Solaris ネーミングの管理

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650–960–1300

Part Number 806-2721–10
2000 年 3 月

原典: *Solaris Naming Administration Guide*

Part No: 806-1387-10

Revision A

Adobe PostScript

目次

はじめに

このマニュアルでは、4 つのネームサービス、すなわち NIS+、NIS、FNS、DNS の構成を一度設定してからカスタマイズする場合の手順を解説します。このマニュアルは Solaris™ 8 システムとネットワークの管理マニュアルセットの一部です。

対象読者

このマニュアルは主に、経験のあるシステム管理者、ネットワーク管理者を対象としています。

このマニュアルは Solaris ネームサービスの基本について説明するものであり、ネットワークの基礎や Solaris 環境で提供される管理ツールについては触れていません。したがってネットワークを管理するためには、ネットワークの動作やツールの選択についての知識が必要となります。

(Solaris の 4 つのネームサービスの設定と構成については、『*Solaris* ネーミングの設定と構成』を参照してください。)

このマニュアルの構成

このマニュアルは次の 7 つのパートに分かれています。

パート I – ネーミングの紹介

名前空間の概念と、Solaris ネームサービスについて説明します。また、nsswitch.conf ファイルを使ってネームサービスを活用する手順を紹介しています。

- 第 1 章 :「名前空間」および「ネームサービス」とは何か、それらが何をするのかを説明し、続いて、4 つの Solaris ネームサービス、すなわち、DNS、NIS、FNS、NIS+ の概要を説明します。
- 第 2 章 : ネームサービススイッチはいくつかのネームサービスを使い分けます。この章ではネームサービススイッチとは何か、それを使用してクライアントがどの情報ソースを使用するのかについて説明します。

パート II — NIS+ の紹介と概要

NIS+ について説明します。

- 第 3 章 : NIS+ (Network Information Service Plus) の概要を説明します。
- 第 4 章 : NIS+ の名前空間の構造、サーバーとクライアントについて説明します。
- 第 5 章 : NIS+ テーブルの構造と、テーブルの設定の概要について説明します。
- 第 6 章 : NIS+ のセキュリティシステムの概要と、それが NIS+ 名前空間全体におよぼす影響について説明します。

パート III —NIS+ の管理

NIS+ 名前空間の管理方法について説明します。

- 第 7 章 : NIS+ 資格とその管理方法について説明します。
- 第 8 章 : NIS+ キーとその管理方法について説明します。
- 第 9 章 : 拡張セキュリティ資格の管理について説明します。
- 第 10 章 : NIS+ のアクセス権とその管理方法について説明します。
- 第 11 章 : 一般のユーザー (NIS+ 主体) の立場から見た passwd コマンドの使用方法について、また、NIS+ 管理者によるパスワードシステムの管理方法について説明します。
- 第 12 章 : NIS+ グループとその管理方法について説明します。
- 第 13 章 : NIS+ ディレクトリオブジェクトとその管理方法について説明します。

- 第 14 章 : NIS+ テーブルとその管理方法について説明します (デフォルトの NIS+ テーブルの詳細は、付録 C を参照)。
- 第 15 章 : NIS+ クライアントがサーバーを選択する動作をカスタマイズする方法を説明します。
- 第 16 章 : NIS+ 名前空間をバックアップしたり、復元したりする方法を説明します。
- 第 17 章 : NIS+ ディレクトリ管理コマンドを使用してクライアントマシン、サーバーから NIS+ を削除する方法、NIS+ 名前空間全体を削除する方法について説明します。

パート IV — NIS の管理

NIS (Network Information Service) とその管理方法について解説します。

- 第 18 章 : NIS について解説します。
- 第 19 章 : NIS の管理方法について解説します。

パート V — FNS の管理

フェデレーテッド・ネーミング・サービス (FNS) とその管理方法について説明します。

- 第 20 章 : 経験のある管理者を対象としています。FNS の概要、設定と構成の手順について説明し、サンプルプログラムを紹介します。
- 第 21 章 : X/Open XFN フェデレーテッド・ネーミング標準を Sun で実装したフェデレーテッド・ネーミング・サービス (FNS) について説明します。
- 第 22 章 : FNS ポリシーについて説明します。
- 第 23 章 : FNS がエンタープライズレベルのネームサービスとどのような関係にあるかを説明します。
- 第 24 章 : 既存のエンタープライズレベルのコンテキストを個々に作成、管理する方法について説明します。
- 第 25 章 : アプリケーション固有のコンテキストの管理方法について説明します。
- 第 26 章 : 2 つのグローバルネーミングシステム (DNS と X.500/LDAP) と、それを FNS でフェデレートする方法について説明します。
- 第 27 章 : FNS 属性とその管理方法について説明します。

パート VI — DNS の管理

ドメイン名システムとその管理方法について説明します。

- 第 28 章 : ドメイン名システムについて説明します。
- 第 29 章 : ドメイン名システムの管理方法について説明します。

パート VII — 付録

参考となる資料や用語解説を提供します。

- 付録 A : NIS+ の管理上発生する可能性のある様々な問題と、その解決方法について説明します。
- 付録 B : 一般的なエラーメッセージについて、アルファベット順に説明します。
- 付録 C : NIS+ 標準テーブルの内容についてまとめます (NIS+ テーブルと、管理に使用するコマンドの概要は、第 14 章を参照)。
- 付録 D : DNS テキスト (TXT) レコードの使用と、XFN リファレンスでの X.500 属性の使用についての補足資料です。
- 用語集 : NIS+ に関連する用語について説明します。

関連マニュアル

- 『*Solaris* ネーミングの設定と構成』 - NIS+、DNS を設定、構成する方法について説明しています。
- 『*NIS+* への移行』 - NIS から NIS+ に移行する方法について説明しています。

Solaris 8 マニュアルセット以外のマニュアル

- 『*DNS and Bind*』 - Cricket Liu & Paul Albitz 共著、浅羽登志也／上水流由香 監訳、アスキー出版社、1995年
- 『*NFS and NIS*』 - Hal Stern 著、倉骨 彰 訳、砂原秀樹監訳、アスキー出版局、1992年

マニュアルの注文方法

SunDocs™ プログラムでは、米国 Sun Microsystems™, Inc. (以降、Sun™ とします) の 250 冊以上のマニュアルを扱っています。このプログラムを利用して、マニュアルのセットまたは個々のマニュアルをご注文いただけます。

マニュアルのリストと注文方法については、米国 SunExpress™, Inc. のインターネットホームページ `http://www.sun.com/sunexpress` にあるカタログセクションを参照してください。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、またはコード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力とは区別して示します。	`system%` **`su`** `password:`
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。

表 P–1 表記上の規則 続く

字体または記号	意味	例
「 」	参照する章、節、ボタンやメニュー名、または強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を越える場合、バックスラッシュは継続を示します。	sun% **grep '^#define \ XV_VERSION_STRING'**

