライアントとサーバーを区別します。 このオペレーティング・システムは、クライアントとサーバーのどちらとしても機能できます。 ただし、構成を単純化するために区別があります。 **PPP** サーバーには、生成中の接続の間に IP/IPv6CP アドレスのプールを割り当てる傾向があります。 メディア・デバイスの間には、相関関係があります。 この **PPP** のインプリメンテーションは、この相関関係を破っています。 すべてのサーバー **PPP** 接続は、最初に使用可能であるかどうかに基づいて割り当てられます。 このため、**PPP** とメディアの分離が容易になります。 接続プロセスは、適切なタイプのリンクへリンクされることを要求しなければなりません。

PPP ユーザー・レベル・プロセス

このオペレーティング・システム上の**非同期 Point-to-Point Protocol** は、次の 3 つのユーザー・レベル・プロセスを使用します。

1. 制御デーモン (**pppcontrold**)。これは、root によってシステム・リソース・コントローラーの下で実行されます (**startsrc -s pppcontrold**)。 制御デーモンの機能には、このサブシステムに関連したすべてのカーネル・エクステンション機能のロードと構成が含まれます。 制御デーモンは、オペレーティング・システムが **PPP** 機能を必要とする限り、実行された状態で残ります。
2. 接続プロセス (**pppattachd**)。これは、TTY ストリームを**リンク制御プロトコル**、**ネットワーク制御プロトコル**、データグラム・プロトコルのインスタンスへバインドします。 **pppattachd** のインスタンスは、システム内のアクティブな **PPP** 接続ごとに 1 つずつ存在します。 接続プロセスのユーザーは、uucp グループに属していなければならず、/usr/sbin を PATH 環境変数の中に含んでいなければなりません。
3. ダイヤラー・プロセス (**pppdial**)。これは、発信接続を確立します。 ダイヤラーは、**pppattachd** によってコネクター・プログラムとして実行されることを想定しています。 ダイヤラーの目的は、**PPP** 折衝に先立って非同期デバイスを通じて対話することです。 この対話は、UUCP チャット対話フォーマットと同じように定義されています。 ダイヤラー機能は、リモート・システムとの接続の確立を支援するためのものです。 実際のセッション確立は、**PPP** の有効範囲外です。

非同期 Point-to-Point Protocol の構成

SMIT を使用して**非同期 Point-to-Point Protocol** を構成できます。

次の表は、システムを構成するときに必要なすべてのタスクを示しています。 この表のタスクを実行するには、root 権限を持っていなければなりません。

システムを初期構成するときには、この表から少なくとも次のタスクを選択することになります。

- リンク構成の追加
- サーバー・インターフェースの 追加 (そのマシンを PPP サーバーとしてセットアップする 場合)
- 要求インターフェースの 追加 (そのマシンにデマンド接続をサポートさせる場合)
- PAP ユーザー/パスワードの操作、または CHAP ユーザー/パスワードの操作 (そのマシンに **PPP** 認証をサポートさせる場合)
- 変更を有効にするための **PPP** の開始 (または **PPP** が現在実行中であれば **PPP** の停止後に **PPP** の開始)

表 *109.* 非同期 *PPP* の構成タスク

タスク	**SMIT 高速パス**
リンク制御の構成の作成	`smit ppplcp`
リンク構成の追加	`smit addlcp`
リンク構成の変更/表示	`smit chglcp`
リンク構成の除去 [1]	`smit rmlcp`
PPP IP インターフェースの 作成	`smit pppip`
サーバー・インターフェースの 追加	`smit addpppserver`

表 *109.* 非同期 *PPP* の構成タスク *(*続き*)*	
タスク	**SMIT 高速パス**
サーバー・インターフェースの 変更/表示	`smit listserver`
サーバー・インターフェースの 除去 [1]	`smit rmlistserver`
要求インターフェースの 追加	`smit addpppdemand`
要求インターフェースの 変更/表示	`smit listdemand`
要求インターフェースの 除去 [1]	`smit rmlistdemand`
PAP ユーザー/パスワードの操作	`smit ppppap`
PAP ユーザーの追加	`smit addpapuser`
PAP ユーザーの変更/表示	`smit listpapuser`
PAP ユーザーの除去	`smit rmpapuser`
CHAP ユーザー/パスワードの操作	`smit pppchap`
CHAP ユーザーの追加	`smit addchapuser`
CHAP ユーザーの変更/表示	`smit listchapuser`
CHAP ユーザーの除去	`smit rmchapuser`
PPP[2] の開始	`smit startppp`
PPP[3] の停止	`smit stopppp`
PPP IPv6 インターフェース	`smit pppipv6`
PPP IPv6 サーバー・インターフェースの 追加	`smit addpppv6server`
PPP IPv6 インターフェースの 表示または変更	`smit listv6server`
PPP IPv6 インターフェースの 除去	`smit rmlistv6server`
PPP IPv6 クライアント・インターフェースの 追加	`smit addpppv6client`
PPP IPv6 クライアント・インターフェースの 表示または変更	`smit listpppv6client`
PPP IPv6 クライアント・インターフェースの 除去	`smit rmlistpppv6client`
PPP IPv6 要求インターフェースの 追加	`smit addpppv6demand`
PPP IPv6 要求インターフェースの 表示または変更	`smit listpppv6demand`
PPP IPv6 要求インターフェースの 除去	`smit rmlistpppv6demand`
PPP IP および IPv6 インターフェース	`smit pppipv4_6`
PPP IP/IPv6 サーバー・インターフェースの 追加	`smit addpppv4_6server`
PPP IP/IPv6 インターフェースの 表示または変更	`smit listv4_6server`
PPP IP/IPv6 インターフェースの 除去	`smit rmlistv4_6server`
PPP IP/IPv6 クライアント・インターフェースの 追加	`smit addpppv4_6client`
PPP IP/IPv6 クライアント・インターフェースの 表示または変更	`smit listpppv4_6client`
PPP IP/IPv6 クライアント・インターフェースの 除去	`smit rmlistpppv4_6client`

表 109. 非同期 PPP の構成タスク (続き)	
タスク	**SMIT 高速パス**
PPP IP/IPv6 要求インターフェースの 追加	`smit addpppv4_6demand`
PPP IP/IPv6 要求インターフェースの 表示または変更	`smit listpppv4_6demand`
PPP IP/IPv6 要求インターフェースの 除去	`smit rmlistpppv4_6demand`

注 :

1. このタスクを選択すると、既存の情報が破棄されます。
2. **PPP** を開始する別の方法は、**startsrc -s pppcontrold** コマンドを入力することです。 しかし、SMIT インターフェースを 使用すると、ブート時に **PPP** を開始するように設定することもできます。
3. **PPP** を停止する別の方法は、**stopsrc -s pppcontrold** コマンドを入力することです。 しかし、SMIT インターフェースを 使用すると、ブート時に **PPP** を開始しないように設定することもできます。

PPP SNMP の使用可能化

PPP は、TCP/IP SNMP デーモンと対話して、PPP リンク層構成情報とアクティブな **リンク制御プロトコル (LCP)** インターフェースに 関する情報を報告できます。

TCP/IP SNMP と SNMP 管理ソフトウェアの両方が正しく構成されていれば、**PPP** SNMP は次のことができます。

- **PPP** リンク構成情報 (最大受信単位サイズ、非同期文字マッピングなど) の検索。
- **PPP** リンク構成情報の設定。
- アクティブな LCP リンクに関する LCP インターフェース 情報の検索。
- アクティブな LCP リンクの状態は、適切な **ifAdminStatus** 管理情報ベース (MIB) オブジェクトを設定することによって「down (下に移動)」に変更できます。

RFC1471 によって **PPP** MIB 用に定義されているすべてのオブジェクトがサポートされているわけではありません。 **PPP** サブシステムには **pppLink** テーブルしか適用できないので、**pppLqr** と **pppTests** の部分はサポートされません。 **pppLink** 部分は、次の例外付きでサポートされます。

- **pppLinkConfigMagicNumber** オブジェクトが読み取り専用。 **PPP** では、マジック・ナンバーの折衝は常に実行され、使用不可にはできません。
- **pppLinkConfigFcsSize** オブジェクトが読み取り専用。 このオペレーティング・システムで **PPP** がサポートする FCS サイズは、16 のみです。

デフォルトでは、**PPP** 用の SNMP は使用不可になっています。 **PPP** SNMP を使用可能にする場合、次の手順を使用できます。この手順を実行するには、root 権限を持っていなければなりません。

注 : 次の手順は、**PPP** リンク構成が既に設定されていることを想定しています。 まだ設定されていない場合は、**PPP** SNMP を使用可能にする前に、680 ページの『非同期 Point-to-Point Protocol の構成』 の手順を実行してください。

1. SMIT インターフェースを 開始し、次のように入力して「Change/Show a Link Configuration (リンク構成の変更/表示)」画面を表示します。

   ```
   smit chglcp
   ```

2. 「Enable **PPP** SNMP subagent (**PPP** SNMP のサブエージェントを使用可能にする)」フィールドを「yes (はい)」に切り替えます。
3. 変更を受け入れ、SMIT を終了します。

PPP SNMP は、**PPP** が再始動されるまで使用可能になりません。

- **PPP** が現在実行中の場合は、次の作業を行います。
  1. `smit stopppp` 高速パスを使用して **PPP** を停止する (680 ページの『非同期 Point-to-Point Protocol の構成』 の表を参照)。

2. サブシステムがシャットダウンしたかどうかを定期的に検査するため、次のように入力します。

   ```
   lssrc -s pppcontrold
   ```

   サブシステムを完全に停止するために要する時間は、**PPP** 構成の中で定義されたリンクの数によって異なります。 サブシステムが、このコマンドの出力で「inoperative」という状況が表示された時点で完全にシャットダウンされます。
3. smit startppp 高速パスを使用して **PPP** を停止する (680 ページの『非同期 Point-to-Point Protocol の構成』の表を参照)。

- **PPP** が現在実行中でない場合は、smit startppp 高速パスを使用して **PPP** を開始します (680 ページの『非同期 Point-to-Point Protocol の構成』の表を参照)。

シリアル回線インターネット・プロトコル

シリアル回線インターネット・プロトコル (SLIP) は、TCP/IP でシリアル接続による操作をするときに使用されるプロトコルです。

これは一般に、1200bps から 19.2Kbps 以上の速度で作動する専用シリアル・リンクとダイヤルアップ接続で使用されます。

注 : 38400 より高いボー・レートを使用する場合は、望ましい TTY 用の /etc/uucp/Devices ファイルにボー・レート 50 を指定し、その TTY が望ましい実際のボー・レートを反映するように、その TTY の SMIT 構成を変更します。

例えば、**cu** コマンドを、ボー・レート 115200 の tty0 で実行する場合は、以下の手順で行います。

1. ハードウェアがこのボー・レートをサポートしていることを確認します。
2. /etc/uucp/Devices を編集して以下の行を組み込みます。

   ```
   Direct tty0 - 50 direct
   ```
3. smit chtty 高速パスを入力します。
4. tty0 を選択します。
5. ボー・レートを 115200 に変更します。
6. SMIT を終了します。

SLIP の構成

以下は、**SLIP** の構成時に実行すべき 2 つの推奨ステップです。

この 2 つのステップの方法を使用すると、ハードウェアとマシンに依存した構成要件が、SLIP ソフトウェアとコマンド構文の問題から分離されます。

1. ATE または **cu** ユーティリティーを 使用して、リモート・システムに正常にログインします。

   この結果、物理リンクの使用可能性と正しさが判明します。

   SLIP リンクに関係するモデムは、セットアップ・フェーズで問題の原因となることが最も多いので、これらの操作容易性を検査することは重要なことです。
2. ATE コマンドまたは **cu** コマンドを使用して、リモート・システムへのエラー・フリーのログインを確立した後、**SLIP** の構成を開始することができます。

SLIP モデムの考慮事項

SLIP 用のモデムを構成する際は、通信リンクの両端でこれらの変更を行う必要があります。

ローカル・モデムとリモート・モデムの両方をまったく同じように構成する必要があります。

1. モデムは、DTR の存在を認知する必要があります。

   ローカル・モデムに問い合わせをするとき、DTR が想定または無視されている場合、モデムはハングアップを実行できません。 これは、回線をクローズするか、もう一方の端末からのキャリアの消失を認識してからハングアップすることしかできません。 つまり、切断は、もう一方の端末によってきっかけが

発生したときにのみ行われます。 AT コマンド &D2 または &D3 は、大半の Hayes 互換モデムに適切な設定です。

2. モデムはデータ・キャリア検出 (DCD) を強制、想定、または無視してはなりません。

   DCD は実際の状態を追跡する必要があります。 このことは、交換電話回線での他方の端末 (モデム) への正式な接続の後もキャリアが存続することを意味します。 これは、専用回線にも当てはまります。 &C1 は、大半の Hayes 互換モデムに対する推奨設定です。

3. モデムは送信可 (CTS) 信号を強制、想定、または無視してはなりません。

   CTS は送信要求 (RTS) を追跡する必要があります。 CTS が強制的に true になっている場合、**getty** がポート上に置かれていたり、RTS フロー制御プロトコルがポートに追加されていたりすると、ポート・オープンは失敗します。

4. モデムは、**slattach** ダイヤル試行中に問題が発生したら、自動繰り返し要求 (ARQ) コードをオフにするように構成する必要があります。

   モデムが、**slattach** ダイヤルイン試行時の接続に繰り返し失敗する場合は、モデム構成を調べ、ARQ コードが現在オンであれば、それをオフにする必要があります。 大半の Hayes 互換モデムでは、これは &A0 設定です。

   ARQ 結果コードを使用不可にしても、エラー制御された接続が影響を受けたり、モデムが (結果コードが使用可能の場合に) **slattach** ダイヤル文字列の必要に応じた標準 CONNECT メッセージが戻されなくなることはありません。

5. ECL (Error Checking on the Link) は重要です。

   BOTH モデムまたは NEITHER モデムがこれを使用できます。 通常は、両方のモデムが、接続セッション中のこの使用法について合意している必要があります。 ECL が選択されている場合、物理電話回線は、**SLIP** リンクを介して送信された最後のデータに対する応答パケットを待っている間、TCP/IP タイマーが満了する前にデータ・エラーからのリカバリーができるものでなければなりません。

6. リンク上でのデータ圧縮

   リンク上でのデータ圧縮の使用は、それが完全にモデムによって処理されるのであれば可能です。 **SLIP** は、どのようなタイプの圧縮も行いません。 データ圧縮を呼び出す場合には、まったく同じタイプの 2 つのモデムを用意することをお勧めします。そうすれば、必ずそれぞれが同じ方法および同じ時間枠で圧縮を実行することになります。

cu コマンドを使用した手動でのモデム・プログラミング

システム装置に接続されたモデムを手動でプログラミングする場合は、以下の手順で行います。

- UNIX-to-UNIX Copy Program (UUCP) がシステムにインストールされている 必要があります。 インストールを確認するには、**lslpp -f | grep bos.net.UUCP** コマンドを使用してください。
- モデムがシステムに 接続され、電源がオンになっている必要があります。
- 該当のファイルを変更するために、root ユーザー権限が必要です。

1. /etc/uucp/Devices ファイルに、以下の行がなければ、この行を追加します (# はポート番号で置き換えてください)。

   ```
   Direct tty# - Any direct
   ```

   注: Devices ファイル内で、左端の列が # 記号で始まる行はすべてコメントです。

2. ファイルを保管して終了します。
3. コマンド・ラインで以下のコマンドを入力します。

   ```
   cu -ml tty#
   ```

4. モデムが接続されていて、プログラム可能であることを示す接続メッセージが画面上に表示されます。
5. AT と入力して、Enter キーを押します。

   モデムは「OK」と応答します。 モデムから応答がない場合や、入力した文字が画面上に表示されない場合は、以下の確認をしてください。

- モデムのケーブル接続を確認します。
- モデムがオンになっていることを確認します。
- Enter キーを押したときのモデムのフロントパネルのライトを観察します。 受信データ (RD) と送信データ (SD) のライトが点滅している場合、モデムはシステムと 通信しており、問題は現在のモデムの設定にあると考えられます。 ライトが点滅していない場合は、モデム接続に問題があります。
- 以下を入力して、状態が変化するかどうかを確認します。

```
ATE1 <enter>
ATQ0 <enter>
```

ATE1 はエコー・モードをオンにし、その結果、入力された文字が画面上に表示されます。 ATQ0 によって結果コードの表示が可能になります。

6. 前のセクション『モデムに関する考慮事項』に示されている設定を使用してモデムをプログラミングします。
以下の例は、Hayes 互換モデムの基本設定をプログラミングして保管する方法を示しています。 次のように入力してください。

```
AT&F    <enter>
AT&D2   <enter>
ATS0=1  <enter>
ATS9=12 <enter>
AT&C1   <enter>
AT&W    <enter>
~.      <enter>
```

&F は、モデムを出荷時のデフォルトにリセットする場合に使用され、&D2 は DTR、S0、および S9 設定登録値を設定し、&C1 はキャリアを設定し、&W はその設定値をモデムに書き込みます。 ティルドとピリオド (~.) は、接続を終了します。

自動モデムの構成

モデムは手動でカスタマイズするか、あるいは **cu** ユーティリティーとその 関連ファイルを使用して、自動モデム構成スクリプトを作成することができます。

- UUCP がシステムにインストールされている 必要があります。 インストールを確認するには、**lslpp -f | grep bos.net.UUCP** コマンドを使用してください。
- モデムがシステムに 接続され、電源がオンになっている必要があります。
- モデムの AT コマンド・ストリング (例えば、at&f&c1&d3) が存在している必要があります。 **cu** コマンドを使用してコマンド・ストリングをまず手動で試行し、それから自動モデム構成を実行してください。
- 該当のファイルを変更するために、root ユーザー権限が必要です。

以下の例は、tty0 に接続された Telebit T3000 モデムを自動構成する方法を示しています。

1. /etc/uucp/Systems ファイルを編集します。
2. ファイルの末尾に以下の行を追加します。 この項目は、ファイルの左端の列で始まっていなければなりません。

```
telebit Nvr TELEPROG 19200
```

3. ファイルを保管して終了します。
4. /etc/uucp/Devices ファイルを編集します。
5. ファイルの末尾に以下の行を追加します。 この項目は、ファイルの左端の列で始まっていなければなりません。

```
TELEPROG tty0 - 19200 TelebitProgram
```

6. ファイルを保管して終了します。
7. /etc/uucp/Dialers ファイルを編集します。
8. ファイルの末尾に以下の行を追加します。
この項目は、ファイルの左端の列で始まっていなければなりません。

注 : 以下の 4 行は、1 つの長い行として入力する必要があります。

```
TelebitProgram  =,-,    "" ¥dAT&F¥r¥c OK
ats0=1s2=255s7=60s11=50s41=2s45=255s51=252s63=1s58=2s64=1¥r¥c OK
ATs69=2s105=0s111=30s255=0M0&C1Q2&D3&Q0&R3&S1&T5¥r¥c OK
ATE0X12&W¥r¥c OK
```

9. ファイルを保管して終了します。
10. 自動構成を開始するため、以下のコマンドを入力します。

    ```
    cu -d telebit
    ```

    このコマンドは、システムに接続していないために失敗します。 コマンドのデバッグ出力を調べて、ATE0X12&W がモデムに送信されていることと、OK が受信されていることを確認します。 これが確認できれば、モデムは正常にプログラミングされています。

Dialers ファイル内にある誤った値のため、またはモデムの既存の構成のために問題が発生することがあります。 その場合は、モデムを手動でプログラミングして、(ステップ 8 の) dialers 文字列を入力してみてください。

モデムによる SLIP の構成

モデムを介して通信する 2 つのシステム間で**シリアル回線インターフェース・プロトコル (SLIP)** を構成する場合、次の手順を使用できます。これにより、システム管理インターフェース・ツール (SMIT) インターフェースとコマンド・ラインを切り替えて構成を完了することができます。

分かりやすくするため、次の説明では、2 つのホストに bronze と gold という名前を使用します。

1. モデムを bronze と gold に物理的に接続します。
2. SMIT を使用して bronze 上に tty を作成するには、以下の手順を実行します。
   a) 次のように入力します。

   ```
   smit maktty
   ```

   b) 作成する tty のタイプとして **rs232** を選択します。
   c) 使用可能なシリアル・ポートを選択します。例えば sa0 (システム・シリアル・ポート 1)。
   d) この tty 用のポート番号をリストから選択します。
   e)「BAUD rate (ボー・レート)」を、使用するモデムのボー・レートに設定します。
   f)「Enable LOGIN (ログインを可能にする)」を使用不可に設定します。
   g) SMIT を終了します。
3. gold 上で tty を作成します。

   bronze について行ったのと同じ手順 (ステップ 2) に従ってください。ただし、「Enable LOGIN (ログインを可能にする)」は**使用可能**に設定してください。

   この手順の残りの部分では、bronze 上と gold 上の tty 番号が両方とも tty1 であると想定しています。
4. ATE を使用して物理接続をテストします。
   a) bronze 上で次のように入力します。

   ```
   ate
   ```

   b)「Unconnected Main Menu (未接続のメイン・メニュー)」で、「**Alter (変更)**」サブコマンドを選択します。 「Rate (速度)」を、使用しているモデムのボー・レートに設定し、「Device (デバイス)」を tty1 に設定します。
   c)「Unconnected Main Menu (未接続のメイン・メニュー)」で、「**Connect (接続)**」サブコマンドを選択します。 ATE が電話番号の入力を求めるプロンプトを表示したら、gold の電話番号を入力して Enter キーを押します。
   d) この時点で、gold のログイン・プロンプトが表示されます。 ログインします。

e) 接続画面に戻り、gold からログアウトし、Ctrl-v を押して「ATE CONNECTED MAIN MENU (ATE 接続メインメニュー)」を表示させ、T キーを押して接続を終了し、次に Q キーを押して ATE を終了します。

注 : ログイン・プロンプトが表示されない場合は、ステップ 1 に戻って構成が正しいかどうかを検査します。 gold にログインできるまで、先へ進まないでください。

5. ATE 用の tty の構成は、SLIP 用の場合の構成とは少し異なるので、次の変更を加えなければなりません。
   a) bronze 上で次のように入力します。

   ```
   smit chgtty
   ```

   b) gold 上で次のように入力します。

   ```
   smit chgtty-pdisable tty1
   ```

   c)「**tty1**」を選択してから、「**Change/Show TTY Program (TTY プログラムの変更/表示)**」を選択します。
   d)「Enable LOGIN (ログインを可能にする)」を使用不可に設定し、SMIT を終了します。
6. bronze と gold の両方で、`/usr/lib/uucp/Devices` ファイルに次の行を追加します。

   ```
   Direct tty1 - 9600 direct
   ```

   または、使用しているモデム・スピードで、`9600` を置き換えます。
7. bronze 上で、**SLIP** ネットワーク・インターフェースを 作成します。
   a) 次のように入力します。

   ```
   smit mkinet1sl
   ```

   b)「TTY PORT for SLIP Network Interface (SLIP ネットワーク・インターフェースの TTY ポート)」に対して「**tty1**」を選択します。
   c)「INTERNET ADDRESS (インターネット・アドレス)」を指定します (例えば、130.130.130.1)。
   d)「DESTINATION address (宛先アドレス)」(gold の) を指定します (例えば、130.130.130.2)。
   e) モデムの「BAUD rate (ボー・レート)」を指定します。
   f)「DIAL STRING (ダイヤル文字列)」を、例えば、次のように指定します。
   - "" AT OK ATDT555-1234 CONNECT ""
   - このコマンドの意味は次のとおりです。**tty1** を 9600 ボーで使用します。 モデムへ AT を送信します。 モデムは「OK」と応答します。 電話番号 555-1234 をダイヤルします。 モデムは CONNECT と応答します。 "" の文字の前後にあるスペースも必ず入力してください。

   g) SMIT を終了します。
8. gold 上で、**SLIP** ネットワーク・インターフェースを 作成します。
   bronze について行ったのと同じ手順 (ステップ 5) に従ってください。ただし、「INTERNET ADDRESS (インターネット・アドレス)」と「DESTINATION address (宛先アドレス)」を入れ替えてください。
9. bronze と gold の両方で、**/etc/hosts** ファイルに次の 2 つの項目を追加します。

   ```
   130.130.130.1   bronze
   130.130.130.2   gold
   ```

   割り当てる名前は固有のものでなければなりません。 つまり、bronze 上のトークンリング・インターフェースに、既に `bronze` という名前が割り当てられている場合は、**SLIP** インターフェースに `bronze_slip` などの名前を割り当てます。

   注 : slattach コマンドへの簡易インターフェースの 場合は、スクリプト `/usr/sbin/slipcall` を使用します。
10. **SLIP** 接続をテストします。

a) bronze 上で次のように入力します。

```
ping gold
```

b) gold 上で次のように入力します。

```
ping bronze
```

両方のテストに合格した場合は、**SLIP** 接続を使用する準備が整いました。 合格しなかった場合は、ステップ 5 に戻って、bronze と gold の両方の構成が正しいかどうかを検査してください。

null モデム・ケーブルによる SLIP の構成

ヌル・モデム・ケーブルを使用して接続された 2 つのシステム間で **SLIP** を構成する場合、次の手順を使用できます。これにより、システム管理インターフェース・ツール (SMIT) インターフェースとコマンド・ラインを切り替えて構成を完了することができます。

分かりやすくするため、この説明では、2 つのホストに bronze と gold という名前を使用します。

1. bronze と gold を null モデム・ケーブルによって物理的に接続します。
   以下のケーブルが必要です。 (このケーブルのリストは、bronze から gold へ接続する順に示してあります。)
   a) ケーブル B (部品番号 00G0943)。 シリアル・ポート・ジャンパー・ケーブル。1 システムにつき 2 本ずつ提供されますが、モデル 220、340、350 には必要ありません。
   b) ケーブル D (部品番号 6323741、フィーチャー・コード 2936)。 非同期ケーブル EIA-232/V.24。
   c) ケーブル E (部品番号 59F2861、フィーチャー・コード 2937)。 プリンター/ターミナル・インターポーザー EIA-232 (null モデム・ケーブル)。
   d) チェンジャー・アダプター (アダプターの両端がソケットになっています)。
2. bronze 上で tty を作成します。
   a) 次のように入力します。

   ```
   smit maktty
   ```

   b) 作成する tty のタイプとして **rs232** を選択します。
   c) 使用可能なシリアル・ポートを選択します。例えば **sa0** (システム・シリアル・ポート 1)。
   d) この tty 用のポート番号をリストから選択します。
   e)「BAUD rate (ボー・レート)」を 19200 に設定します。 (後に、これを 38400 に変更します。 ただし、現時点では 19200 を使用します。)
   f)「Enable LOGIN (ログインを可能にする)」を使用不可に設定し、SMIT を終了します。
3. gold 上で tty を作成します。 bronze について行ったのと同じステップ (ステップ 2) に従ってください。ただし、「Enable LOGIN (ログインを可能にする)」は**使用可能**に設定してください。

   注 : この手順の残りの部分では、bronze 上と gold 上の tty 番号が両方とも tty1 であると想定しています。
4. ATE を使用して物理接続をテストします。
   a) bronze 上で次のように入力します。

   ```
   ate
   ```

   b)「Unconnected Main Menu (未接続のメイン・メニュー)」で、「**Alter (変更)**」サブコマンドを選択します。 「Rate (速度)」を 19200 に設定し、「Device (デバイス)」を tty1 に設定します。
   c)「Unconnected Main Menu (未接続のメイン・メニュー)」で、「**Connect (接続)**」サブコマンドを選択します。
   ATE が電話番号の入力を求めるプロンプトを表示したら、Enter キーを押します。 次のメッセージが表示されます。

   ```
   ate: 0828-010 The Connect command has made a connection through port tty1
   ```

d) Enter キーを押します。

gold のログイン・プロンプトが表示されます。 gold にログインしてください。

e) 最後に、接続画面に戻って gold からログアウトし、Ctrl-v を押して「ATE CONNECTED MAIN MENU (ATE 接続メインメニュー)」を表示させ、T キーを押して接続を終了し、次に Q キーを押して ATE を終了します。

注 : ログイン・プロンプトが表示されない場合は、ステップ 1 に戻って構成が正しいかどうかを検査します。 gold にログインできるまで、先へ進まないでください。

5. ATE 用の tty の構成は、**SLIP** 用の場合の構成とは少し異なるので、次の変更を加えなければなりません。

a) bronze 上で次のように入力します。

```
smit chgtty
```

b)「**tty1**」を選択します。 「BAUD rate (ボー・レート)」を 38400 に設定し、SMIT を終了します。

c) gold 上で次のように入力します。

```
pdisable tty1
```

d) gold 上で次のように入力します。

```
smit chgtty
```

e)「**tty1**」を選択します。 「Enable LOGIN (ログインを可能にする)」を使用不可に設定し、「BAUD rate (ボー・レート)」を 38400 に設定してから SMIT を終了します。