ただし AnswerBook2™ では、ユーザーが入力する文字と画面上のコンピュータ出力は区別して表示されません。

コード例は次のように表示されます。

■ C シェルプロンプト

```
system% command y|n [filename]
```

■ Bourne シェルおよび Korn シェルのプロンプト

```
system$ command y|n [filename]
```

■ スーパーユーザーのプロンプト

```
system# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の場合、*filename* は省略してもよいことを示します。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、「IA」という用語は、Intel 32 ビットのプロセッサアーキテクチャを意味します。これには、Pentium、Pentium Pro、Pentium II、Pentium II Xeon、Celeron、Pentium III、Pentium III Xeon の各プロセッサ、および AMD、Cyrix が提供する互換マイクロプロセッサチップが含まれます。

パート I　Solaris ネーミングの紹介

パート I では、名前空間と Solaris ネームサービスの紹介と概要、nsswitch.conf ファイルを使用してネームサービスの使用法を調整する方法について説明します。

- 第 1 章
- 第 2 章

第 1 章

ネームサービスについて

この章では、「名前空間」と「ネームサービス」の概要と機能について説明します。ネームサービスは、「ネットワーク情報サービス」、「ディレクトリサービス」と呼ぶこともあります。後半では、DNS、NIS、FNS、NIS+ という 4 つの Solaris ネームサービスについて簡単に説明します。

NIS+、NIS、DNS、FNS の名前空間の設定については、『*Solaris* ネーミングの設定と構成』で説明します。用語や略語の定義については 用語集を参照してください。

ネームサービスとは

ネームサービスは、ユーザー、ワークステーション、アプリケーションが、ネットワークを通じてやりとりする必要がある情報を 1 つの場所に格納します。情報にはたとえば、以下のものがあります。

- マシン (ホスト) 名とアドレス
- ユーザー名

- パスワード
- アクセス権

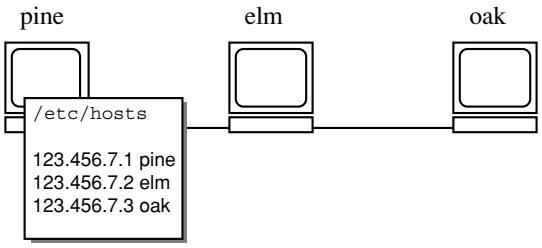
集中化されたネームサービスがなければ、ワークステーションごとに、これらの情報のコピーを管理しなければなりません。ネームサービス情報はファイルまたはマップ、データベーステーブルの形で格納できます。これらのデータを 1 カ所で管理すれば、大規模なネットワークの管理が簡単になります。

ネームサービスは、どのようなコンピュータネットワークにも欠かせないものです。そのほかにもネームサービスには、以下の機能があります。

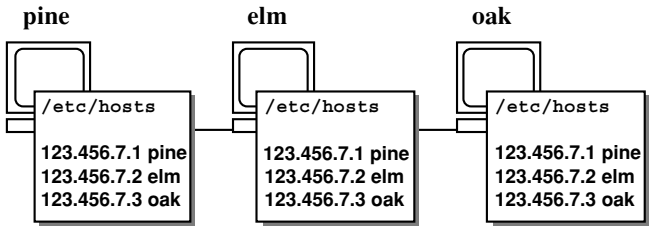
- 名前とオブジェクトを対応づける (結合する)
- オブジェクトの名前を解決する
- 結合を解除する
- 名前を一覧表示する
- 名前を変更する

ネットワーク情報サービスを使用すると、数値アドレスの代わりに一般的な名前でワークステーションを識別できます。そのため、ユーザーは「129.44.3.1」のような難しい数値アドレスを覚えて入力する必要がなくなり、情報のやりとりが簡単になります。

たとえば、pine、elm、oak という 3 台のワークステーションからなる簡単なネットワークを例にとります。pine が elm または oak にメッセージを送信するには、それら 2 台のネットワークアドレスを知る必要があります。そのため pine は、自分自身を含めたネットワーク内のすべてのワークステーションのネットワークアドレスを格納する /etc/hosts ファイルまたは /etc/inet/ipnodes ファイルを持っています。



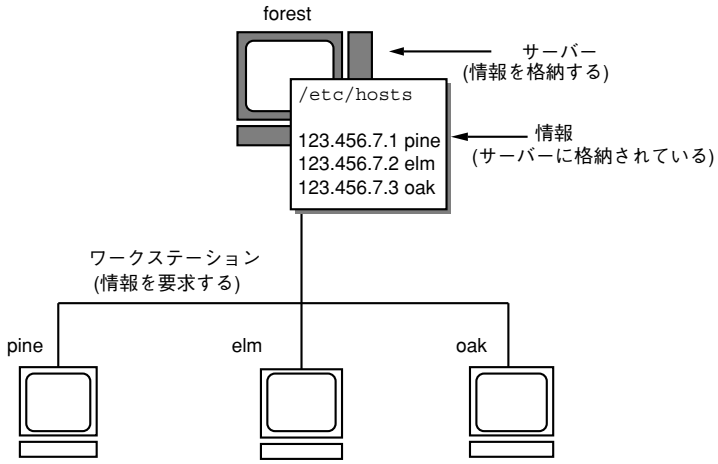
同様に、elm や oak が pine と通信したり、お互いに通信するためには、同じ /etc/hosts ファイルを持つ必要があります。



しかし、ワークステーションが格納しなければならないネットワーク情報は、アドレスだけではありません。セキュリティ情報、メールデータ、Ethernet インタフェースについての情報、ネットワークサービスについての情報、ネットワークの使用を許可されたユーザーグループについての情報、ネットワーク上で提供されるサービスについての情報なども必要です。ネットワークによって提供されるサービスが増えるにつれて、そのファイルも大きくなります。その結果、各ワークステーションに `/etc/hosts` または `/etc/inet/ipnodes` のようなファイルのセット全部を持たせる必要もでてきます。

この情報が変更されるごとに、管理者はネットワーク内のすべてのワークステーションにある情報を最新のものにしなければなりません。小さなネットワークではこれは単純な作業にすぎませんが、中規模または大規模なネットワークでは、この仕事は時間がかかるだけではなく、手に負えないものとなります。

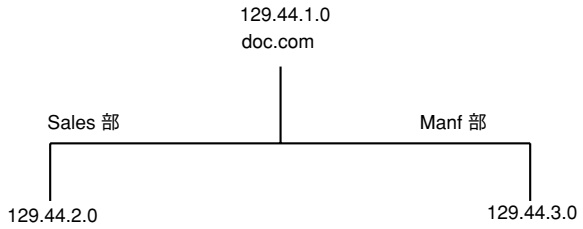
ネットワーク情報サービスがこの問題を解決します。このサービスでは、ネットワーク情報をサーバーに格納し、その情報を要求するワークステーションに提供します。



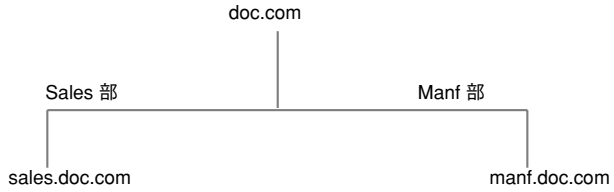
この場合、ワークステーションはサーバーの「クライアント」と呼ばれます。ネットワークについての情報が変更されるたびに、各クライアントのローカルファイルを変更する代わりに、管理者はネットワーク情報サービスが格納する情報だけを更新します。これによって、エラー、クライアント間の不一致、そして作業量を減らすことができます。

このように、サーバーがネットワークを通してサービスをクライアントに提供する方法を「クライアントサーバーコンピューティング」と呼びます。

ネットワーク情報サービスの第一の目的は情報の一元管理ですが、もう 1 つの目的はネットワーク名の簡素化です。たとえば、ある会社がネットワークを設定して、インターネットに接続したと仮定します。インターネットはその会社に 129.44.0.0 というネットワーク番号と、`doc.com` というドメインネームを割り当てました。会社には「営業 (Sales)」と「製造 (Manf)」という 2 つの部門があるため、このネットワークは 1 つのメインネットと、各部門に 1 つずつ、合計 2 つのサブネットに分割されます。各ネットには独自のアドレスがあります。



上に示すように、各部はネットワークアドレスで識別することもできますが、ネームサービスで使用できる名前の方が便利です。

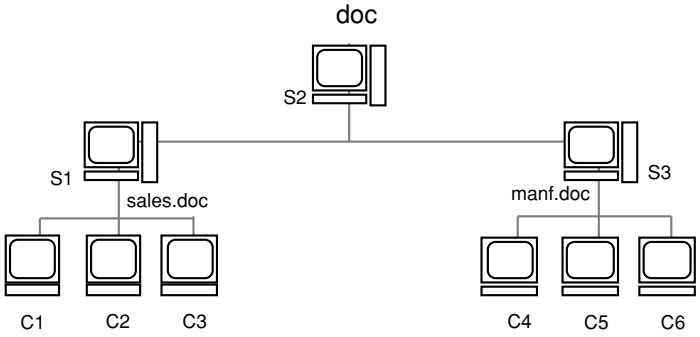


したがって、メールやその他のネットワーク通信の送信先を 129.44.1.0 というアドレスを指定する代わりに、単に `doc` と指定できます。また、129.44.2.0 や 129.44.3.0 と指定する代わりに、`sales.doc` や `manf.doc` と指定できます。

名前は、物理アドレスよりもはるかに柔軟です。物理的なネットワークはめったに変更されませんが、これらを使用する組織はよく変化します。ネットワーク情報

サービスは、組織とその物理的なネットワークとの間のバッファのように機能します。その理由は、ネットワーク情報サービスが物理的ネットワークに実際に接続されているのではなく、マップされているためです。次の例でこれを説明します。

この doc.com ネットワークが、S1、S2、S3 の 3 台のサーバーによってサポートされ、これらのうち 2 台のサーバー (S1 と S3) がクライアントをサポートすると仮定します。

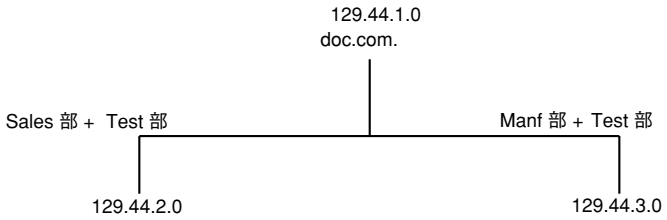


クライアント C1、C2、C3 はネットワーク情報をサーバー S1 から入手します。クライアント C4、C5、C6 はこれをサーバー S3 から入手します。このような形態のネットワークを次の表に示します (表 1–1 は、前記のネットワークを一般化して表現したもので、実際のネットワーク情報マップとは異なる)。

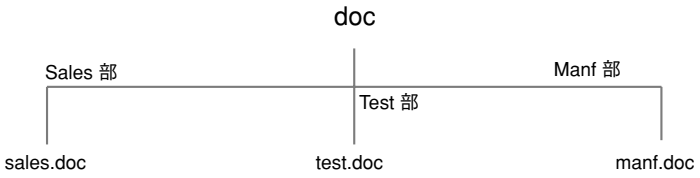
表 1–1 doc.com ネットワークの構成

ネットワークアドレス	ネットワーク	サーバー	クライアント
129.44.1.0	doc	S1	
129.44.2.0	sales.doc	S2	C1, C2, C3
129.44.3.0	manf.doc	S3	C4, C5, C6

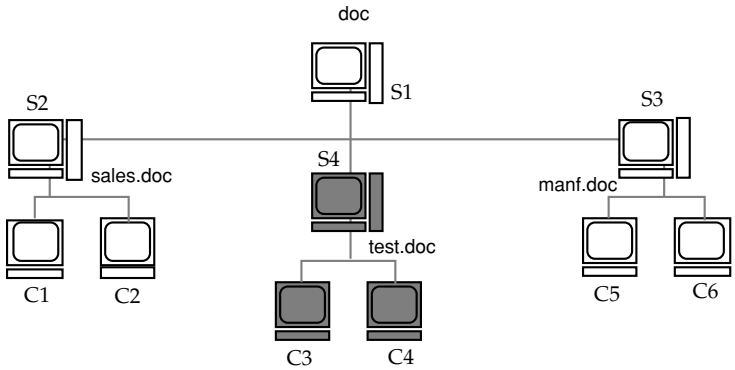
2 つの部門からある人数の人材を借りて第 3 の部門 Test を新設し、第 3 のサブネットは開設しなかったとします。その結果この物理ネットワークは、この企業の組織とは対応しなくなってしまいます。



Test 部門の通信には専用のサブネットがなく、129.44.2.0 と 129.44.3.0 に分割されます。しかし、ネットワーク情報サービスを使用することにより、Test 部門の通信も専用のネットワークを備えることができます。



このように、組織が変更された場合、そのネットワーク情報サービスは単にマッピングを変更するだけで対応できます。



こうして、クライアント C1 と C2 はサーバー S2 から、C3 と C4 はサーバー S4 から、C5 と C6 はサーバー S3 からそれぞれ情報を入手します。

この組織でこれ以降に行われる変更に対しては、ハードウェアのネットワーク構造を再編成することなく、ソフトウェアのネットワーク情報構造を変更することにより対応できます。

Solaris のネームサービス

Solaris 8 リリースには、以下のようなネームサービスがあります。

- DNS (ドメイン各システム、Domain Name Service) - 45ページの「DNS とは」を参照してください。
- /etc ファイル - 本来 UNIX で使用されるネームシステム。46ページの「/etc ファイル」を参照してください。
- NIS (ネットワーク情報サービス、Network Information Service) - 46ページの「NIS とは」を参照してください。
- NIS+ (Network Information Service plus) - 46ページの「NIS+ とは」を参照してください。
- FNS (フェデレーテッド・ネーミング・サービス、Federated Naming Service) - 1 つの Solaris 環境でさまざまな独立したネーミングシステムを使用できます。47ページの「FNS とは」を参照してください。

最近のほとんどのネットワークでは、これらのサービスを 2 つ、またはそれ以上組み合わせて使用します。複数のサービスを使用するときは、nsswitch.confファイルで調整します。nsswitch.conf ファイルについては第 2 章で説明します。

DNS とは

DNS は TCP/IP ネットワーク用にインターネットが提供するネームサービスです。ネットワーク上のワークステーションがインターネットアドレスではなく、普通の名前で識別できるように開発されたものです。DNS は、ローカルの管理ドメイン内と、複数の管理ドメイン間においてホスト名の管理を行います。

DNS を使う、ネットワークに接続されたワークステーションの集合のことを「DNS 名前空間」と呼びます。DNS 名前空間は階層をなす複数の「ドメイン」に分けることができます。1 つの DNS ドメインは複数のワークステーションがまとまったグループです。各ドメインは複数の「ネームサーバー」、つまり、1 つの主サーバーと 1 つまたは複数の副サーバーよってサポートされます。各サーバーは in.named と呼ばれるデーモンを実行することによって DNS を実装しています。クライアント側は、「リゾルバ」によって DNS を実装します。リゾルバの機能は、ユーザーによる参照を解決することです。このためにリゾルバはネームサーバーを参照します。参照を受けたネームサーバーは、要求された情報、または、別のサーバーに向けられた参照内容を返します。

/etc ファイル

最初のホストを基本とした UNIX の命名システムは、スタンドアロンの UNIX マシン用に開発された後、ネットワークで使用されるようになりました。UNIX オペレーティングシステムの旧版の多くや UNIX マシンでは、現在でもこのシステムが使用されていますが、大規模で複雑なネットワークにはあまり適切ではありません。

NIS とは

「NIS」は DNS とは独立して開発され、目的はやや異なっています。DNS はワークステーションアドレスの代わりにワークステーション名を使うことによって、通信を簡略化することに焦点を当てているのに対して、NIS の場合は、多様なネットワーク情報を集中管理することによりネットワーク管理機能を高めることに焦点を絞っています。NIS には、ユーザー、ネットワークそのもの、ネットワークサービスについての情報も格納されます。これらのネットワーク「情報」をまとめて「NIS の名前空間」と呼びます。

NIS 名前空間情報は NIS マップに格納されています。NIS マップは UNIX の `/etc` ファイルやその他の構成ファイルに替わるものとして設計され、実際には名前やアドレス以外の情報も持っています。その結果、NIS 名前空間には非常に大きなマップの集合が含まれることになります (368ページの「NIS マップ」を参照)。

NIS は DNS に似たクライアントサーバーの配列を持っています。複製の NIS サーバーは NIS クライアントへサービスを提供します。主サーバーは「マスター」サーバーと呼ばれ、安全のためのバックアップがあります。これは「スレーブ」サーバーと呼ばれています。どちらのサーバーも NIS 情報検索ソフトウェアを使用し、NIS マップを格納します。NIS アーキテクチャの詳細は、362ページの「NIS アーキテクチャ」を参照してください。

NIS とその管理方法については パート IV「NIS の管理」を参照してください。

NIS+ とは

「NIS+」は、NIS によく似たネットワークネームサービスですが、より多くの機能を備えています。NIS+ は NIS を機能拡張したものではなく、新しいソフトウェアプログラムとなっています。

NIS+ ネームサービスは、ネットワークがどのような構造であっても、その周囲を取り巻くことにより、サービスを設置した組織の形態に適合するように設計されてい

ます。NIS の場合と異なり、NIS+ の名前空間は動的なため更新が可能で、正規ユーザーであればいつでも更新できます。

NIS+ は (ワークステーションのアドレス、セキュリティ情報、メール情報、Ethernet インタフェースおよびネットワークサービスに関する情報などの) 情報を 1 カ所に格納して、ネットワーク上のすべてのワークステーションからアクセスできるようにします。このように構成されたネットワーク情報を、NIS+「名前空間」と呼びます。

NIS+ 名前空間は階層構造となっていて、UNIX のディレクトリファイルシステムによく似ています。階層構造になっていることから、NIS+ 名前空間は企業組織の階層に合わせて構成できます。名前空間の情報のレイアウトは、その「物理的」構成とは無関係です。したがって、NIS+ 名前空間は、独立して管理できる複数のドメインに分割できます。クライアントは、適切なアクセス権があれば、自分のドメインだけではなく、ほかのドメインの情報にもアクセスできます。

NIS+ はクライアントサーバーモデルを使用して、NIS+ 名前空間に情報を格納し、またその情報にアクセスできます。各ドメインは複数のサーバーによってサポートされます。最も重要なサーバーは「マスターサーバー」と呼ばれ、バックアップサーバーは「複製サーバー」と呼ばれます。ネットワーク情報は、内部 NIS+ データベース内にある 16 個の標準 NIS+ テーブルに格納されています。マスターサーバーと複製サーバーは、共に NIS+ サーバーソフトウェアを実行し、NIS+ テーブルのコピーを管理します。マスターサーバー上の NIS+ データに対する変更は、複製サーバーにも自動的に伝達されます。

NIS+ には、名前空間の構造とその情報を保護するために、高度なセキュリティシステムが組み込まれています。NIS+ は認証 (authentication) と承認 (authorization) を使用して、クライアントの情報要求に応えるべきかどうかを検証します。「認証」とは、情報の要求者がネットワークの正当なユーザーであるかどうかを判定することです。「承認」とは、要求された情報に関して特定のユーザーが入手または変更を許可されているかどうかを判定することです。

NIS+ のより詳しい説明については、パート II「NIS+ の紹介と概要」を、使用方法については、パート III「NIS+ の管理」を参照してください。

FNS とは

「FNS」は、1 つの Solaris 環境でさまざまなネームサービスを独立して動作させるための機能です。FNS を使用すれば、ネットワーク上のさまざまなネームサービスすべてに、1 つの簡単なネームシステムインタフェースで対応できます。FNS は、X/Open federated naming (XFN) 規格に適合しています。

FNS は、NIS+、NIS、DNS、/etc ファイルの代わりとして使用することはできません。FNS はむしろこれらのサービスの一番上に位置しており、通常の名前をデスクトップ上のアプリケーションで使用できるようにします。

FNS のより詳しい説明と管理方法については、パート V「FNS の管理」を参照してください。

第 2 章

ネームサービススイッチ

この章では、ネームサービススイッチの機能と、これを使用してクライアントが 1 つまたは複数のソースからネーミング情報を入手する方法について説明します。ネームサービススイッチは、異なるネームサービスの使用方法を調整するために使います。

ネームサービススイッチについて

ネームサービススイッチは `nsswitch.conf` という名前のファイルで、クライアントのワークステーションやアプリケーションがネットワーク情報を得る方法を管理します。ネームサービススイッチは、

- `gethostbyname()`
- `getpwuid()`
- `getpwnam()`
- `getipnodebyname()`

のような getXbyY() インタフェースを呼び出すクライアントアプリケーションによって使用されます。ネームサービススイッチは単に「スイッチ」または「スイッチファイル」と呼ばれることもあります。各ワークステーションは、それぞれの /etc ディレクトリの中にスイッチファイルを持っています。ファイルの各行は、host、passwd、group などの特定タイプのネットワーク情報を識別します。その後には、クライアントがネットワーク情報を探すための 1 つまたは複数のソースが続きます。

クライアントは、1 つまたは複数のスイッチのソースからネーミング情報を入手できます。たとえば、NIS+ のクライアントは、NIS+ テーブルからホスト情報を、ローカルの /etc ファイルからパスワード情報をそれぞれ入手できます。さらに、スイッチが各ソースを使用する条件を指定することもできます。52ページの「検索規準」を参照してください。

Solaris オペレーティング環境では、インストールの過程において、各ワークステーションの /etc ディレクトリに nsswitch.conf ファイルを自動的にロードします。同時に、次に示すスイッチファイルの 4 つの代替 (テンプレート) バージョンも /etc ディレクトリにロードされます。

- /etc/nsswitch.files
- /etc/nsswitch.nis
- /etc/nsswitch.nisplus
- /etc/nsswitch.ldap

これら 4 つのファイルは、代替デフォルトスイッチファイルです。各ファイルはそれぞれ /etc ファイル、NIS、NIS+ 、LDAP という異なる主要なネームサービス用に設計されています。Solaris リリースソフトウェアを初めてワークステーションにインストールするときに、インストール担当者はワークステーションのデフォルトネームサービスを NIS+、NIS、ローカルファイル、LDAP の中から選択します。インストール中に、対応するテンプレートファイルが nsswitch.conf ファイルにコピーされます。たとえば、クライアントが NIS+ を使用しているワークステーションでは、インストールの過程で nsswitch.nisplus ファイルが nsswitch.conf にコピーされます。特殊な名前空間を持っている場合を除き、通常の操作には nsswitch.conf にコピーされるデフォルトのテンプレートファイルを使用します。DNS または IPv6 用のデフォルトファイルは提供されませんが、これら 4 つのファイルを編集して DNS または IPv6 用に使用できます (59ページの「DNS とインターネットでのアクセス」および60ページの「IPv6 とインターネットでのアクセス」を参照してください)。

後からワークステーションの主要なネームサービスを変更する場合は、適切な代替スイッチファイルを nsswitch.conf にコピーするだけで変更できます (56ページの「nsswitch.conf テンプレートファイル」を参照)。また、/etc/nsswitch.conf ファイルの適当な行を編集することによって、クライアントが使用するネットワーク情報のソースを変更することもできます。この構文については以下に説明します。詳細は、『*Solaris* ネーミングの設定と構成』を参照してください。

nsswitch.conf ファイルのフォーマット

nsswitch.conf ファイルは、基本的には 16 種類の情報とそのソース (getXXbyYY() 関数の情報検索先) のリストです。順序は必ずしも以下のとおりではありません。

- aliases
- bootparams
- ethers
- group
- hosts
- ipnodes
- netgroup
- netmasks
- networks
- passwd (シャドウ情報含む)
- protocols
- publickey
- rpc
- services
- automount
- sendmailvars

表 2–1 は、上記の情報タイプのスイッチファイルの中に、一覧表示できるソースの種類の詳細を示しています。

表 2–1 スイッチソースの例

ソース	説明
files	クライアントの /etc ディレクトリに格納されているローカルファイル (/etc/passwd など)
nisplus	NIS+ テーブル (hosts テーブルなど)
nis	NIS マップ (hosts マップなど)
compat	パスワードとグループ情報を対象に、/etc/passwd、/etc/shadow、/etc/group ファイルで旧形式の「+」または「-」構文をサポートする
dns	ホスト情報を DNS から入手するように指定する
ldap	エントリを LDAP ディレクトリから入手するように指定する

検索規準

「単一ソース」

nisplus のような情報のソースが 1 つだけの場合、スイッチを使用している関数は、そのソースだけで情報を検索します。情報が見つかった場合、success という状態メッセージが渡されます。情報が見つからない場合は、検索が停止され、success 以外の状態メッセージが渡されます。状態メッセージに基づいて何をするかは、関数によって異なります。

「複数ソース」

テーブルに複数のソースがある場合、スイッチは最初のソースから情報検索を始めるように関数に指示します。情報が見つかれば success という状態メッセージが返されますが、見つからないときは次のソースが検索されます。関数は必要な情報が見つかるか、return 処理によって中止されるまで全ソースの検索を続けます。必要な情報がどのソースにもなかったとき、関数は検索を停止し、non-success という状態メッセージを返します。

スイッチ状態メッセージ

関数は情報を見つけると、success という状態メッセージを返します。また情報が見つからなかった場合、その理由によって、3 種類のメッセージのうちの 1 つを返します。表示される状態メッセージを、以下の表 2–2 に示します。

表 2–2 スイッチ状態メッセージ

状態メッセージ	意味
SUCCESS	要求されたエントリがソース内で発見された
UNAVAIL	ソースが応答しない、または使用不可。つまり、NIS+ テーブル、NIS マップ、/etc ディレクトリのファイルが見つからなかった (アクセスできなかった)
NOTFOUND	エントリなし。テーブル、マップ、ファイルにアクセスしたが、必要な情報は見つからなかった
TRYAGAIN	ソース使用中のため、再検索の必要あり。テーブル、マップ、ファイルは見つかったが、照会に対して応答しなかった

スイッチの動作に関するオプション

表 2–3 に示すように、状態メッセージに対して次の 2 つの動作のどちらかで応答するようにスイッチに指示できます。

表 2–3 スイッチ状態メッセージへの応答

動作	意味
return	情報の検索を停止する
continue	次のソースがあれば、それを検索する

デフォルト検索基準

nsswitch.conf ファイルの状態メッセージと動作オプションの組み合わせによって、関数の各ステップでの動作が決まります。この状態と動作の組み合わせのことを、「検索基準」と呼びます。

スイッチのデフォルト検索規準は、どのソースについても同じです。これらを上記の状態メッセージに基づいて説明すれば次のようになります。

- SUCCESS=return

  情報の検索を停止し、見つかった情報を使用して処理を続行する

- UNAVAIL=continue

  次のソース (nsswitch.conf ファイルに指定されたもの) を使用して検索を続行する。次のソースがなければ、NOTFOUND という状態メッセージを返す

- NOTFOUND=continue

  次のソース (nsswitch.conf ファイルに指定されたもの) を使用して検索を続行する。次のソースがなければ、NOTFOUND という状態メッセージを返す

- TRYAGAIN=continue

  次のソース (nsswitch.conf ファイルに指定されたもの) を使用して検索を続行する。次のソースがなければ、NOTFOUND という状態メッセージを返す

これらはデフォルトの検索基準であるため、自動的に表示されます。つまり、スイッチファイルで、はっきりと指定する必要はありません。ほかの検索基準を明示してデフォルトの検索基準を変更するには、上記の *STATUS=action* という構文を使用します。たとえば、NOTFOUND 状態に対し、デフォルトの動作では次のソースに移って検索を続行します。networks など、特定の情報を設定して検索すると、検索は NOTFOUND で中止します。スイッチファイルの networks の行を、以下のように編集してください。

```
networks: nis [NOTFOUND=return] files
```

networks: nis [NOTFOUND=return] files は、NOTFOUND に関してデフォルトでない検索基準を設定するものです (デフォルト以外の設定をするときは [ ]を使用します)。

この例では、検索関数は以下のような働きをします。

- networks

  マップが見つかり、必要な情報があった場合、関数は SUCCESS という状態メッセージを返します。

- networks

  マップが見つからなかった場合、関数は UNAVAIL という状態メッセージを返し、デフォルトにより適当な /etc ファイルの検索を続行します。

- networks

  マップは見つかったが必要な情報がなかった場合、関数は NOTFOUND という状態メッセージを返します。そして /etc ファイルの検索を続行する (デフォルトの設定) 代わりに検索を停止します。

- networks

  マップが使用中の場合、関数は TRYAGAIN という状態メッセージを返し、デフォルトで適当な /etc ファイルの検索を続行します。

構文が正しくない場合の処理

クライアントのライブラリ関数には、nsswitch.conf ファイルにおいて「必要なエントリがない」、「エントリの構文が誤っている」といった場合に使用されるコンパイル時に組み込まれるデフォルトエントリが含まれています。これらのエントリは nsswitch.conf ファイルのデフォルトエントリと同じものです。ネームサービススイッチは、テーブル名やソース名のスペルが正しいものとして処理をします。スペルが正しくない場合は、デフォルト値が使用されます。

auto_home と auto_master

auto_home テーブル、auto_master テーブルとマップのスイッチ検索基準は、automount と呼ばれる 1 つのカテゴリに統合されます。

Timezone とスイッチファイル

timezone テーブルはスイッチを使用しないため、スイッチファイルのリストには含まれていません。

nsswitch.conf ファイル中のコメント

nsswitch.conf ファイル中の行のうち、「#」で始まっているものはコメント行として解釈され、ファイルを検索する関数では無視されます。

「#」が行の途中にある場合、その左側の文字は nsswitch.conf ファイルを検索する関数の一部となり、右側の文字はコメント行として無視されます。

表 2–4 スイッチファイルのコメント例

行の種類	コメント例
コメント行 (無視される)	`# hosts: nisplus [NOTFOUND=return] files`
完全に解釈される行	`hosts: nisplus [NOTFOUND=return] file`
部分的に解釈される行 (「files」の部分は解釈されない)	`hosts: nisplus [NOTFOUND=return] # files`

スイッチファイルのキーサーバーと publickey エントリ

キーサーバーは、起動時にだけネームサービススイッチ構成ファイルの publickey エントリを参照します。つまり、スイッチ構成ファイルを更新しても、再起動しない限りキーサーバーはそのことを認識しないということになります。

nsswitch.conf テンプレートファイル

Solaris 8 リリースでは、nsswitch.conf のテンプレートファイルが 3 種類提供されます。デフォルトの情報ソース (一次ソース、およびそれに続くもの) としては、それぞれ異なったものが指定されています。

3 種類のテンプレートファイルの詳細は以下のとおりです。

- 「NIS+ テンプレートファイル」(nsswitch.nisplus ファイル)

  このテンプレートファイルで passwd、group、automount、aliases を除くすべての情報の一次ソースとして指定されているのは、NIS+ です。passwd、group、automount、aliases の一次ソースとして指定されているのは /etc ディレクトリのファイルで、二次ソースとして指定されているのは NIS+ テーブルです。[NOTFOUND=return] という検索基準は、「No such entry」というメッセージを受け取ったら NIS+ テーブルの検索を停

止する」という意味です。また、ローカルファイルを検索するのは NIS+ サーバーを使用できない場合だけです。

- 「NIS テンプレートファイル」(nsswitch.nis ファイル)

  NIS+ テーブルではなく NIS マップを使用するという点を除けば NIS+ テンプレートファイルとほぼ同じです。passwd、group の情報に関しては files nis という順序で検索するよう指定されているため、/etc/passwd と /etc/group に + エントリを指定する必要はありません。

- 「Files テンプレートファイル」(nsswitch.files ファイル)

  このテンプレートファイルでは、ローカルの /etc ディレクトリのファイルだけがワークステーションの情報ソースとして指定されています。netgroup に関する files のソースは存在しないため、クライアントがスイッチファイルでこのエントリを使用することはありません。

要求するものに最も近いテンプレートファイルを nsswitch.conf にコピーし、必要に応じて修正します。この手順の詳細は、『*Solaris* ネーミングの設定と構成』のスイッチに関する章を参照してください。

たとえば、NIS+ テンプレートファイルを使用するには、以下のコマンドを入力します。