6. bronze と gold の両方で、/usr/lib/uucp/Devices ファイルに次の行を追加します。

```
Direct tty1 - 38400 direct
```

7. **bronze** 上で、**SLIP** ネットワーク・インターフェースを 作成します。

a) 次のように入力します。

```
smit mkinet1sl
```

b)「TTY PORT for SLIP Network Interface (SLIP ネットワーク・インターフェースの TTY ポート)」に対して「**tty1**」を選択します。

c)「INTERNET ADDRESS (インターネット・アドレス)」を指定します (例えば、130.130.130.1)。

d)「DESTINATION address (宛先アドレス)」(gold の) を指定し (例えば 130.130.130.2)、「OK」をクリックするか Enter キーを押します。

8. gold 上で、**SLIP** ネットワーク・インターフェースを 作成します。 bronze について行ったのと同じ手順 (ステップ 5) に従ってください。ただし、「INTERNET ADDRESS (インターネット・アドレス)」と「DESTINATION address (宛先アドレス)」を入れ替えてください。
9. bronze と gold の両方で、/etc/hosts ファイルに次の 2 つの項目を追加します。

```
130.130.130.1   bronze
130.130.130.2   gold
```

割り当てる名前は固有のものでなければなりません。 つまり、bronze 上のトークンリング・インターフェースに、既に bronze という名前が割り当てられている場合は、**SLIP** インターフェースに bronze_slip などの名前を割り当てます。

10. bronze と gold の両方で **SLIP** を開始します。 次のように入力します。

```
slattach tty1
```

11. **SLIP** 接続をテストします。

a) bronze 上で次のように入力します。

```
ping gold
```

b) gold 上で次のように入力します。

```
ping bronze
```

両方のテストに合格した場合は、**SLIP** 接続を使用する準備が整いました。 合格しなかった場合は、ステップ 5 に戻って、bronze と gold の両方の構成が正しいかどうかを検査してください。

SLIP 接続の非活動化

SLIP 接続を非活動化するには、以下の手順を使用します。

1. 次のように入力します。

```
ps -ef | grep slatt
```

slattach コマンドと関連付けられているプロセスのプロセス番号を書き留めておいてください。

2. プロセス番号ごとに、次のように入力します。

```
kill process_number
```

kill コマンドの -9 フラグを使用しないでください。

slattach を誤って -9 フラグを使用して強制終了した場合、SLIP ロックが /etc/locks の中に残る可能性があります。 **slattach** のあと、クリーンアップのためにこのロック・ファイルを削除してください。

SLIP 接続を一時的に非活動にするには、ローカル・システムとリモート・システムの両方で以下を行います。

1. 次のように入力します。

```
ifconfig sl# down
```

2. 次のコマンドを使用して、現在実行している **slattach** プロセスをリストします。

```
ps -ef | grep slat
```

出力は以下のようになります。

```
root  1269  1    0   Jun 25  ... slattach
```

3. **slattach** プロセスを、そのプロセス ID を使用して強制終了します。 例えば、上記の例の **slattach** プロセスを強制終了する場合は、次のように入力します。

```
kill 1269
```

1269 は **slattach** プロセス ID です。 **kill** コマンドの -9 フラグを使用して **slattach** プロセスを除去することはしない でください。

この時点でこの **SLIP** 接続は使用不可となります。

SLIP 接続の活動化

一時的に使用不可にされた **SLIP** 接続を活動化するには、以下の手順を使用します。

以下のコマンドをローカル・システムとリモート・システムの両方で実行します。

1. 次のように入力します。

```
ifconfig sl# up
```

2. 最初に使用された **slattach** コマンドを再実行します。

SLIP インターフェースの除去

SLIP インターフェースを 完全に除去するには、以下の手順を使用します。

これらの手順を実行すると、sl# インターフェースとその 関連 **slattach** プロセスの両方が除去されます。 /etc/hosts ファイル内に作成された項目は残るので、手動で除去する必要があります。

1. **SLIP** インターフェースとその 関連 **slattach** プロセスを除去するため、`smit rminet` 高速パスを使用して、「**Available Network Interfaces (使用可能なネットワーク・インターフェース)**」画面にアクセスします。
2. 「**Available Network Interfaces (使用可能なネットワーク・インターフェース)**」画面から適切な項目を選択し、「**Do (実行)**」を選択します。

   注: /etc/hosts ファイル内に作成された項目は残るので、手動で除去する必要があります。

SLIP トラブルシューティング

SLIP の問題をデバッグするには、以下のコマンドが必要です。

各コマンドは、**SLIP** の問題をトラブルシューティングするためにそのコマンドを 使用する例とともに示されています。

さらに、共通問題とエラー・メッセージも参照できます。

netstat コマンド

netstat コマンドは、**ifconfig** コマンドと併用され、TCP/IP ネットワーク・インターフェースの 状況条件を提供します。

例えば、コマンド **netstat -in** は、-i フラグを使用してネットワーク・インターフェースについての 情報を表示しています。一方、-n フラグはホスト名の代わりに IP アドレスを印刷します。 SLIP インターフェース、アドレス、およびホスト名を確認する場合は、このコマンドを使用します。 以下のセクションでは、**netstat -in** の出力について説明します。

683 ページの『SLIP モデムの考慮事項』のセクションに示されている設定を使用して、モデムをプログラミングします。 以下の例は、Hayes 互換モデムの基本設定をプログラミングして保管する方法を示しています。 次のように入力します。

```
Name  Mtu   Network    Address              Ipkts     Ierrs Opkts Oerrs  Col
lo0   1536  <Link>                          2462       0    2462   0     0
lo0   1536  127        localhost.austi      2462       0    2462   0     0
tr0   1492  <Link>                          1914560    0   21000   0     0
tr0   1492  129.35.16  glad.austin.ibm      1914560    0   21000   0     0
sl0   552   1.1.1.0    1.1.1.1              48035      0   54963   0     0
sl1*  552   140.252.1  140.252.1.5          48035      0   54963   0     0
```

sl1 インターフェースの 横の * に注目してください。 これは、このネットワーク・インターフェースが 停止状態または使用不可であることを示しています。 これが有効な **SLIP** インターフェースであれば、これは **ifconfig sl1 up** コマンドを実行することによって訂正できます。

netstat は、**SLIP** 接続のトラブルシューティング 時に役立つ、入力パケット・カウントと出力パケット・カウント、および入力エラーと出力エラーについての統計を提供します。

例えば、ユーザーが **SLIP** リンクを介してリモート・ホストに対する **ping** を実行し、その **ping** コマンドが停止しているように見えます。 ユーザーがすぐに別のコマンド・シェルから **netstat -in** コマンドを実行し、Opkt は増えているのに、リモート・ホストからの Ipkt がないことに気付きます。 これは、リモート・システムが情報を戻していない (または受信していない) ことを示しています。 これらのユーザーはリモート・システム 上で同じ **netstat** コマンドを実行して、**ping** パケットの受信かエラー・カウントの増加を確認する必要があります。

ホスト名対インターネット 番号の変換はネーム・レゾリューションに関係するので、**SLIP** 回線の正しい操作のために重要です。 ホスト名、別名、および経路指定の問題をデバッグするには、**netstat -rn** コマンドを使用します。 ホストまたはホスト名のベース名は、**/etc/hosts** ファイルから戻る唯一の名前です。 マシンが nameserver によってサービスを受けている (すなわち、/etc/resolv.conf が存在する) 場合は、ネーム・サーバーがこのコマンドに完全修飾ドメイン名を戻します。

ifconfig コマンド

ifconfig コマンドは、ネットワーク・インターフェース 構造をカーネル・メモリーにおいて動的に作成または削除できる、ネットワーク・インターフェース 構成ツールです。

このコマンドは、コマンド・ラインからデータを受け入れて、そのパラメーターに従ったメモリー構造を構築します。 デバッグの目的では、**ifconfig** コマンドは、通信インターフェースの 状況を検査するために使用されます。

注 : システムがリブートされると、**ifconfig** コマンドによるインターフェース 属性の変更はすべて失われます。

例えば、sl1 インターフェースの 現在の状況を検査するには、次のようにします。

1. **netstat -i** コマンドを入力し、適切な sl# インターフェース (例えば、sl0、sl1、sl2 など) を選択して出力を検査します。
2. **ifconfig sl#** コマンドを入力し、以下のキー・フィールドの ifconfig 出力を検査します。

項目	説明
POINTTOPOINT フラグ	このフラグは、作動可能 SLIP リンク上に常に存在していなければなりません。 存在しない場合、そのリンクは停止または切断状態にある可能性があります。 この状態が変化したかどうかを調べるには、**ifconfig sl# up** コマンドおよび **ifconfig sl#** コマンドをもう一度実行してみてください。
UP フラグ	これは、ネットワークの sl# インターフェースが 活動化されており、作動可能であることを示します。
RUNNING フラグ	これは、**slattach** コマンドが正常に終了したことを示します。 実際には、リンクがアクセスされ、ダイヤルが完了し、他方の端末が応答し、リモート端末が CARRIER DETECT 状況を戻しています。 このフラグは、CD 状況が発生すると実行ビットで更新されます。

pdisable コマンドおよび *lsdev* コマンド

SLIP 接続に使用される tty ポートは、使用不可または選択不可の状態でなければなりません。

tty1 用のポートが使用不可であることを確認するには、root ユーザー権限を取得して、以下のいずれかのコマンドを入力します。

- `lsattr -El tty1 -a login`

  このコマンドは、システムのオブジェクト ・データベース・マネージャー (ODM) に記録された、tty ポートの永久状態を表示します。 出力が `login disable` 以外の場合は、SMIT を使用して「enable LOGIN (ログインを可能にする)」フィールドを **disable** に変更します。

- `pdisable | grep tty1`

  このコマンドは、パラメーターを指定せずに使用すると、使用不可状態のすべての tty ポートを表示します。 この例では、**pdisable** が、不必要な出力を除去する **grep** コマンドにパイピングされています。 このコマンドの実行後に tty1 が表示されない場合、ポートは使用不可ではありません。

ps コマンド

ps コマンドは、アクティブ・プロセスに関する情報を標準出力に表示します。

このコマンドは、ネットワーク・インターフェースに tty 回線を割り当てるために使用される **slattach** プロセスの存在 (または不在) を確認するために使用します。

netstat -in が、インターフェースが 停止していることを示している場合は、**ps -ef | grep slat** コマンドを実行して、**slattach** プロセスが現在関連 tty ポート上で実行しているかどうかを調べる必要があります。 直接接続された **SLIP** インターフェースの 場合、中断接続は手操作による介入なしに自動的に再試行されます。 モデムで接続された **SLIP** インターフェースの 場合は、中断接続は手動でリダイヤルする必要があります。 **slattach** コマンド・ラインにダイヤル文字列を指定する場合は、コマンドとダイヤル文字列を再入力して中断接続をリストアする必要があります。

ping コマンドとモデム・ライト

ping コマンドとモデム・ライトは、**SLIP** 通信の問題をデバッグするために使用されます。

ping はマシンから送信されるエコー要求パケットであり、エコー応答パケットが戻されます。 管理者にモデム・ライトが見える場合、この一連のイベントは便利です。

例えば、ローカル・システムがエコー要求パケットを作成して、それをリモート・システムに送信します。 ローカル・モデムの送信データ (SD) ライトが発光します。 このことは、ローカル TCP/IP、**slattach**、および tty が情報をグループ化し、それをモデムからリモート・システムへ送信できたことを意味します。

リモート・モデムがそのパケットを受信し、受信データのライトが点滅しますが、この SD ライトは点滅しません。 このことは、リモート・システムがローカル・システムの ping 要求を送信 (またはリターン) できなかったことを意味します。 この結果、ローカル・システム上のユーザーは、**ping** コマンドが停止しており、この状態を終了するために Ctrl-C が必要であることを知ります。

この問題の最も一般的な原因は、一方または両方のモデムの XON/XOFF フロー制御の使用法ですが、システム上の経路指定またはアドレスの競合の可能性も見逃せません。

一般的な *SLIP* 問題とエラー・メッセージ

ここでは、一般的な **SLIP** 問題とエラー・メッセージ、その考えられる原因、および推奨されるユーザー処置を参照することができます。

メッセージ: 0821-296 Cannot set line discipline for `/dev/tty#` to slip.ioctl(TXSETLD). A system call received a parameter that is not valid.

考えられる原因: このタイプのエラーは通常 **slattach** プロセスを開始する際に発生し、誤った SLIP の構成が原因である可能性があります。 この問題の原因となることが最も多いことがらは、tty デバイス番号と sl インターフェース 番号の間の不一致です。 このことは、**slattach** の前に ifconfig が実行されていないことをシステムが報告したことの説明にもなります。

この問題は、**slattach** プロセスが誤ってドロップまたは強制停止されたときや、ユーザーが **SLIP** 接続を別の tty ポートに移動しようとして、sl# インターフェースをその tty に合わせて再構成し忘れたときにも発生します。 まだ実行しているかもしれない実行中の **slattach** プロセスがないか確認してください (例えば、`ps -ef | grep slat`)。

アクション: SLIP 用の tty デバイスは **/dev/tty24** ですが、ユーザーは sl0 インターフェースを 作成しました。 これは誤りです。 ユーザーは tty 番号 (tty24 および sl24) と一致する sl24 インターフェースを 作成する必要があります。 問題が続く場合は、sl インターフェースを 停止し (『SLIP インターフェースを 停止状態にする』を参照)、以下のコマンドを使用して接続を再構成する必要があります。

```
lsdev  -Cc if  -s SL
lsattr -El sl0
```

メッセージ:

network is not currently available

route to remote host not available

考えられる原因: これらのエラーはたいていの場合、ユーザーが **SLIP** リンクを介してホストに ping しようとして、リンクが正しく確立されなかったときに発生します。 最もよくある問題は、sl# インターフェースに関連した一方または両方の tty ポートが使用可能状態にあることです。 ホスト・システム間にアドレスまたは経路の競合がある可能性もあります。

アクション:

- `smit rminet` 高速パスを使用して、sl# インターフェースを 除去してください。これをローカルとリモートの両方の SLIP ホスト上で行う必要があります。
- 各 SLIP ホストに対して、以下を行います。
  1. `pdisable | grep tty#` を入力します。

2. 上記のコマンドの出力に、tty デバイスがリストされていない場合、tty は使用不可になっていません。 SMIT またはコマンド・ラインから tty を使用不可にしてください。 tty ポートを使用不可の状態にして、SMIT を使用し、両方のシステム上に **SLIP** インターフェースを 再作成してください。 問題が続く場合は、ネットワーク・アドレスと経路 (存在する場合) を確認してください。 アドレス、経路指定、およびインターフェースの 情報をすぐに表示するには、**netstat -ir** コマンドを使用します。

問題: リモート・サイトがローカル・ホストにダイヤルインするとき、ローカル・ホスト上のモデムが接続するが、ログイン・プロセスを完了しない。

考えられる原因: 2 つのモデムが接続し、ハンドシェークまたは接続情報の交換を開始した後に切断した場合は、モデムの結果コードが問題の原因である可能性があります。 この問題は、不適切な **slattach** ダイヤル文字列が原因で発生することもあります。 2 つのモデムが接続したのに、ハンドシェーク・プロセスを開始しない場合、問題はモデムに自動応答の設定がされていないことにあると考えられます。

アクション:

1. まず **cu** コマンドを使用してモデム接続をテストしてください。 リモート・ホスト上のモデムは、ユーザーにシステムへのログインを 許可していなければなりません。 ログイン試行中には画面に不要情報が表示されないようにします。表示される場合、それはノイズが 多い電話回線を示し、これが問題の一端となっている可能性があります。 ログイン中は、複数のログイン・ヘラルドが画面をスクロールしているべきではありません。 これらが存在する場合、それらもやはり問題のある電話回線や誤ったモデム設定を示している可能性があります。
2. モデムの構成を確認し、ARQ コードが現在オンになっている場合は、それをオフにしてみてください。 大半の Hayes 互換モデムでは、これは &A0 設定です。 ARQ 結果コードを使用不可にしても、エラー制御された接続が影響を受けたり、モデムが (結果コードが使用可能の場合に) **slattach** ダイヤル文字列の必要に応じた標準 CONNECT メッセージが戻されなくなることはありません。

問題: ユーザーがモデム **SLIP** 接続を介して **ping** できない。 **ping** コマンドが停止しているか、あるいはエラー・メッセージを戻す。

考えられる原因:

1. モデムと tty ポートの両方または一方が、XON/XOFF フロー制御を使用するように構成されている可能性があります。
2. **slattach** プロセスがリモート・ホスト上で終了されているか、モデム接続がドロップされている可能性があります。
3. **SLIP** ホストに割り当てられたアドレスが誤っている可能性があります。

アクション:

1. ローカルとリモートの両方のモデムの構成を検査します。 これらは、RTS/CTS (ハードウェア) フロー制御を使用するように設定するか、またはフロー制御を使用しないように設定する必要があります。 各システムから ping を試行する必要があります。 systemA から systemB に ping してください。
2. ローカルとリモートの両方のシステム上で **slattach** プロセスがまだ実行しているかどうかを確認します。 コマンド `ps -ef |grep slat` を使用してください。 sl# インターフェースが 実行状態であることを確認します。 コマンド `ifconfig sl#` を使用してください。
3. **SLIP** アドレスと、他のネットワーク・インターフェースに 関連したアドレス (存在する場合) の間に競合がないことを確認します。 コマンド **netstat -ir** を使用してください。 アドレスまたはアドレス・クラスが問題となっている場合は、より単純なアドレス方式 (ローカル・ホストに 1.1.1.1 およびリモート・ホストに 1.1.1.2 など) を使用して **SLIP** を再構成してください。

SLIP の質問表

以下の質問表を使用して、**SLIP** 構成についてのデータを記録してください。

SLIP 構成に関する追加支援が必要になったときは、以下のシートに集めた情報をサービス担当者にファックスで送ることができます。

1. この **SLIP** 構成は以前は正しく機能していましたか。 (Y/N) ____
2. マシン・タイプは何ですか。 (例: UNIX/PC、DOS/PC など)

   ローカル・システム: _____ リモート・システム: _____

ホストが IBM UNIX システムでない場合は、**SLIP** 接続の確立に使用するソフトウェアのタイプを指定してください。

_____

3. 各システム装置上の IBM UNIX オペレーティング・システムのバージョンは何ですか? /bin/oslevel コマンドを実行してください。 このコマンドが認識されない場合は、次のメソッドを使用してください。

```
lslpp -h bos.rte
```

active commit 行 のリリース・レベルを調べてください。

ローカル・システム: _____ リモート・システム: _____

4. 両方のシステム上の使用可能なすべてのインターフェース (例えば、sl0、sl1) をリストしてください。 そのためには、`lsdev -Cc if` コマンドを使用します。

ローカル・システム: _____ リモート・システム: _____

_____ _____

_____ _____

_____ _____

SLIP インターフェース 番号は tty デバイス番号と一致している必要があります。 例えば、/dev/tty53 は sl53 と共に使用する必要があります。

5. **SLIP** は、SMIT とコマンドのどちらを使用して構成されていますか。 コマンドを使用した SLIP 構成は永続的ではなく、システム・リブートの後にはなくなります。

_____

6. **SLIP** は、モデムと直接シリアル回線のどちらを介して構成されていますか。

_____

7. モデムが使用されている場合は、ローカルとリモートの両方のシステムのメーカーおよびモデム ・タイプをリストしてください。

```
        タイプ     ボー・レート  IBM のケーブルか。IBM のケーブルでない場合、
                                (はい/いいえ)     タイプは何か。
ローカル:  _________ ________________    _______      ________________________
リモート:  _________ ________________    _______      ________________________
```

8. モデムが使用されている場合、電話のキャリア・タイプは何か。 (専用回線または標準交換)

_____

9. **SLIP** 回線は、どのようなハードウェア上で使用されていますか。

128 ポート・アダプター (および 16 ポート RAN): ___

2-Port Adapter: ___

8 ポート・アダプター: ___

ネイティブ、S1 または S2 シリアル・ポート: ___

10. ローカル・システムからリモート ・システムに ping できますか。

```
(Y/N) ______    (ローカル・システムで、<remote address> を入力します)
```

11. リモート・システムからローカル・システムに ping できますか。

```
(Y/N) ______    (リモート・システムで、<local address> を入力します)
```

12. ローカルとリモートの両方のシステムで tty ポートが使用不可になっていますか。

```
(Y/N) ______
```

コマンド pdisable | grep tty# を使用してください。 このコマンドの出力には、使用不可の tty 番号のみが表示されます。

13. エラー・メッセージは表示されていますか。 表示されている場合は、それらを以下にリストしてください。

非同期端末エミュレーション

非同期端末エミュレーション (ATE) プログラムにより、オペレーティング・システム上の端末は、端末をエミュレートすることができます。これにより、非同期端末をサポートする他のほとんどのシステムへの接続が可能になります。

ATE は、リモート・システムに、端末をシステム・ディスプレイまたは DEC VT100 端末と見なすようにさせて、これを可能にします。 VT100 オプションを使用する場合、ユーザーは、ユーザーの端末をサポートしていないシステムにも、それが VT100 端末をサポートしていればログインできます。

下の図が示すように、ATE はユーザー・システムとリモート・システムの間の通信に、直接 (ケーブル) 接続とモデム接続の両方を使用します。

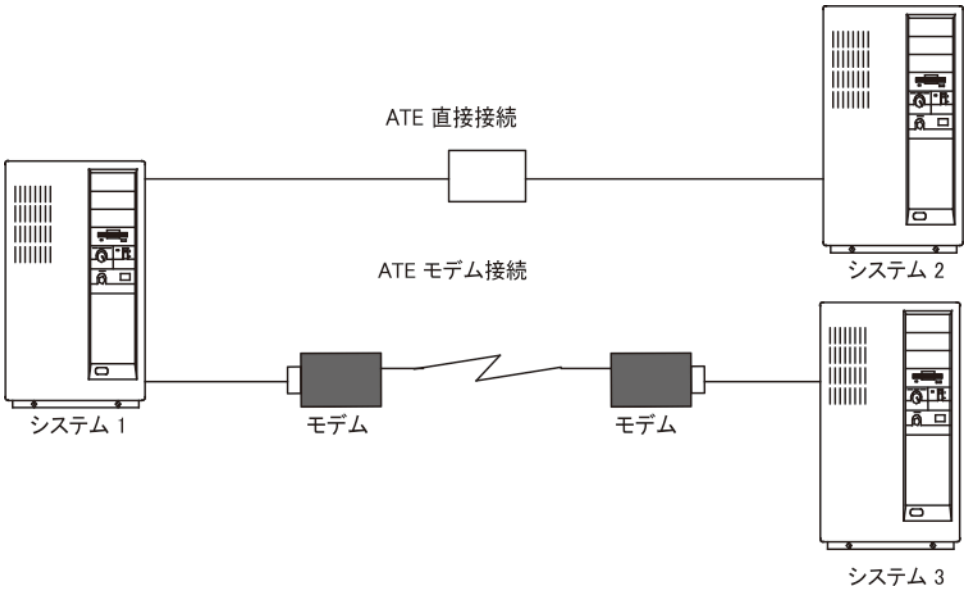


図 *41. ATE* の接続タイプ

ATE は、使用される接続タイプに応じて、隣室のシステムにも、国内の各地のシステムに接続するようにも構成できます。 直接接続の場合は、システムで使用するポートが分かっていなければなりません。 モデ

ム接続の場合は、システムで使用するポートと、リモート・システムの電話番号が分かっていなければなりません。 さらに、リモート・システムのログイン ID とパスワードも持っていなければなりません。

ATE を使用すると、リモート・システム上でコマンドを実行したり、 ファイルを送受信できます。 また、**xmodem** プロトコルを使用してシステムで転送されたファイル内のデータの保全性を検査できます。 ユーザーはまた、リモート・システムからの着信データをキャプチャーしてファイルすることもできます。

> **注:** ATE を使用するには、UNIX 間コピー・プログラム (UUCP) グループのメンバーであることが条件です。 root 権限を付与されたユーザーは、システム管理インターフェース ・ツール (SMIT) を使用して、グループ内の個々のユーザーをインストールします。

ATE のセットアップ

ATE を実行する前に、システム管理者が適切なソフトウェア (必要な場合) をインストールし、tty ポートと接続を構成する必要があります。

- ATE は、オプションのプログラム製品です。 ATE の操作に必要なファイルはすべて、インストール・メディアに収録された **bos.net.ate** プログラム製品の中に含まれています。 システムで ATE が使用可能かどうかを確認するには、次のコマンドを使用してください。

  ```
  lslpp -h | more   <return>
  /bos.net.ate      <return>
  ```

  システムで ATE が使用可能でない場合は、インストール・メディア (磁気テープ、ディスケット、またはネットワーク・サーバー) から **bos.net.ate** イメージをインストールしてください。
- システムに ATE がインストールされている場合は、以下のコマンドを使用して、このプログラムに関連したファイルのリストを表示することができます。

  ```
  lslpp -f | more   <return>
  /bos.net.ate      <return>
  ```
- 通信デバイスのためのポートをセットアップするには、ユーザーは root ユーザー権限を持つ必要があります。

ATE は、直接 (ケーブル) 接続とモデム接続の両方を使用します。 ローカルの RS-232C 接続では最長 15 メートル (50 フィート) 離れたコンピューター同士、また、RS-422A 接続では最長 1200 メートル (4000 フィート) 離れたコンピューター同士を接続できます。

ATE を使用してリモート・システムをコールする 前に、リモート・システムの tty デバイスでコールを受け入れ可能かどうかを確認してください。

ATE をシステム上で実行する準備をするために、次のステップを実行してください。

1. システムに組み込みシリアル・ポートがない場合は、非同期通信アダプター・カードをシステム装置の適切なスロットに取り付けます。
2. アダプター・カードまたは組み込みシリアル・ポートに RS-232C ケーブルか RS-422A ケーブルを接続します。
3. `smit mkdev` 高速パスを使用して、通信ポート用の tty デバイスを追加します。
4. ATE を使用してエミュレートする端末タイプを選択し、環境に合わせて必要な調整を行います。

   最も一般的な変更対象は、回線速度、パリティー設定、1 文字当たりのビット数、および回線がリモート回線で動作するかまたはローカル回線で動作するかです。 各国語サポート (NLS) が必要な場合は、bpc 8 と no parity を使用してください。
5. デバイスのためのポートをセットアップします。

   ATE で呼び出すポートをセットアップするには、**pdisable** コマンドを使用します。 例えば、ポート tty1 をセットアップするには、次のように入力します。

   ```
   pdisable tty1
   ```

他のプログラムが呼べるようにポートをセットアップするには、**penable** コマンドを使用します。 例えば、他のシステムが tty2 ポートを呼べるようにするには、次のように入力します。

```
penable tty2
```

6. デバイスがリモート・システムに対して既に定義されていることを確認します。
   デバイスを定義したあと、ATE プログラムをカスタマイズして、リモート・システムのデバイス 設定を反映させなければなりません。 デフォルト設定をカスタマイズするには、alter および modify サブコマンドを使用するか、`ate.def` デフォルト・ファイルを編集します。 電話接続のデフォルト設定を変更するには、ダイヤル・ディレクトリー・ファイルの項目を使用します。

ATE メインメニュー

ATE は使用されるサブコマンドに応じてメニューを表示します。

ate コマンドを使用して ATE を始動すると、「Unconnected Main」メニューが表示され、次の操作を実行できます。

- ATE の特性を一時的に変更する (**modify**、**alter**)。
- 別のシステムに接続する (**directory**、**connect**)。
- ヘルプを表示する (**help**)。
- システム上でワークステーションのオペレーティング・システム・コマンドを実行する (**perform**)。
- ATE を終了する (**quit**)。

「Unconnected Main」メニューから出すサブコマンドに応じて、ATE は各種のサブメニューを表示します。

表 *110. ATE* サブメニュー

使用するもの	**表示されるもの**
modify サブコマンド	「Modify (変更)」メニュー (詳しくは、**ate** コマンドを参照)
alter サブコマンド	「Alter (変更)」メニュー (詳しくは、**ate** コマンドを参照)
リモート・システムに接続するための **connect** または **directory** サブコマンド	「Connected (接続されました)」メインメニュー
directory サブコマンド	ダイヤル・ディレクトリー (電話番号のリスト)

「Connected (接続されました)」メインメニューからは、サブコマンドを出して次の操作を実行できます。

- リモート・システムとの間でファイルを送受信する (**send**、**receive**)。
- リモート・システムにブレーク・シグナルを送信する (**break**)。
- リモート・システムへの接続を終了する (**terminate**)。