```
mymachine# cp nsswitch.nisplus nsswitch.conf
```

デフォルトスイッチテンプレートファイル

3 つのスイッチファイルは、Solaris 8 リリースでは、以下のようになります。

例 2–1 NIS+ スイッチファイルテンプレート (nsswitch.nisplus)

```
#
# /etc/nsswitch.nisplus:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS+ (NIS Version 3) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd: files nisplus
group: files nisplus
# consult /etc "files" only if nisplus is down.
hosts: nisplus [NOTFOUND=return] files
```

(続く)

```
# Uncomment the following line, and comment out the above, to use
# both DNS and NIS+. You must also set up the /etc/resolv.conf
# file for DNS name server lookup. See resolv.conf(4).
# hosts: nisplus dns [NOTFOUND=return] files
services: nisplus [NOTFOUND=return] files
networks: nisplus [NOTFOUND=return] files
protocols: nisplus [NOTFOUND=return] files
rpc: nisplus [NOTFOUND=return] files
ethers: nisplus [NOTFOUND=return] files
netmasks: nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files
publickey: nisplus
netgroup: nisplus
automount: files nisplus
aliases: files nisplus
sendmailvars: files nisplus
```

例 **2–2** NIS スイッチファイルテンプレート

```
#
# /etc/nsswitch.nis:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS (YP) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
#
# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd: files nis
group: files nis
# consult /etc "files" only if nis is down.
hosts: nis [NOTFOUND=return] files
networks: nis [NOTFOUND=return] files
protocols: nis [NOTFOUND=return] files
rpc: nis [NOTFOUND=return] files
ethers: nis [NOTFOUND=return] files
netmasks: nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey: nis [NOTFOUND=return] files
netgroup: nis
automount: files nis
aliases: files nis
# for efficient getservbyname() avoid nis
services: files nis
sendmailvars: files
```

例 **2–3** Files スイッチファイルテンプレート

```
#
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
passwd: files
group: files
hosts: files
networks: files
protocols: files
rpc: files
ethers: files
netmasks: files
bootparams: files
publickey: files
# At present there isn't a 'files' backend for netgroup;
# the system will figure it out pretty quickly, and won't use
# netgroups at all.
netgroup: files
automount: files
aliases: files
services: files
sendmailvars: files
```

デフォルト **nsswitch.conf** ファイル

Solaris の環境を初めてインストールするときのデフォルトの nsswitch.conf ファイルは、Solaris のソフトウェアをインストールする際に選択したネームサービスで決まります。ネームサービスが選択されると、そのサービスのスイッチテンプレートファイルがコピーされ、新しい nsswitch.conf ファイルが作成されます。たとえば、NIS+ が選択された場合は、nsswitch.nisplus ファイルがコピーされて nsswitch.conf ファイルが作成されます。

DNS とインターネットでのアクセス

nsswitch.conf ファイルでは、以下のいくつかのセクションで説明するとおり、クライアントの DNS 転送も制御されます。DNS 転送によって、クライアントへのインターネットでのアクセスが可能になります。

注 - NIS+ クライアントには、正しく設定された `/etc/resolv.conf` ファイルが必要です (577ページの「DNS クライアントとリゾルバ」を参照)。

NIS+ および NIS クライアントによる DNS 転送ができるよう設定するための手順については、『*Solaris* ネーミングの設定と構成』 のスイッチファイルに関する説明を参照してください。

NIS+ クライアントでの DNS 転送

NIS+ クライアントには、NIS クライアントにあるような DNS 転送機能はありません。その代わりに、スイッチを使用した DNS 転送が可能です。NIS+ のクライアントに DNS 転送機能を持たせるには、`hosts` のエントリを以下のように変えます。

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

NIS クライアントでの DNS 転送

NIS ネームサービスでは、本来 DNS 転送が可能です。DNS 転送を可能にするためには、NIS 一次スイッチファイルの `hosts` の行に以下の書式を使用 します。

```
hosts: nis [NOTFOUND=return] files
```

注意 - NIS のクライアントが、NIS と互換性のある NIS+ サーバーの DNS 転送機能を使用している場合、ホストファイルの構文として `nsswitch.conf` ファイルに `hosts: nis dns files` を使用することはできません。使用した場合、DNS 転送によってホスト要求が自動的に DNS に転送され、この構文によって NIS+ サーバーが DNS サーバーにエラーステータスメッセージを 2 度転送することになり、性能が低下します。DNS 転送機能を十分に利用するには `nsswitch.nis` ファイルのデフォルト構文を使用してください。

IPv6 とインターネットでのアクセス

`nsswitch.conf` ファイルは、IPv6 のアドレスの検索基準を制御します。IPv6 は、32 ビットから 128 ビットまで IP アドレスサイズを大きくして、より多くのアドレス階層をサポートし、より多くのノードにアドレス指定できるようにします。 IPv6

の構成と実装の詳細は、『*Solaris* のシステム管理 *(*第 *3* 巻*)*』の「IPv6 の概要」と「IPv4 から IPv6 への移行」を参照してください。

IPv6 アドレスには、新しい ipnodes ソースを使用してください。/etc/inet/ipnodes ファイルには、 IPv4 と IPv6 の両方のアドレスが格納されています。 /etc/inet/ipnodes ファイルは、 /etc/hosts ファイルと同じフォーマットを使用しています。

IPv6 のネームサービスでは、 検索用に新しい ipnodes ソースを使用しています。たとえば、 LDAP でIPv6 のアドレスを認識させる場合には、次のように指定します。

```
ipnodes: ldap [NOTFOUND=return] files
```

注意 - ipnodes は、デフォルトでは files です。 IPv4 から IPv6 への変更中には、すべてのネームサービスが、IPv6 のアドレスを認識できるわけではないので、デフォルトの files を使用します。 このデフォルトを使用しない場合には、アドレスの解決中に不必要な遅延が生じることがあります (ブート時の遅延など) 。

注意 - アプリケーションは、、IPv4 のアドレスを ipnodes データベースで検索してから、hosts データベースを検索します。 ipnodes を指定する前に、IPv4 アドレスの両方のデータベースを検索する時間を考慮にいれる必要があります。

+/- 構文との互換性を確保する

nsswitch.conf ファイルに、/etc/passwd、/etc/shadow、/etc/group などでときおり使用される +/- 構文との互換性を持たせる方法を以下に示します。

- 「NIS+ の場合」に NIS+ で +/- 構文と同じ効果を得るには、passwd および groups のソースを compat に変更し、nsswitch.conf ファイルの passwd あるいは group エントリの後に passwd_compat: nisplus というエントリを追加します (下記参照)。

```
passwd: compat
passwd_compat: nisplus
group: compat
```

(続く)

```
group_compat: nisplus
```

この指定により、クライアント関数のネットワーク情報獲得先は、ファイル中に +/- エントリを指定したのと同じように、/etc ディレクトリのファイルと NIS+ テーブルになります。

- 「NIS の場合」に Sun OS 4.x リリースの構文と同じ効果を得るには、passwd と groups のソースを compat に変更します。

```
passwd: compat
group: compat
```

この指定により、クライアント関数のネットワーク情報獲得先は、ファイル中に +/- エントリを指定したのと同じように、/etc ファイルと NIS マップになります。

注 - NIS+ サーバーが NIS 互換モードで動作している場合、クライアントマシンでは netgroup テーブルに対して ypcat を実行できません。実行してもテーブルにエントリがない場合と同じような動作をします。

---

nsswitch.conf ファイルに +/- 構文と同じ意味を持つ指定をする手順の詳細は、『*Solaris* ネーミングの設定と構成』のスイッチファイルに関する説明を参照してください。

スイッチファイルとパスワード情報

passwd 情報では、常に files が初めに検索されるソースになります。

たとえば、NIS+ の環境では、nsswitch.conf ファイルの passwd 行は以下のようになります。

```
passwd: files nisplus
```

NIS の環境では、nsswitch.conf ファイルの passwd 行は以下のようになります。