また、**modify**、**alter**、**help**、**perform**、**quit** の各サブコマンドは、「Unconnected Main」メニューから実行するのと同じ機能を実行します。

制御キー・シーケンスを使用すると、ATE の特定の処理を制御できます。 これらのキー・シーケンスは、CAPTURE_KEY、MAINMENU_KEY、PREVIOUS_KEY と呼ばれます。 キー・シーケンスについては、700 ページの『ATE 制御キー・シーケンス』で説明します。 これらのキーは ATE のインストール時にデフォルトのキーの組み合わせになりますが、ATE のデフォルト・ファイル `ate.def` を変更して、キーの組み合わせを変えることができます。

ATE の「Unconnected (接続されていません)」メインメニュー

ATE の「Unconnected (接続されていません)」メインメニューを表示するには、**ate** コマンドを使用します。

接続が確立された後、ATE の **connect** サブコマンドを使用して、「Unconnected (接続されていません)」メインメニューを表示します。

ATE の「Unconnected (接続されていません)」メインメニューからは、次のサブコマンドを発行できます。サブコマンドを出すには、メニュー上のコマンド・プロンプトからサブコマンドの頭文字を入力します。例えば、**directory** サブコマンドを発行するには、**d** と入力します。

項目	説明
alter	送信速度などのデータ送信特性を一時的に変更します。
connect	接続を確立します。
directory	ダイヤル・ディレクトリーを表示します。
help	ヘルプ情報を表示します。
modify	着信データのキャプチャー・ファイルなどのローカル設定を一時的に変更します。
perform	ATE でワークステーションのオペレーティング・システム・コマンドを実行できるようにします。
quit	ATE プログラムを終了します。

注 : 制御キー・シーケンス CAPTURE_KEY、MAINMENU_KEY、PREVIOUS_KEY のうちで、ATE の「Unconnected (接続されていません)」メインメニューから使用できるのは、PREVIOUS_KEY だけです。

ATE の「Connected (接続されました)」メインメニュー

「Connected (接続されました)」メインメニューを表示するには、ATE の「Unconnected (接続されていません)」メインメニューで **connect** サブコマンドを使用します。

または、リモート・システムに接続している場合は MAINMENU_KEY キーを押します。

ATE の「Connected (接続されました)」メインメニューからは、次のサブコマンドを発行できます。 これらのサブコマンドの定義については、**ate** コマンドを参照してください。 サブコマンドを出すには、メニュー上のコマンド・プロンプトからサブコマンドの頭文字を入力します。 例えば、**alter** サブコマンドを発行するには、a と入力します。

項目	説明
alter	送信速度などのデータ送信特性を一時的に変更します。
break	リモート・システムにブレーク・シグナルを送信します。
help	ヘルプ情報を表示します。
modify	着信データのキャプチャー・ファイルなど、エミュレーターに使用されるローカル設定を一時的に変更します。
perform	ATE でワークステーションのオペレーティング・システム・コマンドを実行できるようにします。
quit	ATE プログラムを終了します。
receive	リモート・システムからファイルを受信します。
send	リモート・システムにファイルを送信します。
terminate	ATE 接続を終了します。

3 つの ATE 制御キー・シーケンスはすべて、ATE の「Connected Main」メニューから使用できます。

ATE 制御キー・シーケンス

ATE では次の制御キーを使用します。 ate.def ファイルを編集して、 各機能のキー・シーケンスを変更してください。

項目	説明
CAPTURE_KEY	接続中に画面に表示されるデータの保存を開始または停止します。 CAPTURE_KEY のデフォルト・キー・シーケンスは Ctrl-B です。 CAPTURE_KEY にはスイッチ、すなわちトグル効果があります。 この制御キーを押すと、 データの保存が開始されます。 この制御キーを 2 度目に押すと、データの保存が停止されます。 データは、ate.def ファイルに定義されたキャプチャー・ファイルに保存されます。 デフォルトのキャプチャー・ファイル名は $HOME/kapture ファイルです。 キャプチャー・ファイル名を一時的に変更するには、**modify** サブコマンドを使用します。 キャプチャー・ファイル名を永続的に変更するには、ATE のデフォルト・ファイルを編集します。 709 ページの『ATE のデフォルト・ファイルの編集』を参照してください。 CAPTURE_KEY キー・シーケンスは、端末がファイル転送操作を実行している間は機能せず、接続が確立されているときにのみ有効です。 接続が確立される前に CAPTURE_KEY キー・シーケンスを押すと、次に入力するコマンドは失敗に終わり、エラー・メッセージが表示されます。
PREVIOUS_KEY	直前に表示されていた画面に戻ります。 PREVIOUS_KEY はファイル転送操作を停止するときにも使用します。 PREVIOUS_KEY のデフォルトのキー・シーケンスは Ctrl-R です。 PREVIOUS_KEY は ATE の「Main」メニューからも使用できます。
MAINMENU_KEY	「Connected (接続されました)」メインメニューが表示され、そこで ATE のサブコマンドを発行できます。 MAINMENU_KEY のデフォルトのキー・シーケンスは Ctrl-V です。 リモート・システムへの接続が確立されたら、 この制御キーを使用して「Connected Main」メニューを表示します。 接続が確立される前に MAINMENU_KEY キー・シーケンスを押すと、次に入力するコマンドは失敗に終わり、エラー・メッセージが表示されます。 ATE デフォルト・ファイルをカスタマイズすると、制御キーの設定とキャプチャー・ファイル名を永続的に変更できます。 709 ページの『ATE のデフォルト・ファイルの編集』を参照してください。

ATE のカスタマイズ

ATE を最初に実行するとき、ATE は現行ディレクトリーに ate.def デフォルト・ファイルを作成します。 ATE のさまざまな性質をカスタマイズするには、この ate.def ファイルを編集します。

例えば、ダイヤル・ディレクトリー・ファイルの名前、リモート・システムとの間でファイルを送受信するために使用する転送プロトコルのタイプ、およびモデムが使用することを ATE が予期するボー・レートを変更することができます。 ate.def ファイルの詳細については、709 ページの『ATE のデフォルト・ファイルの編集』を参照してください。

modify サブコマンドおよび **alter** サブコマンドを使用して、ATE の特定の性質を一時的に変更することもできます。 これらのサブコマンドは、すべての ATE のデフォルト値を変更できます。ただし制御キー・シーケンス (デフォルト・ファイルを編集することによってのみ変更可能) とダイヤル・ディレクトリーの名前 (**directory** サブコマンドを使用するか、デフォルト・ファイルを編集することによって変更可能) は除きます。 **modify**、**alter**、または **directory** サブコマンドを使用して加えられた変更は、その ATE のセッション中のみ有効です。 次に ATE を実行するときに使用される設定は、デフォルト・ファイルで定義されている設定です。

ATE にモデムを使用する場合は、20 桁以内の電話番号のダイヤル・ディレクトリーを作成できます。 **directory** サブコマンドは、電話番号をメニュー形式で表示するので、ユーザーはコールしたいシステムを選択することができます。 詳しくは、705 ページの『ATE ダイヤル・ディレクトリーの設定』 を参照してください。

特定のシステムをコールするとき、ダイヤル・ディレクトリーを使用するなら、電話番号を調べる必要はなくなります。 ダイヤル・ディレクトリー・ファイルに特定のデータ伝送特性を指定することもできます。これは、一部の接続が ATE のデフォルトと異なる特性を使用する場合に役立ちます。

専用のダイヤル・ディレクトリーを作成できます。また、システム管理者はシステム全体のダイヤル・ディレクトリーを作成できます。 使用するダイヤル・ディレクトリーを ATE のデフォルト・ファイルに指定します。 詳しくは、705 ページの『ATE ダイヤル・ディレクトリーの設定』 を参照してください。

ate.def 構成ファイル

ate.def ファイルは、非同期接続とファイル転送に使用するデフォルトを設定します。

このファイルは、ATE の最初の実行時に現行ディレクトリー内に作成されます。 ate.def ファイルには、ATE プログラムが以下のために用いるデフォルト値が含まれています。

- データ送信特性
- ローカル・システムの機能
- ダイヤル・ディレクトリー・ファイル
- 制御キー

ATE プログラムは、特定のディレクトリーから初めて呼び出されるときに、そのディレクトリー内に ate.def ファイルを作成します。

```
LENGTH          8
STOP            1
PARITY          0
RATE            1200
DEVICE          tty0
INITIAL         ATDT
FINAL
WAIT            0
ATTEMPTS        0
TRANSFER        p
CHARACTER       0
NAME            kapture
LINEFEEDS       0
ECHO            0
VT100           0
WRITE           0
XON/XOFF        1
DIRECTORY       /usr/lib/dir
CAPTURE_KEY     002
MAINMENU_KEY    026
PREVIOUS_KEY    022
```

これらの特性の値に永続的な変更を加えるには、任意の ASCII テキスト・エディターを使用して ate.def ファイルを編集します。 これらの特性の値を一時的に変更する場合は、ATE メインメニューから利用できる、ATE の **alter** および **modify** サブコマンドを使用します。

ate.def ファイルには、パラメーター名を大文字で入力します。 パラメーターは、元のデフォルト・ファイルに表示されているとおりのスペルにします。 パラメーターは、1 行につき 1 つのみ定義します。 パラメーターの値が間違って定義されていると、ATE がシステム・メッセージを戻します。 しかし、プログラムはデフォルト値を使用して実行し続けます。 以下は、ate.def ファイルのパラメーターです。

LENGTH

データ文字内のビット数を指定します。 この長さは、リモート・システムが期待する長さと一致している必要があります。

```
Options:  7 or 8
Default:  8
```

STOP
文字に付加され、データ伝送時にその文字の終了を示すストップ・ビットの数を指定します。 この数は、リモート・システムが使用するストップ・ビットの数と一致している必要があります。

```
Options:  1 or 2
Default:  1
```

PARITY
文字がリモート・システムとの間で正常に送受信されているどうかを検査します。 リモート・システムのパリティーと一致している必要があります。

例えば、偶数パリティーを選択している場合、文字内の 1 ビットの数が奇数であると、パリティー・ビットがオンになって 1 ビットの数を偶数にします。

```
Options: 0 (none), 1 (odd), or 2 (even)
Default: 0.
```

RATE
ボー・レート、または 1 秒あたりに送信されるビットの数 (bps) を決定します。 速度は、モデムおよびリモート・システムの速度と一致している必要があります。

```
Options: 50,75,110,134,150,300,600,1200,1800,2400,4800,9600,19200
Default: 1200
```

DEVICE
リモート・システムに接続するために使用される非同期ポートの名前を指定します。

```
Options: Locally created port names.
Default: tty0.
```

INITIAL
アクセス番号 (モデムを使用して自動ダイヤルする場合に電話番号の前に付いていなければならないストリング) を定義します。 正しいダイヤル・コマンドについては、モデムの資料を参照してください。

```
Options: ATDT, ATDP, or others, depending on the type of modem.
Default: ATDT.
```

FINAL
ダイヤル接尾部 (モデムを使用して自動ダイヤルする場合に電話番号の後ろに付いていなければならないストリング) を定義します。 正しいダイヤル・コマンドについては、モデムの資料を参照してください。

```
Options: Blank (none) or a valid modem suffix.
Default: No default.
```

WAIT
リダイヤル試行間の待機時間を指定します。 待機期間は、接続試行がタイムアウトになるか、中断されるまでは始まりません。 ATTEMPTS パラメーターが 0 に設定されている場合は、リダイヤルは試行されません。

```
Options: 0 (none) or a positive integer designating the number of seconds to wait.
Default: 0
```

ATTEMPTS
ATE プログラムが接続のためのリダイヤルを試行する最大回数を指定します。 ATTEMPTS パラメーターが 0 に設定されている場合は、リダイヤルは試行されません。

```
Options: 0 (none) or a positive integer designating the number of attempts.
Default: 0
```

TRANSFER
接続時にファイルを転送する非同期プロトコルのタイプを定義します。

p (pacing)
ファイル転送プロトコルは、指定された文字を待機するか、各行伝送の間に一定の秒数間待機することにより、データ伝送速度を制御します。 これにより、伝送ブロックが大きすぎたとき、または送信速度が速すぎてシステムが処理できなかったときのデータの損失が防止されます。

x (xmodem)
データ伝送エラーを検出してデータを再送するための 8 ビット・ファイル転送プロトコル。

```
Options: p (pacing), x (xmodem)
Default: p.
```

CHARACTER
使用するペーシング・プロトコルのタイプを指定します。 行を送信するための信号。 文字を 1 つ選択します。

send サブコマンドは、データの送信中に改行 (LF) 文字を検出すると、ペーシング文字の受信を待ってから次の行を送信します。

receive サブコマンドは、データを受信する準備ができるとペーシング文字を送信し、その後 30 秒待ってからデータを受信します。 **receive** サブコマンドは、データ内で復帰文字を検出するたびにペーシング文字を再度送信します。 **receive** サブコマンドは、30 秒間データを受信しないと終了します。

```
Options: any character
Default: 0
```

Interval
送信する各行の間のシステムの待機秒数。 Interval 変数の値は整数でなければなりません。 デフォルト値は 0 で、0 秒のペーシング遅延を示します。

```
Default: 0.
```

NAME
着信データ用のファイル名 (キャプチャー・ファイル)。

```
Options: A valid file name less than 40 characters long.
Default: kapture
```

LINEFEEDS
着信データ・ストリーム内のすべての復帰文字の後ろに改行 (LF) 文字を追加します。

```
Options: 1 (on) or 0 (off).
Default: 0.
```

ECHO
ユーザーの入力した入力データを表示します。 エコーをサポートするリモート・コンピューターでは、送信された各文字が戻され、画面上に表示されます。 ECHO パラメーターがオンの場合、各文字は 2 回表示されます (まず入力時に表示され、接続を介して戻り時に再び表示されます)。 ECHO パラメーターがオフの場合、各文字は接続を介して戻るときにのみ表示されます。

```
Options: 1 (on) or 0 (off).
Default: 0.
```

VT100
ローカル・コンソールが DEC VT100 端末をエミュレートし、リモート・システムで DEC VT100 コードが使用できるようになります。 VT100 パラメーターがオフの場合、ローカル・コンソールはワークステーションのように機能します。

```
Options: 1 (on) or 0 (off).
Default: 0.
```

WRITE
着信データを取り込み、それを NAME パラメーターで指定されたファイルとディスプレイに 送信します。 復帰と改行 (LF) の組み合わせは、キャプチャー・ファイルに書き込まれる前に改行文字に変換されます。 既存ファイルでは、データはファイルの末尾に追加されます。

接続中にキャプチャー・モードのオン/オフを切り替える場合は、CAPTURE_KEY (通常は Ctrl-B キー・シーケンス) を使用することができます。

```
Options: 1 (on) or 0 (off).
Default: 0.
```

XON/XOFF
ポートでのデータ伝送を、次のように制御します。

- XOFF 信号が受信されると、伝送は停止します。
- XON 信号が受信されると、伝送は再開します。
- 受信バッファーがほぼいっぱいになると、XOFF 信号が送信されます。
- バッファーがいっぱいでなくなると、XON 信号が送信されます。

```
Options: 1 (On), or 0 (Off).
Default: 1.
```

DIRECTORY
ユーザーのダイヤル・ディレクトリーが含まれるファイルの名前を指定します。

```
Default: the /usr/lib/dir file.
```

CAPTURE_KEY
キャプチャー・モードを切り替える制御キー・シーケンスを定義します。 CAPTURE_KEY (通常は Ctrl-B キー・シーケンス) は、押されると、活動状態の接続の期間中に画面に表示されるデータの取り込み (保管) を開始または停止します。

```
Options: Any ASCII control character.
Default: ASCII octal 002 (STX).
```

MAINMENU_KEY
「Connected Main Menu (接続メインメニュー)」を戻すことにより、活動状態の接続の期間中にユーザーがコマンドを実行できるようにする制御キー・シーケンスを定義します。 MAINMENU_KEY (通常は Ctrl-V キー・シーケンス) は、接続された状態からのみ機能します。

```
Options: Any ASCII control character.
Default: ASCII octal 026 (SYN).
```

PREVIOUS_KEY
プログラム期間中にいつでも前の画面を表示する制御キー・シーケンスを定義します。 表示される画面は、PREVIOUS_KEY (通常は Ctrl-R キー・シーケンス) を押したときに使用されている画面に応じて異なります。

```
Options: Any ASCII control character.
Default: ASCII octal 022 (DC2).  The ASCII control character is mapped to the interrupt
signal.
```

ATE ダイヤル・ディレクトリーの設定

ATE ダイヤル・ディレクトリー・ファイルは、モデムによるリモート接続を確立するために ATE プログラムが使用する電話番号がリストされています。

ATE ダイヤル・ディレクトリーをセットアップするには、以下の前提条件を満たす必要があります。

- 非同期端末エミュレーション (ATE) プログラムがシステムにセットアップされていること 。
- システム全体のダイヤル・ディレクトリーを設定する場合は、/usr/lib/dir ファイルへの書き込みアクセス権をもっていること。

ダイヤル・ディレクトリー・ファイルには、任意の有効なファイル名を付け、読み取りおよび書き込みアクセスを持つ任意のディレクトリーに置きます。 ダイヤル・ディレクトリー・ファイルは、任意の ASCII テキスト・エディターを使用して編集します。 ATE プログラムのデフォルトのダイヤル・ディレクトリー情報は、 /usr/lib/dir ファイルに含まれています。以下にそれを示します。

注 : 以下の内容において、いくつかの ATE 項目は、読みやすくする目的で複数の行に分けられています。 しかし、実際のダイヤル・ディレクトリー・ファイルでは、項目のすべてのエレメントが単一の一続きの行で宣言されています。

```
#   COMPONENT_NAME: BOS dir
#
#   FUNCTIONS:
#
#   ORIGINS: 27
#
#
#   (C) COPYRIGHT International Business Machines Corp. 1985, 1989
#   Licensed Materials - Property of IBM
#
#   US Government Users Restricted Rights - Use, duplication or
#   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#   dir - sample dialing directory
#
#
# Micom   9,555-9400 1200 7 1 2 0 0
# R20     9,555-9491 1200 7 1 2 0 0
# QT      9,555-8455 1200 7 1 2 0 0
# Dallas1 9,555-7051 1200 8 1 0 0 0
```

ATE 内のダイヤル・ディレクトリー情報には、「**UNCONNECTED MAIN MENU (未接続のメイン・メニュー)**」にある **directory** サブコマンドを使用することによってアクセスできます。 この画面には、ATE プログラム内に表示されるのと同じディレクトリー情報が表示されます。

複数のダイヤル・ディレクトリーがある可能性もあります。 ATE プログラムが使用するダイヤル・ディレクトリー・ファイルを変更するには、現行ディレクトリー内の ate.def ファイルを変更する必要があります。

> **注:** ダイヤル・ディレクトリー・ファイルには、最高 20 行まで入れることができます (1 行に 1 項目)。 ATE は 21 行目以降を無視します。

ダイヤル・ディレクトリー・ファイルは、ATE プログラムでコールされるリモート・システムの項目が含まれた電話帳のページのようなものです。 ダイヤル・ディレクトリー項目のフォーマットは次のとおりです。

```
Name Phone Rate Length StopBit Parity Echo Linefeed
```

フィールドは、1 つ以上のスペースで区切る必要があります。 各項目を見やすくするために複数のスペースを使用することもできます。 フィールドは次のとおりです。

Name
電話番号を識別します。 名前は 20 文字以内の任意の文字の組み合わせです。 名前の中のワード間には、例えば data_bank のように、スペースではなく _ (下線) を使用してください。

Phone
ダイヤルされる電話番号。 番号は 40 桁以内です。 受け入れ可能な数字および文字のリストについては、モデムの資料を参照してください。 例えば、外線にアクセスするために 9 をダイヤルしなければならない場合は、9,1112222 のように電話番号の前に 9, (数字の 9 とコンマ) を付けます。

電話番号の長さは最大で 40 桁まで可能ですが、directory サブコマンドは最初の 26 桁しか表示しません。

Rate
ビット/秒 (bps) の送信速度またはボー・レート。 1 秒あたりに送信される文字数を決定します。 使用されている通信回線との互換性のあるボー・レートを選択してください。 受け入れ可能なレートは以下のとおりです。

50、75、110、134、150、300、600、1200、1800、2400、4800、9600、19200

非 POSIX のボー・レートの場合、レートを 50 に設定すると、ATE は SMIT を通してそのデバイス用に設定された構成済みのボー・レートを使用します。

Length
1 文字を構成するビット数。 「Length」フィールドの項目は、7 または 8 になります。

StopBit
ストップ・ビットは、文字の終了を示します。 「StopBit」フィールドの項目は、1 または 2 になります。

Parity
文字がリモート・システムとの間で正常に送受信されたかどうかを検査します。 「Parity」フィールドの項目は、0 (なし)、1 (奇数)、または 2 (偶数) になります。

Echo
入力された文字がローカルで表示されるかどうかを決定します。 「Echo」フィールドの項目は、0 (オフ) または 1 (オン) になります。

Linefeed
リモート・システムから着信するデータの各行の末尾に改行 (LF) 文字を入れます。 改行 (LF) 文字は、機能の点で回帰文字と改行文字に似ています。 「Linefeed」フィールドの項目は、0 (オフ) または 1 (オン) になります。

注 : アプリケーション間で制御キーが競合する場合は、変更や再マップが必要である可能性があります。 例えば、ATE プログラム用にマップされた制御キーがテキスト・エディター内の制御キーと競合している場合は、ATE の制御キーを再マップします。

注 : 選択される ASCII 制御文字は、次のような、8 進、10 進、または 16 進の形式になります。

8 進数
000 から 037。 先行ゼロが必要です。

10 進数
0 から 31。

16 進数
0x00 から 0x1F。 先頭の 0x が必要です。 x は大文字でも小文字でもかまいません。

これらの特性を定義する `ate.def` ファイルを作成して、ATE エミュレーションの特性を変更します。 例えば、RATE を 300 bps に、DEVICE を tty3 に、TRANSFER モードを x (Xmodem プロトコル) に、DIRECTORY を my.dir に変更するには、ATE プログラムを実行するディレクトリー内に、以下の項目を含む `ate.def` を作成します。

```
RATE          300
DEVICE        tty3
TRANSFER      x
DIRECTORY     my.dir
```

プログラムは、このディレクトリーから ATE プログラムが開始された時点から、定義された値を使用するようになります。

1. 次のようにして、ダイヤル・ディレクトリー・ファイルを作成します。
   a) ダイヤル・ディレクトリー・ファイルが入っているディレクトリーに変更します。
   b) /usr/lib/dir ファイルをコピーして、テンプレートとして使用します。 有効なファイル名を指定してファイル名を変更します。
   c) ダイヤル・ディレクトリー・ファイル・フォーマットで示されているフォーマットで、電話番号エントリーを作成します。
   d) ファイルを保存します。

   注: 新しいダイヤル・ディレクトリー・ファイルをシステム全体のデフォルト・ファイルにする場合は、ファイルを /usr/lib/dir という名前で保存します。
2. ダイヤル・ディレクトリー・ファイル名がデフォルト名 (/usr/lib/dir) でない場合は、実行する ATE プログラムが入っているディレクトリー内の ate.def ファイルを編集します。 ate.def ファイルの *DIRECTORY* パラメーターを新しいダイヤル・ディレクトリー・ファイルに変更します。
   709 ページの『ATE のデフォルト・ファイルの編集』 を参照してください。
3. ATE を始動し、**directory** サブコマンドでダイヤル・ディレクトリーを表示します。

ATE を使用したダイヤルアウト

ATE とカスタマイズした /usr/lib/dir ダイヤル・ディレクトリー・ファイルを使用してシステムからダイヤルアウトするには、この手順で行います。

ダイヤルアウトを試行する前に、以下の前提条件がすべて満たされていることを確認してください。

- ATE がシステムにインストールされている 。
- モデムが接続および構成され、使用可能になっている。
- ユーザーが UUCP グループのメンバーである (詳細については、697 ページの『ATE のセットアップ』 を参照してください)。
- /usr/lib/dir ダイヤル・ディレクトリー・ファイルが既に正しい情報でカスタマイズされている。
- ユーザーの現在の作業ディレクトリー (pwd) に、正しく更新された ate.def ファイルが含まれている。
- SMIT で /dev/tty ポートの「ENABLE login (ログインを可能にする)」フィールドが disable、share、または delay に設定されている。

1. 次のように入力します。

```
ate
```

2. メインメニューで d を入力し、Enter キーを押します。
3. 表示するディレクトリーのファイル名を入力し、Enter キーを押します。 現行ディレクトリーを使用する場合は、そのまま Enter キーを押します。
4. **#** 列に適切なディレクトリー・エントリー・ナンバーを 入力して、対応する電話番号をダイヤルします。

ATE を使用したファイルの転送

ローカル・ホストからリモート・システムにファイルを転送するには、以下の手順を使用します。

ATE を使用したファイル転送を試みる前に、以下の前提条件がすべて満たされていることを確認してください。

- 接続が **ATE** プログラムを使用して事前に確立されている必要があります。
- Xmodem ファイル転送プロトコルが、ローカルとリモートの両方のシステム上に既に存在している必要があります。 オペレーティング・システム上では、Xmodem は /usr/bin ディレクトリー内に置かれます。

1. ログイン後にリモート・システム上で以下の **xmodem** コマンドを実行します。

```
xmodem -r newfile
```

r は受信する Xmodem フラグであり、*newfile* は受信されるファイルの名前です。 この名前は、転送されるファイルと同じである必要はありません。

2. Enter キーを押します。
3. 次のメッセージが表示されます。

```
ate: 0828-005 The system is ready to receive file newfile. Use Ctrl-X to stop xmodem.
```

このメッセージが表示されない場合は、システムに **xmodem** プログラムがインストールされていないか、このプログラムがシステムのコマンド・パス内に置かれていない可能性があります。

4. Ctrl-V を押して、「ATE CONNECTED MAIN MENU (ATE 接続メインメニュー)」に戻ります。
5. S キーを押して、ファイルを送信します。
6. 次のメッセージが表示されます。

```
Type the name of the file you wish to send and press Enter. To use the last
file name (), just press Enter.
```

7. 転送するファイルの名前と絶対パスを入力します。
8. Enter キーを押します。
9. ATE が次のメッセージを表示し、ファイルの転送を開始します。

```
ate: 0828-024 The program is ready to send file newfile. You will receive another
message when the file transfer is complete.
ate: 0828-025 The system is sending block 1.
ate: 0828-025 The system is sending block 2.
ate: 0828-015 The file transfer is complete.
ate: 0828-040 Press Enter
```

10. 転送が完了したら、Enter キーを押します。

ATE を使用したファイルの受信

リモート・ホストから転送されたファイルを受信するには、以下の手順を使用します。

ATE を使用したファイル受信を試みる前に、以下の前提条件がすべて満たされていることを確認してください。

- 接続が ATE プログラムを使用して事前に確立されている必要があります。
- Xmodem ファイル転送プロトコルが、ローカルとリモートの両方のシステム上に既に存在している必要があります。 オペレーティング・システム上では、Xmodem は /usr/bin ディレクトリー内に置かれます。

1. ログイン後にリモート・システム上で以下の **xmodem** コマンドを実行します。

```
xmodem -s newfile
```

s は送信のための **xmodem** コマンドであり、*newfile* は転送するファイルの名前と絶対パスです。

2. Enter キーを押します。
3. 次のメッセージが表示されます。

```
ate: 0828-005 The system is ready to send file newfile. Use ctrl-X to stop xmodem.
```

このメッセージが表示されない場合は、システムに **xmodem** プログラムがインストールされていないか、このプログラムがシステムのコマンド・パス内に置かれていない可能性があります。

4. Ctrl-V を押して、「ATE CONNECTED MAIN MENU (ATE 接続メインメニュー)」に戻ります。
5. R キーを押して、ファイルを受信します。
6. 次のメッセージが表示されます。

```
Type the name of the file you wish to store the received data in and press
Enter.  To use the last file name (), just press Enter.
```

7. 転送するファイルの名前と絶対パスを入力します。
8. Enter キーを押します。
9. ATE が次のメッセージを表示し、ファイルの転送を開始します。

```
ate: 0828-020 The program is ready to receive file newfile. You will
receive another message when the file transfer is complete.
ate: 0828-028 The system is receiving block 1.
ate: 0828-028 The system is receiving block 2.
ate: 0828-040 Press Enter.
```

10. 転送が完了したら、Enter キーを押します。

ATE のデフォルト・ファイルの編集
ATE デフォルト・ファイルを編集する場合は、ATE プログラムがシステムにセットアップされていなければなりません。

ate.def ファイルの設定を変更するには、

1. ASCII テキスト・エディターを使用して、ate.def ファイルを開きます。
2. 変更したいパラメーターの新しい値を入力します。 他の値は削除するか、または、そのままにしておくことができます。 削除したパラメーターについては、デフォルト値が使用されます。
3. 変更した ate.def ファイルを保存します。

ate.def ファイルの変更結果は、 カスタマイズ済みの ate.def ファイルが入っているディレクトリーから次に ATE を実行したときから有効になります。

読み取りおよび書き込み許可をもつ任意のディレクトリー内に、ate.def ファイルのコピーを保持できます。 例えば、別々の時点で別々のデフォルト値を使用して ATE プログラムを実行したい場合は、適切な設定にした ate.def ファイルの複数のコピーを $HOME ディレクトリーの別々のサブディレクトリー内に保持します。 しかし、ate.def ファイルの複数のコピーはシステム・ストレージを使用します。 代わりの方法として、ATE の **alter** および **modify** サブコマンドを使用して、ほとんどの設定を一時的に変更してください。 個々のモデム接続の設定はダイヤル・ディレクトリーのエントリーを使用して変更します。 705 ページの『ATE ダイヤル・ディレクトリーの設定』を参照してください。

ATE のトラブルシューティング
次の一般的な ATE 問題が発生した場合は、以下のように解決策を検討してください。

問題:
ファイルの転送または受信時に、**xmodem** コマンドが停止しているように見える。 Ctrl-X によってこの問題は解決される。

解決策:
「Alter (変更)」メニューを検査して、Xmodem プロトコル (または転送メソッド) が使用されていることを確認してください。

問題:
ファイルの転送または受信時に、画面でファイルをスクロールすると、実際には完了していないのに、転送または受信が完了したというメッセージが表示される。

解決策:
「Alter (変更)」メニューを検査して、**xmodem** プロトコル (または転送メソッド) が使用されていることを確認してください。

問題:
ATE を開始するときに、以下のエラーを受信する。

```
ate: 0828-008 The system tried to open port /dev/tty0 but failed. If the port name is not
correct,
change it using the Alter menu. Or, take the action indicated by the system message shown
below.

Connect: The file access permissions do not allow the specified action.
ate: 0828-040 Press Enter.
```

解決策:
エラー・メッセージ内の「Connect:」の行で問題点は絞られています。 ATE を実行しようとしているユーザーが UUCP グループのメンバーであることを確認してください。 これを確認するには、コマンド・ラインから *id* を入力します。出力リストに uucp が表示される必要があります。

問題:
ATE に接続しようとすると、以下のエラーを受信する。

```
ate: 0828-008 The system tried to open port /dev/tty0 but failed. If the port name is not
correct, change it using the Alter menu. Or, take the action indicated by the system
message shown below.

Connect: A file or directory in the path name does not exist.
ate: 0828-040 Press Enter.
```

解決策:
誤りまたは選択不可の tty が ATE 用に選択されています。 ATE の「Alter (変更)」画面を検査してください。

問題:
ファイルは正常に転送されるが、ファイル・サイズが元のファイルより大きくなる。

解決策:
転送中に、Xmodem プロトコルによってファイルに埋め込みが行われています。 これを防止するには、**tar** コマンドを使用してファイルを圧縮して転送してください。 これは、一度に 1 ファイルしか送信されないという xmodem のもう 1 つの制約事項を克服する手段にもなります。 いくつかのファイルを単一の tar イメージ内に **tar アーカイブ処理し**、xmodem モデムを使用してそれを転送することができます。

ATE コマンドとそのサブコマンド
ここでは、ATE コマンドとサブコマンドを簡単に説明します。

詳細は、711 ページの『ATE のファイル・フォーマット』を参照してください。

項目 **説明**

ate ATE プログラムを始動します。 以下のサブコマンドの定義については、**ate** コマンドを参照してください。

break
リモート・システムの現行のアクティビティーを 入力します。

connect
リモート・コンピューターに接続します。

directory
ATE のダイヤル・ディレクトリーを表示し、リモート・システムへの接続用エントリーをディレクトリーから選択できるようにします。

help
ATE サブコマンドに関するヘルプを表示します。

perform
ATE の使用中にワークステーション・オペレーティング・システム・コマンドを出せるようにします。

quit
ATE プログラムを終了します。

receive
リモート・システムからファイルを受信します。

send
リモート・ファイルシステムにファイルを 送信します。

terminate
リモート・システムへの ATE 接続を終了します。

さらに、**xmodem** コマンドは、非同期伝送中にデータ伝送エラーを検出する xmodem プロトコルを使用してファイルを転送する場合に有効です。

ATE のファイル・フォーマット

非同期端末エミュレーション (ATE) ファイル・フォーマットには、ate.def とダイヤル・ディレクトリー・フォーマットが含まれます。

項目	説明
ate.def	接続の設定をデフォルト値にします。
Dialing directory	特定のモデム接続用の電話番号と設定を定義します。

詳細は、710 ページの『ATE コマンドとそのサブコマンド』を参照してください。

ダイナミック・スクリーン・ユーティリティー

ダイナミック・スクリーン・ユーティリティー、つまり **dscreen** コマンドは、一度に複数の仮想端末セッション (画面) に単一の物理端末を接続するためのユーティリティーです。

このユーティリティーは、主として、2 ページ以上の画面メモリーがある端末 (例えば、拡張用カートリッジ付き IBM 3151 モデル 310 または 410 表示装置) を持つ端末で使用するためのものです。 このような端末では、仮想画面間で切り替えを行うと、物理端末の画面ページ間でも切り替えが行われ、各仮想画面のイメージの保管と復元を行うことができます。 複数ページ分の画面メモリーがない端末でも、**dscreen** コマンドを使用して仮想画面セッション間の切り替えを行えますが、画面の外観は保たれません。

注 : dscreen ユーティリティーを 完全にサポートするには、端末が内部画面ページをコマンドに応じて切り替えられなければならず、さらにページごとのカーソル位置を記憶していなければなりません。
dscreen ユーティリティーはスマート 端末とダム端末の両方で作動しますが、ダム端末では画面変更時に画面イメージの保管が行われません。

dscreen 端末構成情報ファイル

dscreen ユーティリティーの 端末構成情報ファイル (または dsinfo ファイル) は、**dscreen** ユーティリティーのための異なるキー・セットを定義するのに使用されます。

このような定義を行うのは、例えば、当初定義されていた **dscreen** ユーティリティー・キーが、システム上で使用中のソフトウェア・アプリケーションと 競合するような場合です。

dsinfo ファイルの端末タイプは、1 ページの画面メモリーを想定しています。 したがって、追加の画面メモリー・ページを端末がサポートする場合は、ページ・メモリー制御用の適切なシーケンスを使用するために、dsinfo ファイルをカスタマイズする必要があります。 特定の制御シーケンスについては、該当する端末の解説書を参照してください。

デフォルトの dsinfo ファイルは、/usr/lbin/tty/dsinfo です。 異なる dsinfo ファイルを指定するには、**-i** フラグを使用してください。 このセクションでは、これ以降、デフォルト・ファイルを参照します。 ただし、ユーザーが作成し、カスタマイズした dsinfo ファイルのすべてに同じ情報が適用されます。

dsinfo ファイルの詳細については、713 ページの『ダイナミック・スクリーンの割り当て』を参照してください。

dscreen キー・アクションの割り当て

dscreen コマンドは、実行時に仮想画面を始動します。 端末のキーボードのキーの一部は、仮想画面に渡されなくなります。代わりに、**dscreen** がそれらのキーを代行受信し、そのキーを押すと特定のアクションが行われます。

そのアクションを次に示します。

項目	説明
Select (712 ページの『dscreen 選択キー』を参照)	指定画面を選択します。

項目	説明
Block (712 ページの『dscreen ブロック・キー』を参照)	すべての入出力をブロックします。
New (712 ページの『dscreen 新規画面キー』を参照)	新しい画面セッションを開始します。
End (712 ページの『dscreen End キーと終了キー』を参照)	**dscreen** ユーティリティーを 終了します。
Quit (712 ページの『dscreen End キーと終了キー』を参照)	**dscreen** ユーティリティーを 終了します。
Previous (713 ページの『dscreen 前画面キー』を参照)	前画面に切り替えます。
List (713 ページの『dscreen リスト・キー』を参照)	**dscreen** が割り当てたキーとそのアクションをリストします。

各キーの機能は、端末と /usr/lbin/tty/dsinfo ファイル内の端末記述によって異なります。

dscreen 選択キー
新しい仮想画面を作成すると、その画面に選択キーが割り当てられます。

選択キーを押すと、次のアクションが実行されます。

- 物理端末から、特定の仮想画面に対応するビデオ・ページに切り替わる。
- 物理端末と仮想画面の間で、適宜、入出力が行われる。

dsinfo ファイルに定義されているすべての選択キーに仮想画面が割り当てられた後は、それ以上の画面の作成は不可能になります。 個々の画面セッションは、最初のシェル・プロセスの終了時に終了します。 これによって、別の仮想画面で使用するために、対応する選択キーが解放されます。 **dscreen** ユーティリティーは、アクティブ画面が存在しなくなると終了します。

dscreen ブロック・キー
ブロック・キーは、IXON フロー制御の使用時に Ctrl-S キーと同様のフォーマットで出力を停止します。

ブロック・キーの目的は、2 つのシリアル・ポートがある端末を使用して 2 台のコンピューター上で端末セッションを透過的にセットアップすることです。

dscreen 新規画面キー
新規画面キーを押すと新しい論理画面が作成され、選択キーの 1 つにその画面が割り当てられます。

新規画面ごとに次のものが必要です。

- dsinfo ファイルに定義されている選択キー。
- **dscreen** 疑似端末デバイス。
- 画面追跡で使用される各種の構造用の十分なメモリー量。
- シェルを実行するプロセス。

これらのいずれかが使用できない場合、新規画面操作は失敗し、失敗の原因を示すメッセージが表示されます。

dscreen End キーと終了キー
End キーと終了キーを押すと、一連のアクションが実行されます。

End キーを押すと、次のアクションが実行されます。

- **SIGHUP** シグナルがすべての画面セッションに送信されます。
- クリーンアップされます。
- 状況 0 で終了します。

終了キーを押しても同じアクションが実行されますが、状況 1 で終了します。

dscreen 前画面キー
前画面キーを押すと、直前に表示されていた画面に端末が切り替わります。

注 :

1. 現在の画面に書き込みが行われているときは、画面を切り替えないでください。切り替えるとエスケープ・シーケンスが切り捨てられ、端末は未知状態になります。
2. 端末ディスプレイの 中には、個々の画面のカーソル位置は保存できても、挿入モードや逆転表示といった、その他の状態を保存できないものがあります。このようなケースでは、画面の切り替え中はこれらのモードを避ける必要があります。

dscreen リスト・キー
リスト・キーを押すと、キーとそのアクションのリストが端末ディスプレイに 表示されます。

表示されるのは **dscreen** ユーティリティーによって 認識されるキーだけです。 **dscreen** ユーティリティーを使用して新規画面を作成すると、「Press KEY for help」というメッセージが端末に表示されます。ここで KEY は、端末に表示されるリスト・キーの名前です。 このメッセージは、リスト・キーが定義されている場合にのみ 表示されることに注意してください。

ダイナミック・スクリーンの割り当て
/usr/lbin/tty/dsinfo ファイル内の端末記述項目には、端末の物理画面ページの数と同じ数の画面選択キーがあります。 物理画面ページの数より多い画面選択キーを定義すると、**dscreen** ユーティリティーは物理画面ページを仮想画面に動的に割り当てます。

関連した画面メモリー・ページを持たない仮想画面を選択すると、**dscreen** ユーティリティーは 最も使用頻度の少ない物理画面をその仮想画面に割り当てます。 /usr/lbin/tty/dsinfo 記述ファイル内に保持されている指定によっては、その物理画面が別の仮想画面に接続されていることが分かる場合があり、そのような場合は画面のクリアなどが行われます。

dsinfo ファイル
dsinfo ファイルは、**dscreen** 複数画面ユーティリティーが 使用する端末記述のデータベースです。

このファイルには、次の情報があります。

- **dscreen** ユーティリティー ・キーおよびそれらが実行する機能。
- 端末の画面メモリー・ページ数。
- 上記の機能を使用するために送受信されるコード・シーケンス。

デフォルトの dsinfo ファイル内にある端末タイプの項目は、次の 3151 ASCII 端末値に類似しています。

```
# The Cartridge for Expansion (pn: 64F9314) needed for this entry
ibm3151|3151|IBM 3151,
dsks=¥E!a^M|Shift-F1|,            # Selects first screen
dsks=¥E!b^M|Shift-F2|,            # Selects second screen
dsks=¥E!c^M|Shift-F3|,            # Selects third screen
dsks=¥E!d^M|Shift-F4|,            # Selects fourth screen
dskc=¥E!e^M|Shift-F5|,            # Creates a new screen
dske=¥E!f^M|Shift-F6|¥E pA¥EH¥EJ, # Go to screen 1 and end
dskl=¥E!g^M|Shift-F7|,            # Lists function keys (help)
dskp=¥E!h^M|Shift-F8|,            # Go to previous screen
dskq=¥E!i^M|Shift-F9|¥E pA¥EH¥EJ, # Go to screen 1 and quit
dsp=¥E pA|¥EH¥EJ,                 # Terminal sequence for screen 1
dsp=¥E pB|¥EH¥EJ,                 # Terminal sequence for screen 2
dsp=¥E pC|¥EH¥EJ,                 # Terminal sequence for screen 3
dsp=¥E pD|¥EH¥EJ,                 # Terminal sequence for screen 4
dst=10,                           # Allow 1 second timeout buffer
```

dsinfo の項目のフォーマット
dsinfo ファイルの項目は、コンマで区切られたフィールドで構成されます。

最初のフィールドは、端末の代替名のリストであり、それぞれの名前は、パイプ (|) 文字で区切られます。前にポンド (#) 記号の付いているテキストはすべて、**dscreen** によってコメントと見なされ、無視されま

す。 残りのフィールドは、**dscreen** ユーティリティーに 対して端末の機能を記述する文字列です。 その文字列内では、次のエスケープ・コードが認識されます。

表 *111. dsinfo* ファイルのフィールド

エスケープ・シーケンス	**説明**
¥E,¥e	エスケープ文字
¥n,¥l	改行 (または行送り) 文字
¥r	復帰
¥t	タブ文字
¥b	バックスペース文字
¥f	用紙送り文字
¥s	スペース文字
¥*nnn*	8 進値 *nnn* を持つ文字
^*x*	該当する *x* 値用の Ctrl-X

その他の ¥ (円記号) が前に付く文字は、すべてその文字自体を生成します。 文字列は *type=string* として入力します。この *type* は次に示す文字列の型であり、*string* は文字列値です。

`dsinfo` ファイルのエントリー・フィールドをコンマで区切ることが重要です。 コンマを省略したり、`dsinfo` ファイルのエントリーの末尾で切り捨てると、そのファイルは **dscreen** ユーティリティーが 読み取れなくなり、エラー・メッセージが表示されます。

disinfo 文字列の型

ここでは、disinfo 文字列の型を示します。

文字列の型を次に示します。

項目 **説明**

dskx dsk で始まる文字列の型は、キーについての記述です。 型の長さは 4 文字にする必要があり、4 番目の文字 *x* は、キーが受信されると実行されるアクションを示します。 キーの型は次のとおりです。

Type
アクション

dsks
画面を切り替えます。

dskb
入出力をブロックします。

dske
dscreen を終了します。

dskq
dscreen を終了します (終了状況 = 1)。

dskc
新規画面を作成します。

dskp
前画面に切り替えます。

dskl
キーとアクションをリストします。

その他のキーの型 (つまり、s、b、e、q、p、および l で終らない文字列型 dsk*x*) はすべて、内部 **dscreen** アクションを発生しませんが、キー・リストに表示され、認識されて機能します。 内部 **dscreen** アクションが不要なときは、dskn (n は操作なしを表す) の型を使用してください。

各キーの値の文字列には 3 つのサブストリングがあり、それらのサブストリングはパイプ (|) 文字で区切られます。

注 : サブストリングの 1 つに | 文字を含めるには、¥| を使用してください。

最初のサブストリングは、キーを押すと端末が送信する文字シーケンスです。 2 番目のサブストリングは、キー・リストが表示されると出力されるキーのラベルです。 3 番目のサブストリングは、このキーを押したとき、このキーの要求するアクションを実行する前に、**dscreen** が端末に送信する文字シーケンスです。

dsp dsp 型の文字列は、端末の物理画面についての記述です。 端末の物理画面ごとに 1 つの dsp 文字列が対応しなければなりません。 各物理画面の値の文字列には 2 つのサブストリングがあり、それらのサブストリングは、パイプ (|) 文字で区切られます。

最初のサブストリングは端末に送信され、端末上の物理ページに表示されて出力される文字シーケンスです。

2 番目のサブストリングは、ページが新しいことに使用されるときに端末に送信されます。 この 2 番目のサブストリングは、多くの場合、画面をクリアするシーケンスに設定されます。 このサブストリングの送信には、次の 2 つの条件が必要です。

1. 新しい仮想端末セッションが生成される場合。
2. 物理画面の数より仮想端末の数が多い場合。 **dscreen** が物理画面の 1 つを再使用しなければならない仮想端末を選択すると、このシーケンスが画面に送信され、画面内容と接続された仮想端末の出力が一致しないことが示されます。

注 : 物理画面より多い仮想端末での実行は混乱を生じる恐れがあり、お勧めできません。これを避けるには、dsinfo エントリーに物理画面 (dsp=) より多い画面選択キー (dsks=) を定義しないようにします。

項目	説明
dst A	dst アジャスト型の文字列は、**dscreen** の入力タイムアウトを調整します。 文字列の値は 10 進数です。 タイムアウト値は 10 分の 1 秒単位であり、最大値は 255 です (デフォルトは **1**、つまり 0.1 秒)。 **dscreen** は、入力キー・シーケンスの接頭語を認識してもそのシーケンスのすべての文字を取得していないとき、シーケンスを認識できるまで文字がさらに送信されるのを待機します。 それらの文字の受信前にタイムアウトが発生すると文字は仮想画面に送信され、**dscreen** はそれを入力キー・シーケンスの一部として認識しなくなります。 **dscreen** がトリガーとなるキーの 1 つ以上が実際は多数のキー・ストロークである場合 (画面選択用に Ctrl-Z 1、Ctrl-Z 2、Ctrl-Z 3 などを割り当て、新規画面用に Ctrl-Z N を割り当てるような場合)、この値を大きくすることが必要になるかもしれません。

dysinfo 例

次の dysinfo 例は、3 つの画面セッションを持つ Wyse-60 についての例です。

```
wy60|wyse60|wyse model 60,
dsks=^A`^M|Shift-F1|,
dsks=^Aa^M|Shift-F2|,
dsks=^Ab^M|Shift-F3|,
dskc=¥200|Ctrl-F1|,
dske=¥201|Ctrl-F2|¥Ew0¥E+,
dskl=¥202|Ctrl-F3|,
dsp=¥Ew0|¥E+,
dsp=¥Ew1|¥E+,
dsp=¥Ew2|¥E+,
```

このエントリーの場合、

- Shift-F1 から Shift-F3 までのキー・ストロークは、画面 1 から 3 の選択に使用します。
- Ctrl-F1 は、新規画面を作成します。
- Ctrl-F2 は、「Esc w 0 Esc +」を画面に送信し (ウィンドウ 0 に切り替えて画面をクリアする)、**dscreen** を終了します。
- Ctrl-F3 は、キーとその機能をリストします。

物理画面が新規画面のために使用されるたびに、シーケンス Esc + が端末に送信され、画面がクリアされます。

次の例は、3 画面セッションの Wyse-60 に適用されますが、画面の 1 つは端末上の 2 番目のシリアル・ポートを介して通信するもう 1 台のコンピューターの画面です。

```
wy60-1|wyse60-1|wyse model 60 - first serial port
dsks=^A`^M|Shift-F1|,
dsks=^Aa^M|Shift-F2|,
dsks=^Ab^M|Shift-F3|¥Ed#^Ab¥r^T¥Ee9,
dskc=¥200|Ctrl-F1|,
dske=¥201|Ctrl-F2|¥Ed#¥201^T¥Ew0¥E+,
dskl=¥202|Ctrl-F3|,
dsp=¥Ew0|¥E+,dsp=¥Ew1|¥E+,
wy60-2|wyse60-2|wyse model 60 - second serial port
dsks=^A`^M|Shift-F1|¥Ed#^A`¥r^T¥Ee8,
dsks=^Aa^M|Shift-F2|¥Ed#^Aa¥r^T¥Ee8,
dsks=^Ab^M|Shift-F3|,
dskc=¥200|Ctrl-F1|,
dske=¥201|Ctrl-F2|¥Ed#¥201^T¥Ew0¥E+,
dskl=¥202|Ctrl-F3|,
dsp=¥Ew2|¥E+,
```

dscreen は両方のコンピューターで実行される必要があり、端末タイプ wy60-1 は最初のコンピューター上、端末タイプ wy60-2 は 2 番目のコンピューター上にあります (**-t** オプションを **dscreen** に使用)。 wy60-1 エントリーが最初に検査されます。

最初の 2 つのキー・エントリーは、始動時の wy60 エントリーから変更されていません。 ただし、3 番目のキーは dskb 型です。入力および出力の両方をブロックします。 このキーを押すと、次のシーケンスが端末に送信されます。

```
Esc d # Ctrl-A b CR Ctrl-T Esc e 9
```

この出力がブロックされたあと、**dscreen** は入力をスキャンし続けてキー・シーケンスを求めますが、他のすべての入力を放棄します。

シーケンス `Esc d #` は、端末を透過表示モードにします。このモードでは、Ctrl-T までのすべての文字を他方のシリアル・ポートを介してエコーします。

文字 `Ctrl-A b CR` は他方のシリアル・ポートから送信され、他方のコンピューター上の **dscreen** プロセスに、Shift-F3 キーに対応するウィンドウをアクティブにするように 通知します。

Ctrl-T キー・シーケンスは、透過表示モードを終了します。 Esc 9 キー・シーケンスは、データ通信のために他方の AUX シリアル・ポートに端末を切り替えます。

この時点で、他方のコンピューターがアクティブになり、「`Esc w 2`」を送信し、3 番目の物理画面に切り替えたあと、通常の通信を再開します。

wy60-2 エントリーは、Shift-F1 キーと Shift-F2 キーに対して同様の通常パターンに従います。

- 透過表示モードに切り替えます。
- ファンクション・キー文字列を他方のコンピューターに送信します。
- 透過表示をオフに切り替えます。
- 他方のシリアル・ポートに切り替えます。

End キー Ctrl-F2 は、両方のコンピューターで同様に機能します。つまり、他方のコンピューターに透過出力機構を介して End キー・シーケンスを送信し、端末をウィンドウ 0 に切り替え、画面をクリアしたあと、終了します。

Serial over Ethernet デバイス・ドライバー

Request for Comments (RFC) 2217 プロトコルによってサポートされている Ethernet Device Server (EDS) を使用して、仮想シリアル・デバイスおよびテレタイプ (tty) デバイスを AIX オペレーティング・システムで作成します。

Serial over Ethernet (SoE) デバイス・ドライバーを使用すると、Request for Comments (RFC) 2217 プロトコルによってサポートされている Ethernet Device Server (EDS) を使用して、仮想シリアル・デバイスおよびテレタイプ (tty) デバイスを AIX オペレーティング・システムで作成できます。 EDS の例として、`Digi` デバイスや `Perle` デバイスがあります。 SoE デバイス・ドライバーの機能性は、実 COM (通信) ポート (例えば、2 ポート、8 ポート、および 128 ポートの各アダプター) と似ています。

EDS は、イーサネット・シリアル・サーバーまたはイーサネット端末サーバーとも呼ばれます。 EDS は、イーサネットが接続されている外付けの IBM 以外の機器で、この機器には、外付けモデムを接続できる 1 つ以上のシリアル・ポート (RS/232) が用意されています。 Telnet プロトコルの拡張版である RFC 2217 (Telnet 通信ポート制御プロトコル) をサポートしていれば、その EDS には互換性があります。 このプロトコルを使用することによって、EDS は RFC 2217 サーバーとして動作します。 EDS は、RFC 2217 クライアント・システムからの Telnet セッションを受け入れることができ、また受信した Telnet データを COM ポートに送信できます。 COM ポートで受信されたデータは、RFC 2217 クライアント・システムに送信されます。

さらに EDS は、シリアル・デバイスの状況の変更に関する情報を RFC 2217 クライアント・システムに送信することもできます。 RFC 2217 クライアント・システムは、EDS とのフロー制御を管理し、そのプロトコルを使用して構成情報を EDS に送信します。

AIX LPAR は、RFC 2217 クライアント・システムとして動作します。 AIX LPAR は、EDS (つまり、RFC 2217 サーバー) への Telnet セッションを確立します。 次の図は、どのように AIX LPAR が仮想シリアル・ポートで EDS との通信を行うかを示しています。

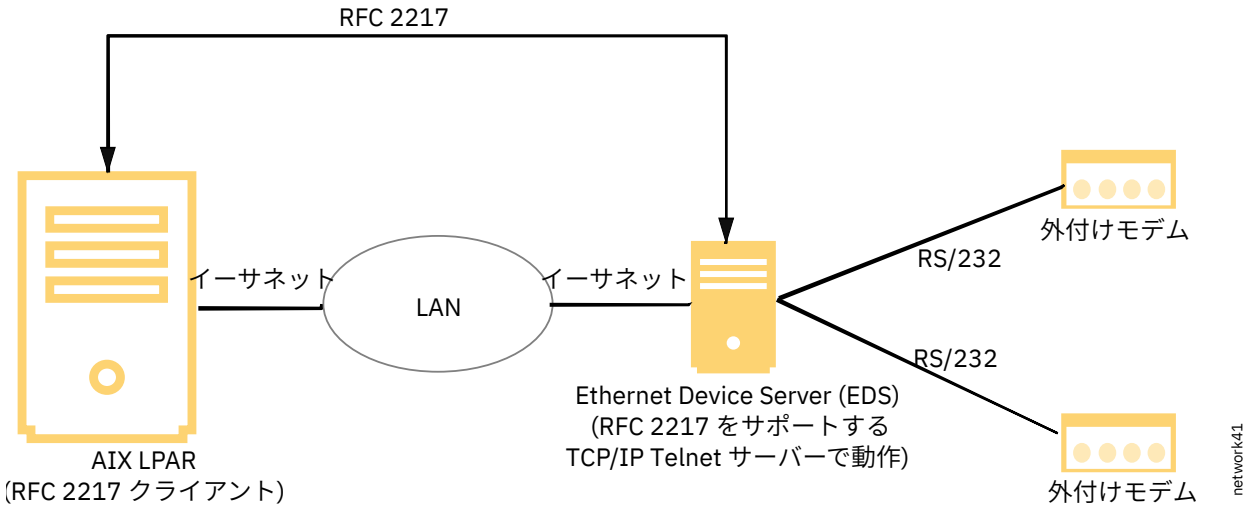


図 *42. Ethernet Driver Server* の構成

Ethernet Device Server の構成

Ethernet Device Server (EDS) には一般に、構成および管理を目的とした Web ベースのインターフェースが用意されています。 例えば、`Digi PortServer` デバイスの IP アドレスが `9.5.80.73` であれば、Web ブラウザーに URL `http://9.5.80.73` を入力することによって、構成および管理のためのインターフェースにアクセスできます。 EDS 上のそれぞれのシリアル・ポートに、TCP ポート番号が割り当てられています。 ある特定のシリアル・ポートには、EDS IP アドレスとそのシリアル・ポートに割り当てられている TCP ポート番号 (例えば、`9.5.80.73:2001`) への Telnet セッションを確立することによってアクセスできます。 この Telnet セッションは、SoE ドライバーによって内部的に作成されます。

AIX LPAR を構成して仮想シリアル・ポートを作成

SoE デバイス・ドライバーは、デバイスが物理シリアル・デバイス用に作成されるときと同じように、SoE アダプター (sa) デバイスおよびテレタイプ (tty) デバイスを作成するためのセマンティクスを維持しています。

仮想シリアル・ポートを AIX LPAR に作成するには、以下の手順に従ってください (以下のコマンドを入力するか、または `smit soe` メニュー・オプションを使用してください)。

1. 次のコマンドを実行して、sa デバイスを作成します。 EDS の IP アドレスを指定してください。

   ```
   # mkdev -c adapter -s pseudo -t soe -a netaddr=EDS_IP_address
   ```

   例:

   ```
   # mkdev -c adapter -s pseudo -t soe -a netaddr=9.126.88.123
   sa2 Available
   ```