```
passwd: files nis
```

注意 - passwd 情報の nsswitch.conf ファイルでは、files を 1 番目のソースにしてください。files が 1 番目のソースでない場合は、ネットワークセキュリティが低くなり、ログの扱いが難しくなります。

FNS とネームサービススイッチ

FNS (フェデレーテッド・ネーミング・サービス) の詳細は、パート V「FNS の管理」を参照してください。

FNS (XFN API を Solaris 上に実装したもの) は、クライアントがネーム情報を照会するためのネームサービスを指定するときにも使用できます。XFN API は、X、Y 両次元において、スイッチファイルを使用する最新の getXbyY() インタフェースよりも一般的です。たとえば、XFN API を使用して、NIS+ と NIS の両方からホストとユーザーに関する情報を調べることができます。アプリケーションは、getXbyY()、XFN、あるいはその両方のクライアントとして使用できます。

FNS とスイッチファイルに一貫性を持たせる

FNS による名前空間データの変更を、スイッチファイルを使用して名前空間情報を入手しているクライアントが常に把握できるようにするために、スイッチと FNS には常に同じネームサービスを設定してください。

名前空間の更新

XFN API によるデータ更新は、getXbyY() インタフェースによる更新よりも優れています。ほとんどの名前空間は複数ソースのデータで構成されています。たとえば、groups の名前空間には、/etc/group ファイルと NIS+ group.org_dir オブジェクトの両方の情報があるかもしれません。しかし、スイッチファイルは、グループデータの特定の部分のソースや更新するソースを識別するための、アプリケーションの更新関数に十分な情報を提供しません。

各 FNS の従属名前空間は、すべてが 1 つのネームサービスから取られます。更新がどのネームサービスに行われるかというような混乱がないため、更新は簡単明瞭なものになります。

パート II　NIS+ の紹介と概要

パート II では、NIS+ について説明します。

- 第 3 章
- 第 4 章
- 第 5 章
- 第 6 章

第 3 章

NIS+ の紹介

この章では、NIS+ の概要を述べます。

NIS+ および DNS 名前空間を設定する方法については、『*Solaris* ネーミングの設定と構成』を参照してください。用語や略語の定義については、用語集を参照してください。

NIS+ について

NIS+ は NIS によく似たネットワークネームサービスですが、より多くの機能を備えています。NIS+ は NIS を機能拡張したものではなく、新しいソフトウェアプログラムとなっています。

NIS+ ネームサービスは、ネットワークがどのような構造であっても、その周囲を取り巻くことにより、サービスを設置した組織の形態に適合するように設計されています。

NIS+ はワークステーションのアドレス、セキュリティ情報、メール情報、Ethernet インタフェース、ネットワークサービスなどの情報を 1 カ所に格納して、ネットワーク上のすべてのワークステーションからアクセスできるようにします。このように構成されたネットワーク情報を、NIS+「名前空間」と呼びます。

NIS+ 名前空間は階層構造となっていて、UNIX のディレクトリファイルシステムによく似ています。階層構造になっていることから、NIS+ 名前空間を企業組織の階層に合わせて構成できます。名前空間の情報のレイアウトは、その「物理的」構成とは無関係です。したがって、NIS+ 名前空間は、独立して管理できる複数のドメインに分割できます。クライアントは、適切なアクセス権があれば、自分のドメインだけではなく、ほかのドメインの情報にもアクセスできます。

NIS+ はクライアントサーバーモデルを使用して、NIS+ 名前空間に情報を格納し、またその情報にアクセスできます。各ドメインは複数のサーバーによってサポートされます。最も重要なサーバーは「マスター」サーバーと呼ばれ、バックアップサーバーは「複製」サーバーと呼ばれます。ネットワーク情報は、内部 NIS+ データベース内にある 16 個の標準 NIS+ テーブルに格納されています。マスターサーバーと複製サーバーは、共に NIS+ サーバーソフトウェアを実行し、NIS+ テーブルのコピーを管理します。マスターサーバー上の NIS+ データの変更は、複製サーバーにも自動的に伝達されます。

NIS+ には、名前空間の構造とその情報を保護するために、高度なセキュリティシステムが組み込まれています。NIS+ は認証 (authentication) と承認 (authorization) を使用して、クライアントの情報要求に応えるべきかどうかを検証します。「認証」とは、情報の要求者がネットワークの正当なユーザーであるかどうかを判定することです。「承認」とは、要求された情報に関して特定のユーザーが入手または変更を許可されているかどうかを判定することです。

Solaris のクライアントは、ネームサービススイッチ (`/etc/nsswitch.conf` ファイル) を使用して、ワークステーションがどこからネットワーク情報を取り出すかを決定します。この種の情報はローカル側の `/etc` ファイルや、NIS、DNS、NIS+ に格納されます。ネームサービススイッチでは、情報の種類ごとに異なるソースを指定できます。

NIS+ のメリット

NIS と比較すると、NIS+ には次のようなメリットがあります。

- 安全性の高いデータアクセス
- 階層構造を使用した分散型のネットワーク管理
- 非常に大きな名前空間の管理
- 異なるドメインにあるリソースへのアクセス
- 増分 (incremental) 更新

74ページの「NIS+ のセキュリティ」で説明したセキュリティシステムを使用すれば、特定のテーブルの個々のエントリに対する特定のユーザーのアクセスを制御できます。このようなセキュリティ方式では、システムを安全に維持でき、しかも NIS+ 名前空間全体またはテーブル全体が損傷する危険性がなく、管理業務を広く分散させることができます。

NIS+ の階層構造により、1 つの名前空間に複数のドメインを置くことができます。ドメインに分割することにより、管理が容易になります。個々のドメインは完全に独立して管理できるため、システム管理者の負担が軽減され、非常に大きな名前空間の管理責任から解放されます。このように、セキュリティシステムを分散型のネットワーク管理と組み合わせることによって、管理作業の負担を分担することができます。

ドメインが個別に管理されているとしても、すべてのクライアントに名前空間内のすべてのドメインの情報を参照するアクセス権を与えることができます。クライアントは自分のドメイン内のテーブルしか見ることができないため、クライアントがほかのドメイン内のテーブルにアクセスするには、そのテーブルについて明示的にアドレスを指定しなければなりません。

増分更新とは、名前空間内での情報の高速の更新を意味します。ドメインは独立して管理されるため、マスターサーバーテーブルへの変更は、名前空間全体にではなく、その複製サーバーだけに伝達しますが、伝達が行われると、これらの変更はすぐに名前空間全体で有効になります。

NIS+ と NIS の違い

「NIS+」と「NIS」はいくつかの点で違いがあります。NIS+ は多くの新機能を備え、同じような概念でも用語が異なっています。わからない用語があれば用語集を参照してください。NIS と NIS+ の主な相違点を表 3–1 にまとめます。

表 3–1 NIS と NIS+ の違い

NIS	NIS+
フラットなドメイン - 階層なし	階層構造 - データを名前空間内の異なるレベルに格納
データを 2 列のマップに格納	データを複数の列のテーブルに格納
認証を使用しない	DES 認証を使用
ネットワーク情報源は 1 つ	ネームサービススイッチ - クライアントは NIS、NIS+、DNS、またはローカル側の `/etc` ファイルから情報源を選択できる
バッチ伝達のため更新は遅い	すぐに伝達される増分更新

NIS+ は NIS に代わるものとして設計されました。NIS は、1980 年代に普及していたクライアントサーバーコンピューティングネットワークの管理要件に応えるものです。その当時のクライアントサーバーネットワークは、通常、クライアント数が数百を超えることはなく、多目的サーバーの数もわずかでした。これらのネットワークは数カ所のリモートサイトを結んでいるだけであり、しかもユーザーに専門知識があり信頼できたため、セキュリティを必要としませんでした。

しかしクライアントサーバーネットワークは 1980 年代半ば頃から急速な成長を遂げました。現在では、世界中のサイトに配置された 10～100 台の専用サーバーにサポートされた 100～10,000 台のマルチベンダークライアントが存在し、複数の公衆網に接続されています。さらに、ネットワークが格納する情報は、NIS の時代よりもはるかに急速に変化しています。このようなネットワークの規模と複雑性に対処するため、新しい管理方式が必要になりました。NIS+ はこれらの必要性に焦点をあてて設計されました。

NIS の名前空間は、フラットな状態で管理機能を集中管理しています。1990 年代に入りネットワークはスケーラビリティと管理の分散化を求めるようになったため、NIS+ の名前空間は DNS の場合のように階層ドメインをベースに設計されました。

たとえば、図 3–1 は `doc` という名前の親ドメインと、`sales` と `manf` という 2 つのサブドメインを持つ会社の例を示しています。

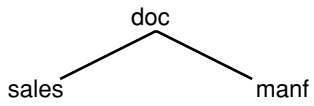


図 3–1　階層型ドメインの例

これによって NIS+ は、小規模から大規模まで、広い範囲のネットワークで使用できます。また、NIS+ のサービスを組織の成長に適合させることもできます。たとえば、ある会社が 2 つの部門に分かれた場合、それに対する NIS+ の名前空間を 2 つのドメインに分割し、これらを自律的に管理できます。インターネットがドメインの管理を下のレベルに委譲するように、NIS+ のドメインも程度の差はあっても互いに独立して管理できます。

NIS+ は DNS と似たドメイン階層を使用しますが、NIS+ のドメインは DNS のドメインよりずっと多くの情報を持っています。DNS のドメインは、そのクライアントの名前とアドレスの情報を格納するだけです。一方 NIS+ のドメインには、組織の一部の中でのワークステーション、ユーザー、およびネットワークサービスについての「情報」が集められています。

ドメインをこのように分割することで、管理はより自律的になり、規模が拡大した場合にもうまく対処できますが、情報へのアクセスが以前より困難になることもありません。クライアントは他のドメインの情報にも、同じドメインと同じようにアクセスできます。あるドメインを別のドメインの内部から管理することもできます。

NIS+ のクライアントサーバーの配置は、各ドメインが一組のサーバーによってサポートされるという点では、NIS や DNS の配置と似ています。 主サーバーは「マスター」サーバーと呼ばれ、バックアップサーバーは「複製」サーバーと呼ばれます。マスターサーバーと複製サーバーは、両方とも NIS+ のサーバーソフトウェアを実行し、NIS+ テーブルのコピーを保持します。主サーバーはオリジナルのテーブルを、バックアップサーバーはコピーを、それぞれ格納します。

しかし NIS+ は、NIS とはまったく異なる更新方式を使用します。NIS が開発された当時は、格納される情報の型はめったに変化しなかったため、NIS は安定性に重点を置いた更新方式によって開発されました。NIS テーブルの更新は手作業で処理

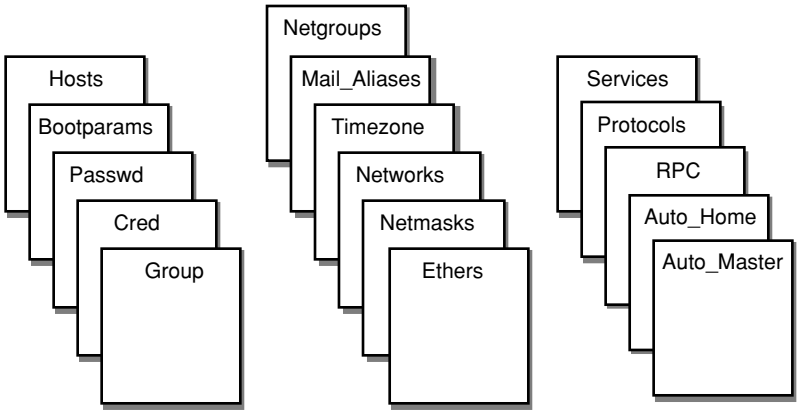
され、大規模な組織では、すべての複製サーバーへの伝達に 1 日以上かかることもあります。この原因の一つは、マップ内の情報が変化するたびにマップ全体を再作成して伝達しなければならなかったからです。

しかし NIS+ は、複製サーバーに対する「増分 (incremental)」更新が可能です。マスターサーバー上での変更は依然として必要ですが、いったん変更すれば、複製サーバーに自動的に伝達され、すぐに名前空間全体から使用できるようになります。マップを「作成」したり、伝達を待つ必要はありません。

NIS+ のドメイン構造、サーバー、およびクライアントの詳細は、第 4 章を参照してください。

NIS+ のドメインは、次に示すようなネームサービススイッチを使用して、その NIS+ クライアントを経由してインターネットに接続できます (74ページの「NIS+ とネームサービススイッチ」を参照)。このクライアントが DNS のクライアントでもある場合、自分のスイッチ構成ファイルを設定して、NIS+ テーブルだけでなく、DNS ゾーンファイルまたは NIS マップ内の情報を検索できます。

NIS+ は、マップやゾーンファイルではなく、「テーブル」に情報を格納します。NIS+ は 16 種類のあらかじめ定義したテーブル (つまり「システム」テーブル) を提供します。



各テーブルにはそれぞれ異なる情報が格納されています。たとえば、`hosts` テーブルにはワークステーションアドレスについての情報が、`password` テーブルにはネットワークのユーザーについての情報が格納されています。

NIS+ のテーブルは、NIS で使用されるマップと比較して 2 つの点で大きく改良されています。第 1 に、NIS+ のテーブルは、最初の列 (キーと呼ばれることもある) だけではなく、任意の列ごとにアクセスできます。これによって、NIS で使用される `hosts.byname` や `hosts.byaddr` マップなどの二重マップが必要なくなります。

第 2 に、NIS+ のテーブル内の情報は、3 つの細分化されたレベルでアクセスし、操作できます。テーブルレベル、エントリレベル、および列レベルです。NIS+ のテーブルとそこに格納されている情報については、第 5 章で説明します。

NIS 管理者は、以下に示す原則および条件下で NIS を NIS+ と共に使用できます。

- 同一ドメインに NIS サーバーと NIS+ サーバーの両方が存在する

  NIS 管理者は同一ドメインで NIS サーバーと NIS+ サーバーの両方を動作させることは可能ですが、このような動作を長時間行わせることは望ましくありません。一般に、同一ドメイン内での NIS サーバーと NIS+ サーバーを両方使用するのは、NIS から NIS+ への短い移行期間だけに制限するべきです。

- サブドメイン

  NIS 管理者のルートドメインのマスターサーバーで NIS+ が動作している場合は、NIS 管理者は、すべてのサーバーで NIS が動作しているサブドメインを設定できます。NIS 管理者のルートドメインのマスターサーバーで NIS が動作している場合は、NIS 管理者はサブドメインの設定はできません。

- 同一ドメインに複数のワークステーションが存在する

  - 同一ドメイン内のサーバーで NIS+ が動作している場合は、NIS+、NIS、または `/etc` ファイルを使ってネームサービス情報を取得できるように、ドメイン内の各マシンを設定できます。NIS+ サーバーが NIS クライアントのニーズを満たすには、この NIS+ サーバーが NIS 互換モードで動作している必要があります。
  - 同一ドメイン内の異なる複数のサーバーで NIS が動作している場合は、NIS または `/etc` ファイルを使ってネームサービス情報を取得できるように、このドメイン内の各マシンを設定することができます (各マシンは NIS+ の使用不可)。

  さまざまなネームサービス情報を取得するためにワークステーションがどのサービスを使用するかは、そのマシンの `nsswitch.conf` ファイルで制御されます。このファイルは、「スイッチ」ファイルと呼ばれています。詳細は、第 2 章を参照してください。

NIS+ のセキュリティ

NIS+ は、「承認と認証」という相補的なプロセスによって、名前空間の構造と格納する情報を保護します。

- 「承認」は、名前空間の構成要素に対して、「誰がどのような操作を行えるのか」ということを指定することです。すべての構成要素について行われます。
- 「認証」は、名前空間へのアクセス要求をしている NIS+「主体」(プロセス、マシン、ルート、ユーザーなど) が「正当なものであるかどうかを確認」することです。「正当な NIS+ 主体」とは、NIS+「資格」を持っているものを指します。名前空間へのアクセスが行われると、必ずその要求者 (主体) の認証が行われます。

NIS+ が要求どおりの動作をするのは、「主体が認証されていて (正当な資格を持っていて)、要求された操作が、該当する構成要素において承認されている」という場合のみです。資格がない (または完全でない)、要求された操作が承認されていないという場合、NIS+ はアクセス要求を拒否します。NIS+ セキュリティシステムの詳細は、第 6 章パート II「NIS+ の紹介と概要」を参照してください。

NIS+ とネームサービススイッチ

NIS+ は、「ネームサービススイッチ」と呼ばれる別の機能と連係して動作します。ネームサービススイッチは、単に「スイッチ」と呼ばれることもあります。これを使用することにより、Solaris 8 リリースベースのワークステーションは、複数のネームサービスから情報を入手できます。具体的には、ローカルファイル、つまり `/etc` 内のファイル、NIS マップ、DNS ゾーンファイル、または NIS+ テーブルからです。このスイッチによって、ソースの選択が可能となるだけでなく、ワークステーションは情報の「種類」ごとに別のソースを指定できます。このスイッチの詳細は、第 2 章を参照してください。

Solaris 1.x と NIS 互換モード

Solaris リリース 1.x または 2.x では、NIS+ を NIS が動作しているワークステーションで使用できます。つまり、NIS+ ドメイン内にあるマシンはそれぞれの

nsswitch.conf ファイルを nisplus ではなく nis に設定できます。NIS を実行中のマシン上で NIS+ のサービスにアクセスする場合は、NIS+ のサーバーを「NIS 互換モード」で動作させる必要があります。