2. 次のコマンド実行して、tty デバイスを作成します。 ステップ 1 でのコマンド出力に表示された SoE `adapter` (sa) デバイスと、TCP ポートを指定してください。

   ```
   # mkdev -t tty -s rs232 -p sa_device -w tty_port_number -a -a port_num=TCP_port
   ```

   例:

   ```
   # mkdev -t tty -s rs232 -p sa2 -w 0 -a port_num=2002
   tty1 Available
   ```

   このコマンドで、tty デバイスが /dev ディレクトリーに作成されます。 どのアプリケーションも、新しく作成された tty デバイスを使用して、EDS 上のシリアル・ポートに接続されているターゲット・デバイスとの通信を行うことができます。

注 : EDS 上のシリアル・ポートはそれぞれ、固有の TCP ポートで構成されている必要があります。また、SoE デバイス・ドライバーを使用して構成された tty デバイスはそれぞれ、この固有のポートにマッピングされている必要があります。 EDS 上の tty ポートを、AIX LPAR 上の複数の tty デバイスが共有することはできません。

非同期シリアル・デバイスにわたって tty 端末装置を移動

テレタイプ (tty) 端末装置は、ある非同期 SoE adapter (sa) シリアル・デバイス (バッキング・デバイス) から別の非同期デバイスに移動することができます。 また、同じ非同期デバイス上で、ある物理ポートから別の物理ポートに移動することもできます。 AIX オペレーティング・システムでは、tty 端末装置を移動するために、**smitty** とコマンド・ライン・オプションがどちらもサポートされています。

注 :

- tty 端末装置固有の構成設定 (ボー・レートや実行モードなど) が、非同期デバイスへの移動の後で変わることはありません。
- tty 端末装置は、移動操作が進行中であるときに、アプリケーションによってオープンすることも 、使用することもできません。

非同期デバイスとして、実通信ポート (PCI 2 ポート、8 ポート、および 128 ポートのアダプター) または RFC2217 準拠の通信ポート・ドライバーが考えられます。 SoE デバイス・ドライバーは、EDS の列挙型です。

tty デバイスは、PCI ベース物理非同期アダプターから別の PCI ベース物理デバイスに、PCI ベース物理非同期デバイスから SoE デバイスに (またはその逆)、またはある SoE デバイス・タイプから別の SoE デバイス・タイプに移動することができます。

SoE デバイス・ドライバーの tty 端末装置が IP アドレス 192.168.1.1 で、その SoE デバイス・ドライバーの IP アドレスを 10.1.1.1 に変更する、というシナリオを想定します。 関連付けられている tty 端末装置のすべてが **rmdev** コマンドの実行によって完全に削除されるまで、または関連付けられている tty 端末装置のすべてが **rmdev** コマンドの実行によって「defined」状態になるまで、**chdev** コマンドを実行して SoE デバイス・ドライバーの IP アドレスを変更することはできません。 EDS によってバッキングされている SoE デバイス・ドライバーの IP アドレスを変更するには、tty 端末装置を非同期シリアル・デバイスに移動してください。

SoE デバイス・ドライバーの IP アドレスを変更するには、以下の手順に従ってください。

1. SoE デバイス・ドライバーを IP アドレス 10.1.1.1 で作成します。
2. **smitty** または **chdev** コマンドを使用して、IP アドレス 192.1681.1 の SoE デバイス内の tty 端末装置ドライバーのすべてを、IP アドレス 10.1.1.1 の SoE デバイスに移動します。
3. どの tty 端末装置も使用中ではないこと、またオープン状態ではないことを確認してください。 tty 端末装置を同じ SA デバイス上で、あるポートから別のポートに移動するには、-w フラグのオプションとして新しいポート番号を指定し、**chdev** コマンドを実行します。 **chdev** コマンドの構文は次のとおりです。

   ```
   chdev -1 <tty device> -w <destionation port number>
   ```

例えば、tty 端末装置をポート 0 からポート 1 に移動するには、次のコマンドを実行してください。

```
chdev -1 ttyX -w 1
```

tty 端末装置を、あるバッキング・デバイスから別のバッキング・デバイスに移動するには、**-p** フラグのオプションとしてターゲット・デバイスの名前を指定する必要があります。 **chdev** コマンドの構文は次のとおりです。

```
chdev -1 <tty device> -p <destination parent>
```

例えば、tty 端末装置 tty0 を SA1 シリアル・デバイスから SA3 シリアル・デバイスに移動するには、次のコマンドを実行してください。

```
chdev -1 tty0 -p sa2
```

tty 端末装置を、ある物理アダプター・デバイス (例えば PCI 2 ポート、8 ポート、および 128 ポートのアダプター) から SoE デバイス・ドライバー (RFC2217 準拠) に移動するには、**-a** フラグで **port_num** 属性として TCP ポート番号を指定する必要があります。

例えば、tty 端末装置 tty0 を SA2 シリアル・デバイスから SA3 シリアル・デバイスに移動するには、次のコマンドを実行してください。

```
chdev -1 tty0 -p sa3 -a 2001
```

tty 端末装置を、ある EDS でバッキングされた SoE デバイス・ドライバーから、別の EDS でバッキングされた別の SoE デバイスに移動するときのコマンド構文は次のとおりです。

```
chdev -1 <tty device> -p <destination parent>
```

例えば、tty 端末装置を SA1 シリアル・デバイス (EDS1 によってバッキング) から SA2 シリアル・デバイス (EDS2 によってバッキング) に移動するには、次のコマンドを実行してください。

```
chdev -1 tty0 -p sa2
```

チューナブル・パラメーター

SoE デバイス・ドライバーが使用する属性のいくつかを調整するために、以下のチューナブル・パラメーターが用意されています。

- **idle_timeout**: TCP キープアライブ・プローブがデバイスに送信されるまでに、SoE デバイス・ドライバーと EDS と間の TCP 接続がアイドルである時間 (0.5 秒単位) を指定します。 この値は、TCP 接続用に SoE ドライバーによって設定されている TCP ネットワーク・オプション `tcp_keepidle` に対応します。 デフォルト値は 360 です。
- **probe_interval**: EDS への TCP 接続の妥当性を検査するために送信される、TCP キープアライブ・パケット間の間隔を 0.5 秒単位で指定します。 この値は、TCP 接続用に SoE ドライバーによって設定されている TCP ネットワーク・オプション `tcp_keepintvl` に対応します。 デフォルト値は 10 です。
- **probe_count**: EDS との間に確立された TCP 接続の終了前に、デバイスに送信できる TCP キープアライブ・プローブの数を指定します。 この値は、TCP 接続用に SoE ドライバーによって設定されている TCP ネットワーク・オプション `tcp_keepcnt` に対応します。 デフォルト値は 24 です。

一般的なエラーのトラブルシューティング

EDS または SoE デバイス・ドライバーのどちらかが正しく構成されていない場合、tty デバイスが AIX LPAR 上に作成されると、その tty デバイスの状態は DOWN または ERROR になる可能性があります。 デバイス・ドライバーが正しく構成されていれば、tty デバイスの状態は UP です。 tty デバイス状態は、問題のトラブルシューティングを行うための **soestat** コマンドによって表示されます。

以下の理由のために、tty デバイスの状態は DOWN である可能性があります。

- Soe デバイス・ドライバーまたは EDS で、IP アドレスまたはポート番号が誤っている。
- ネットワーク構成またはネットワーク・トポロジーが不適切であることが原因で、SoE デバイス・ドライバーが構成されている AIX LPAR から EDS に到達できない。
- EDS 上の同じ TCP ポート番号を使用して、複数の tty デバイスが作成されている。

以下の理由のために、tty デバイスの状態は ERROR である可能性があります。

- RFC 2217 モードが EDS で選択されていない。 RFC 2217 モードを構成する方法を調べるには、EDS 製造メーカーの資料を参照してください。
- 指定された IP アドレスが EDS のアドレスではなく、SoE デバイス・ドライバーが構成されている the AIX LPAR から到達可能な他のマシンのアドレスである。

注 : tty デバイスは ERROR 状態からリカバリーできず、これ以降、使用することはできません。 問題を修正した後、手動で ERROR 状態の tty デバイスを削除して tty デバイスを再作成するか、または tty を `defined` 状態にしてから、もう一度 `available` 状態にする必要があります。

汎用データ・リンク・コントロール環境

汎用データ・リンク・コントロール (GDLC) とは、アプリケーションとカーネルのそれぞれのユーザーが共通のコマンド・セットを使用して、オペレーティング・システム内のデータ・リンク制御 (DLC) デバイス・マネージャーを制御できるようにする汎用インターフェース 定義です。

問題判別については、*Communications Programming Concepts* の 『GDLC Problem Determination』 を参照してください。

汎用データ・リンク・コントロール (GDLC) とは、アプリケーション・ユーザーとカーネル・ユーザーが共通コマンド・セットを使用して、オペレーティング・システム内の DLC デバイス・マネージャーを制御できるようにする汎用インターフェース 定義です。

GDLC インターフェースによって、エントリー・ポイント定義に必要な条件、提供される機能、およびすべての DLC デバイス・マネージャーのデータ構造が指定されます。 GDLC インターフェースに準拠する DLC は次のとおりです。

- 8023 (イーサネット用の IEEE 802.3)
- ETHER (標準イーサネット)
- SDLC (同期データ・リンク制御)
- TOKEN (トークンリング)
- FDDI (光ファイバー分散データ・インターフェース)

DLC デバイス・マネージャーは、カーネル・デバイス・ドライバーの機能の範囲を超えた上位層のプロトコルと機能を実行します。 ただし、このマネージャーは、最大限のパフォーマンスを得るためにカーネル内に存在し、アダプターへの入出力要求のためにカーネル・デバイス・ドライバーを使用します。 DLC ユーザーは、カーネルより上位またはカーネル内に位置します。

DLC デバイス・マネージャーの例として、同期データ・リンク制御 (SDLC) と IEEE 802.2 データ・リンク制御があります。 各 DLC デバイス・マネージャーは、特定のデバイス・ドライバーまたはデバイス・ドライバーのグループとともに動作します。 例えば、SDLC は、システム・プロダクトとそれに関連するアダプター用のマルチプロトコル・デバイス・ドライバーを使用して動作します。

DLC 環境の基本構造を、次の「DLC デバイス・マネージャー環境」の図に示します。 カーネル内のユーザーは、通信メモリー・バッファー (mbufs) へアクセスし、**fp** カーネル・サービスを介して追加エントリー・ポイントをコールします。 カーネルより上位のユーザーは、インターフェースとカーネル 間の標準デバイス・ドライバーにアクセスし、ファイルシステムは、**dd** エントリー・ポイントをコールします。 データを転送するには、ユーザーとカーネル・スペース間のデータの移動が必要になります。

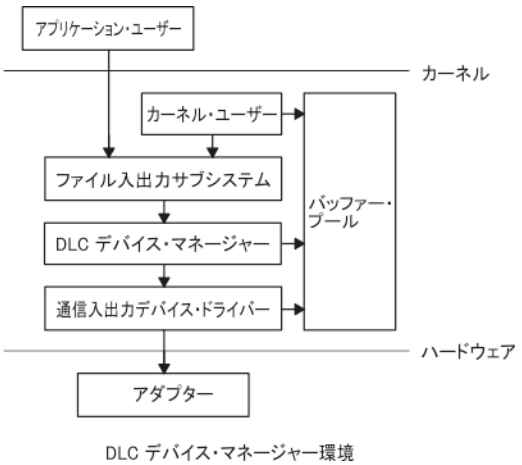


図 *43. DLC* デバイス・マネージャー環境

この図には、アプリケーション・ユーザーとアダプターとの間のハードウェア・レベルのリンクが示されています。 その中間の領域には、カーネル・ユーザー、ファイル入出力サブシステム、DLC デバイス・マネージャー、通信入出力デバイス・ドライバー、およびバッファー・プールが存在します。 これらの「中間」エンティティーはカーネル・レベルに位置します。

DLC デバイス・マネージャー環境のコンポーネントは、次のとおりです。

項目	説明
アプリケーション・ユーザー	アプリケーションまたはアクセス方式としてカーネルより上位に位置します。
カーネル・ユーザー	カーネル・プロセスまたはデバイス・マネージャーとしてカーネル内に位置します。
ファイル入出力サブシステム	ファイル・ディスクリプター・サブルーチンとファイル・ポインター・サブルーチンを、スイッチ・テーブルのファイル・ポインター・アクセスに変換します。
バッファー・プール	通信サブシステム用のデータ・バッファー・サービスを提供します。
通信入出力デバイス・ドライバー	ハードウェア・アダプターの入出力と DMA (ダイレクト・メモリー・アクセス) レジスターを制御し、受信パケットを複数の DLC に経路指定します。
アダプター	通信メディアに接続します。

GDLC 仕様に準拠して作成されたデバイス・マネージャーは、通信デバイス・ドライバーとそのターゲット・アダプターを含んでいるすべてのオペレーティング・システムのハードウェア構成上で稼働します。各デバイス・マネージャーは、上位の複数ユーザーとともに、下位の複数のデバイス・ドライバーとアダプターをサポートします。一般に、複数のユーザーが 1 つのアダプターを介して並行して操作するか、または各ユーザーが複数のアダプターを介して操作します。DLC デバイス・マネージャーは、そのプロトコルの制約に応じてさまざまなものがあります。

722 ページの図 44 は、複数ユーザー構成を示したものです。

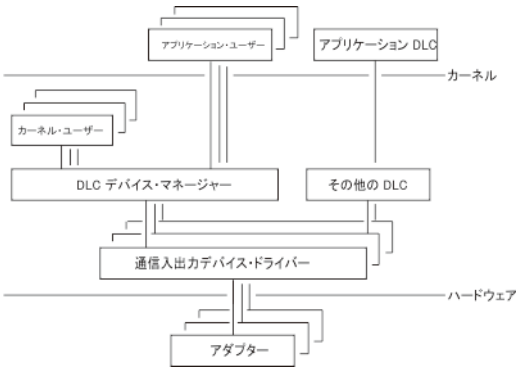


図 *44.* 複数のユーザーと複数アダプターの構成

この図は、アプリケーション・ユーザーとアダプターの間のカーネル・レベルを別の視点から示したものです。この図には、複数のユーザーを表す複数のエンティティーが示されています。

GDLC 基準

GDLC インターフェースは、次の基準を満たすことが必要です。

- 柔軟であり、アプリケーション・ユーザーとカーネル・ユーザーの両方からアクセスできること。
- 複数ユーザーと複数アダプターへの適応性があり、プロトコルが複数のセッションとポートを使用できること。
- 可能な場合は、コネクション指向およびコネクションレスの両方のサービスをサポートしていること。
- 使用中の DLC デバイス・マネージャーの機能の範囲を超える特別な必要条件が絡む場合は、透過的データ転送が可能であること。

GDLC インターフェース

各 DLC デバイス・マネージャーは、指定したプロトコル用の複数のデバイス・マネージャーとしてカーネル内で動作する標準 /dev エントリーです。

DLC が使用していないアダプターの場合、DLC デバイス・マネージャーに対する各 open サブルーチンが、カーネル・プロセスを作成します。 open サブルーチンは、ターゲット・アダプターのデバイス・ハンドラーに対しても発行されます。 必要があれば、同じプロトコルの複数 DLC アダプター・ポートに対して open サブルーチンを追加発行してください。 同じポートを対象として open サブルーチンを発行しても、カーネル・プロセスの追加ではなく、open サブルーチンは既存プロセスと連結します。 使用中のポートごとに、1 つのカーネル・プロセスが必ず存在します。

DLC デバイス・マネージャーの内部構造は、カーネル・プロセスが非同期イベントで割り込みハンドラーを置き換える点を除いて、カーネル・デバイス・ハンドラーと基本構造が同じです。 読み取り、書き込み、入出力制御、およびブロック選択機能を、次の「標準カーネル・デバイス・マネージャー」の図に示します。

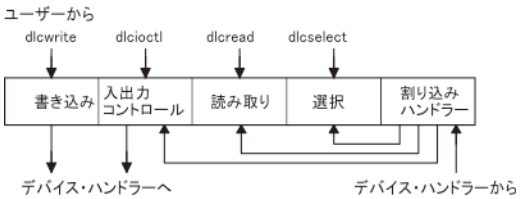


図 *45.* 標準カーネル・デバイス・マネージャー

この図には、DLC デバイス・マネージャーの内部構造が示されています。 この構造は、書き込み、入出力制御、読み取り、選択、および割り込みハンドラーから構成されています。 デバイス・マネージャーは、ユーザーからの情報が各種の領域に渡される際にこれを受け取ります。この情報は、その後デバイス・ハンドラーに渡されます。

GDLC データ・リンク制御

DLC は、個別にインストールすることも、グループ単位でインストールすることもできます。 DLC デバイス・マネージャーは、インストールした DLC のタイプごとに、自動的にカーネルに追加され、「Available (使用可能)」状態に設定されます。

次のように **lslpp** コマンドを入力すると、インストールを確認できます。

```
lslpp -h dlctype
```

この *dlctype* は、次のいずれかです。

項目	説明
bos.dlc.8023	IEEE イーサネット (802.3) データ・リンク制御
bos.dlc.ether	標準イーサネット・データ・リンク制御
bos.dlc.fddi	FDDI データ・リンク制御
bos.dlc.sdlc	SDLC データ・リンク制御
bos.dlc.token	トークンリング・データ・リンク制御

インストールした DLC についての情報は、システム管理インターフェース・ツール (SMIT)、またはコマンド・ラインによって表示できます。使用頻度の高いシステムや通信ポートに関しては、DLC 属性を変更して DLC のパフォーマンスを微調整することが必要な場合があります。 受信パフォーマンスが低く、かつ DLC とそのデバイス・ハンドラーとの間にリング・キューのオーバーフローが発生していることをシステム・エラー・ログが示している場合は、着信データ用の DLC キューの項目数を増やしてください。 インストールした DLC を長期間使用しない場合は、その DLC をカーネルから除去することをお勧めします。 除去しても、その DLC はシステムから撤去されるわけではありません。DLC がまた必要になるまで、カーネル・リソースを他のタスクのために解放するだけです。 これらのタスクのすべての説明は、727 ページの『DLC デバイス・ドライバーの管理』にあります。

GDLC インターフェース ioctl エントリー・ポイントの操作

汎用データ・リンク・コントロール (GDLC) インターフェースは、次の **ioctl** サブルーチン操作をサポートしています。

項目	説明
DLC_ENABLE_SAP	サービス・アクセス・ポイント (SAP) を使用可能にします。
DLC_DISABLE_SAP	SAP を使用不可にします。
DLC_START_LS	特定の SAP 上のリンク・ステーションを呼び出し側またはリスナーとして始動します。
DLC_HALT_LS	リンク・ステーションを一時停止します。
DLC_TRACE	リンク・ステーションの短期間または長期間のアクティビティーをトレースします。
DLC_CONTACT	特定のローカル・リンク・ステーションからリモート・ステーションと接続します。
DLC_TEST	特定のローカル・リンク・ステーションからリモートへのリンクをテストします。
DLC_ALTER	リンク・ステーションの構成パラメーターを変更します。
DLC_QUERY_SAP	特定の SAP の統計を照会します。
DLC_QUERY_LS	特定のリンク・ステーションの統計を照会します。
DLC_ENTER_LBUSY	特定のリンク・ステーション上でローカル・ビジー・モードに入ります。
DLC_EXIT_LBUSY	特定のリンク・ステーション上でローカル・ビジー・モードを終了します。
DLC_ENTER_SHOLD	特定のリンク・ステーション上で短期保留モードに入ります。
DLC_EXIT_SHOLD	特定のリンク・ステーション上で短期保留モードを終了します。
DLC_GET_EXCEP	非同期例外通知をアプリケーション・ユーザーに戻します。 **注:** すべての例外条件は、例外ハンドラーを介してカーネル・ユーザーに渡されるので、この `ioctl` サブルーチン操作をカーネル・ユーザーが使用することはありません。
DLC_ADD_GRP	グループ受信アドレスまたはマルチキャスト受信アドレスをポートに追加します。
DLC_DEL_GRP	グループ受信アドレスまたはマルチキャスト受信アドレスをポートから除去します。
DLC_ADD_FUNC_ADDR	グループ受信機能アドレスまたはマルチキャスト受信機能アドレスをポートに追加します。
DLC_DEL_FUNC_ADDR	グループ受信機能アドレスまたはマルチキャスト受信機能アドレスをポートから除去します。
IOCINFO	GDLC デバイス・マネージャーを記述する構造を戻します。 詳しくは、`/usr/include/sys/devinfo.h` ファイルの形式を参照してください。

GDLC サービス・アクセス・ポイント

サービス・アクセス・ポイント (SAP) によって、特定クラスのデータを送受信する特定のユーザー・サービスが識別されます。

このため、異なるクラスのデータを対応するサービス・ハンドラーに個別に経路指定できます。 複数のコンカレント SAP をサポートする DLC には、宛先 SAP および送信元 SAP と呼ばれるアドレスがそのパケット・ヘッダーに埋め込まれています。 1 つの SAP だけをサポートできる DLC に、SAP アドレッシングは不

要であるか、または使用されません。 しかし、1 つの SAP を使用可能にするという概念はまだ残っているため、一般に、各ポート上の DLC ユーザーごとに使用可能な SAP が 1 つあります。

ほとんどの SAP アドレス値は、IEEE 標準化ネットワーク管理エンティティーによって定義されたものか、「*Token-Ring Network Architecture Reference*」に指定されたユーザー定義の値です。 一般的な SAP アドレスの一部を次に示します。

項目	説明
Null SAP (0x00)	SAP が使用可能でない状態でも、リモート・ノードに応答できるようにします。 この SAP は、コネクションレス・サービスのみをサポートし、XID および TEST リンク・プロトコル・データ単位 (LPDU) のみに応答します。
SNA Path Control (0x04)	SNA (システム・ネットワーク体系) ノードが使用するデフォルトの個々の SAP アドレスを示します。
PC Network NETBIOS (0xF0)	NetBIOS (ネットワーク基本入出力システム) エミュレーションによって実行されるすべての DLC 通信に使用されます。
Discovery SAP (0xFC)	ローカル・エリア・ネットワーク (LAN) 名前ディスカバリー・サービスによって使用されます。
Global SAP (0xFF)	アクティブなすべての SAP を識別します。

GDLC リンク・ステーション

リンク・ステーション (LS) は、1 対の SAP に対する 2 つのノード間の接続を識別します。

この接続は、コネクションレス・サービス (データグラム) としても、コネクション指向サービス (エラー・リカバリーによって完全にシーケンス化されたデータ転送) としても動作できます。 一般に、リモート接続ごとに 1 つずつ LS が始動されます。

GDLC ローカル・ビジー・モード

LS がコネクション指向モードで動作しているとき、リソースの停止などの理由から、リモート・ステーションに対して情報パケットの送信を停止させる必要があります。 この場合、リモート・ステーションに通知を送信し、ローカル・ステーションをローカル・ビジー・モードにすることができます。

リソースが使用可能になると、ローカル・ステーションはリモート・ステーションに対してビジーでなくなったことと、情報パケットを再び送信できることを通知します。 シーケンス化された情報パケットのみが、ローカル・ビジー・モードで一時停止されます。 それ以外のタイプのデータはすべて影響を受けません。

GDLC 短期保留モード

特定のデータ・ネットワークを介して操作するときは、短期保留モードの操作を行います。

短期保留モードは、次の特性を持つデータ・ネットワークに有用です。

- 短期呼び出しセットアップ時間
- 接続時間に対する料金と比べて、少額の呼び出しセットアップ料金が指定されている料金構造

短期保留モードでは、2 つのステーション間の接続は両者間に転送できるデータがある間だけ維持されます。 送信するデータがなくなると、指定のタイムアウト期間後に接続が切断され、転送する新しいデータが発生したときにのみ再確立されます。

GDLC リンクのテストとトレース

2 つのステーション間の接続をテストするには、LS に対してローカル・ステーションからテスト・パケットを送信するように指示します。 接続が正しく行われていれば、このパケットは、リモート・ステーションからエコー・バックされます。

一部のデータ・リンクでは、プロトコルの制約によりこの機能のサポートが制限されています。 例えば、SDLC の場合、テスト・パケットはホストまたは 1 次ステーションからしか発行できません。 一方、他のほとんどのプロトコルの場合は、ホスト 1 次ステーション以外からもテスト・パケットを発信できます。

リンク、回線データ、および特殊イベント (ステーションのアクティブ化、終了、タイムアウトなど) をトレースするには、汎用トレース・チャネルを取得し、LS に対して、各 LS の汎用トレース機能にトレース・ログを書き込むように指示します。 この機能は、特定の通信接続問題の原因を判別するのに役立ちます。 簡略トレース・エントリーと詳細トレース・エントリーの両方がサポートされます。

GDLC 統計情報
GDLC ユーザーは、SAP と LS の両方の統計を照会できます。

SAP の統計は、現在の SAP の状態とデバイス・ハンドラーに関する情報で構成されます。 LS の統計は、現在のステーションの状態、およびステーションのアクティビティーを 始動時からモニターしている各種の信頼性、可用性、保守性の度合いを示すカウンターで構成されます。

GDLC 特殊カーネル・サービス

汎用データ・リンク・コントロール (GDLC) は、カーネル・ユーザーに特殊サービスを提供します。

ただし、トラステッド環境がカーネル内に存在しなければなりません。 非同期イベント・データをユーザー・スペースにコピーする DLC デバイス・マネージャーの代わりに、カーネル・ユーザーは、関数ハンドラーと呼ばれる特殊ルーチンを指す関数ポインターを指定する必要があります。 関数ハンドラーは、DLC によって実行時にコールされます。 これによって、カーネル・ユーザーと DLC 層間で最大限のパフォーマンスが得られます。 各カーネル・ユーザーは、関数ハンドラーの数を最小パス長に制限し、通信メモリー・バッファー (mbuf) 方式を使用する必要があります。

関数ハンドラーは、別の DLC エントリーを直接コールすることはできません。 これは、直接のコールがロック状態で行われ、致命的なスリープ状態が発生するからです。 この唯一の例外は、カーネル・ユーザーが 4 つの受信データ関数のいずれかを使用しているときに `dlcwritex` エントリー・ポイントをコールすることができます。 `dlcwritex` エントリー・ポイントをコールすると、途中でタスクを切り替えずに応答を即時に生成できます。 書き込み操作をコールしているユーザーのプロセス識別コードを検査するには、特殊な論理が DLC デバイス・マネージャー内になければなりません。 そのプロセスが DLC プロセスであり、DLC 内部のキューイング容量を超えている場合には、呼び出しプロセス (DLC) をスリープ状態にする代わりに、書き込み操作が、正しくない戻りコード (`EAGAIN` 戻り値) とともに戻されます。 その場合、受信データ関数からの特殊な通知を DLC に戻すのは、コール側ユーザー・サブルーチンの作業です。これにより、あとで受信バッファーの処理を再試行できます。

ユーザーが作成する関数ハンドラーは、次のとおりです。

項目	説明
データグラム・データ受信ルーチン	カーネル・ユーザーに対するデータグラム・パケットが受信されるとコールされます。
例外条件ルーチン	SAP Closed や Station Contacted など、カーネル・ユーザーに通知しなければならない非同期イベントが発生するとコールされます。
I フレーム・データ受信ルーチン	カーネル・ユーザーに対する通常のシーケンス化データ・パケットが受信されるたびにコールされます。
ネットワーク・データ受信ルーチン	カーネル・ユーザーに対するネットワーク固有のデータが受信されるとコールされます。
XID データ受信ルーチン	カーネル・ユーザーに対する交換識別 (XID) パケットが受信されるとコールされます。

非同期関数エントリーは、DLC デバイス・マネージャーによって直接コールされるので、カーネル・ユーザーは DLC の `dlcread` エントリー・ポイントと `dlcselect` エントリー・ポイントをコールしません。 一般に、これらのイベントをキューイングするには、ユーザーの関数ハンドラー内で実行する必要があります。 しかし、カーネル・ユーザーがある受信パケットを処理できない場合、DLC デバイス・マネージャーが最後の受信バッファーを保持し、次の 2 つの特殊ユーザー・ビジー・モードのいずれかに入ることがあります。

ユーザー終了ビジー・モード (I フレームのみ)

カーネル・ユーザーが受信した I フレームを (キューのブロックなどの問題により) 処理できない場合、DLC_FUNC_BUSY 戻りコードが戻され、DLC はバッファー・ポインターを保持してローカル・ビジー・モードに入り、リモート・ステーションによる I フレームの送信を停止します。 カーネル・ユーザーは、Exit Local Busy 関数をコールしてローカル・ビジー・モードをリセットし、I フレームの受信を再開する必要があります。 停止できるのは、通常のシーケンス化 I フレームのみです。 XID、データグラム、およびネットワーク・データは、ローカル・ビジー・モードに影響されません。

タイマー終了ビジー・モード (すべてのフレーム・タイプ)

カーネル・ユーザーが特定の受信パケットを処理できず、DLC に受信バッファーを短期間保持させたあとでユーザーの受信関数を再度コールしたい場合、DLC_FUNC_RETRY 戻りコードが DLC に戻ります。受信パケットがシーケンス化 I フレームである場合、ステーションは、その期間だけローカル・ビジー・モードになります。 いずれの場合もタイマーが始動され、さらにタイムアウトになると、再び受信データ関数エントリーがコールされます。

DLC デバイス・ドライバーの管理

DLC を使用する前に、DLC をシステムに追加する必要があります。

インストールした各 DLC は、インストール後およびシステムを再始動するたびに、自動的に追加されます (723 ページの『GDLC データ・リンク制御』を参照してください)。 DLC の除去後に再始動を行っていない場合は、その DLC を再び追加できます。

表 *112. DLC* デバイス・ドライバーの管理タスク

タスク	SMIT 高速パス	コマンドまたはファイル
インストールした DLC の追加	次のいずれかを (デバイス・ドライバー名で) 選択します。`smit cmddlc_sdlc` `smit cmddlc_token` `smit cmddlc_qllc` `smit cmddlc_ether`[1] `smit cmddlc_fddi` 次に **Add** を選択します。	**mkdev**[2]
DLC 属性の変更 [3,4]	次のいずれかを (デバイス・ドライバー名で) 選択します。`smit cmddlc_sdlc_ls` `smit cmddlc_token_ls` `smit cmddlc_qllc_ls` `smit cmddlc_ether_ls`[1] `smit cmddlc_fddi_ls`	**chdev**[2]
DLC ローカル・エリア・ネットワークのモニター・トレースの開始 [5]	`smit trace`	**trace -j** *nnn*。ここで、値 *nnn* はトレースされるフック ID です。
DLC ローカル・エリア・ネットワークのモニター・トレースの中止	`smit trcstop`	**trcstop**[2]
DLC ローカル・エリア・ネットワークのモニター・トレース・レポートの生成	`smit trcrpt`	**trcrpt -d** *nnn*。ここで、値 *nnn* は報告されるフック ID です。

表 112. DLC デバイス・ドライバーの管理タスク (続き)		
タスク	**SMIT 高速パス**	**コマンドまたはファイル**
現在の DLC 情報のリスト表示 [3]	次のいずれかを (デバイス・ドライバー名で) 選択します。`smit cmddlc_sdlc_ls` `smit cmddlc_token_ls` `smit cmddlc_qllc_ls` `smit cmddlc_ether_ls`[1] `smit cmddlc_fddi_ls`	**lsdev**[2] または **lsattr**[2]
DLC の除去 [3,6]	次のいずれかを (デバイス・ドライバー名で) 選択します。`smit cmddlc_sdlc_rm` `smit cmddlc_token_rm` `smit cmddlc_qllc_rm` `smit cmddlc_ether_rm`[1] `smit cmddlc_fddi_rm`	**rmdev**[2]

注:

1. イーサネット・デバイス・マネージャー用の SMIT 高速パスには、標準イーサネットと IEEE 802.3 イーサネットの両方のデバイス・マネージャーが含まれます。
2. コマンド・ライン・オプションの詳細については、**mkdev**、**chdev**、**trace**、**trcstop**、**trcrpt**、**lsdev**、**lsattr**、および **rmdev** の各コマンドの説明を参照してください。
3. DLC の現行属性の列挙、表示、変更、除去を行うには、前もって DLC がインストールおよび追加されていなければなりません (723 ページの『GDLC データ・リンク制御』を参照してください)。 変更対象の DLC に対してオープン操作が行われていない場合に限り、属性を変更できます。 変更操作を実行する前に、SNA、OSI、NetBIOS などのサービスを停止して DLC を使用しないようにする必要があります。
4. 受信キューのサイズを変更すると、システム・リソースに直接影響します。 この変更は、パフォーマンスが低い、DLC とそのデバイス・ハンドラー間のオーバーフローなど、DLC に受信キューに関する問題が起きている場合に限り行ってください。
5. モニター・トレースは、DLC とその関連機能のパフォーマンスに直接影響するので、使用可能にするときには注意してください。
6. 対象の DLC に対してオープン操作が行われていない場合に限り、DLC を除去できます。 除去操作を実行する前に、SNA、OSI、NetBIOS などのサービスを停止し、DLC を使用しないようにする必要があります。

通信およびネットワーク・アダプター・リファレンス

ここでは、PCI アダプターと 非同期通信アダプターの 異なった構成シナリオを参照できます。

PCI アダプター

ここでは、PCI アダプターのインストールと 構成に関する情報を記載します。

ここで説明する項目は、PCI 広域ネットワーク (WAN) アダプター (728 ページの『2 ポート・マルチプロトコル HDLC ネットワーク・デバイス・ドライバー 』と 729 ページの『ARTIC960Hx PCI アダプター 』) のサポートと構成です。

2 ポート・マルチプロトコル HDLC ネットワーク・デバイス・ドライバー

2 ポート・マルチプロトコル・アダプター (PCI) のハイレベル・データ・リンク制御 (HDLC) デバイス・ドライバーは、通信入出力サブシステムのコンポーネントです。 このデバイス・ドライバーは、1.544 Mbps までの速度で 2 ポート・マルチプロトコル・アダプター上の HDLC の動作をサポートします。

次のオプションによって、2 ポート・マルチプロトコル HDLC ネットワーク・デバイス・ドライバーにアクセスできます。

- システム・ネットワーク体系 (SNA)
- GDLC プログラミング・インターフェースの 同期データ・リンク制御 (SDLC) バージョン
- SDLC MPQP-API (マルチプロトコル・クワッド・ポート - アプリケーション・プログラミング・インターフェース) と互換性のあるユーザー作成アプリケーション

注 : これらのオプションでは、mpc*n* スペシャル・ファイルを使用する必要があります。このファイルを使用すると、SDLC COMIO デバイス・ドライバー・エミュレーション・サブシステムを介して 2 ポート・マルチプロトコル・アダプターの HDLC デバイス・ドライバーにアクセスできます。 このサブシステムは、HDLC ネットワーク・デバイスごとにインストールして構成する必要があります。

- HDLC 共通データ・リンク・インターフェース (CDLI) API と互換性のあるユーザー作成アプリケーション

2 ポート・マルチプロトコル・アダプター・デバイス・ドライバーでは、2 ポート・マルチプロトコル・アダプターを使用して、リモート・ホスト・システムに専用回線を介して直接接続することも、交換回線を介して接続することもできます。 このデバイス・ドライバーによって、作業グループ環境とリモート・データ処理機能の間にゲートウェイが提供されます。

2 ポート・マルチプロトコル・アダプターの構成
2 ポート・マルチプロトコル・アダプターを 構成するには、以下の説明を参照してください。

表 *113. 2* ポート・マルチプロトコル・アダプターの構成タスク

タスク	**SMIT 高速パス**
アダプターへのデバイス ・ドライバーの追加	`smit mkhdlcdpmpdd`
アダプター上のデバイス・ドライバーの再構成	`smit chhdlcdpmpdd`
アダプター上のデバイス・ドライバーの除去	`smit rmhdlcdpmpdd`
定義済みデバイス・ドライバーの使用可能化	`smit cfghdlcdpmpdd`
アダプターへの SDLC COMIO エミュレーターの追加	`smit mksdlcsciedd`
アダプター上の SDLC COMIO エミュレーターの再構成	`smit chsdlcsciedd`
アダプター上の SDLC COMIO エミュレーターの除去	`smit rmsdlcsciedd`
定義済み SDLC COMIO エミュレーターの使用可能化	`smit cfgsdlcsciedd`

ARTIC960Hx PCI アダプター
ARTIC960Hx PCI アダプターの MPQP COMIO デバイス・ドライバー・エミュレーターは、通信入出力サブシステムのコンポーネントです。 このデバイス・ドライバーは、最高速度 2M bps で ARTIC960HX PCI アダプターをサポートします。

サポートされるのは外部クロックのみなので、使用するモデムでクロックを提供する必要があります。

次のオプションによって、ARTIC960Hx PCI アダプター MPQP COMIO デバイス・ドライバーにアクセスできます。

- システム・ネットワーク体系 (SNA)
- 汎用データ・リンク・コントロール (GDLC) プログラミング・インターフェース
- MPQP-API (マルチプロトコル・クワッド・ポート - アプリケーション・プログラミング・インターフェース) と互換性のあるユーザー作成アプリケーション (例えば、SDLC および BiSync アプリケーション)

これらのオプションでは、mpq*x* スペシャル・ファイルを使用することが必要です。このファイルを使用すると、ARTIC960Hx PCI アダプターに MPQP COMIO エミュレーション・デバイス・ドライバーを介してアクセスできます。 このデバイス・ドライバーは、ARTIC960Hx PCI アダプター上のポートごとにインストールして構成する必要があります。 mpq*x* スペシャル・ファイルは、/dev ディレクトリーにあります。

注 : mpq*x* の *x* は、デバイス・ドライバーのインスタンス (mpq0 など) を指定します。

MPQP COMIO エミュレーション・デバイス・ドライバーでは、ARTIC960Hx PCI アダプターを 使用して、リモート・ホスト・システムに専用回線を介して直接接続することができます。 このデバイス・ドライバーによって、作業グループ環境とリモート・データ処理機能の間にゲートウェイが提供されます。

ARTIC960Hx PCI アダプターの MPQP COMIO エミュレーション・ドライバーの構成
ARTIC960Hx PCI アダプターの MPQP COMIO エミュレーション・ドライバーを構成するには、以下の説明を参照してください。

表 *114. MPQP COMIO* エミュレーション・ドライバーの構成タスク

タスク	**SMIT 高速パス**
デバイス・ドライバーの追加	`smit mktsdd`
MPQP COMIO エミュレーション・ドライバーの再構成	`smit chtsdd`
デバイス・ドライバーの除去	`smit rmtsdd`
定義済みデバイス・ドライバーの構成	`smit cfgtsdd`
ポートの追加	`smit mktsdports`
MPQP COMIO エミュレーション・ポートの再構成	`smit chtsdports`
ポートの除去	`smit rmtsdports`
定義済みポートの構成	`smit cfgtsdports`
MPQP COMIO エミュレーション・ドライバーのトレース	`smit trace_link`

非同期通信アダプター

次の表にリストされている標準、8 ポート、および 16 ポートの各非同期通信アダプター。

次の表に、これらの製品の要約を示します。

表 *115.* 非同期通信アダプター

非同期接続	**バス・タイプ**	**フィーチャー・コードまたはマシン・タイプ (モデル)**	**ポートごとの最大データ速度 (KBit/秒)**	**主要な機能**
8 ポート EIA 232	Micro Channel	2930	76.8	広範囲な標準機能
8 ポート EIA 422A	Micro Channel	2940	76.8	長距離
8 ポート MIL-STD 188	Micro Channel	2950	UART のボー・レート生成プログラム・クロック速度に基づいて選択可能。	不均衡電圧デジタル・インターフェース用の MIL-STD 188-114
8 ポート EIA 232	ISA	2931	115.2	高能率
8 ポート EIA 232	ISA	2932	115.2	高能率
8 ポート EIA 422	PCI	2943	230	高能率
16 ポート EIA 232	Micro Channel	2955	76.8	ローカル接続フォーカス
16 ポート EIA 422A	Micro Channel	2957	76.8	長距離

表 115. 非同期通信アダプター (続き)				
非同期接続	**バス・タイプ**	**フィーチャー・コードまたはマシン・タイプ (モデル)**	**ポートごとの最大データ速度 (KBit/秒)**	**主要な機能**
-	ISA	2933	-	-
-	PCI	2944	-	-

次の表は、詳細な製品特性を示しています。

表 116. 非同期接続製品の特性						
	ネイティブ・シリアル・ポート	**8 ポート**		**16 ポート**	**128 ポート (RAN あり)**	
		MC	**ISA**		**MC**	**ISA**
アダプターごとの非同期ポート数	該当なし	8	8	16	128	128
アダプターの 最大数	該当なし	8	7	8	7	7
非同期ポートの最大数	2 または 3	64	56	128	896	896
RAN ごとの非同期ポート数	該当なし	該当なし	該当なし	該当なし	16	16
RAN の最大数	該当なし	該当なし	該当なし	該当なし	56	56
最大スピード (KBit/秒)	UART のボー・レート生成プログラム・クロック速度に基づいて選択可能。	76.8	115.2	76.8	230	230
接続方式	標準	direct	direct	direct	ノード	ノード
サポートされる非同期エレクトリカル・インターフェース	EIA 232	EIA 232 EIA 422A[4]MIL-STD[4] 188-114[4]	EIA 232 EIA 422A	EIA 232 EIA 422A	EIA 232 EIA 422	EIA 232 EIA 422
標準コネクター	DB25M/ MODU	DB25M	DB25M	DB25M	RJ-45[2]	RJ-45[2]
DB25 ケーブル・オプション	該当なし	該当なし	該当なし	該当なし	RJ-45-DB25	RJ-45-DB25
ラック・マウント・オプション	該当なし	該当なし	該当なし	該当なし	はい	はい
電源装置	該当なし	該当なし	該当なし	該当なし	外付け	外付け
サポートされるシグナル (EIA 232)	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS[3]-DTR - DCD -	TxD RxD RTS CTS DTR DSR DCD RI	TxD RxD RTS CTS DTR DSR DCD RI

注 :

1. ソケットは、信号での縮小がサポートされている場合、8p RJ-45、6p RJ-11、または 4p RJ-11 プラグを受け入れます。
2. 16 ポート・インターフェース・ケーブル EIA 232 (FC 2996) のファンアウト・コネクター・ボックスでは、RTS は高くなります (+12V)。
3. Micro Channel のみ。

各製品オファリングは、製品の長所を引き出す代表的なシナリオによって表現されています。 各アダプターの推奨事項は次のとおりです。

関連概念

TTY 端末デバイス
tty 端末デバイスは、文字単位で入出力を行うキャラクター型デバイスです。

関連情報

2-Port Asynchronous EIA-232 PCI Adapter Installation and Using Guide

8 ポート・マイクロチャネル

8 ポート Micro Channel アダプターには、非同期入出力に使用可能な Micro Channel バス・スロットがあります。

その他の機能は、以下のとおりです。

- 拡張なしで 8 ポート未満のポート。
- すべてのローカル端末の位置は、システムから 61 メートル (200 フィート) 以内。
- リモート端末が必要である (OEM マルチプレクサー/モデムを通してサポート)。
- デバイス帯域幅要求が低から中 (76.8 Kbps まで) である。

8 ポート ISA バス EIA 232 または EIA 232/EIA 422

8 ポート ISA バス EIA 232 または EIA 232/EIA 422 アダプターの 機構には、ISA スロットが含まれます。

その他の機能は、以下のとおりです。

- 拡張がほとんどまたはまったくなしで 8 ポート未満。
- すべて EIA 232、すべて EIA 422、または EIA 232 と EIA 422 混用のポートが必要。
- メイン CPU から文字割り込みおよび端末 I/O 処理をオフロードする。
- 115.2 Kbps までの非同期速度。
- 高速 (28.8 Kbps) モデム用の最大パフォーマンス (データ圧縮あり)。

16 ポート・マイクロチャネル

16 ポート Micro Channel アダプターには、非同期入出力に使用可能な Micro Channel バス・スロットがあります。

その他の機能は、以下のとおりです。

- 拡張なしで 8 以上 16 未満のポート。
- すべてのローカル端末の位置は、システムから 61 メートル (200 フィート) 以内。
- リモート端末が必要である (OEM マルチプレクサー/モデムを通してサポート)。
- デバイスにすべての EIA 232 信号は必要ない。
- デバイス帯域幅要求が低から中 (非同期デバイスの場合は 38.4 Kbps まで) である。

128 ポート・アダプター (マイクロチャネル、ISA)

128 ポートの Micro Channel または ISA アダプターには、追加スロットなしで 128 ポートまで拡張できる 16 個のポートがあります。

その他の機能は、以下のとおりです。

- Micro Channel、ISA、または PCI バス・スロットを非同期入出力に使用可能。 (PCI の詳細については、637 ページの『製品選択の考慮事項』を参照してください。)

- 最も遠い端末を Micro Channel アダプターおよび ISA アダプターの最大データ速度でシステムから約 300 メートル (1000 フィート) の位置に配置可能。
- 端末の計画: 構内の近い位置、構内の遠い位置、リモート。
- 低プロセッサー要求で高い非同期スループットが必要。
- 端末接続プリンター機能が必要。
- 光ファイバーまたは同期モデムによるリモート接続が必要。

SMIT を使用した、定義済みのマイクロチャネル 128 ポート非同期通信アダプターのリスト
使用可能かどうかに関係なく、定義済みのすべての 128 ポート非同期通信アダプターをリストするには、以下の手順を使用します。

1. `smit lsd128psync` 高速パスを使用します。 システムが情報をスキャンして、それを表示します。
2. SMIT インターフェースを 終了します。

8 ポート非同期 ISA/PCI アダプター
8 ポート非同期 ISA アダプターは、POWER プロセッサー・ベース コンピューターで使用可能なマルチチャネルのインテリジェント・シリアル通信フィーチャーです。

ISA アダプターには、プログラム・コードおよびデータ・バッファリングに使用される、128K のデュアル・ポート高速ランダム・アクセス・メモリー (RAM) が組み込まれています。 非同期ポートは、115 Kbps のスループット速度をサポートする 32 ビット 16 MHz IDT 3041 プロセッサーで稼働します。

3041 プロセッサーとデュアル・ポート RAM により、システムの多くの文字処理の負荷が軽減されます。 大きなデータ・ブロックは、アダプターに直接転送されてから、シリアル・ポートへ一度に 1 文字ずつ送信されます。

デュアル・ポート RAM には、アダプターおよびコンピューターの両方が、読み取り操作および書き込み操作のためにアクセスできます。 コンピューターはデュアル・ポート RAM をその所有メモリーと見なし、内部メモリーに使用するのと同じ高速メモリー参照コマンドを使用してこれにアクセスします。

8 ポート EIA 232 ISA アダプターは、EIA 232 デバイスのみをサポートします。 このアダプターは、システムにデバイス・パッケージ `devices.isa.cxia` がインストールされていることを要求します。

8 ポート EIA 232/422 ISA アダプターは、EIA 232 および EIA 422 デバイスをサポートします。 ポートごとに、これらの両方のデバイス・タイプを任意の組み合わせで構成できます。 このアダプターは、システムにデバイス・パッケージ `devices.isa.pc8s` がインストールされていることを要求します。

上記のパッケージには、`devices.common.IBM.cx` パッケージが必要です。

8 ポート・アダプターのインストール
ISA アダプターはオペレーティング・システムによって自動検出されないので、手動でインストールする必要があります。

1. IBM 8 ポート非同期 EIA 232/EIA 422 ISA アダプターを 構成するた め、`smit mkdev_isa` 高速パスを使用して「**Add an ISA Adapter (ISA アダプターの 追加)**」画面にアクセスします。
2. **pcxr** (8 ポート EIA 232 アダプターの 場合) または **pc8s** (8 ポート EIA 232/EIA 422 アダプターの 場合) を選択し、Enter キーを選択します。
3. 適切なバスを選択し、Enter キーを押します。
4. 「Bus I/O Address (バス I/O のアドレス)」フィールドで、アダプターのアドレス (アダプターの DIP スイッチによって設定されています) へのアドレスを設定します。 DIP スイッチの詳細については、*8 Port Asynchronous ISA Adapter Installation Guide* を参照してください。 システムが `saX Available` を表示している場合、残りのアダプター構成は自動的に行われます。
5. 完了したら、「**Do (実行)**」を選択します。

`stty-cxma` は、Micro Channel 128 ポート・アダプターおよび ISA 8 および 128 ポート・アダプターのための端末オプションを設定および表示するユーティリティー・プログラムで、`/usr/lbin/tty` ディレクトリーにあります。 フォーマットは次のとおり。

```
stty-cxma [-a] [option(s)] [ttyname]
```

stty-cxma は、オプションを指定しないと、すべての特殊なドライバーの設定、モデム・シグナル、および標準入力によって参照される tty デバイスの **stty(1)** によって 表示される標準パラメーターを表示します。 コマンド・オプションによって、フロー制御の変更、透過印刷オプションの設定、モデム制御回線の強制、およびすべての tty 設定の表示を行うことができます。 認識されなかったオプションは、**stty(1)** に渡されて解釈されます。 オプションは、PCI アダプターに 使用されるオプションと同じです。 詳しくは、677 ページの『stty-cxma 端末オプション』を参照してください。

標準 I/O ポート
ほとんどのシステム装置モデルには、2 つの内蔵 (標準) EIA 232 非同期シリアル・ポートがあります。

モデル M20/M2A は単一の内蔵の非同期シリアル・ポートを備えており、これはオプションの fanout ケーブルを使用して、2 つのシリアル・デバイスをサポートするように変換することができます。 EIA 232 非同期シリアル・デバイスは、標準シリアル・ケーブルと 9 ピンまたは 25 ピンの D シェル・コネクターを使用して、標準シリアル・ポートに直接接続できます。

> **注:** Itanium ベースのプラットフォームの 場合、EIA 232 非同期シリアル・デバイスは、標準シリアル・ケーブルと 9 ピン D シェル・コネクターを使用して、標準シリアル・ポートに直接接続することができます。

マルチプロセッシング対応のマシンには、3 つのシリアル・ポートがあります。

EIA 232 非同期端末デバイスの構成
標準シリアル・ポート、8 ポート、または 16 ポートの非同期通信アダプターに接続された tty デバイスの定義および構成は、以下の手順で行えます。

1. `smit mktty` 高速パスを使用して、「**Add a TTY (TTY の追加)**」メニューにアクセスします。
2. 「**Add a TTY (TTY の追加)**」を選択します。
3. 「**tty rs232 Asynchronous Terminal (tty rs232 非同期端末)**」を選択します。
4. 画面に表示された使用可能な標準 I/O、8 ポート、または 16 ポートのアダプターの中から選択をします。

   アダプターが 表示されない場合、またはそれらが定義済み状態にある場合は、構成、配線、およびセットアップをもう一度確認してください。
5. 表示されたダイアログ・フィールドで、TTY 属性を追加または変更することができます。
6. 完了したら、「**Do (実行)**」を選択します。

EIA 232 非同期プリンター/プロッター・デバイスの構成
標準シリアル・ポート、8 ポート非同期通信アダプター、または 16 ポート非同期通信アダプターに 接続されたプリンター/プロッター・デバイスの定義および構成は、以下の手順で行えます。

1. 非同期通信アダプター 上にプリンター/プロッター・デバイスを作成するため、`smit pdp` 高速パスを使用して「**Printer/Plotter Devices (プリンター/プロッター・デバイス)**」メニューにアクセスします。
2. 「**Add a Printer/Plotter (プリンター/プロッターの追加)**」を選択します。
3. 画面上に表示されたプリンターとプロッターのタイプのリストから 選択をし、Enter キーを押します。 この例では、以下を選択しました。

   ```
   osp Other serial printer
   ```
4. **rs232** オプションを選択します。
5. 画面上の使用可能な 8 ポート・コントローラーの中から選択をします。 コントローラーが表示されない場合、またはそれらが定義済み状態として表示されている場合は、構成、配線、およびセットアップをもう一度確認してください。
6. 表示されたダイアログ・フィールドで、プリンター/プロッターの属性を追加または変更することができます。
7. 完了したら、「**Do (実行)**」を選択します。

8 ポート・マイクロチャネル非同期通信アダプター

非同期通信アダプターのファミリーは、共通の機能設計に基づきます。 しかし、個々のアダプター特性は、サポートされるデバイス・インターフェースによって決まります。

注 : 以下のセクションは、Itanium ベースのプラットフォームには該当しません。

このファミリーは、以下の 3 つのアダプターで 構成されています。

- 8 ポート非同期通信アダプター - EIA 232
- 8 ポート非同期通信アダプター - MIL-STD-188
- 8 ポート非同期通信アダプター - EIA 422A

8 ポート・アダプターのファミリーは、2 つのシリアル通信チャネルを備えた dual universal asynchronous receiver and transmitter (DUART) チップに基づきます。

以下のセクションでは、8 ポート・アダプターの詳細について説明します。

8 ポート非同期通信アダプター - EIA 232

EIA 232 は、システム装置への最大で 8 つの EIA 232D 非同期シリアル・デバイス (モデム、端末、プロッター、およびプリンターなど) の接続をサポートする 8 ポート非同期通信アダプターです。

システムは、Micro Channel バスまたは ISA バスを基礎とし、8 ポート・アダプターを 8 つまでサポートしている必要があります。

このアダプターは、完全にプログラマブルであり、非同期通信のみをサポートします。 さらに、スタート・ビットとストップ・ビットを追加および除去することもでき、シリアル・データに対する偶数パリティー、奇数パリティー、またはパリティーなしもサポートします。 プログラマブル・ボー・レート・ジェネレーターは、Micro Channel バスでは 50 から 38,400 bps まで、ISA バスでは 50 から 115,200 bps までの操作を許可します。アダプターは、1、1.5、または 2 ストップ・ビットを含む 5、6、7、または 8 ビット文字をサポートします。 優先割り込みシステムは、送信、受信、エラー、回線状況、およびデータ・セット割り込みを制御します。

8 ポート非同期通信アダプターのインストール

8 ポート非同期通信アダプターは、システムの単一の Micro Channel スロットに収まります。アダプターをインストールするには、以下の手順を使用します。

1. すべてのユーザーがシステムからログオフしていることを確認し、次のコマンドを実行します。

   ```
   shutdown -F
   ```

2. **shutdown** コマンドが完了したら、システムの電源スイッチをオフの位置に切り替えます。
3. システムのケースを 開き、8 ポート非同期通信アダプターを 空いている Micro Channel スロットに挿入します。
4. 78 ピン D シェル・コネクターを 8 ポート・インターフェース ・ケーブルから 8 ポート・アダプターに接続します。
5. システム 装置にカバー・パネルを戻します。
6. システムの電源スイッチをオンの位置まで押します。 ブート・プロセス中にシステムが 8 ポート・アダプターを認識して構成します。
7. ブートの完了後、root ユーザー ID ブート・プロセスを使用してログインします。

   ```
   lsdev -Cc adapter | pg
   ```

システムは、使用可能状態のアダプターのみを 使用することができます。

新しくインストールしたアダプターが 使用不可 である場合は、以下を確認してください。

- アダプターが Micro Channel スロットに正しく取り付けられている。
- 必要なすべてのケーブルが接続され、所定の位置にぴったり収まっている。
- コマンド **errpt -a | pg** を実行し、アダプターに関連した問題についてのシステム・エラー・レポートを調べます。

- コマンド **cfgmgr -v | pg** を実行します。 このコマンドは、リブートなしでアダプターの 再構成を試行します。 ページ送りされる出力にエラーがないか調べます。

実行中の **cfgmgr** が失敗する場合は、リブートが必要になります。

8 ポート非同期通信アダプターのハードウェア情報

システム・インターフェースは、3 ビットのアドレスと 8 ビットのデータ、および DUART チップへの制御回線を提供します。 システム・インターフェースからのデータは、外部デバイスに送信するために直列化されます。 シリアル・データには、バイト境界にパリティー・ビットが含まれることがあります。 反対に、外部デバイスからのデータは、システム・インターフェースに送信するために非直列化されます。 このデータにも、パリティー・ビット (これはオプションで検査できる) が含まれる可能性があります。 オプションで、チャネルは先入れ先出し法 (FIFO) モードでも作動できます。

FIFO モードでは、送信側と受信側の両方において最大 16 バイトをバッファーに入れることができます。 シリアル・インターフェースは、データの送信と受信の両方に start-stop プロトコルを使用します。 すなわち、各バイト (およびパリティー・ビット) が 1 つ以上のスタート・ビットとストップ・ビットでフレーム化され、それにより、個々の文字 (バイト) を基本とした同期が可能になります。

DUART チップは、12.288MHz 発振器を使用して、送信および受信論理を駆動するためのその内部時間を生成します。 このチャネルは全二重操作をサポートします。 各 8 ポート・アダプターには、4 つの DUART チップが実装されています。

13 のシステム・アクセス可能レジスターが用意されています。 各チャネルのプログラマブル・フィーチャーには、以下が含まれます。

- 文字長: 5、6、7、または 8 ビット
- パリティー生成/検出: 奇数、偶数、または「なし」
- ストップ・ビット数: 1、1.5、または 2
- 割り込みの使用可能/使用不可。 有効受信データ
- 送信側保持レジスターの空き
- 回線状況
- オーバーラン・エラー
- パリティー・エラー
- フレーム・エラー
- 中断