NIS+ は NIS 互換モードを提供します。NIS 互換モードを使用すれば、Solaris 8 リリースを実行している NIS+ サーバーは、NIS+ クライアントからの要求に応答しながら、NIS クライアントからの要求にも応答できます。NIS+ は 2 つのサービスインタフェースを提供することによってこれを実現します。1 つが NIS+ クライアントの要求に応答し、もう 1 つが NIS クライアントの要求に応答します。

このモードでは、NIS クライアントに対してさらに設定や変更を行う必要はありません。実際、NIS クライアントは、応答しているサーバーが NIS サーバーではないことを意識する必要はありません。ただし、NIS 互換モードで動作している NIS+ サーバーは ypupdate と ypxfr のプロトコルをサポートしないため、複製またはマスターの NIS サーバーとしては使用できません。NIS 互換モードの詳細は、『*NIS+ への移行*』を参照してください。

さらに 2 つの相違を指摘しておく必要があります。1 つは、NIS 互換モードでサーバーを設定する命令が標準の NIS+ サーバーの設定に使用される命令とは少し異なるということです。詳細は、『*Solaris ネーミングの設定と構成*』を参照してください。もう 1 つの相違としては、NIS 互換モードは、NIS+ の名前空間内のテーブルに対するセキュリティにも関係があります。NIS のクライアントソフトウェアは、NIS+ サーバーが NIS+ クライアントに与える資格 (credential) を提供する機能がないため、NIS クライアントからの要求はすべて「未認証」として分類されます。したがって、NIS クライアントが NIS+ テーブル内の情報にアクセスできるように、NIS+ の名前空間内のテーブルは未認証の要求に対してアクセス権を提供しなければなりません。これはパート II で説明するように、サーバーを NIS 互換モードで設定するために使用されるユーティリティによって自動的に処理されます。認証プロセスと NIS 互換モードについては、第 6 章を参照してください。

NIS+ の管理コマンド

NIS+ は、名前空間を管理するためのコマンドをフルセットで提供します。表 3–2 は、これらのコマンドの概要をまとめたものです。

表 3–2 NIS+ の名前空間管理コマンド

コマンド	説明
nisaddcred	NIS+ 主体用の資格を作成し、これらを cred テーブルに格納する
nisaddent	/etc 内のファイルまたは NIS マップからの情報を NIS+ のテーブルに追加する
nisauthconf	オプションで Diffie - Hellman 鍵の長さを設定する
nisbackup	NIS ディレクトリのバックアップコピーを取る
nis_cachemgr	NIS+ クライアント上で NIS+ キャッシュマネージャを起動する
niscat	NIS+ テーブルの内容を表示する
nis_checkpoint	ログには入力されたが、ディスクにチェックポイントが実行されていないデータについて強制的にチェックポイントを実行する
nischgrp	NIS+ オブジェクトのグループ所有者を変更する
nischmod	オブジェクトのアクセス権を変更する
nischown	NIS+ オブジェクトの所有者を変更する
nischttl	NIS+ オブジェクトの生存期間の値を変更する
nisclient	NIS+ 主体を初期化する
nisdefaults	NIS+ オブジェクトのデフォルト値 (ドメイン名、グループ名、ワークステーション名、NIS+ 主体名、アクセス権、ディレクトリ検索パス、および生存期間) を表示する
nisgrep	NIS+ テーブル内のエントリを検索する
nisgrpadm	NIS+ グループの作成または削除、あるいはそのメンバーリストを表示する。また、グループへのメンバー追加、メンバー削除、またはグループメンバーかどうかのテストを行う
nisinit	NIS+ のクライアントまたはサーバーを初期設定する

表 3–2 NIS+ の名前空間管理コマンド 続く

コマンド	説明
nisln	2 つの NIS+ オブジェクト間でシンボリックリンクを作成する
nislog	NIS+ 処理用ログの内容を表示する
nisls	NIS+ ディレクトリの内容を表示する
nismatch	NIS+ テーブル内のエントリを検索する
nismkdir	NIS+ のディレクトリを作成し、そのマスターサーバーと複製サーバーを指定する
nispasswd	NIS+ の passwd テーブルに格納されているパスワード情報を変更する (nispasswd よりは、passwd または passwd -r nisplus を使用する)
nis_ping	複製サーバーのデータをマスターサーバーのデータに強制更新する
nispopulate	新しい NIS+ ドメインに NIS+ テーブルを生成する
nisprefadm	クライアントが NIS+ サーバーから NIS+ 情報を検索する順序を指定する
nisrestore	以前にバックアップを取った NIS+ ディレクトリを復元する。新しい NIS+ 複製サーバーを急速にオンラインにするときにも使用する
nisrm	名前空間から NIS+ のオブジェクト (ディレクトリを除く) を削除する
nisrmdir	名前空間から NIS+ のディレクトリと複製を削除する
nisserver	新しい NIS+ サーバーを設定するときに使うシェルスクリプト
nissetup	org_dir ディレクトリと groups_dir ディレクトリ、および NIS+ ドメイン用の完全セットの (未生成) NIS+ テーブルを作成する
nisshowcache	NIS+ キャッシュマネージャによって管理される NIS+ 共有キャッシュの内容を表示する

表 3–2　NIS+ の名前空間管理コマンド　続く

コマンド	説明
nisstat	NIS+ サーバーに関する統計やその他の情報を報告する
nistbladm	NIS+ テーブルの作成または削除と、NIS+ テーブル内のエントリの追加、修正、または削除を行う
nistest	NIS+ 名前空間の現在の状態を報告する
nisupdkeys	NIS+ オブジェクトに格納されている公開鍵を更新する
passwd	NIS+ パスワードテーブルに保管されているパスワード情報を変更する。またパスワードの経過時間やその他のパスワード関連のパラメタを管理する

NIS+ の API

NIS+ の API (アプリケーションプログラミングインタフェース) とは、アプリケーションが NIS+ のオブジェクトへのアクセスと修正を行うために呼び出す関数群です。NIS+ の API には、9 つのカテゴリに分類される 54 種類の関数があります。

- オブジェクト操作関数 (nis_names())
- テーブルアクセス関数 (nis_tables())
- ローカル名関数 (nis_local_names())
- グループ操作関数 (nis_groups())
- アプリケーションサブルーチン関数 (nis_subr())
- その他の関数 (nis_misc())
- データベースアクセス関数 (nis_db())
- エラーメッセージ表示関数 (nis_error())
- トランザクションログ関数 (nis_admin())

第 4 章

NIS+ の名前空間

この章では、NIS+ の名前空間の構造、これをサポートするサーバー、およびこれを利用するクライアントについて説明します。

NIS+ ファイルとディレクトリ

表 4–1 は、NIS+ ファイルが格納される UNIX ディレクトリの一覧です。

表 4–1 NIS+ ファイルが存在する場所

ディレクトリ	存在するマシン	内容
/usr/bin	すべて	NIS+ ユーザーコマンド
/usr/lib/nis	すべて	NIS+ 管理者コマンド
/usr/sbin	すべて	NIS+ デーモン
/usr/lib/	すべて	NIS+ 共有ライブラリ
/var/nis/data	NIS+ サーバー	サーバーの使用できるデータファイル
/var/nis	NIS+ サーバー	NIS+ ワーキングファイル
/var/nis	NIS+ クライアントマシ	NIS+ の使用するマシン固有のデータ

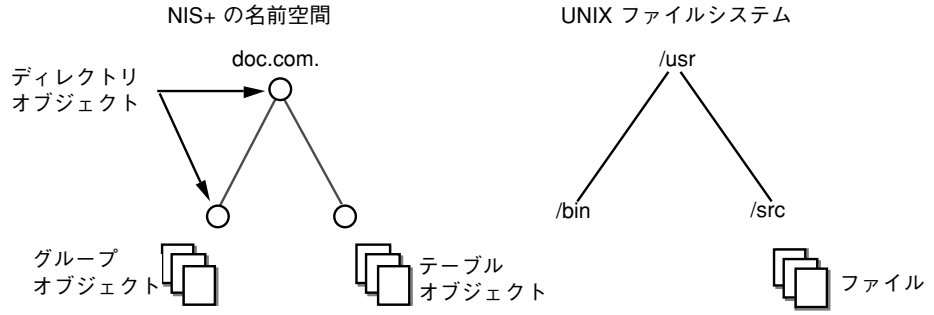
注意 - nisinit など NIS+ 設定プロシージャーによって作成された /var/nis、/var/nis/data といったディレクトリ、およびその下のファイルは、名前を変更しないでください。Solaris 2.4 以前では、/var/nis ディレクトリに *hostname*.dict、*hostname*.log という 2 つのファイルが含まれていました。またサブディレクトリ /var/nis/*hostname* もありました。Solaris 2.5 を起動すると、2 つのファイル名は trans.log、data.dict となり、サブディレクトリ名は /var/nis/data となります。ファイルの「内容」も変更されており、Solaris 2.4 以前との互換性はなくなっています。したがって、これらのファイルやディレクトリを Solaris 2.4 での名前にしてしまうと、Solaris 2.4、2.5 双方の rpc.nisd で機能しなくなりますので名前を変更しないようにしてください。

注 - Solaris オペレーティング環境では、NIS+ データディレクトリ (/var/nis/data.dict) は、マシンに依存しません。そのため、NIS+ 名を簡単に変えることができます。また、NIS+ のバックアップコピーを使用したり機能を復元したりして、NIS+ のデータをひとつのサーバーから別のサーバーに転送もできます。349ページの「バックアップと復元を使用して複製サーバーを設定する」を参照してください。

NIS+ 名前空間の構造

NIS+ の名前空間は、NIS+ によって格納された情報を配置したものです。この名前空間は、組織のニーズに合わせていろいろな方法で配置することができます。たとえば、3 つの部門を持つ組織の場合、その NIS+ の名前空間も各部門ごとに 1 つずつで 3 つの部分に分割されます。分割した各部分は、その部門のユーザー、ワークステーション、およびネットワークサービスについての情報を格納しますが、互いに通信することも簡単です。このような配置により、ユーザーによる情報へのアクセスや、管理者による情報の管理が更に容易に行えます。

NIS+ の名前空間の配置はサイトごとに異なりますが、ディレクトリ、テーブル、グループという構成はすべてのサイトが使用します。これらの構成は NIS+「オブジェクト」と呼ばれます。NIS+ オブジェクトは、UNIX のファイルシステムに似た階層形式に配置できます。たとえば、次の図を参照してください。左側の名前空間は、3 つのディレクトリオブジェクト、3 つのグループオブジェクト、および 3 つのテーブルオブジェクトから構成されています。右側の UNIX ファイルシステムは、3 つのディレクトリと 3 つのファイルから構成されています。



NIS+ の名前空間は UNIX ファイルシステムに似ていますが、重要な違いが 5 つあります。

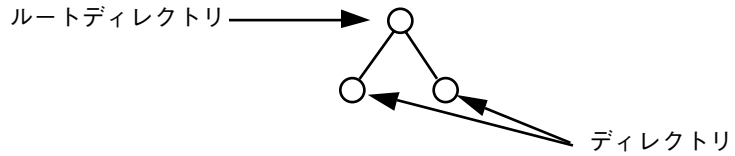
- 両方ともディレクトリを使用するが、NIS+ の名前空間でのオブジェクトはテーブルとグループであり、ファイルではない
- NIS+ の名前空間は、その目的 (システム管理ツール) のために設計された NIS+ の管理コマンド、または Solstice™ AdminSuite™

  ツールなどのグラフィカルユーザーインタフェース (GUI) を通じてだけ管理され、標準の UNIX ファイルシステムコマンドや GUI では管理できない

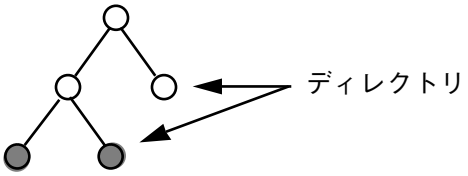
- UNIX ファイルシステム構成要素の名前はスラッシュで区切られる (/usr/bin) が、NIS+ の名前空間オブジェクト名はドットで区切られる (doc.com.)
- UNIX ファイルシステムの「ルート」へは、ディレクトリを右から左にたどっていけば到達する (/usr/src/file1) が、NIS+ の名前空間のルートは左から右にたどって到達する (sales.doc.com.)
- NIS+ オブジェクト名は、左から右に構成されているので、完全指定名は常にドットで終わる。また最後に「.」がないものは完全な名前とはみなされず、相対的な名前 (まだ上の階層があるという意味) であるとみなされる。

ディレクトリ

ディレクトリオブジェクトは、名前空間の骨格を形成します。ツリー構造に配置した場合、名前空間を別々に分けます。ディレクトリ階層を理解するには、木の根を一番上に置き、逆さまの木として見るとよく分かります。名前空間の一番上のディレクトリが「ルート」ディレクトリです。名前空間が 1 つの層だけの場合、ディレクトリは 1 つしかありませんが、そのディレクトリはルートディレクトリです。ルートディレクトリの下にあるディレクトリオブジェクトは、単に「ディレクトリ」と呼ばれます。



1 つの名前空間は複数の階層のディレクトリを持つことができます。

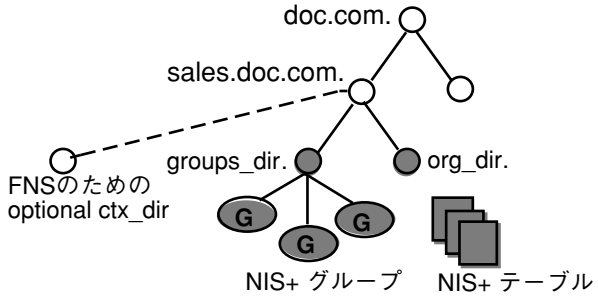


あるディレクトリと他のディレクトリの関係を考える場合、下側のディレクトリは「子」ディレクトリと呼ばれ、上側のディレクトリは「親」ディレクトリと呼ばれます。

UNIX のディレクトリは UNIX のファイルを入れるように設計されていますが、NIS+ のディレクトリは NIS+ オブジェクト (他のディレクトリ、テーブル、および

グループ) を入れるように設計されています。各 NIS+ のドメインレベルのディレクトリには、以下のサブディレクトリが含まれています。

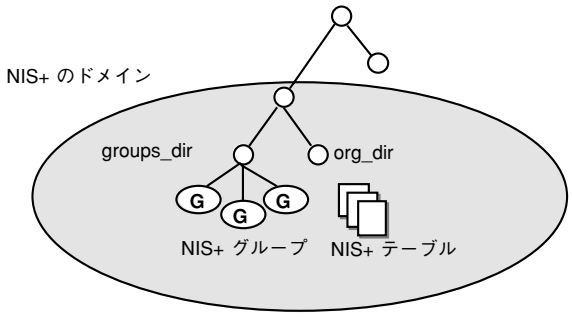
- `groups_dir` - NIS+ グループ情報を格納する
- `org_dir` - NIS+ システムテーブルを格納する
- `ctx_dir` - FNS を使用しているときだけ使用する



技術的には、ユーザーはディレクトリ、テーブル、およびグループをどんな構造にでも配置することができます。しかし、名前空間内の NIS+ のディレクトリ、テーブル、およびグループは、「ドメイン」と呼ばれる構成に配置されるのが普通です。ドメインは、名前空間の別々の部分をサポートするよう設計されています。たとえば、あるドメインが会社の営業部門 (Sales Division) をサポートし、別のドメインが製造部門 (Manufacturing Division) をサポートできます。

ドメイン

1 つの NIS+ のドメインは、ディレクトリオブジェクト、その `org_dir` ディレクトリ、その `groups_dir` ディレクトリ、および NIS+ テーブルのセットから構成されます。



NIS+ のドメインは、実際に存在する名前空間の構成要素ではありません。これらは単に、現実の組織をサポートするために使用される名前空間の「一部分」を示すための便利な方法にすぎません。

たとえば、DOC 社に営業部門と製造部門があるとします。これらの部門をサポートするため、DOC 社の NIS+ 名前空間は、以下に示すような構造によって、3 つの主要ディレクトリグループに配置されることになります。

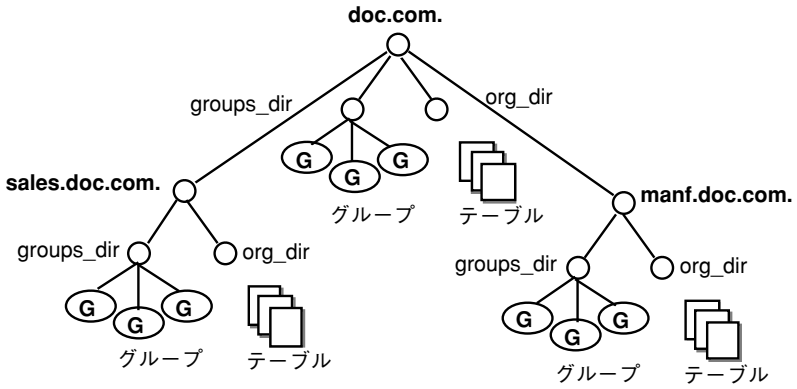


図 4–1　NIS+ ディレクトリ構造の例

このような構造を、3 つのディレクトリ、6 つのサブディレクトリ、およびいくつかの追加オブジェクトと表現するよりも、3 つのドメインと表現する方が便利です。

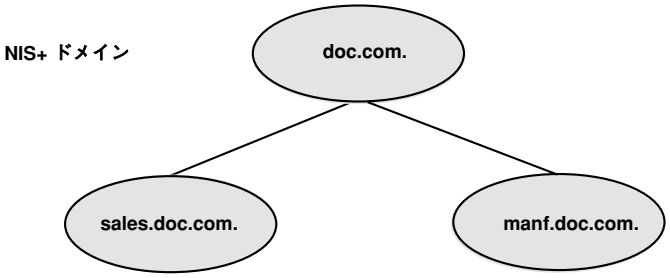
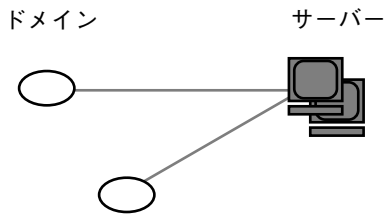


図 4–2　NIS+ ドメインの例

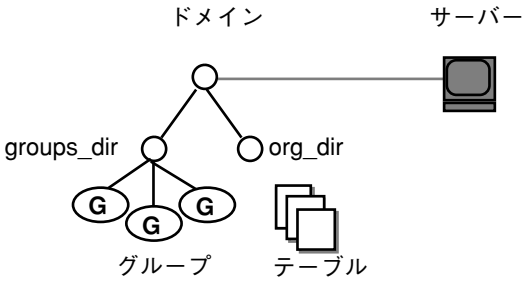
サーバー

すべての NIS+ ドメインは複数の NIS+「サーバー」によってサポートされます。サーバーは、ドメインのディレクトリ、グループ、およびテーブルを格納し、ユー

ザー、管理者、およびアプリケーションからのアクセス要求に応答します。各ドメインは、1 組の複数のサーバーだけによってサポートされます。ただし、この 1 組のサーバーは複数のドメインをサポートできます。



ドメインはオブジェクトではなく、オブジェクトの集合を意味するものであることを忘れないでください。したがって、ドメインをサポートするサーバーは、実際にはドメインにではなく、ドメインのメイン「ディレクトリ」に関連付けられています。



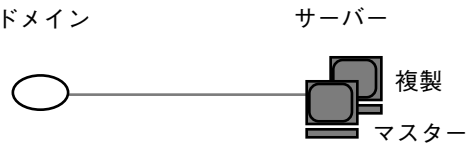
サーバーとディレクトリオブジェクトとのこうした接続は、ドメインを設定するときに行われます。操作説明についてはパート II「NIS+ の紹介と概要」で説明しますが、ここでは大切なことを 1 つ指摘しておきます。それは、この接続が確立されたときに、ディレクトリオブジェクトはそのサーバー名と IP アドレスを格納するということです。後述しますが、クライアントはこの情報を使用してサービス要求を送信します。

Solaris 8 リリースベースのワークステーションは、どれも NIS+ サーバーとなることができます。Solaris 2.4 のリリースには、NIS+ のサーバー用とクライアント用の両方のソフトウェアが入っています。したがって、Solaris 2.x がインストールされているワークステーションであれば、サーバーにもクライアントにも、あるいはその両方にもなることができます。クライアントとサーバーを区別するのは、その果たす役割です。ワークステーションが NIS+ サービスを提供している場合、それは NIS+ サーバーとして機能しています。ワークステーションが NIS+ サービスを要求している場合、これは NIS+ クライアントとして機能しています。

多数のクライアント要求にサービスを提供する必要があるため、NIS+ サーバーとして機能するワークステーションは、平均的なクライアントよりも高いコンピュー

ティング能力と多くのメモリーを装備して構成されます。そして、NIS+ データを格納する必要があるため、より大型のディスクも装備することになります。しかし、性能を高めるためのハードウェアを除いては、サーバーと NIS+ クライアントとの本質的な違いはありません。

NIS+ ドメインをサポートするのは、マスターとその複製の 2 種類のサーバーです。