次の表は、アダプターのポート (デバイス・インターフェース) の特性の要約です。

表 *117. 8* ポート非同期通信アダプターのポート特性

パラメーター	**EIA 232**	**MIL-STD 188**	**EIA 422A**
トポロジー	Point to Point	Point to Point	Point to Point
最大データ速度	138.4Kbps (MC)/115.2 (ISA)	138.4Kbps	138.4Kbps
送信メディア	マルチコンダクター	マルチコンダクター	マルチコンダクター
ケーブル・ワイヤー数	9 (信号用接地を含む)	9 (信号用接地を含む)	5 (信号用接地を含む)
最大ケーブル長	61m (200 フィート)	38.4Kbps で 130m	1200 m < 90Kbps
デバイス・コネクター	25 ピン D	25 ピン D	25 ピン D
電気インターフェース	不平衡	不平衡	平衡
ビット・エンコード	デジタル 2 レベル	デジタル 2 レベル	デジタル 2 レベル

割り込み調停論理は、以下の図式に従ってアダプターの 優先順位を設定します。

```
Adapter   Priority
    1     Highest
    2       |
    3       |
    4       |
    5       |
    6       |
    7       |
    8     Lowest
```

通信チャネルの優先順位

保留中の割り込みのある DUART チャネルは、一定の優先順位に従って処理されます。

最も高い優先順位はポート 0 に割り当てられます。 次に優先順位が高いのはポート 1 となります。 優先順位が最も低いのはポート 7 となります。

8 ポート非同期通信アダプター割り込み論理の説明

割り込み論理は、割り込み生成論理と割り込み調停論理に分かれています。

これらの両方の論理セクションが、すべての 8 ポート・アダプターに実装されています。 割り込み生成論理は、システムにインターフェースを 提供します。 この論理は、システム割り込み要求を生成し、割り込み共有回路を包含します。

割り込み調停論理の機能は、優先順位が最も高い割り込みが保留になっている 8 ポート・アダプターを 識別することです。 この論理は、優先順位の最も高いポートの割り込み情報を割り込み調停レジスターに入れます。 これは、1 回の読み取り操作で行われます。

割り込み調停論理は 8 ポート・アダプターに 固有なものです。Micro Channel の調停論理と混同しないでください。

8 ポート割り込み生成論理

非同期通信アダプターには、8 つのシステム 割り込み要求回線が実装されます。

アダプターには、以下の 8 つのシステム 割り込み要求回線が実装されています。

- IRQ 3
- IRQ 5
- IRQ 9
- IRQ 10
- IRQ 11
- IRQ 12
- IRQ 14
- IRQ 15

通常の操作時は、要求回線が 1 つだけ活動状態になります。 最適なシステム・パフォーマンスのためには、1 つのシステムの中のすべての 8 ポート・アダプターが 同一の割り込みレベルを使用しているべきです。 アクティブ回線は、セットアップ・サイクル中に適切な POS レジスターに書き込みを行うことによって選択します。 アダプターは、割り込み共有をサポートし、オープン・コレクター構成を実装しています。 この構造では、システム引き上げレジスターによって割り込み回線のレベルが引き上げられます。 アダプターは回線のレベルを引き下げて、アクティブ割り込み要求を示します。

8 ポート割り込み調停論理

割り込み調停論理は、複数の 8 ポート・アダプターまたは 16 ポート・アダプターが 割り込みを生成した場合に、ソフトウェア・サービスの優先順位を決定します。

最大で 8 つの 8 ポート・アダプターがシステムに 共存して並行操作できます。 この論理は、1 回の読み取り操作でシステムにアダプターとポートの 識別と割り込みタイプを提供します。 割り込み要求が検出されると、システムは I/O アドレス 0130 にある 16 ビット割り込み調停レジスターを読み取ります。

8 ポート非同期通信アダプター MIL-STD 188 のインターフェース信号

アダプターの 各ポートには、以下のインターフェース 信号が実装されています。

シグナル	定義
Tx Data	送信データ
RTS	送信要求
CTS	送信可
DSR	データ・セット・レディー
Rx Data	受信データ
DCD	データ・キャリア検出
DTR	データ端末レディー
RI	リング・インディケーター
Sig Gnd	信号用接地

8 ポート MIL-STD 188 信号の電圧レベル

MIL-STD 188 アダプターの 電圧レベルについては、通常のマークおよびスペースの極性、またはマークおよびスペースの極性の反転を通じて説明することができます。

MIL-STD 188 アダプターの 電圧レベルについては、以下のセクションで説明しています。

- 通常のマークおよびスペースの極性
- マークおよびスペースの極性の反転

インターフェース・ポイントで測定した交換回路の電圧が信号用接地に関して -4 V dc より低い場合、信号はマーク状態になります。 信号用接地に関して電圧が +4 V dc より高い場合、信号はスペース状態になります。 +4 V dc と -4 V dc の間の領域は、遷移領域として定義されており、有効レベルではありません。 -6 V dc より低い電圧や +6 V dc より高い電圧も、有効レベルではありません。

データの送信中は、マーク状態がバイナリー 1 を示し、スペース状態がバイナリー 0 を示します。

インターフェース 制御回路の場合、機能は信号用接地に関して電圧が +4 V dc より高いと「オン」になり、電圧が -4 V dc より低いと「オフ」になります。 MIL-STD 188 信号のレベルを、次の表に示します。

表 *118. MIL-STD 188* の信号レベル

交換電圧	バイナリーの状態	信号の状態	インターフェース 制御機能
+ 電圧	0	スペース	オン
- 電圧	1	マーク	オフ

軍の標準 MIL-STD 188 では、アダプターは、送信および受信回線のマーク状態とスペース状態の極性をオプションで反転する機能を提供する必要があります。 この機能は、各ポート上で独立して提供されます。

DUART モデム制御レジスター・ビット 3 (Out 2) は、この目的に使用されます。 ビット 3 が値 1 に設定されている場合、マーク状態とスペース状態の極性は通常の状態に設定されます。 ビット 3 が値 0 に設定されている場合、マーク状態とスペース状態の極性は反転します。

信号用接地に関して電圧が -4 V dc より低い場合、信号はスペース状態 になります。 信号用接地に関して電圧が +4 V dc より高い場合、信号はマーク状態になります。

+4 V dc と -4 V dc の間の領域は、遷移領域 として定義されており、有効レベルではありません。 -6 V dc より低い電圧や +6 V dc より高い電圧も、有効レベルではありません。

8 ポート非同期 MIL-STD 188 アダプター・ポートの電気特性は、MIL-STD 188-114 の不均衡電圧インターフェースについてのセクションの 規格と合致しています。 この標準は、1976 年 3 月 24 日付けのものです。

このアダプター・ポートは、1969 年 10 月付けの EIA 標準 232C、および 1987 年 1 月付けの EIA 標準 232D で記述されている非同期操作の機能要件 (start-stop プロトコル) を満たしています。

8 ポート非同期通信アダプター *EIA 422A* のインターフェース信号

アダプターの各ポートには、以下の EIA 422A インターフェース信号が実装されています。

シグナル	定義
TxA	送信データ
TxB	送信データ
RxA	受信データ
RxB	受信データ
Sig Gnd	信号用接地

8 ポート *EIA 422A* 信号の電圧レベル

回線ドライバーは、2 から 6 ボルト (ジェネレーター・インターフェース・ポイントで測定) の範囲の差動電圧を生成します。 受信側の差動電圧の絶対値は、200mV (ミリボルト) から 6 ボルト (ロード・インターフェース・ポイントで測定) の範囲でなければなりません。

測定値は、端末 B (負のリード線) に関して端末 A (正のリード線) で測定されます。 以下の表は、電圧レベルに関連した信号の状態を示しています。

表 *119. 8* ポート *EIA 422A* 信号の状態

交換電圧	バイナリーの状態	信号の状態
+ 電圧	0	スペース
- 電圧	1	マーク

8 ポート非同期 EIA 422A アダプターは、1200m (4000ft) までの長さの室内ケーブルをサポートします。 このような長さのケーブルは、間接的な雷撃などの誘発電圧による突然の電圧サージの影響を受けやすくなります。 EIA 422A アダプターには、これらの電圧サージからこれを保護するために、副次的なサージ保護回路が実装されています。 このサージ保護回路は、アダプターのインターフェース ・データ回線に実装されています。

受信装置がドライバーに接続されていないとき (オープン・ケーブル) の障害状態を防ぐため、各 EIA 422A 受信装置の入力リード線に、フェイルセーフ回路が追加されています。 フェイルセーフ回路は、受信装置がドライバーに接続されていないときは常に受信装置をマーク状態 (バイナリー 1) に設定します。

8 ポート非同期 EIA 422A アダプター・ポートの電気特性は、1978 年 12 月付けの EIA 標準 422A に準拠しています。

8 ポート非同期通信アダプター *EIA 232* のインターフェース信号

8 ポート非同期通信アダプターの各ポートには、以下のインターフェース信号が実装されています。

アダプターの各ポートには、以下のインターフェース信号が実装されています。

シグナル	定義
TxD	送信データ
RTS	送信要求
CTS	送信可
DSR	データ・セット・レディー
RxD	受信データ
DCD	データ・キャリア検出
DTR	データ端末レディー

シグナル	定義
RI	リング・インディケーター
Sig Gnd	信号用接地

8 ポート *EIA 232* 信号の電圧レベル

インターフェース・ポイントで測定した交換回路の電圧が信号用接地に関して -3 V dc より低い場合、信号はマーク状態になります。 信号用接地に関して電圧が +3 V dc より高い場合、信号はスペース状態になります。 +3 V dc と -3 V dc の間の領域は、遷移領域 として定義されており、有効レベルではありません。 -15 V dc より低い電圧や +15 V dc より高い電圧も、有効レベルではありません。

データの送信中は、マーク状態がバイナリー状態 1 を示し、スペース状態がバイナリー状態 0 を示します。

インターフェース 制御回路の場合、機能は信号用接地に関して電圧が +3 V dc より高いとオンになり、電圧が -3 V dc より低いとオフになります。 EIA 232 の信号レベルについては、以下の表を参照してください。

表 *120. EIA 232* の信号レベル

交換電圧	バイナリーの状態	信号の状態	インターフェース 制御機能
+ 電圧	0	スペース	オン
- 電圧	1	マーク	オフ

8 ポート非同期 EIA 232 アダプター・ポートの電気特性は、1969 年 10 月付けの EIA 標準 232C、および 1987 年 1 月付けの EIA 標準 232D の規格と合致しています。

このアダプター・ポートは、1969 年 10 月付けの EIA 標準 232C、および 1987 年 1 月付けの EIA 標準 232D で記述されている非同期操作の機能要件 (start-stop プロトコル) を満たしています。

8 ポート非同期通信アダプターの制御論理

PAL ベースの制御論理セクションは、すべての主要なアダプター機能のアクティビティーを 調整します。

これは 40 MHz 方形波ジェネレーターで刻時されます。 これは Micro Channel と相互作用し、その機能にはアドレスのデコード、アドレス・パリティーの検査、適切な I/O 制御信号での応答、および選択された割り込み要求 (IRQ) 回線 (8 つの IRQ 回線のいずれか) の駆動が含まれます。

制御論理は他のアダプター論理ブロックと相互作用し、その間に通信チャネル (DUART) への制御回線と割り込み調停論理を提供します。 また、制御論理はデータ・バス・ドライバー論理とも相互作用し、データ・フローの方向と、ローカル・バスに置かれる選択データ・バイトの制御を提供します。 これは、データ・パリティー・ジェネレーター、パリティー・チェッカー、およびラッチを制御します。

16 ポート非同期通信アダプター

アダプターのファミリーは、共通の機能設計に基づきます。 しかし、個々のアダプター特性は、サポートされるデバイス・インターフェースによって 決まります。 ファミリーは、16 ポート EIA 422A 非同期通信アダプターと 16 ポート EIA 232 非同期通信アダプターの 2 つのアダプターから 構成されます。

注 : 以下のセクションは、Itanium ベースのプラットフォームには 該当しません。

16 ポート・アダプターのファミリーは、2 つのシリアル通信チャネルを備えた dual universal asynchronous receiver and transmitter (DUART) チップに基づきます。 DUART チップとその機能の詳細については、『16 ポート非同期通信アダプターのハードウェア 情報』を参照してください。

16 ポート非同期通信アダプター *- EIA 422A*

16 ポート非同期通信アダプター - EIA 232 は、システム装置への最大で 16 の EIA 232 非同期シリアル・デバイス (プリンターおよび端末) の接続をサポートします。

単一のシステム 装置では、最大で (ファミリー内の任意の組み合わせの) 8 つのアダプターが 使用できます。

このアダプターは、完全にプログラマブルであり、非同期通信のみをサポートします。 これは、スタート・ビットとストップ・ビットを追加および除去します。 このアダプターは、シリアル・データに対する偶数パリティー、奇数パリティー、またはパリティーなしをサポートします。 プログラマブル・ボー・レート・ジェネレーターは、50 から 38400 bps の操作を許可します。 アダプターは、1、1.5、または 2 ストップ・

ビットを含む 5、6、7、または 8 ビット文字をサポートします。 優先割り込みシステムは、送信、受信、エラー、回線状況、およびデータ・セット割り込みを制御します。 EIA 422A 16 ポート・ケーブル・アセンブリーの中には、デバイス接続用の 16 のコネクターが用意されています。

EIA 422A 16 ポート・アダプターには、以下の特性があります。

- 標準 Micro Channel フォーム・ファクター・カード
- 38.4K bps/ポートまでのデータ速度
- 送信および受信の 16 バイト・バッファリング
- 単一 78 ピン出力コネクター (マルチポート・インターフェース・ケーブルは、このコネクターに接続します)
- サージ保護回路
- 1200m (4000ft) までのケーブルをサポート
- TxD および RxD インターフェース 信号をサポート
- 8 ビット/16 ビット Micro Channel 従属インターフェース

16 ポート非同期通信アダプターのインストール

16 ポート非同期通信アダプターは、サーバーの単一 の Micro Channel スロットに収まります。アダプターをインストールするには、以下の手順で使用します。

1. すべてのユーザーがシステムからログオフしていることを確認し、次のコマンドを実行します。

   ```
   shutdown -F
   ```

2. **shutdown** コマンドが完了したら、システムの電源スイッチを「オフ」の位置に切り替えます。
3. サーバー・ケースを開き、16 ポート非同期通信アダプターを 空いている Micro Channel スロットに挿入します。
4. 78 ピン D シェル・コネクターを 16 ポート・インターフェース・ケーブルから 16 ポート・アダプターに接続します。
5. システム装置にカバー・パネルを戻します。
6. システムの電源スイッチをオンの位置まで押します。 ブート・プロセス中にシステムが 16 ポート・アダプターを認識して構成します。

ブートの完了後、root ユーザー ID を使用してログインし、以下のコマンドを実行してアダプターの 可用性を確認してください。

```
lsdev -Cc adapter | pg
```

システムは、使用可能状態のアダプターのみを 使用することができます。

新しくインストールしたアダプターが 使用不可である場合は、以下を確認してください。

1. アダプターが Micro Channel スロットに正しく取り付けられている。
2. 必要なすべてのケーブルが接続され、所定の位置にぴったり収まっている。
3. コマンド **errpt -a | pg** を実行し、アダプターに関連した問題についてのシステム・エラー・レポートを調べます。
4. コマンド **cfgmgr -v | pg** を実行します。 このコマンドは、リブートなしでアダプターの 再構成を試行します。 ページ送りされる出力にエラーがないか調べます。
5. 実行中の **cfgmgr** が失敗する場合は、リブートが必要になります。

16 ポート非同期通信アダプターのハードウェア情報

システム・インターフェースは、3 ビットのアドレスと 8 ビットのデータ、およびチップへの制御回線を提供します。 システム・インターフェースからのデータは、外部デバイスに送信するために直列化されます。 シリアル・データには、バイト境界にパリティー・ビットが含まれることがあります。 反対に、外部デバイスからのデータは、システム・インターフェースに 送信するために非直列化されます。 このデータにも、パリティー・ビット (これはオプションで検査できる) が含まれる可能性があります。 オプションで、チャネルは先入れ先出し法 (FIFO) モードでも作動できます。

FIFO モードでは、送信側と受信側の両方において最大 16 バイトをバッファーに入れることができます。 シリアル・インターフェースは、データの送信と受信の両方に start-stop プロトコルを使用します。 すなわち、各バイト (およびパリティー・ビット) がスタート・ビットとストップ・ビットでフレーム化され、それにより、個々の文字 (バイト) を基本とした同期が可能になります。

DUART チップは、12.288MHz 発振器を使用して、送信および受信論理を駆動するためのその内部時間を生成します。 このチャネルは全二重操作をサポートします。 各 16 ポート・アダプターには、8 つの DUART チップが実装されています。

13 のシステム・アクセス可能レジスターが用意されています。 各チャネルのプログラマブル・フィーチャーには、以下が含まれます。

- 文字長: 5、6、7、または 8 ビット
- パリティー生成/検出: 奇数、偶数、または「なし」
- ストップ・ビット数: 1、1.5、または 2
- 割り込みの使用可能/使用不可。 有効受信データ
- 送信側保持レジスターの空き
- 回線状況
- オーバーラン・エラー
- パリティー・エラー
- フレーム・エラー
- 中断

次の表は、アダプターのポート (デバイス・インターフェース) の特性の要約です。

表 *121. 16* ポート非同期通信アダプターのポート特性

パラメーター	**EIA 232**	**EIA 422A**
トポロジー	Point to Point	Point to Point
最大データ速度 (標準)	20Kbps	2Mbps
最大データ速度 (ボード)	38.4Kbps	38.4Kbps
送信メディア	マルチコンダクター	マルチコンダクター
ケーブル・ワイヤー数	5 (信号用接地を含む)	5 (信号用接地を含む)
最大ケーブル長	61m (200ft)	1200 m < 90Kbps
デバイス・コネクター	25 ピン D	25 ピン D
電気インターフェース	不平衡	平衡
ビット・エンコード	デジタル 2 レベル	デジタル 2 レベル

16 ポート非同期通信アダプター・ボード優先順位

割り込み調停論理は、特定の図式に従ってアダプターの 優先順位を設定します。

```
Adapter   Priority
    0     Highest
    1       |
    2       |
    3       |
    4       |
    5       |
    6       |
    7       |
    8       |
    9       |
    10      |
    11      |
    12      |
    13      |
```

```
    14      |
    15    Lowest
```

保留中の割り込みのある DUART チャネルは、一定の優先順位に従って処理されます。 最も高い優先順位はポート 0 に割り当てられます。 次に優先順位が高いのはポート 1 となります。 優先順位が最も低いのはポート 15 となります。

16 ポート非同期通信アダプター割り込み論理

16 ポート非同期通信アダプターの 割り込み論理は、割り込み生成論理と割り込み調停論理に分かれています。

これらの両方の論理セクションが、すべての 16 ポート・アダプターに実装されています。 割り込み生成論理は、システムにインターフェースを 提供します。 この論理は、システム割り込み要求を生成し、割り込み共有回路を包含します。

割り込み調停論理の機能は、優先順位が最も高い割り込みが保留になっている 16 ポート・アダプターを識別することです。 この論理は、優先順位の最も高いポートの割り込み情報を割り込み調停レジスターに入れます。 これは、1 回の読み取り操作で行われます。

割り込み調停論理は 16 ポート・アダプターに固有なものです。Micro Channel の調停論理と混同しないでください。

16 ポート割り込み生成論理

16 ポート非同期通信アダプターには、8 つのシステム割り込み要求回線が実装されます。

アダプターには、以下の 8 つのシステム割り込み要求 (IRQ) 回線が実装されています。

- IRQ 3
- IRQ 5
- IRQ 9
- IRQ 10
- IRQ 11
- IRQ 12
- IRQ 14
- IRQ 15

通常の操作時は、要求回線が 1 つだけ活動状態になります。 最適なシステム・パフォーマンスのためには、1 つのシステムの中のすべての 16 ポート・アダプターが同一の割り込みレベルを使用しているべきです。アクティブ回線は、セットアップ・サイクル中に適切な POS レジスターに書き込みを行うことによって選択します。 アダプターは、割り込み共有をサポートし、Micro Channel アーキテクチャーで定義されているオープン・コレクター構成を実装しています。この構造では、システム引き上げレジスターによって割り込み回線のレベルが引き上げられます。 アダプターは回線のレベルを引き下げて、アクティブ割り込み要求を示します。

16 ポート割り込み調停論理

割り込み調停論理は、複数の 8 ポート・アダプターまたは 16 ポート・アダプターが割り込みを生成した場合に、ソフトウェア・サービスの優先順位を決定します。

最大で 8 つの 8 ポート・アダプターまたは 16 ポート・アダプターがシステムに 共存して並行操作できます。 この論理は、1 回の読み取り操作でシステムにアダプターとポートの 識別と割り込みタイプを提供します。 割り込み要求を検出すると、システムは I/O アドレス 0130 にある 16 ビット割り込み調停レジスターを読み取ります。

16 ポート非同期通信アダプター *EIA 232* のインターフェース信号

16 ポート非同期通信アダプターの 各ポートには、以下のインターフェース 信号が実装されています。

シグナル	定義
TxD	送信データ
DCD	データ・キャリア検出

シグナル	定義
DTR	データ端末レディー
RxD	受信データ
Sig Gnd	信号用接地

16 ポート *EIA 232* 信号の電圧レベル

インターフェース・ポイントで測定した交換回路の電圧が信号用接地に関して -3 V dc より低い場合、信号はマーク状態になります。 信号用接地に関して電圧が +3 V dc より高い場合、信号はスペース状態になります。 +3 V dc と -3 V dc の間の領域は、遷移領域として定義されており、有効レベルではありません。 -15 V dc より低い電圧や +15 V dc より高い電圧も、有効レベルではありません。

データの送信中は、マーク状態がバイナリー状態 1 を示し、スペース状態がバイナリー状態 0 を示します。

インターフェース 制御回路の場合、機能は信号用接地に関して電圧が +3 V dc より高いとオンになり、電圧が -3 V dc より低いとオフになります。 EIA 232 の信号レベルについては、以下の表を参照してください。

表 *122. EIA 232* の信号レベル

交換電圧	バイナリーの状態	信号の状態	インターフェース 制御機能
+ 電圧	0	スペース	オン
- 電圧	1	マーク	オフ

16 ポート非同期 EIA 232 アダプター・ポートの電気特性は、1969 年 10 月付けの EIA 標準 232C、および 1987 年 1 月付けの EIA 標準 232D の規格と合致しています。

このアダプター・ポートは、1969 年 10 月付けの EIA 標準 232C、および 1987 年 1 月付けの EIA 標準 232D で記述されている非同期操作の機能要件 (start-stop プロトコル) を満たしています。

16 ポート非同期通信アダプター *EIA 422A* のインターフェース信号

16 ポート非同期通信アダプターの 各ポートには、以下の EIA 422A インターフェース 信号が実装されています。

シグナル	定義
TxA	送信データ
TxB	送信データ
RxA	受信データ
RxB	受信データ
Sig Gnd	信号用接地

16 ポート *EIA 422A* 信号の電圧レベル

回線ドライバーは、2 から 6 ボルト (ジェネレーター・インターフェース・ポイントで測定) の範囲の差動電圧を生成します。 受信側の差動電圧の絶対値は、200mV (ミリボルト) から 6 ボルト (ロード・インターフェース・ポイントで測定) の範囲でなければなりません。

測定値は、端末 B (負のリード線) に関して端末 A (正のリード線) で測定されます。 以下の表は、電圧レベルに関連した信号の状態を示しています。

表 *123. 16* ポート *EIA 422A* 信号の状態

交換電圧	バイナリーの状態	信号の状態
+ 電圧	0	スペース
- 電圧	1	マーク

16 ポート非同期 EIA 422A アダプターは、長さが 1200m (4000ft) までの室内ケーブルをサポートします。 このような長さのケーブルは、間接的な雷撃などの誘発電圧による突然の電圧サージの影響を受けやすくなります。 EIA 422A アダプターには、これらの電圧サージからこれを保護するために、副次的なサージ保護回路が実装されています。 このサージ保護回路は、アダプターのインターフェース ・データ回線に実装されています。

受信装置がドライバーに接続されていないとき (オープン・ケーブル) の障害状態を防ぐため、各 EIA 422A 受信装置の入力リード線に、フェイルセーフ 回路が追加されています。 フェイルセーフ回路は、受信装置がドライバーに接続されていないときは常に受信装置をマーク状態 (バイナリー 1) に設定します。

16 ポート非同期 EIA 422A アダプター・ポートの電気特性は、1978 年 12 月付けの EIA 標準 422A に準拠しています。

ASCII、10 進数、16 進数、8 進数、およびバイナリーの変換表

この表では、ASCII、10 進数、16 進数、8 進数、およびバイナリー値を変換する際に役立つ情報を参照できます。

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換

ASCII	10 進数	16 進数	8 進数	バイナリー
null	0	0	0	0
ヘッダー開始	1	1	1	1
テキスト開始	2	2	2	10
テキスト終結	3	3	3	11
伝送終了	4	4	4	100
ENQ (問い合わせ)	5	5	5	101
ACK (応答)	6	6	6	110
ベル	7	7	7	111
バックスペース	8	8	10	1000
水平タブ	9	9	11	1001
改行	10	A	12	1010
垂直タブ	11	B	13	1011
用紙送り	12	C	14	1100
復帰	13	D	15	1101
シフトアウト	14	E	16	1110
シフトイン	15	F	17	1111
伝送制御拡張	16	10	20	10000
装置制御 1/Xon	17	11	21	10001
装置制御 2	18	12	22	10010
装置制御 3/Xoff	19	13	23	10011
装置制御 4	20	14	24	10100
NAK (否定応答)	21	15	25	10101
同期信号	22	16	26	10110
伝送ブロック終結	23	17	27	10111
キャンセル	24	18	30	11000

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換 *(*続き*)*				
ASCII	**10 進数**	**16 進数**	**8 進数**	**バイナリー**
メディア終端	25	19	31	11001
ファイル終わり/置換	26	1A	32	11010
escape	27	1B	33	11011
ファイル分離	28	1C	34	11100
グループ分離	29	1D	35	11101
レコード分離	30	1E	36	11110
ユニット分離	31	1F	37	11111
スペース	32	20	40	100000
!	33	21	41	100001
"	34	22	42	100010
#	35	23	43	100011
$	36	24	44	100100
%	37	25	45	100101
&	38	26	46	100110
'	39	27	47	100111
(	40	28	50	101000
)	41	29	51	101001
*	42	2A	52	101010
+	43	2B	53	101011
,	44	2C	54	101100
-	45	2D	55	101101
.	46	2E	56	101110
/	47	2F	57	101111
0	48	30	60	110000
1	49	31	61	110001
2	50	32	62	110010
3	51	33	63	110011
4	52	34	64	110100
5	53	35	65	110101
6	54	36	66	110110
7	55	37	67	110111
8	56	38	70	111000
9	57	39	71	111001
:	58	3A	72	111010

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換 *(*続き*)*

ASCII	10 進数	16 進数	8 進数	バイナリー
;	59	3B	73	111011
<	60	3C	74	111100
=	61	3D	75	111101
>	62	3E	76	111110
?	63	3F	77	111111
@	64	40	100	1000000
A	65	41	101	1000001
B	66	42	102	1000010
C	67	43	103	1000011
D	68	44	104	1000100
E	69	45	105	1000101
F	70	46	106	1000110
G	71	47	107	1000111
H	72	48	110	1001000
I	73	49	111	1001001
J	74	4A	112	1001010
K	75	4B	113	1001011
L	76	4C	114	1001100
M	77	4D	115	1001101
N	78	4E	116	1001110
O	79	4F	117	1001111
P	80	50	120	1010000
Q	81	51	121	1010001
R	82	52	122	1010010
S	83	53	123	1010011
T	84	54	124	1010100
U	85	55	125	1010101
V	86	56	126	1010110
W	87	57	127	1010111
X	88	58	130	1011000
Y	89	59	131	1011001
Z	90	5A	132	1011010
[	91	5B	133	1011011
¥	92	5C	134	1011100

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換 *(*続き*)*

ASCII	10 進数	16 進数	8 進数	バイナリー
]	93	5D	135	1011101
^	94	5E	136	1011110
_	95	5F	137	1011111
`	96	60	140	1100000
a	97	61	141	1100001
b	98	62	142	1100010
c	99	63	143	1100011
d	100	64	144	1100100
e	101	65	145	1100101
f	102	66	146	1100110
g	103	67	147	1100111
h	104	68	150	1101000
i	105	69	151	1101001
j	106	6A	152	1101010
k	107	6B	153	1101011
l	108	6C	154	1101100
m	109	6D	155	1101101
n	110	6E	156	1101110
o	111	6F	157	1101111
p	112	70	160	1110000
q	113	71	161	1110001
r	114	72	162	1110010
s	115	73	163	1110011
t	116	74	164	1110100
u	117	75	165	1110101
v	118	76	166	1110110
w	119	77	167	1110111
x	120	78	170	1111000
y	121	79	171	1111001
z	122	7A	172	1111010
{	123	7B	173	1111011
|	124	7C	174	1111100
}	125	7D	175	1111101
~	126	7E	176	1111110

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換 *(*続き*)*

ASCII	10 進数	16 進数	8 進数	バイナリー
DEL	127	7F	177	1111111
	128	80	200	10000000
	129	81	201	10000001
	130	82	202	10000010
	131	83	203	10000011
	132	84	204	10000100
	133	85	205	10000101
	134	86	206	10000110
	135	87	207	10000111
	136	88	210	10001000
	137	89	211	10001001
	138	8A	212	10001010
	139	8B	213	10001011
	140	8C	214	10001100
	141	8D	215	10001101
	142	8E	216	10001110
	143	8F	217	10001111
	144	90	220	10010000
	145	91	221	10010001
	146	92	222	10010010
	147	93	223	10010011
	148	94	224	10010100
	149	95	225	10010101
	150	96	226	10010110
	151	97	227	10010111
	152	98	230	10011000
	153	99	231	10011001
	154	9A	232	10011010
	155	9B	233	10011011
	156	9C	234	10011100
	157	9D	235	10011101
	158	9E	236	10011110
	159	9F	237	10011111
	160	A0	240	10100000

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換 *(*続き*)*

ASCII	10 進数	16 進数	8 進数	バイナリー
	161	A1	241	10100001
	162	A2	242	10100010
	163	A3	243	10100011
	164	A4	244	10100100
	165	A5	245	10100101
	166	A6	246	10100110
	167	A7	247	10100111
	168	A8	250	10101000
	169	A9	251	10101001
	170	AA	252	10101010
	171	AB	253	10101011
	172	AC	254	10101100
	173	AD	255	10101101
	174	AE	256	10101110
	175	AF	257	10101111
	176	B0	260	10110000
	177	B1	261	10110001
	178	B2	262	10110010
	179	B3	263	10110011
	180	B4	264	10110100
	181	B5	265	10110101
	182	B6	266	10110110
	183	B7	267	10110111
	184	B8	270	10111000
	185	B9	271	10111001
	186	BA	272	10111010
	187	BB	273	10111011
	188	BC	274	10111100
	189	BD	275	10111101
	190	BE	276	10111110
	191	BF	277	10111111
	192	C0	300	11000000
	193	C1	301	11000001
	194	C2	302	11000010

表 *124. ASCII*、*10* 進数、*16* 進数、*8* 進数、およびバイナリー値間の変換 *(*続き*)*				
ASCII	**10 進数**	**16 進数**	**8 進数**	**バイナリー**
	195	C3	303	11000011
	196	C4	304	11000100
	197	C5	305	11000101
	198	C6	306	11000110
	199	C7	307	11000111
	200	C8	310	11001000
	201	C9	311	11001001
	202	CA	312	11001010
	203	CB	313	11001011
	204	CC	314	11001100
	205	CD	315	11001101
	206	CE	316	11001110
	207	CF	317	11001111
	208	D0	320	11010000
	209	D1	321	11010001
	210	D2	322	11010010
	211	D3	323	11010011
	212	D4	324	11010100
	213	D5	325	11010101
	214	D6	326	11010110
	215	D7	327	11010111
	216	D8	330	11011000
	217	D9	331	11011001
	218	DA	332	11011010
	219	DB	333	11011011
	220	DC	334	11011100
	221	DD	335	11011101
	222	DE	336	11011110
	223	DF	337	11011111
	224	E0	340	11100000
	225	E1	341	11100001
	226	E2	342	11100010
	227	E3	343	11100011
	228	E4	344	11100100

表 124. ASCII、10 進数、16 進数、8 進数、およびバイナリー値間の変換 (続き)

ASCII	10 進数	16 進数	8 進数	バイナリー
	229	E5	345	11100101
	230	E6	346	11100110
	231	E7	347	11100111
	232	E8	350	11101000
	233	E9	351	11101001
	234	EA	352	11101010
	235	EB	353	11101011
	236	EC	354	11101100
	237	ED	355	11101101
	238	EE	356	11101110
	239	EF	357	11101111
	240	F0	360	11110000
	241	F1	361	11110001
	242	F2	362	11110010
	243	F3	363	11110011
	244	F4	364	11110100
	245	F5	365	11110101
	246	F6	366	11110110
	247	F7	367	11110111
	248	F8	370	11111000
	249	F9	371	11111001
	250	FA	372	11111010
	251	FB	373	11111011
	252	FC	374	11111100
	253	FD	375	11111101
	254	FE	376	11111110
	255	FF	377	11111111

uDAPL (ユーザー・レベルの Direct Access Programming Library)

uDAPL (ユーザー・レベルの Direct Access Programming Library) は、InfiniBand や RNIC などの直接データ・アクセスをサポートするトランスポートで実行される直接アクセス・フレームワークです。

DAT Collaborative は、**uDAPL** API http://www.datcollaborative.org を指定します。

Open Fabrics からの uDAPL コードベースは AIX に移植され、現在、GX++ HCA および 4X DDR 拡張カード (CFFh) InfiniBand アダプターによりサポートされます。

uDAPL 1.2 バージョンは、AIX 6.1 (6100-06 適用) 以降でサポートされます。 **uDAPL** インストール・イメージは、*udapl.rte* として拡張パックで出荷されます。 このインストール・イメージは、**/usr/include/dat**

のもとにある DAT ヘッダー・ファイルを提供します。 このインストール・イメージは、*libdat.a* と *libdapl.a* の 2 つのライブラリーも提供します。

アプリケーションには DAT ヘッダー・ファイルが含まれ、DAT ライブラリー (**/usr/include/dat** にある *libdat.a*)とリンクします。 DAT レイヤーは、基礎となる該当のトランスポート固有のライブラリーを決定します。

AIX **uDAPL** Provider は、*dat.conf* エントリーを使用して DAT レジストリーに登録します。 */etc/dat.conf* ファイルはデフォルトのエントリーを入れて出荷され、このファイルにはエントリーのフォーマットに関する詳細が含まれています。

デバッグ目的で、**uDAPL** ライブラリーは AIX システム・トレースをサポートします。 **uDAPL** システム・トレース・フック ID には、5C3 (DAPL イベントの場合)、5C4 (DAPL エラー・イベントの場合)、5C7 (DAT イベントの場合)、および 5C8 (DAT エラー・イベントの場合) が含まれます。 初期トレース・レベルは、0 から 10 の範囲の数値を指定できる環境変数 *DAT_TRACE_LEVEL* および *DAPL_TRACE_LEVEL* を使用して変更できます。 イベントの数およびトレースされるデータの量は、このレベルにつれて増加します。主要なトレース・レベルは次のとおりです。

```
TRC_LVL_ERROR  = 1,
TRC_LVL_NORMAL = 3,
TRC_LVL_DETAIL = 7
```

その他の標準 AIX 保守容易性機能 (AIX エラー・ログなど) は、問題判別に役立てることができます。 さらに、基礎となるトランスポート層の保守容易性機能 (ibstat コマンドや InfiniBand コンポーネント・トレースなど) も、問題の診断に役立ちます。

DAT API は、*/usr/include/dat/dat_error.h* ファイルの助けによりデコードできる標準戻りコードを戻します。 戻りコードの詳細な説明は、DAT Collaborative からの **uDAPL** 仕様内にあります。

448 ページの『InfiniBand (IPoIB) を介したインターネット・プロトコル』

AIX でサポートされる uDAPL API

DAT Collaborative によって指定される多くの **uDAPL** API のうちで、AIX ではサポートされない幾つかの API があります。

一般的業界の **uDAPL** 実装環境ではサポートされず、かつ AIX でもサポートされない API を以下に示します。

項目	説明
dat_cr_handoff	// In DAT 1.2
dat_ep_create_with_srq	// In DAT 1.2
dat_ep_recv_query	// In DAT 1.2
dat_ep_set_watermark	// In DAT 1.2
dat_srq_create	// In DAT 1.2
dat_srq_post_recv	// In DAT 1.2
dat_srq_resize	// In DAT 1.2
dat_srq_set_lw	// In DAT 1.2
dat_srq_free	// In DAT 1.2
dat_srq_query	// In DAT 1.2

AIX がサポートしない追加の API は、次のとおりです。

- dat_lmr_sync_rdma_read

- dat_lmr_sync_rdma_write
- dat_registry_add_provider
- dat_registry_add_provider

サポートされないすべての API の場合、AIX は、そのサポートの欠如を示すために DAT 仕様に記されている特定のメカニズムに従います。 これには、属性値 (max_srq 等価ゼロなど) および特定の戻りコード (DAT_MODEL_NOT_SUPPORTED など) が含まれます。 業界の実装環境および DAT 仕様との整合性を取るために、DAT_NOT_IMPLEMENTED も、サポートされない機能に対して戻されることがあります。

RMR 関連の API (*dat_rmr_create*、*dat_rmr_bind*、*dat_rmr_free*、*dat_rmr_query* など) のサポートは、基礎となる HCA 機能に応じて決まり、その成否は基礎となる IB フレームワークによって決まります。 現在、GX++ HCA および 4X DDR 拡張カード (CFFh) InfiniBand アダプターは、これらの RMR 操作をサポートしません。

752 ページの『uDAPL (ユーザー・レベルの Direct Access Programming Library)』

754 ページの『uDAPL のベンダー固有の属性』

448 ページの『InfiniBand (IPoIB) を介したインターネット・プロトコル』

uDAPL のベンダー固有の属性

AIX でサポートされる幾つかのベンダー固有の属性があります。 属性名は、**delayed_ack_supported**、**vendor_extension**、**vendor_ext_version**、**debug_query**、および **debug_modify** です。

delayed_ack_supported

InfiniBand (IB) トランスポートの AIX プロバイダーには、**delayed_ack_supported** という名前のベンダー固有のインターフェース・アダプター (IA) 属性が含まれます。 この属性の値は、**true** または **false** のいずれかです。 **true** の場合、IA に関連付けられたエンドポイントは、**delayed_ack** という名前の変更可能なプロバイダー固有の属性を持ちます。 **delayed_ack_supported** 属性が **false** の場合は、エンドポイントの **delayed_ack** というプロバイダー固有の属性を変更することはできません。 エンドポイントの **delayed_ack** 属性のデフォルト値は **false** です。 この値を **true** に設定すると (dat_ep_modify を使用して)、エンドポイントに関連付けられた特定の IB キュー・ペアに対して、基礎となる IB ホスト・チャネル・アダプター (HCA) の delayed_ack 機能が使用可能になります。 このハードウェア機能はすべての HCA によってインプリメントされるわけではないため、すべての IA に使用できるわけではありません。 この機能を使用可能にすると、データ転送操作がサーバーのシステム・メモリー内で明確になるまで、HCA は確認応答の送信を遅らせます。 これは、待ち時間の微少の増加という潜在的な犠牲を払うことで、IB 仕様で規定されるものより少し強い意味合いがあります。

vendor_extension、vendor_ext_version、debug_query、および debug_modify

デバッグ目的で、**uDAPL** ライブラリーは AIX システム・トレースをサポートします。 初期トレース・レベルは、環境変数 *DAT_TRACE_LEVEL* および *DAPL_TRACE_LEVEL* を使用して変更できます。 API を使用してこれらのトレース・レベルを動的に変更するために、AIX 上で動的トレース・レベル・サポートが提供されます。 ライブラリーに動的トレース・レベル・サポートがあるかどうかを検査するために、アプリケーションは、ベンダー固有の IA 属性 **vendor_extension** の照会を行うことができます。 照会から戻る際、**vendor_extension** 属性の存在は動的トレース・レベル・サポートを示します。 この属性の値は **true** に設定されます。ただし、この設定に関係なく、この属性の存在は動的トレース・レベル・サポートを示します。 **vendor_extension** 属性が存在する場合、アプリケーションは、ベンダー固有の IA 属性 **debug_query** および **debug_modify** の照会を行うことにより、**dat_trclvl_query()** および **dat_trclvl_modify()** への関数ポインターを得ることができます。 これらの属性の値には、対応する機能へのポインターがあります。 この **vendor_extension** インターフェースを 将来用に拡張可能にするために、別のベンダー固有の IA® 属性 **vendor_ext_version** があります。現在、1 つのバージョンのみをサポートするため、この属性の値は **1.0** に設定されます。 **vendor_extension** 属性が存在しない場合、アプリケーションは、トレース・レベルを動的に変更することはできません。

これらの属性を取り扱う方法の例は、AIX 実装環境でインストールされた **uDAPL** サンプル・コードに組み込まれています。

752 ページの『uDAPL (ユーザー・レベルの Direct Access Programming Library)』

448 ページの『InfiniBand (IPoIB) を介したインターネット・プロトコル』

PCIe2 10 GbE RoCE アダプター・サポート

PCIe2 10GbE RDMA Over Converged Ethernet (RoCE) アダプターは、最初は AIX オペレーティング・システムで、リモート・ダイレクト・メモリー・アクセス (RDMA) 対応のデバイスとしてのみサポートされていました。 これをサポートするソフトウェアは、AIX InfiniBand スタックに基づいた IBM 専用ソフトウェアでした。 このサポートは、AIX RoCE と呼ばれていました。AIX 7 (7100-02 適用) 以降では、このアダプターを 2 つのモード、すなわち AIX RoCE と 10G イーサネット・サポート (ネットワーク・インターフェース・カード (AIX NIC) とも呼ばれている) でサポートします。 新しい AIX 7 (7100-03 適用) は、RDMA を NIC モードおよび OpenFabrics Enterprise Distribution (OFED) でサポートするようになりました。 どちらのモードが使用可能であるかは、以前のバージョンの AIX オペレーティング・システムで使用可能でなかったホスト・バス・アダプター (HBA) が管理します。

以下の表に、PCIe2 10GbE アダプター・ソフトウェアの進化状況を示します。

AIX レベル	**モード 1**	**モード 2**
AIX 7 (7100-02 適用) より前	AIX RoCE	NA
AIX 7 (7100-02 適用)	AIX RoCE	AIX NIC
AIX 7 (7100-03 適用)	AIX RoCE	AIX NIC + OFED RoCE

このアダプターの最新のデバイス・ドライバーをダウンロードするには、以下の手順を実行します。

1. IBM Web サイト (www.ibm.com) にアクセスします。
2. 「**サポート&ダウンロード**」をクリックします。
3. 最新のファームウェアを AIX ホスト・ロケーション (`/etc/microcode`) にダウンロードします。
4. 以下のいずれかの手順を選択して、ファームウェアを更新するために **diag** ツールを実行します。
   - 短いパスの手順
     a. 次のコマンドを入力します。
     ```
     *diag -d entX -T download
     ```
     注 : 以前のバージョンの RoCE スタックを使用している場合は、**entX** を **roceX** に置き換えます。
     b. `/etc/microcode` ディレクトリーに保存されているマイクロコードを選択します。
   - 長いパスの手順
     a. 次のコマンドを入力します。
     ```
     *diag
     ```
     b. 次の順にクリックします。
     「**タスク選択**」 > 「**マイクロコード・タスク (Microcode tasks)** > 「**マイクロコードのダウンロード**」
     c. 「**entX**」または「**roceX**」を選択します。
     d. `/etc/microcode` ディレクトリーに保存されているマイクロコードを選択します。

デフォルトでは、アダプターは AIX RoCE モードをサポートするように構成されます。モードを他のモードに変更するには、755 ページの『AIX NIC + OFED RDMA』セクションの手順を実行します。

AIX NIC + OFED RDMA

AIX 7 (7100-02 適用) 以降、PCIe2 10 GbE RoCE アダプターを AIX NIC 構成で稼働するように構成できます。 AIX 7 (7100-03 適用) 以降、OFED RDMA 機能も AIX NIC 構成に追加されています。 RDMA の利点を活用した、ネットワークを集中的に使用するアプリケーションがない場合は、このアダプターをネットワーク・インターフェース・カード (NIC) のみとして使用できます。

PCIe2 10 GbE RoCE アダプターを AIX NIC + OFED RoCE 構成または AIX RoCE 構成で使用するには、以下のファイルセットが必須です。これらは AIX 7 (7100-03 適用) 基本オペレーティング・システム CD にあります。

devices.ethernet.mlx
AIX NIC + OFED RoCE 構成をサポートするための Converged Ethernet Adapter のメイン・デバイス・ドライバー (mlxentdd)。

devices.pciex.b315506b3157265
NGP ITE Converge Ethernet Adapter ASIC2 のパッケージ化サポート。

devices.pciex.b3155067b3157365
NGP ITE Converge Ethernet Adapter ASIC1 のパッケージ化サポート。

devices.pciex.b315506714101604
スモール・フォーム・ファクター・プラグ可能 (SFP+) トランシーバー付き Mellanox 2 ポート 10 GbE Converge Ethernet Adapter のパッケージ化。

devices.pciex.b315506714106104
すべての SFP+ トランシーバーをサポートする Mellanox 2 ポート 10 GbE Converge Ethernet Adapter のパッケージ化。

devices.common.IBM.ib
AIX RoCE 構成を使用するために必要な ICM デバイス・ドライバー。

devices.pciex.b3154a63
AIX RoCE 構成を使用するために必要な、Mellanox 10 GbE Converge Ethernet Adapter のデバイス・ドライバー。

ofed.core
OFED RDMA が必要な場合にのみ必要となる OFED Core Runtime Environment ファイルセット。

既存の AIX RoCE ファイルセットを新しいファイルセットで更新した後、roce と ent の両方のデバイスが構成対象として表示される場合があります。 アダプターに対して **lsdev** コマンドを実行したときに、両方のデバイスが構成対象として表示された場合は、以下の手順を実行します。

1. 次のコマンドを入力して、PCIe2 10 GbE RoCE アダプターに関連する *roceX* インスタンスを削除します。

```
# rmdev -dl roce0[, roce1][, roce2,...]
```

2. 次のコマンドを入力して、PCIe2 10 GbE RoCE アダプターに関連する *entX* インスタンスを削除します。

```
# rmdev -dl ent1[,ent2][, ent3...]
```

3. PCIe2 10 GbE RoCE アダプターに関連する 1 つ以上のコンバージド・ホスト・バス・アダプター (hbaX) が存在する場合は、次のコマンドを入力して、それらを削除します。

```
# rmdev -dl hba0[, hba1][,hba2...]
```

4. 次のコマンドを入力して構成マネージャーを実行し、変更を取り込みます。

```
# cfgmgr
```

以下の手順を実行して、AIX RoCE 構成から AIX NIC + OFED RoCE 構成に切り替えます。

1. PCIe2 10 GbE RoCE アダプター上で稼働しているすべての RDMA アプリケーションを停止します。
2. 以下のいずれかのコマンドを入力して、*roceX* インスタンスを削除するか再定義します。
   - `# rmdev -d -l roce0`
   - `# rmdev -l roce0`

   `rmdev -l roce0` コマンドは `roce0` 構成の定義を保存するので、次回にその定義を使用して、インスタンスを作成することができます。
3. 次のコマンドを入力して、hba stack_type 設定の属性を `aix_ib` (AIX RoCE) から `ofed` (AIX NIC + OFED RoCE) に変更します。

```
# chdev -l hba0 -a stack_type=ofed
```

4. 次のコマンドを入力して構成マネージャー・ツールを実行し、ホスト・バス・アダプターが PCIe2 10 GbE RoCE アダプターを NIC アダプターとして 構成できるようにします。

```
# cfgmgr
```

5. 次のコマンドを入力して、現時点でアダプターが NIC 構成で稼働していることを確認します。

```
# lsdev -C -c adapter
```

次の例は、AIX NIC + OFED RoCE モードで構成されたアダプターに対して **lsdev** コマンドを実行したときの結果を示しています。

```
ent1 Available 00-00-01 PCIe2 10GbE RoCE Converged Network Adapter
ent2 Avaliable 00-00-02 PCIe2 10GbE RoCE Converged Network Adapter
hba0 Available 00-00 PCIe2 10GbE RoCE Converged Host Bus Adapter (b315506714101604)
```

図 *46. AIX NIC + OFED RoCE* 構成のアダプターに対する ***lsdev*** コマンドの出力の例

AIX 7 (7100-03 適用) 以降、AIX は OFED RDMA を AIX NIC モードでもサポートしているため、OFED RDMA を使用可能にする必要がある場合には、以下の 2 つの手順を追加で実行する必要があります。

1. パッケージ ofed.core をインストールします。
2. 以下のコマンドを実行して ent1、ent2 デバイスで RDMA モードを設定します。

```
# chdev -l ent1 -a rdma=desired
# chdev -l ent2 -a rdma=desired
```

RDMA モードは、インターフェース en1 または en2 が構成される前に設定されます。

3. 次のコマンドを実行すると、RDMA モードを使用不可にすることができます。

```
# chdev -l ent1 -a rdma=disabled
# chdev -l ent2 -a rdma=disabled
```

AIX RoCE

PCIe2 10 GbE RoCE アダプターは、AIX RoCE モードで作動するように事前構成されています。 RDMA を使用するネットワークは、ネットワークを集中的に使用するアプリケーションに対して NIC として使用されるアダプターよりも 高いパフォーマンスを提供します。 このモードは、多くの場合、ネットワーク・ストレージやハイパフォーマンス・コンピューティングに役立ちます。

AIX RoCE 構成では、以下のようなライブラリーまたはインターフェースを 使用する必要があります。

- Direct Access Programming Library (uDAPL)。これは、DB2® データベース・システムによって使用されます。
- Message Passing Interface (MPI)。これは、ハイパフォーマンス・コンピューティング (HPC) に使用されます。

roce_rdmaconfiguration.dita#roce_rdmaconfiguration/figscreenrdma は、アダプターが AIX RoCE モードで動作している場合の出力を示しています。

PCIe2 10 GbE RoCE アダプターは、AIX RoCE モードでは 1 つのアダプター・インスタンスだけを表示しますが、最大 2 つのポートを持つことができます。 構成されているポートの数を判別するには、以下の手順を使用して、**ibstat** コマンドを実行します。

1. 次のコマンドを入力して、icm カーネル・エクステンションが構成されているかどうかを判別します。

```
# lsdev -C | grep icm
```

2. icm カーネルが構成されていない場合は、次のコマンドを入力して構成します。

```
# mkdev -c management -s infiniband -t icm
```

3. 次のコマンドを入力して、**ibstat** コマンドを実行します。

```
# ibstat roce0
```

PCIe2 10 GbE RoCE アダプターは、AIX RoCE モードを使用するように初期構成されていますが、AIX NIC + OFED RoCE 構成からの切り替えが必要な場合もあります。 AIX NIC + OFED RoCE 構成から AIX RoCE 構成に切り替えるには、以下の手順を実行します。

1. 次のコマンドを入力して、アダプターが AIX NIC + OFED RoCE モードであることを確認します。

```
# lsdev -C -c adapter
```

lsdev コマンドの出力は、roce_nicconfiguration.dita#roce_nicconfiguration/figscreennic に示した例のようになります。

2. 次のコマンドを入力して、TCP/IP トラフィックを停止し、IP インターフェースを 切り離します。

```
# ifconfig en1 down detach; ifconfig en2 down detach
```

3. 以下のいずれかのコマンドを入力して、NIC インスタンスを削除するか定義状態に入れておきます。
   - # `rmdev -d -l ent1; rmdev -d -l ent2`
   - # `rmdev -l ent1; rmdev -l ent2`

   `rmdev -l ent1; rmdev -l ent2` コマンドはイーサネット・デバイスの定義を保存するので、次回にインスタンスを作成するときに、その定義を使用できます。
4. 次のコマンドを入力して、hba stack_type の属性を `ofed` (AIX NIC + OFED RoCE) から `aix_ib` (AIX RoCE) に変更します。

```
# chdev -l hba0 -a stack_type=aix_ib
```

5. 次のコマンドを入力して構成マネージャー・ツールを実行し、ホスト・バス・アダプターが PCIe2 10 GbE RoCE アダプターを AIX RoCE アダプターとして構成できるようにします。

```
# cfgmgr
```

6. 次のコマンドを入力して、現時点でアダプターが AIX RoCE 構成で稼働していることを確認します。

```
# lsdev -C -c adapter
```

次の例は、アダプターに対して **lsdev** コマンドを実行し、そのアダプターが AIX RoCE モードで構成されているときの結果を示しています。

```
roce0 Avaliable 00-00-00 PCIe2 10GbE RoCE Converged Network Adapter
hba0 Available  00-00-00 PCIe2 10GbE RoCE Converged Host Bus Adapter (b315506714101604)
```

図 *47. AIX RoCE* 構成を使用しているアダプターに対する ***lsdev*** コマンドの出力の例

PCIe3 40 GbE RoCE アダプター・サポート

PCIe3 40 GbE RDMA Over Converged Ethernet (RoCE) アダプターは、OpenFabrics Enterprise Distribution (OFED) を使用したリモート・ダイレクト・メモリー・アクセス (RDMA) を通常の NIC モードでサポートします。 RDMA は、OpenFabrics ソフトウェアがインストールされている場合、デフォルトでサポートされ使用可能になります。

このアダプターの 最新のデバイス・ドライバーをダウンロードするには、以下の手順を実行します。

1. IBM Web サイト (www.ibm.com) にアクセスします。
2. 「**サポート&ダウンロード**」をクリックします。
3. 最新のファームウェアを AIX ホスト・ロケーション (`/etc/microcode`) にダウンロードします。
4. 以下のいずれかの手順を選択して、ファームウェアを更新するために **diag** ツールを実行します。

- 短いパスの手順
  a. 次のコマンドを入力します。

     ```
     *diag -d entX -T download
     ```

     注 : イーサネット・デバイスが同じホスト・バス・アダプター (例えば、hba0、hba1 など) に属している場合は、ファームウェアをいずれかの **ent** デバイスにダウンロードします。
  b. /etc/microcode ディレクトリーに保存されているマイクロコードを選択します。
- 長いパスの手順
  a. 次のコマンドを入力します。

     ```
     *diag
     ```

  b.「**タスク選択 (Task selection)**」 >「**マイクロコード・タスク (Microcode tasks)**」 >「**マイクロコードのダウンロード**」
     をクリックします。
  c.「**entX**」を選択します。
  d. /etc/microcode ディレクトリーに保存されているマイクロコードを選択します。

PCIe3 40 GbE RoCE アダプターと AIX NIC + OFED RoCE を使用するには、以下のファイルセットが必要です。 これらのファイルセットは、AIX 7 (7100-03 適用) 基本オペレーティング・システム CD にあります。

devices.ethernet.mlx	AIX NIC + OFED RoCE 構成をサポートするための Converged Ethernet Adapter のメイン・デバイス・ドライバー (mlxentdd)。
devices.pciex.b31503101410b504	Quad Small Form-factor Pluggable (QSFP) の受動銅線ポートを使用する Mellanox 2 Ports 40 Gb Converged Ethernet Adapter のパッケージ化。
ofed.core	OFED RDMA 機能が必要な場合にのみ必要となる OFED Core Runtime Environment ファイルセット。

RDMA 機能を使用不可にするには、次のコマンドを入力します。

```
chdev  -l  <Ethernet_device> rdma=disabled
```

例:

```
     # chdev -l ent1 -a rdma=disabled
     # chdev -l ent2 -a rdma=disabled
```

RDMA 機能を使用可能にするには、次のコマンドを入力します。

```
chdev -l <Ethernet_device> rdma=desired
```

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 *103-8510*
東京都中央区日本橋箱崎町 *19* 番 *21* 号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。 国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。 実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述は、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。 記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (年).

このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも 含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。 ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie などの各種テクノロジーの使用について詳しくは、『IBM オンラインでのプライバシー・ステートメントのハイライト』(http://www.ibm.com/privacy/jp/ja/)、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』というタイトルのセクション、および『IBM Software Products and Software-

as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

INFINIBAND、InfiniBand Trade Association、および INFINIBAND デザイン・マークは、INFINIBAND Trade Association の商標またはサービス・マークです。

Intel、Intel (ロゴ)、Intel Inside、Intel Inside (ロゴ)、Intel Centrino、Intel Centrino (ロゴ)、Celeron、Intel Xeon、Intel SpeedStep、Itanium、および Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

登録商標 Linux® は、独占的ライセンシーである Linux Foundation (世界規模でのマークの所有者) からのサブライセンスに従って使用されます。

Microsoft、Windows、Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

P

Q

R

S

T

U

V

W

X

Z

[特殊文字]