ルートドメインのマスターサーバーを「ルートマスター」サーバーと呼びます。1 つの名前空間には 1 つのルートマスターサーバーしか存在しません。他のドメインのマスターサーバーは、単にマスターサーバーと呼びます。同様に、ルート複製サーバーと単なる複製サーバーがあります。

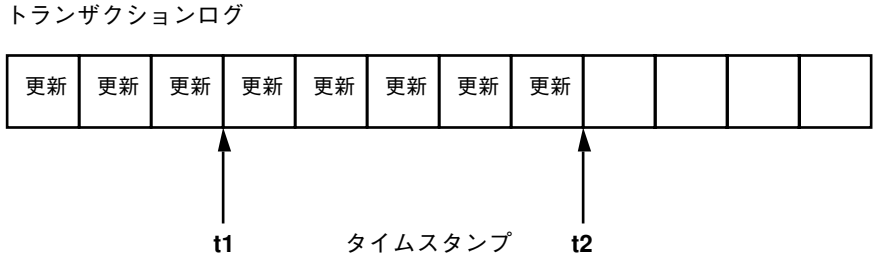
マスターサーバーと複製サーバーは、両方とも NIS+ テーブルを格納し、クライアント要求に応答します。ただし、マスターサーバーはドメインのテーブルのマスターコピーを格納し、複製サーバーは複製だけを格納します。管理者は、情報をマスターサーバー内のテーブルにロードし、マスターサーバーはそれを複製サーバーに伝達します。

この配置には 2 つのメリットがあります。第 1 に、マスターテーブルが 1 組しか存在しないことから、テーブル間の不一致を避けることができます。複製サーバーによって格納されたテーブルは、マスターのコピーにすぎません。第 2 に、これによって NIS+ サービスの「信頼性」が大幅に向上します。マスターまたはスレーブのどちらかがダウンした場合、他のサーバーがバックアップとして機能し、サービス要求に応えることができます。

サーバーが変更を伝達する方法

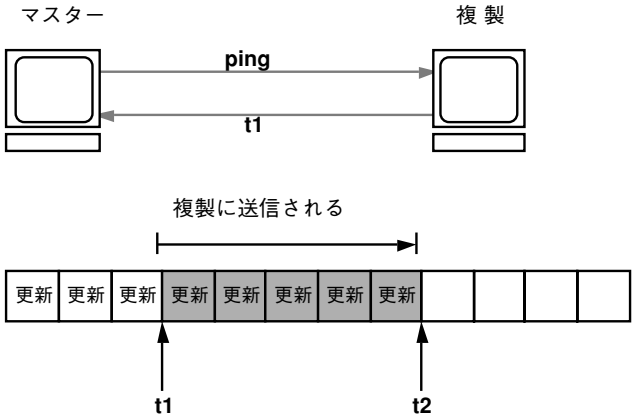
NIS+ のマスターサーバーは、そのオブジェクトをすぐに更新します。しかし、更新内容を複製サーバーに伝達する前に、複数の更新をまとめ (バッチ) ようと試みます。マスターサーバーが、ディレクトリ、グループ、リンク、またはテーブルといったオブジェクトへの更新を受信した場合、約 2 分間ほかの更新が到着しないかどうかを待機して確認します。待機時間が終了すると、これらの更新をディスクと「トランザクションログ」の 2 カ所に格納します (メモリーにはすでに更新を格納)。

マスターサーバーはトランザクションログを使用して、名前空間への変更内容が複製に伝達されるまで、これを格納します。トランザクションログには、更新とタイムスタンプという 2 つの要素が記録されます。



更新とは、変更されたオブジェクトの実際のコピーです。たとえば、ディレクトリが変更された場合、更新はそのディレクトリオブジェクトの完全なコピーです。テーブルエントリが変更された場合、更新は実際のテーブルエントリのコピーです。タイムスタンプは、マスターサーバーによって更新が行われた時間を示します。

その変更内容をトランザクションログに記録したのち、マスターは自分の複製にメッセージを送信し、送信すべき更新があることを知らせます。各複製は、マスターから受信した最後の更新のタイムスタンプでこれに応答します。するとマスターは、各複製のタイムスタンプ以降にログに記録した更新を複製に送信します。



マスターサーバーがその複製サーバーをすべて更新すると、トランザクションログをクリアします。ドメインに新しい複製が追加された場合などには、マスターは、

トランザクションログに記録されている最も早い時間のタイムスタンプよりもさらに前のタイムスタンプを複製から受け取ることがあります。この場合、マスターサーバーは全面的な「再同期」、つまり「resync」を行います。resync では、マスターに格納されているすべてのオブジェクトと情報を該当複製にダウンロードします。resync 実行中には、マスターと複製の両方がビジー状態となります。複製は要求に応答できません。マスターは、読み取り要求には応答できますが、更新要求は受け付けできません。両方とも「Server Busy - Try Again」というメッセージで応答します。

NIS+ 主体 (クライアント)

NIS+ 主体とは、NIS+ サービスに要求をするエンティティ (クライアント) のことです。

主体

一般ユーザーであってもスーパーユーザー (root) であってもログインをすれば NIS+ 主体になります。実際の要求は、前者の場合はクライアントユーザーから、後者の場合はクライアントワークステーションから送られます。つまり NIS+ 主体は、クライアントユーザーである場合と、クライアントワークステーションである場合とがあります。

また NIS+ サーバーから NIS+ サービスを提供するエンティティも、NIS+ 主体になることができます。NIS+ サーバーは、すべて NIS+ クライアントと考えることができるので、ここでの説明のほとんどは NIS+ サーバーにも当てはまります。

クライアント

NIS+ クライアントとは、NIS+ サービスを要求するように設定されたワークステーションです。NIS+ クライアントの設定では、セキュリティ資格 (credential) を設定し、クライアントを適切な NIS+ グループのメンバーとし、そのホームドメインを確認し、スイッチ構成ファイルを確認し、最後に NIS+ 初期設定ユーティリティを実行します (詳細は、パート II「NIS+ の紹介と概要」を参照)。

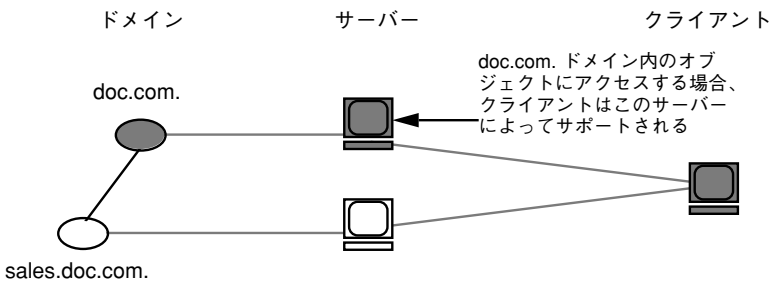
NIS+ クライアントは、セキュリティ上の制約に従って、名前空間のどの部分にもアクセスできます。つまり、認証されており、しかも適切なアクセス権が与えられている場合、名前空間内のどのドメインの情報やオブジェクトにもアクセスできます。

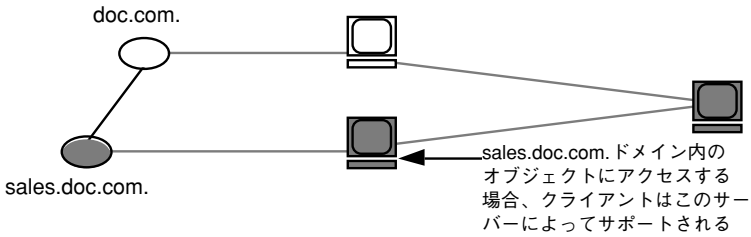
クライアントは名前空間全域にアクセスできますが、1 つのクライアントが所属するドメインは 1 つだけであり、これをその「ホーム」ドメインと呼びます。クライアントのホームドメインは、インストール時に指定されるのが普通ですが、インストール後でも変更または指定できます。クライアントの IP アドレスや資格など、クライアントについてのすべての情報は、そのホームドメインの NIS+ テーブルに格納されています。

NIS+ クライアントであることと、NIS+ テーブルに登録されていることは、微妙な違いがあります。あるワークステーションについての情報を NIS+ テーブルに入力しても、そのワークステーションが自動的に NIS+ クライアントになるわけではありません。これは単に、すべての NIS+ クライアントがそのワークステーションの情報を使用できるようにするだけです。そのワークステーションを実際に NIS+ クライアントとして設定しない限り、NIS+ サービスを要求することはできません。

逆に、あるワークステーションを NIS+ クライアントにしても、そのワークステーションについての情報は NIS+ テーブルに入力されません。これは単に、そのワークステーションで NIS+ サービスを受信できるようにするだけです。管理者がそのワークステーションについての情報を明確に NIS+ テーブルに入力しないと、他の NIS+ クライアントはその情報を入手できません。

クライアントが名前空間へのアクセスを要求した場合、実際には、クライアントは名前空間内の特定ドメインへのアクセスを要求していることになります。したがって、クライアントは、アクセスしようとしているドメインをサポートするサーバーに自分の要求を送信します。簡単に表現すると次のようになります。



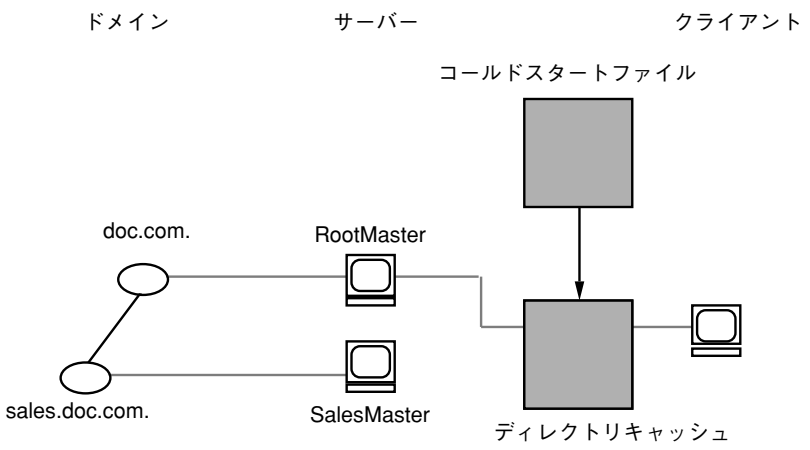


クライアントはこのサーバーを、単純な試行錯誤によって認識します。クライアントは、自分のホームサーバーから始めて、正しいサーバーが見つかるまでサーバーを 1 台ずつ試していきます。サーバーがクライアントの要求に応じられない場合、サーバーは適切なサーバーの検索に役立つ情報をクライアントに送信します。やがて、クライアントは自分の情報キャッシュを構築し、適切なサーバーをより効率的に検索できるようになります。このプロセスの詳細を次に示します。

コールドスタートファイルとディレクトリキャッシュ

クライアントが初期設定されるとき、クライアントには「コールドスタートファイル」が与えられます。コールドスタートファイルの目的は、名前空間内のサーバーと連絡をとるための起点として使用できるディレクトリオブジェクトのコピーをクライアントに提供することです。ディレクトリオブジェクトには、アドレス、公開鍵、およびそのディレクトリをサポートするマスターサーバーと複製サーバーについての情報が収められています。通常、コールドスタートファイルには、クライアントのホームドメインのディレクトリオブジェクトが収められています。

コールドスタートファイルは、クライアントの「ディレクトリキャッシュ」を初期設定するためにだけ使用されます。ディレクトリキャッシュは、「キャッシュマネージャ」と呼ばれる NIS+ 機能によって管理され、クライアントが自分の要求を適切なサーバーに送信できるようにするためのディレクトリオブジェクトを格納しています。



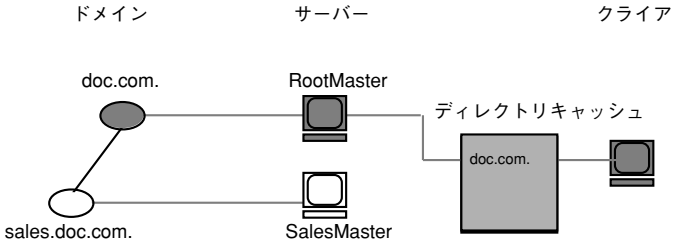
名前空間のディレクトリオブジェクトのコピーを自分のディレクトリキャッシュに格納することによって、クライアントは、どのサーバーがどのドメインをサポートするかを知ることができます。クライアントのキャッシュの内容を表示するには、`nisshowcache` コマンドを使用してください (269ページの「nisshowcache コマンド」を参照)。簡単な例を次に示します。

ドメイン名とディレクトリ名が同じ	サポートするサーバー	**IP** アドレス
`doc.com.`	rootmaster	123.45.6.77
`sales.doc.com.`	salesmaster	123.45.6.66
`manf.doc.com.`	manfmaster	123.45.6.37
`int.sales.doc.com.`	Intlsalesmaster	111.22.3.7

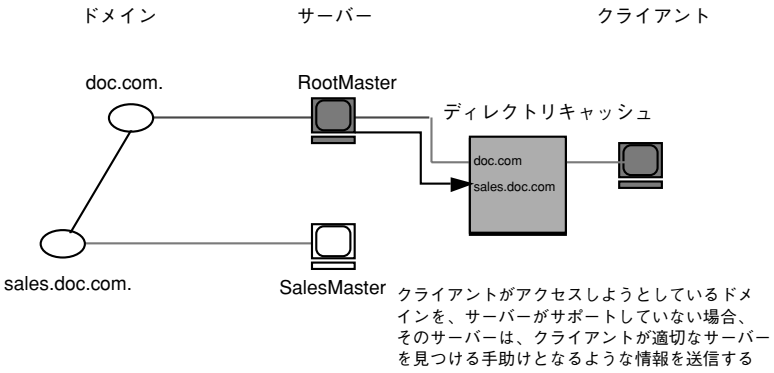
これらのコピーを最新の状態に保つため、各ディレクトリオブジェクトには「生存期間」フィールドがあります。このデフォルト値は 12 時間です。クライアントがディレクトリオブジェクトを求めてディレクトリキャッシュを調べ、最近の 12 時間以内にこれが更新されていないことを発見した場合、キャッシュマネージャはオブジェクトの新しいコピーを獲得します。275ページの「nischttl コマンド」で説明するように、ディレクトリオブジェクトの生存期間の値は、`nischttl` コマンドで変更できます。ただし、生存期間を長くするほどオブジェクトのコピーが古くなる可

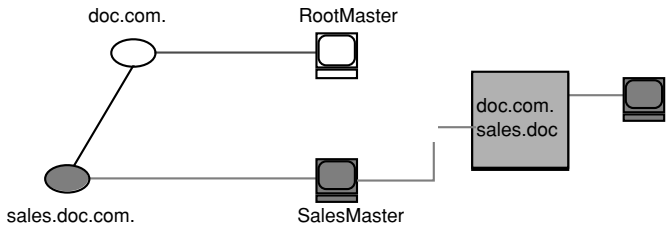
能性が高くなり、生存期間を短くするほどネットワークトラフィックとサーバーのロードが増加することに注意してください。

ディレクトリキャッシュは、これらのディレクトリオブジェクトをどうやって蓄積するのでしょうか。前に説明したように、コールドスタートファイルはキャッシュ内の最初のエントリを提供します。したがって、クライアントが最初の要求を送信するとき、コールドスタートファイルによって指定されたサーバーに要求を送信します。その要求がそのサーバーによってサポートされるドメインへのアクセスである場合、そのサーバーは要求に応答します。



その要求が別のドメイン (たとえば、`sales.doc.com.`) へのアクセスである場合、サーバーは、クライアントが適切なサーバーを探すのを助けようとします。サーバーが、自分のディレクトリキャッシュ内に該当するドメイン用のエントリを保有している場合、そのドメインのディレクトリオブジェクトのコピーをクライアントに送信します。クライアントは、その情報を今後の参照用として自分のディレクトリキャッシュにロードし、そのサーバーに要求を送信します。





ほとんどありえないことですが、クライアントがアクセスしようとしているディレクトリオブジェクトのコピーを、サーバーが保有していない場合、そのサーバーは、自分のホームドメインのディレクトリオブジェクトのコピーをクライアントに送信します。ここには、そのサーバーの親のアドレスが登録されています。クライアントは、親サーバーを使ってこのプロセスを繰り返し、適切なサーバーを見つけるか、または名前空間内の全サーバーを試し終わるまで、これを繰り返します。クライアントがドメイン内の全サーバーを試し終わった後でどうするかは、そのネームサービススイッチ構成ファイルの指示によって決まります。詳細は、第 2 章を参照してください。

やがてクライアントは、名前空間内のすべてのディレクトリオブジェクトのコピーをキャッシュ内に蓄積します。したがって、これらディレクトリオブジェクトをサポートするサーバーの IP アドレスも蓄積することになります。クライアントが他のドメインへのアクセス要求を送信する必要があるとき、通常は、自分のディレクトリキャッシュの中から自分のサーバー名を見つけ、そのサーバーにアクセス要求を直接送信できます。

NIS+ サーバーはクライアントでもある

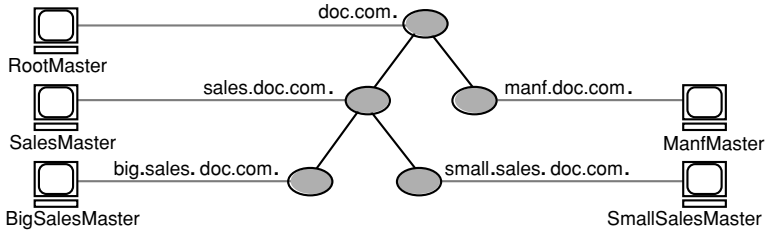
NIS+ サーバーは NIS+ クライアントでもあります。実際、ワークステーションをサーバーとして設定する (パート II「NIS+ の紹介と概要」を参照) には、その前にクライアントとして初期設定しなければなりません。唯一の例外はルートマスターサーバーであり、このサーバーには独自の設定を行う必要があります。

つまり、サーバーはドメインをサポートするだけではなく、ドメインにも「所属する」のです。つまり、クライアントになることによって、サーバーにはホームドメインがあるということです。サーバーのホスト情報は自分のホームドメインの `hosts` テーブルに格納され、その DES 資格は自分のホームドメインの `cred` テーブルに格納されます。他のクライアントと同様、サーバーは、自分のディレクトリキャッシュに記録されているサーバーに対して、サービス要求を送信します。

忘れてはならない重要な点は、ルートドメインを除いて、サーバーのホームドメインは、そのサーバーがサポートするドメインの「親」ドメインであるということです。

つまり、サーバーは 1 つのドメイン内のクライアントをサポートしますが、別のドメインの「クライアント」になるのです。ルートドメインを除いて、サーバーは自分のサポートするドメインのクライアントにはなれません。ルートドメインをサポートするサーバーには親ドメインがないため、これらはルートドメイン自体に所属します。

たとえば、次のような名前空間を考えてみます。



各サーバーがどのドメインをサポートし、どのドメインに所属するかを次に示します。

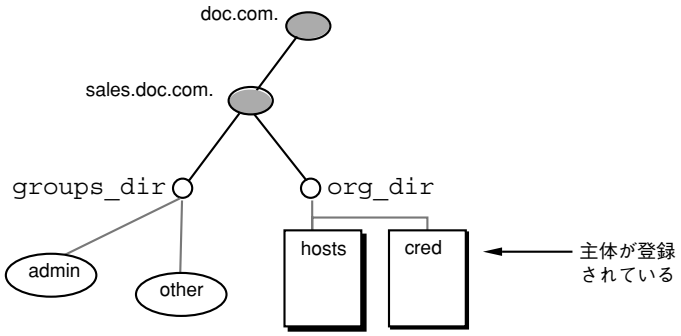
サーバー	サポートするドメイン	所属するドメイン
RootMaster	`doc.com.`	`doc.com.`
SalesMaster	`sales.doc.com.`	`doc.com.`
IntlSalesMaster	`intl.sales.doc.com.`	`sales.doc.com.`
ManfMaster	`manf.doc.com.`	`doc.com.`

命名規則

NIS+ の名前空間内のオブジェクトは、「部分指定」と「完全指定名」という 2 種類の名前によって識別できます。部分指定名は「単純」名とも呼ばれ、単なるオブジェクトの名前か、または完全指定名の一部分です。ある管理業務を行なっている間にユーザーがオブジェクトまたは主体の部分指定名を入力した場合、NIS+ はその名前を完全指定名に展開しようと試みます。詳細は、100ページの「NIS+ の名前展開」を参照してください。

完全指定名とは、名前空間内でオブジェクトを探すために必要なすべての情報を含んだオブジェクトの完全な名前です。必要なすべての情報とはオブジェクトの親ディレクトリ (もしあれば)、最後のドットも含んだ完全なドメイン名などです。

これはオブジェクトの種類によって異なるため、各タイプと NIS+ の主体ごとの規則を別個に説明します。次の名前空間を例として使用します。



NIS+ の主体を含めて、この名前空間内の全オブジェクトの完全指定名を図 4–3 に要約します。

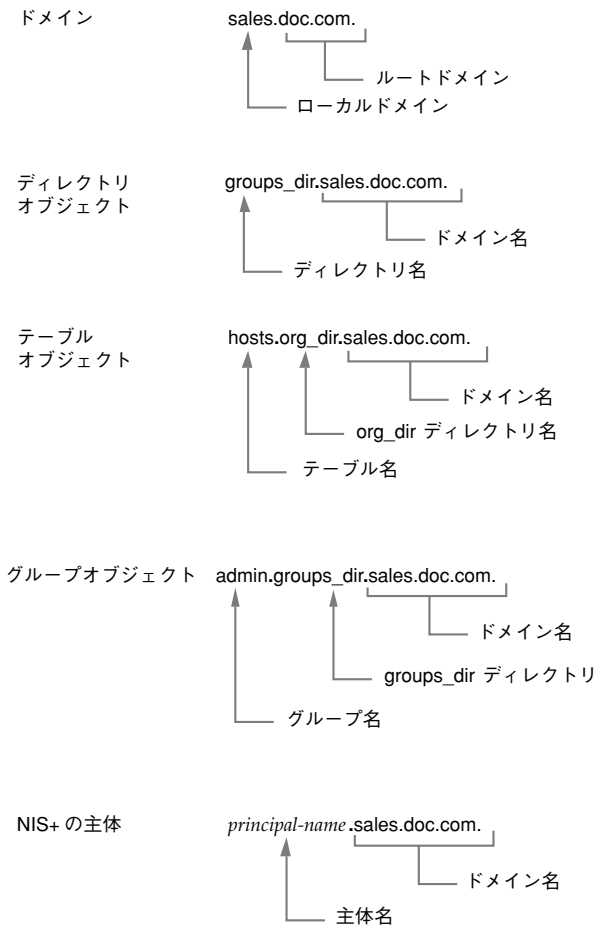


図 4–3　名前空間の構成要素の完全指定名

NIS+ ドメイン名

完全指定ドメイン名は左から右に作成され、ローカルドメインで始まってルートドメインで終わります。たとえば次のようになります。

`doc.com.` (ルートドメイン)

`sales.doc.com.` (サブドメイン)

`intl.sales.doc.com.` (サブドメイン)

最初の行はルートドメインの名前を示します。ルートドメインは、少なくとも 2 つのラベルを持ち、ドットで終わらなければなりません。最後 (右端) のラベルは自由

につけられますが、インターネット上の互換性を維持するためには、表 4–2 に示されるようなインターネットの組織名、または 2、3 文字の地域識別子 (日本であれば .jp) をつける必要があります。

表 4–2 インターネットの組織ドメイン

ドメイン	種類
com	営利団体
edu	教育機関
gov	行政機関
mil	軍事組織
net	ネットワークサポートセンター
org	非営利団体
int	国際組織

上の 2 行目と 3 行目は下位レベルドメインの名前です。

ディレクトリオブジェクト名

ディレクトリの単純名は、単にディレクトリオブジェクトの名前です。この完全指定名は、単純名にそのドメインの完全指定名を加えたものです (常に最後のドットを含む)。

groups_dir (単純名)

groups_dir.manf.doc.com. (完全指定名)

複数のディレクトリ層が 1 つのドメインを形成しないような変則的な階層を設定する場合、中間ディレクトリ名を含めるようにしてください。たとえば、次のようにします。

lowest_dir.lower_dir.low_dir.mydomain.com.

通常単純名は、同じドメイン内から使用され、完全指定名は、リモートドメインから使用されます。しかし、ドメインの NIS_PATH 環境変数に検索パスを指定することによって、リモートドメインから単純名を使用できます (100ページの「NIS+ の名前展開」を参照してください)。

テーブルおよびグループ名

完全指定されたテーブル名とグループ名は、オブジェクト名から始まり、ディレクトリ名をつけ、その後に完全指定されたドメイン名が続いて作られます。すべてのシステムテーブルオブジェクトは `org_dir` ディレクトリに格納されて、すべてのグループオブジェクトは `groups_dir` ディレクトリに格納されていることを忘れないでください。独自の NIS+ テーブルを作成した場合は、どこでも好きなところに格納できます。グループ名とテーブル名の例を次に示します。

```
admin.groups_dir.doc.com.
admin.groups_dir.doc.com.
admin.groups_dir.sales.doc.com.
admin.groups_dir.sales.doc.com.
hosts.org_dir.doc.com.
hosts.org_dir.doc.com.
hosts.org_dir.sales.doc.com.
hosts.org_dir.sales.doc.com.
```

テーブルエントリ名

NIS+ テーブル内のエントリを識別するには、その中にあるエントリとテーブルオブジェクトを識別する必要があります。このような名前を「インデックス付き」名と呼びます。この構文を次に示します。

```
[column=value,column=value,...],tablename
```

Column はテーブル列の名前です。*Value* はその列の実際の値です。*tablename* はテーブルオブジェクトの完全指定名です。`hosts` テーブルのエントリ例を次に示します。

```
[addr=129.44.2.1,name=pine],hosts.org_dir.sales.doc.com.
[addr=129.44.2.2,name=elm],hosts.org_dir.sales.doc.com.
[addr=129.44.2.3,name=oak],hosts.org_dir.sales.doc.com.
```

括弧の中には、テーブルエントリを一意に識別するために必要な最小限の列の値のペアを使用できます。

一部の NIS+ 管理コマンドは、この構文のバリエーションを利用できます。詳細は、パート II「NIS+ の紹介と概要」の `nistbladm`、`nismatch`、`nisgrep` の各コマンドの説明を参照してください。

ホスト名

ホスト名の長さは 24 文字までで、使用できるのは文字、数字、ダッシュ (-)、および下線 (_) です。大文字と小文字の区別はありません。また先頭は英文字とします。スペースは使用できません。

注 - ドット (.) を含むホスト名 (`myco.2` など) は使用できません。これは、`'myco.2'` のように引用符を使用しても同じです。ドットを使用するのは、完全指定のホスト名の一部として表記するときだけです (`myco-2.sales.doc.com` など)。

またドメイン名とホスト名が同じにならないようにします。たとえば、`sales` ドメインがある場合は、ホスト名に `sales` を使用することはできません。同様に、`home` というホスト名がある場合には、ドメイン名に `home` を使用できません。これは、サブドメインについても同様です。たとえば、すでにホスト名として `west` がある場合には、`sales.west.myco.com` というサブドメインを作ることはできません。

NIS+ 主体の名前

NIS+ の主体名は、Secure RPC のネット名とよく混同することがあります。これらの名前については、パート II「NIS+ の紹介と概要」のセキュリティに関する章で説明します。念のためここで 1 つだけ違いを指摘しておきます。NIS+ の主体名は必ずドットで終わりますが、Secure RPC のネット名はドットで終わることは絶対ありません。

表 **4–3** NIS+ の主体名

`olivia.sales.doc.com.`	NIS+ 主体名
`unix.olivia@sales.doc.com`	secure RPC ネット名

また、たとえ主体の資格が `cred` テーブルに格納されていても、`cred` テーブル名も `org_dir` ディレクトリ名も主体名には含まれません。

使用できる記号

名前空間名は、ISO ラテン 1 セットの印刷可能文字を自由に使用して作成できます。ただし、名前の先頭には次に示す文字は使用できません。`@ < > + [ ]- / = . , : ;`

文字列を使用するには、二重引用符で囲みます。名前の中に引用符を使用するには、その記号も引用符で囲みます (たとえば、`o'henry` を使用するには `o''''henry` と入力します)。John Smith などのように空白スペースを組み込むには、次のように一重引用符の中で二重引用符を使用します。

```
'''John Smith'''
```

ホスト名に適用される制限については 99ページの「ホスト名」を参照してください。

NIS+ の名前展開

NIS+ のコマンドで完全指定名を入力することは面倒です。この作業を簡単にするため、NIS+ は名前展開機能を提供します。部分指定名が入力されると、NIS+ は別のディレクトリのもとでこのオブジェクトを見つけようと試みます。最初は、デフォルトドメインから探します。これは、このコマンドを入力したクライアントのホームドメインです。デフォルトドメインでオブジェクトが見つからなかった場合、このオブジェクトが見つかるまで、NIS+ はデフォルトドメインの親ディレクトリを下から上へ探していきます。2 つのラベルしかない名前に到達すると、検索を停止します。次にその例を示します。この例では、`software.big.sales.doc.com.` ドメインに所属するクライアントにログインしたと仮定します。

環境変数 NIS_PATH

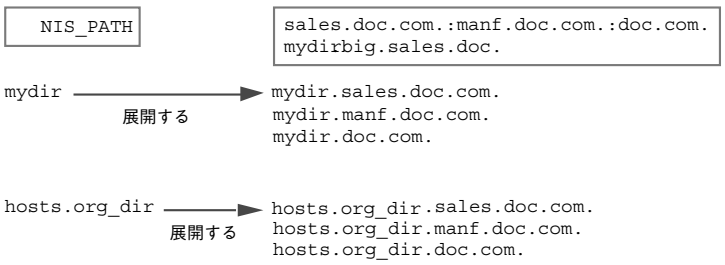
環境変数 NIS_PATH の値を変更することによって、NIS+ が検索するディレクトリのリストを変更または拡張できます。NIS_PATH には、コロンで区切って複数のディレクトリ名を設定できます。次に例を示します。

```
setenv NIS_PATH directory1:      directory2: directory3 ...
```

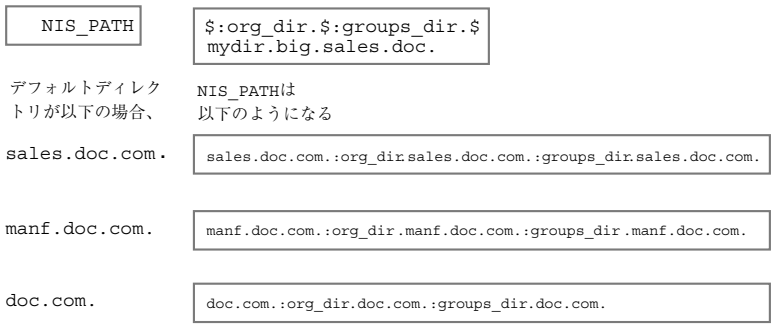
または

```
NIS_PATH=directory1:      directory2: directory3 ...;export NIS_PATH
```

NIS+ は、これらのディレクトリを左から右へ検索していきます。たとえば、次のようになります。



$PATH や $MANPATH のように、変数 NIS_PATH には特殊記号 $ を使用できます。$ 記号は、ディレクトリ名に追加するか、または単独で追加できます。ディレクトリ名に追加した場合、NIS+ はその名前にデフォルトディレクトリを追加します。たとえば、次のようになります。



$ 記号を単独で使用する (たとえば、org_dir.$:$) 場合、NIS+ は前述のような標準の名前展開を実行します。つまり、デフォルトディレクトリの検索から始め、親ディレクトリへと進めていきます。NIS_PATH のデフォルト値は $ です。

注 - NIS_PATH に追加や変更を加えると、NIS+ がより多くの検索を行わなければならないので、性能が低下するという点に注意してください。

第 5 章

NIS+ のテーブルと情報

この章では、これらのテーブルの構造と設定方法の概要について説明します。

NIS+ テーブルの構造

NIS+ は、さまざまなネットワーク情報をテーブルに格納します。NIS+ テーブルには、単純なテキストファイルやマップには見られない機能がいくつかあります。これらのテーブルは列エントリ構造をもち、検索パスを使用することや、リンクすることができ、しかも複数の設定方法があります。NIS+ にはあらかじめ構成された 16 個のシステムテーブルがあり、独自のテーブルを作成することもできます。表 5–1 は、事前構成した NIS+ テーブルを示します。

表 5–1 NIS+ のテーブル

テーブル	テーブル内の情報
`hosts`	ドメイン内の全ワークステーションのネットワークアドレスとホスト名
`bootparams`	ドメイン内の全ディスクレスクライアントのルート、スワップ、およびダンプパーティションの位置
`passwd`	ドメイン内の全ユーザーに関するパスワード情報

表 5–1　NIS+ のテーブル　続く

テーブル	テーブル内の情報
cred	ドメインに所属する主体の資格
group	ドメイン内の全 UNIX グループのグループ名、グループパスワード、グループ id、およびメンバー
netgroup	ドメイン内のワークステーションやユーザーが所属できるネットグループ
mail_aliases	ドメイン内のユーザーのメール別名に関する情報
timezone	ドメイン内の全ワークステーションの時間帯
networks	ドメイン内のネットワークとこれらの標準の名前
netmasks	ドメイン内のネットワークとこれらに関連付けられているネットマスク
ethers	ドメイン内の全ワークステーションの Ethernet アドレス
services	ドメイン内で使用される IP サービス名とそれらのポート番号
protocols	ドメイン内で使用される IP プロトコルのリスト
RPC	ドメイン内で使用可能な RPC サービスの RPC プログラム番号
auto_home	ドメイン内の全ユーザーのホームディレクトリの位置
auto_master	自動マウンタのマップ情報

cred テーブルには NIS+ のセキュリティに関係する情報だけが入っています。これについては第 7 章で説明します。

これらのテーブルには、ユーザー名からインターネットサービスまでの幅広い情報が格納されています。この情報の多くは、設定時または構成の手順の中で生成されます。たとえば、password テーブル内のエントリは、ユーザーアカウントが設定されたときに作成されます。hosts テーブル内のエントリは、ワークステーションがネットワークに加えられたときに作成されます。また、networks テーブル内のエントリは、新しいネットワークが設定されたときに作成されます。

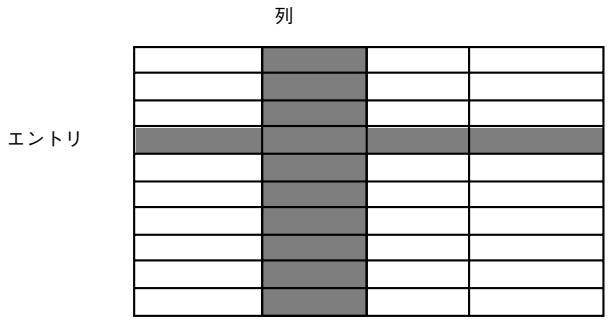
この情報はこのような広範囲にわたる操作によって生成されるため、このマニュアルでは詳細を説明していません。ただし、テーブルの各列に含まれる情報については付録 C で要約し、NIS+ のグループとネットグループとを区別する場合など、混乱を避けるために必要な場合にだけ詳細を示します。テーブルの情報の詳細は、Solaris の「システム管理マニュアルセット」と「ネットワーク管理マニュアルセット」を参照してください。

注 - ドメインにさらに多くのオートマウントマップを作成できますが、必ず NIS+ テーブルとして格納し、`auto_master` テーブルの中に入れてください。オートマウントマップを作成して、ユーザー用に作成された `auto_master` に追加するときは、列名にはキーと値が必要です。オートマウントの詳細は、オートマウントまたは NFS ファイルシステムの関連マニュアルを参照してください。

注 - ネームサービスとして、NIS+ テーブルはオブジェクト自体ではなく、オブジェクトの参照情報を格納するように設計されています。そのため、NIS+ はエントリの大きなテーブルをサポートできません。テーブルに非常に大きなエントリがある場合、`rpc.nisd` は機能しません。

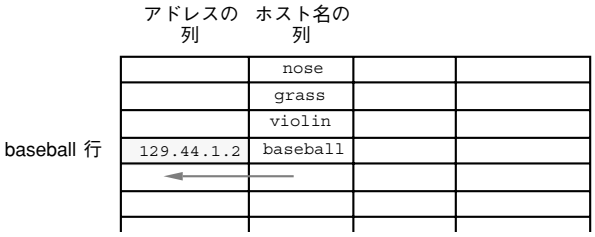
列とエントリ

NIS+ テーブルにはさまざまな種類の情報が格納されていますが、これらの基本構造はすべて同じで、行と列から構成されます。行は「エントリ」または「エントリオブジェクト」と呼ばれます。



クライアントは、キーだけではなく、任意の検索可能な列によっても情報にアクセスできます。たとえば、baseball という名前のワークステーションのネットワークアドレスを見つけるには、ホスト名の列を探して baseball を見つけます。

次に、baseball のエントリ行を移動して、そのネットワークアドレスを見つけます。



クライアントは、任意のレベルでテーブル情報にアクセスできる (つまり、テーブルに全体としてアクセスできる) ため、NIS+ は 3 つのレベルすべてにセキュリティ機構を提供しています。たとえば、管理者は、オブジェクトレベルで全員にテーブルの読み取り権を割り当て、列レベルで所有者に変更権を割り当て、エントリレベルでグループに変更権を割り当てることができます。テーブルのセキュリティの詳細は、第 10 章で説明しています。

検索パス

テーブルには、その「ローカル」ドメインについての情報だけが収められています。たとえば、doc.com. ドメイン内のテーブルには、doc.com. ドメインのユーザー、クライアント、およびサービスについての情報だけが収められています。sales.doc.com. ドメイン内のテーブルには、sales.doc.com. ドメイン

のユーザー、クライアント、およびサービスについての情報だけが収められています。

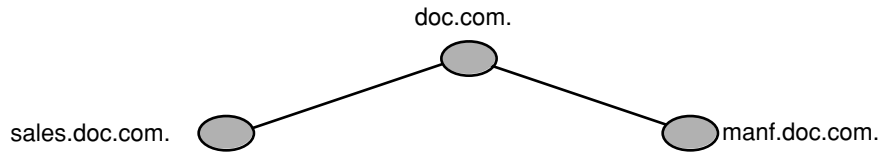
あるドメイン内のクライアントが、別のドメインに格納されている情報を見つけようとした場合、完全指定名を使用しなければなりません。100ページの「NIS+ の名前展開」で説明したように、環境変数 NIS_PATH が正しく設定されていれば、NIS+ サービスがこれを自動的に処理します。

情報を探すときにサーバーがたどる「検索パス」をどの NIS+ テーブルでも指定できます。この検索パスは、NIS+ テーブルをコロンで区切って順番に並べたものです。

table:*table*:*table*...

検索パス内のテーブル名は、完全指定をする必要はありません。テーブル名は、コマンド行で入力された名前と同じように展開されます。サーバーが自分のローカルテーブル内に情報を検出できなかった場合、サーバーはクライアントにテーブルの検索パスを返します。クライアントはそのパスを使用して、その情報が見つかるかまたは最後まで、検索パスに名前があるすべてのテーブルを順番に探していきます。

検索パスのメリットを次に示します。次のようなドメイン階層があるとします。



下位にある 3 つのドメインの hosts テーブルの内容は次のようになります。

表 5–2 hosts テーブルの例

sales.doc.com.		manf.doc.com.	
127.0.0.1	localhost	127.0.0.1	localhost
111.22.3.22	luna	123.45.6.1	sirius
111.22.3.24	phoebus	123.45.6.112	rigel

表 **5–2** hosts テーブルの例 続く

sales.doc.com.		manf.doc.com.	
111.22.3.25	europa	123.45.6.90	antares
111.22.3.27	ganymede	123.45.6.101	polaris
111.22.3.28	mailhost	123.45.6.79	mailhost

ここで、sales.doc.com. ドメイン内の luna にログインしたユーザーが、別のクライアントにリモートログインする場合を考えてみます。このユーザーが完全指定名を指定しない場合、そのユーザーがリモートログインできるのは localhost、phoebus、europa、ganymede、および mailhost の 5 台のワークステーションだけです。

sales.doc.com. ドメインにある hosts テーブルの検索パスが、次のように、manf.doc.com. ドメインの hosts テーブルを指定していると仮定します。

```
hosts.org_dir.manf.doc.com.
```

これで sales.doc.com. ドメイン内のユーザーは、たとえば、rlogin sirius と入力することができ、NIS+ サーバーはこれを検出します。NIS+ サーバーはまずローカルドメインで sirius を探し、見つからなければ manf.doc.com. ドメイン内を探します。クライアントは manf.doc.com. ドメインをどのように認識するのでしょうか。第 4 章で説明したように、この情報はディレクトリキャッシュに格納されています。これがディレクトリキャッシュにない場合、クライアントは、第 4 章で説明したプロセスに従ってこの情報を入手します。

ただし、検索パスを指定する方法には、若干の欠点があります。ユーザーがたとえば、rlogin luba (luna でなく) などと間違った名前を入力した場合、サーバーは 1 つのテーブルではなく、3 つのテーブルを探さなければエラーメッセージを返せません。名前空間の全域に検索パスを設定した場合、1 つの操作は 2、3 のドメインではなく、10 個のドメイン内のテーブルを検索してから終了します。もう 1 つの欠点は、多数のクライアントが NIS+ テーブルにアクセスする必要があるとき、複数のサーバーのセットと通信するために、性能低下を招くという点です。

また、mailhost は別名として使用されることが多いため、特定のメールホストについての情報を探したいときは、その完全指定名 (mailhost.sales.doc.com.

など)を使用しなければなりません。そうしないと、NIS+ は検索したすべてのドメインで見つけたすべてのメールホストを返します。

テーブルの検索パスを指定するには、280ページの「nistbladm コマンド」で説明するように、nistbladm コマンドに –p オプションを付けて実行します。

テーブルの設定方法

NIS+ テーブルの設定には 3 つまたは 4 つの作業が伴います。

1. org_dir ディレクトリの作成
2. システムテーブルの作成
3. システムテーブル以外のテーブルの作成 (必要に応じて)
4. 情報を入力してテーブルを生成 (populate) する

第 4 章で説明したように、NIS+ のシステムテーブルは org_dir ディレクトリに格納されます。したがって、テーブルを作成するには、その前にテーブルを入れる org_dir ディレクトリを作成しなければなりません。これには 3 つの方法があります。

- nisserver スクリプトを使用。nisserver スクリプトは、ディレクトリとシステムテーブのルフルセットを作成します。nisserver スクリプトを実行することを推奨します。
- nismkdir コマンドを使用。nismkdir コマンドは単にディレクトリを作成するものです。
- /usr/lib/nis/nissetup ユーティリティを使用。nissetup ユーティリティは org_dir および groups_dir ディレクトリとシステムテーブルのフルセットを作成します。

nisserver スクリプト、nissetup および nismkdir ユーティリティについては、『*Solaris* ネーミングの設定と構成』に説明があります。nismkdir コマンドに関するその他の情報については、257ページの「nismkdir コマンド」に説明があります。

nissetup ユーティリティのメリットは、サーバーが NIS 互換モードで動作しているドメインのテーブルに対して、適切なアクセス権を割り当てる機能です。–Y フラグを付けて実行すると、このユーティリティは作成するオブジェクトの「未認証」カテゴリに読み取りアクセス権を割り当てます。その結果、未認証の NIS クライアントは、そのドメインの NIS+ テーブルから情報を得ることができます。

NIS+ の 16 個のシステムテーブルと、これらに格納される情報の種類については、付録 C で説明しています。これらを作成するには、前述の 3 つの方法を使用できます。nistbladm コマンドは NIS+ テーブルの作成と変更を行います。nistbladm コマンドで名前空間内のすべてのテーブルを作成できますが、入力量が多くなり、列の正確な名前やアクセス権を知っている必要があります。nisserver スクリプトを使う方が、はるかに簡単です。

システムテーブル以外のテーブル、つまり NIS+ によってまだ構成されていないテーブルを作成するには、nistbladm コマンドを使用します。オートマウントマップを追加する場合は、キーの列と値の列が必要です。

NIS+ テーブルを生成するには、NIS マップから行う方法、ASCII ファイル (/etc 内のファイルなど) から行う方法、および手作業の 3 つの方法があります。

NIS サービスからアップグレードしている場合、すでにネットワーク情報の大部分が NIS マップに格納されています。この情報を NIS+ テーブルに手作業で再入力する必要はありません。nispopulate スクリプトか nisaddent ユーティリティで自動的に転送できます。

他のネットワーク情報サービスを使用していない場合で、しかもネットワークデータを /etc 内のファイルで管理しているときも、この情報を再入力する必要はありません。nispopulate スクリプトか nisaddent ユーティリティを使用すれば自動的に転送できます。

ネットワークを初めて設定する場合、多くのネットワーク情報をすでにどこかに格納しているということはおそらくないでしょう。この場合、まず情報を入手して、次に NIS+ テーブルに手作業で入力する必要があります。これを行うには、nistbladm コマンドを使用できます。また、特定のテーブルの全情報を「入力ファイル」(これは本質的に /etc 内のファイルと同じです) に入力し、次に、nispopulate スクリプトか nisaddent ユーティリティを使用してファイルの内容を転送することによっても同じことが行えます。

テーブルの更新方法

ドメインが設定されると、サーバーはそのドメインの NIS+ テーブルの最初のバージョンを受け取ります。これらのバージョンはディスクに格納されますが、サーバーは起動されると、これらをメモリーにロードします。サーバーがテーブルに対する更新を受信すると、サーバーはそのメモリーにあるテーブルをすぐに更新します。サーバーが情報の要求を受信すると、その応答のためにメモリーにあるコピーを使用します。

もちろん、サーバーはその更新をディスクにも格納する必要があります。ディスクにあるテーブルを更新するには時間がかかるため、NIS+ の全サーバーはテーブル用の「ログ」ファイルを持っています。このログファイルは、テーブルに対して行われた変更内容を、ディスク上で更新できるまで、一時的に格納するように設計されています。ログファイルは、テーブル名を接頭辞として使用し、これに `.log` を追加します。たとえば、次のようになります。

```
hosts.org_dir.log
bootparams.org_dir.log
password.org_dir.log
ipnodes.org_dir.log
```

ログファイルが大きくなりすぎて大量のディスク領域を占有しないように、ディスクにあるテーブルのコピーは毎日更新しなければなりません。このプロセスは「チェックポイントを実行する」と呼ばれます。これには、`nisping –C` コマンドを使用します。

第 6 章

セキュリティの概要

この章では、NIS+ のセキュリティシステムと、システムが NIS+ 名前空間全体に与える影響について説明します。

Solaris のセキュリティ - 概要

Solaris のセキュリティは、ユーザーが Solaris 環境に入るために通過しなければならないゲートと、内部でそのユーザーが何をできるのかを判定するアクセス権マトリックスの 2 つによって確保されています。(図 6–1 参照)

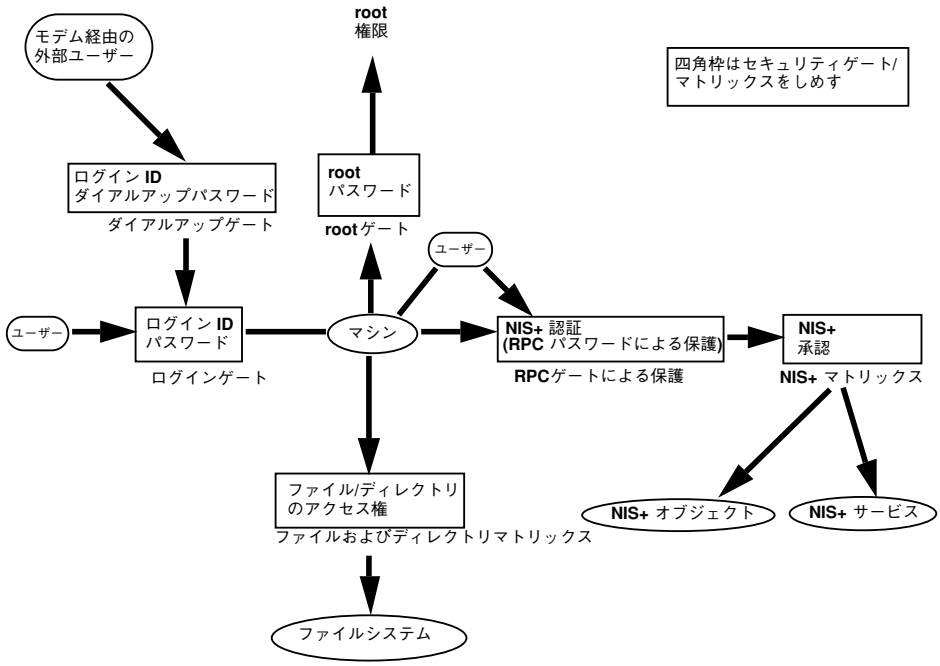


図 6–1　Solaris のセキュリティゲートとフィルタ

図 6–1 に示すように、4 つのゲートと 2 つのアクセス権マトリックスによってシステム全体が構成されています。

- 「ダイアルアップゲート」

  モデムや電話回線により Solaris 環境にアクセスするには、正しいログイン ID とダイアルアップパスワードが必要です。

- 「ログインゲート」

  Solaris 環境に入るには、正しいログイン ID とユーザーパスワードが必要です。

- 「ファイルおよびディレクトリマトリックス」

  Solaris 環境に入ってから、ファイルやディレクトリに関して、読み取り、実行、変更、作成、削除等の各種操作を行う権限はアクセス権マトリックスによってコントロールされます。

- 「root ゲート」

  root 権限にアクセスするには、正しいスーパユーザー (root) パスワードが必要です。

- 「RPC ゲートによる保護」

セキュリティレベル 2 (デフォルト値) の NIS+ 環境においては、NIS+ サービスを使って NIS+ オブジェクト (サーバー、ディレクトリ、テーブル、テーブルエントリなど) にアクセスしようとする場合、ユーザー ID は SECURE RPC プロセスを使った NIS+ によって確認されます。

ダイヤルアップ、ログイン、ルートゲート、およびファイルとディレクトリのアクセス権マトリクスの詳細は、Solaris の各マニュアル及び資料を参照してください。

SECURE RPC ゲートをパスするには、SECURE RPC パスワードが必要です。「SECURE RPC パスワード」は、「ネットワークパスワード」と呼ばれる場合もあります。SECURE RPC パスワードとログインパスワードは通常同一なので、ゲートをパスした場合はパスワードを再入力する必要がありません (2 つのパスワードが異なった場合についての説明は、141ページの「Secure RPC パスワードとログインパスワードの問題」 を参照)。

ユーザー要求が SECURE RPC ゲートをパスするには、1 組の「資格」セットが使われます。ユーザーの資格を、作成、提示、判定するプロセスを、そのプロセスがユーザーが誰であり、正しい RPC パスワードを所持しているか否かを確認することから、「認証 (authentication)」と呼びます。この認証プロセスは、ユーザーが NIS+ サービスを要求するごとに自動的に実行されます。

NIS 互換モード (YP 互換モードでもある) の NIS+ 環境では、ユーザーが正しい資格を所持しているか否かにかかわらず (認証プロセスによって、ID が確認され、SECURE RPC パスワードがチェックされたか否かにかかわらず)、誰でもすべての NIS+ オブジェクトを読み取ることができ、またそれらエントリを変更できるので、SECURE RPC ゲートによるガードは極めて弱くなります。誰もがすべての NIS+ オブジェクトを読み取りかつそれらエントリを変更できるので、互換モードの NIS+ ネットワークは通常モードのよりも安全性が低くなります。

NIS+ の認証と資格の作成および管理については、第 7 章を参照してください。

- 「NIS+ オブジェクトマトリックス」

  一度正しく認証されると、ユーザーが NIS+ オブジェクトを読み取り、変更、作成、削除する権限はアクセス権マトリックスに管理されます。このプロセスを NIS+ の「承認 (authorization)」と呼びます。

  NIS+ のアクセス権および承認の詳細は、第 10 章を参照してください。

NIS+ のセキュリティ - 概要

NIS+ のセキュリティは NIS+ の名前空間全体にかかわっています。セキュリティと名前空間を別に設定することはできません。このため、セキュリティの設定方法は名前空間の各構成要素を設定する手順と密接に関係することになります。一度 NIS+ のセキュリティ環境を設定すると、その後はユーザーの追加および削除、アクセス権の変更、グループメンバーの再割当やネットワーク展開を管理するために必要なその他すべての管理ルーチンタスクを実行できます。

NIS+ のセキュリティ機能は名前空間内の情報と、名前空間そのものの構造を不正アクセスから保護するものです。このセキュリティ機能がなければ、どんな NIS+ クライアントであっても名前空間に保存された情報を獲得することも変更することもできないだけでなく、ダメージを与える可能性があります。

NIS+ セキュリティには次の 2 つの機能があります。

- 「認証 (authentication)」

  認証は NIS+ 主体を識別するのに使われます。主体 (ユーザーまたはマシン) が NIS+ オブジェクトにアクセスしようとするときはいつも、ユーザー ID と SECURE RPC パスワードが確認されチェックされます。

- 「承認 (authorization)」

  承認はアクセス権を識別するのに使われます。NIS+ 主体が NIS+ オブジェクトにアクセスしようとするときはいつも、所有者、グループ、その他、未認証の 4 つのクラスの 1 つに位置付けられています。NIS+ セキュリティシステムは NIS+ 管理機能により、各クラスに NIS+ オブジェクトに対する読み取り、変更、作成、削除の各権限を指定します。たとえば、あるクラスは `passwd` テーブルの特定列を変更できるが、その列を読み取ることはできないとか、別のクラスはいくつかのエントリを読み取ることはできるが、それ以外はできないというように設定できます。

NIS+ のセキュリティ機能は 2 段階のプロセスを用意しています。

1. 「認証 (authentication)」

   NIS+ は資格 (credential) を使ってユーザーを確認します。

2. 「承認 (authorization)」

   承認プロセスがユーザー ID を確認すると、NIS+ はクラスを判定します。NIS+ オブジェクトまたはサービスに対して何が行えるのかは、ユーザーがどのクラスに属しているかに依存します。これは UNIX の標準的なファイルおよびディレク

トリのアクセス権システムと同様の考え方に基づいています (クラスの詳細は、122ページの「承認クラス」を参照)。

このプロセスは、マシン A のルート権限により su コマンドを使って、第 2 の NIS+ アクセス権で NIS+ オブジェクトにアクセスすることを防ぐものです。

しかしながら NIS+ は、他人のユーザーログインパスワードを知っている者が、そのユーザーになりすましてユーザー ID と NIS+ アクセス権を使用することを妨げることはできません。またルート権限を持つユーザーが同一マシンからログインしている別のユーザーになりすますのを防ぐこともできません。

図 6–2 に、このプロセスの詳細を示します。

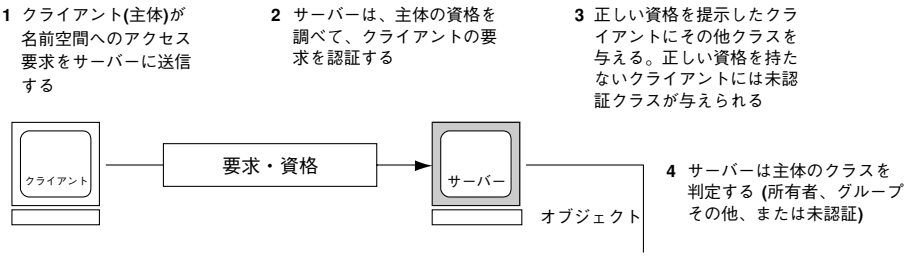


図 6–2　NIS+ セキュリティプロセスの概要

NIS+ の主体について

NIS+ の主体とは、NIS+ サービス要求を依頼するエンティティ (クライアント) です。NIS 主体には、クライアントマシンに一般ユーザーとしてログインしたユーザー、スーパーユーザーとしてログインしたユーザー、NIS+ クライアントマシンにおいてスーパーユーザー特権で動作しているプロセスを含みます。つまり NIS+ の主体はクライアントユーザーの場合もクライアントワークステーションの場合もあります。

NIS+ の主体は、NIS+ サーバーから NIS+ サービスを提供する主体を指すこともあります。すべての NIS+ サーバーは NIS+ クライアントにもなるので、サーバーが NIS+ の主体となる場合もあります。

NIS+ のセキュリティレベルについて

NIS+ サーバーは 2 つのセキュリティレベルの 1 つで動作できます。これらのレベルが、主体がその要求の認証を受ける際に提示しなければならない資格種類を判定

します。NIS+ は最も高度のセキュリティレベル (レベル 2) で動作するように設計されています。レベル 0 はテスト、設定、デバッグ用です。セキュリティレベルについては表 6–1 にまとめてあります。

表 6–1 NIS+ セキュリティレベル

セキュリティレベル	説明
0	セキュリティレベル 0 は、NIS+ 名前空間の初期テストと初期設定を行うレベル。セキュリティレベル 0 で動作している NIS+ サーバーは、ドメイン内の全 NIS+ オブジェクトに対する完全なアクセス権をどの NIS+ 主体にも与える。レベル 0 はシステム管理者だけが設定管理のためにだけ使用する。レギュラーユーザーはネットワークにおける通常のオペレーションに使用できない
1	セキュリティレベル 1 は AUTH_SYS セキュリティを利用する。このレベルは NIS+ のサポートを受けられないため、利用すべきではない
2	セキュリティレベル 2 はデフォルトで、NIS+ が現在提供している最高レベルのセキュリティ。これは DES 資格を持たない要求には未認証クラスが与えられ、未認証クラスに割り当てられたアクセス権が与えられる。無効な DES 資格を使用する要求だけを認証する。資格を使用する要求は再試行される。有効な DES 資格獲得に繰り返し失敗すると、無効資格を持った要求として承認エラーになる (主体が原因となって資格が無効になる場合、その原因として要求をした主体に対してそのマシンで `keylogin` が実行されていなかったり、クロックの同期がとれていなかったり、鍵が不一致であるなどの原因が考えられる)。

セキュリティレベルとパスワードコマンド

Solaris リリース 2.0 から 2.4 の環境では、`nispasswd` を使用してパスワードを変更します。ただし、資格がなければ、`nispasswd` は機能しません (より上位のレベルでの資格があった場合を除き、セキュリティレベル 0 以下では機能しない)。Solaris リリース 2.5 以降の場合は、セキュリティレベルや資格ステータスにかかわらず、`passwd` コマンドを使ってパスワードを変更できます。

NIS+ の認証と資格 - 紹介

NIS+ 資格の目的は、NIS+ サービスや NIS+ オブジェクトへのアクセスを要求する各主体の ID を「認証」(確認) することにあります。つまり、NIS+ 資格および承認プロセスは SECURE RPC システムを実装することです。

資格および認証システムは他人になりすますのを防ぐシステムです。あるマシンのルート権限を持つユーザーが `su` コマンドを使って、ログインしていないもしくは別のマシンにログインしている第 2 のユーザーになりすまし、NIS+ アクセス権を使用して NIS+ オブジェクトにアクセスすることを妨げます。

サーバーが主体を認証すると、主体は 4 つの認証クラスのどれかに置かれ、次にサーバーは承認されている動作を知るためにアクセスしようとする NIS+ オブジェクトをチェックします。承認の詳細は、122ページの「NIS+ の承認とアクセス - 紹介」を参照してください。

ユーザーおよびマシンの資格

主体には、「ユーザー」と「マシン」の 2 つの基本的な種類があります。

- 「ユーザー (user) 資格」。レギュラーユーザーとして NIS+ クライアントにログインした場合、NIS+ サービス要求には当人の「ユーザー」資格を含みます。
- 「マシン (machine) 資格」。スーパーユーザーとして NIS+ クライアントにログインした場合、サービス要求は「クライアントワークステーション」資格を使います。

DES と LOCAL 資格

NIS+ 主体には DES と LOCAL の 2 つの資格が用意されています。

DES 資格 (credential)

注 - 認証を行う仕組みとしては DES 資格が唯一のものです。将来は他の仕組みも利用できるようになるかもしれません。DES 資格と NIS+ 資格は同等のものではありません。

このマニュアルでは、DES 資格という用語は、鍵の長さにかかわりなく Diffie - Hellman 鍵をベースにした認証を表すものとして一般的に使われています。鍵の長さはあらかじめ決められたセットから指定することができます。Diffie - Hellman 鍵の長さの設定や表示を行うには `nisauthconf(1M)` を使用します。

DES (データ暗号化規格) 資格は資格の 1 種であり、保護認証を提供するものです。このマニュアルで NIS+ 主体を認証するために資格をチェックすると記述している場合は、NIS+ がチェックしている DES 資格のことを意味します。

主体が NIS+ サービスを要求したり、NIS+ オブジェクトにアクセスするごとに、ソフトウェアは格納してある資格情報を使って主体の資格を作成します。DES 資格は NIS+ 管理機能によって各主体に作られた情報から作成されます。第 7 章を参照してください。

- 主体の DES 資格の正当性を NIS+ が確認すると、その主体は「認証」されます。
- 主体は所有者、グループ、その他のいずれかの承認クラスを得るために認証されなければなりません。つまり、これらクラスの 1 つを得るために、有効な DES 資格を持つ必要があるのです。有効な DES 資格を持たない主体は自動的に未認証クラスに割り当てられます。
- 当該主体がクライアントユーザーであるかクライアントワークステーションであるかにかかわらず、DES 資格情報は常に主体のホームドメインの `cred` テーブルに格納されます。

LOCAL 資格

LOCAL 資格は、ユーザー ID 番号とホームドメイン名を含む NIS+ 主体名とのマップです。ユーザーがログインすると、システムは DES 資格が格納されているホームドメインを特定する LOCAL 資格をチェックします。システムはその情報を使ってユーザーの DES 資格情報を獲得します。

ユーザーがリモートドメインにログインした場合、その要求はユーザーのホームドメインを示す LOCAL 資格を使います。NIS+ は次にユーザーの DES 資格情報を知るためにユーザーのホームドメインを問い合わせます。こうして、たとえユーザー

の DES 資格情報がリモートドメインに格納されていなくても、リモートドメインで認証されます。

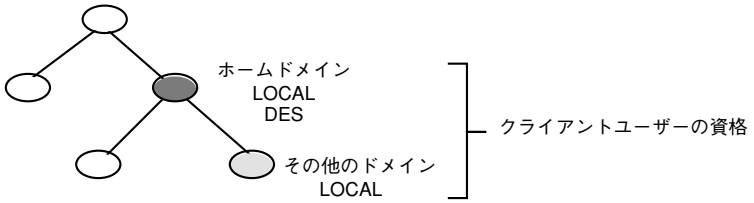


図 6–3　資格とドメイン

LOCAL 資格情報はどのドメインにも格納できます。実際、リモートドメインにログインし認証されるためには、クライアントユーザーはリモートドメインの `cred` テーブルに LOCAL 資格を持っている必要があります。ユーザーが、アクセスしようとするリモートドメインに LOCAL 資格を持っていなかった場合、NIS+ はユーザーの DES 資格を獲得するためにユーザーのホームドメインに入ることができなくなります。そのような場合、ユーザーは認証されず未認証クラスを与えられることになります。

ユーザー種類と資格種類

ユーザーは両方の資格を持つことができますが、マシンは DES 資格しか持てません。

すべてのマシンのルート UID は常に 0 であるため、ルートは他のマシンに対して NIS+ アクセスできません。もしマシン A のルート (UID=0) がマシン B にアクセスしようとすると、すでに存在しているマシン B のルート (UID=0) と衝突します。これでは、クライアント「ワークステーション」の LOCAL 資格が意味をなさないのでクライアント「ユーザー」にだけ認められています。

表 6–2　資格のタイプ

資格のタイプ	クライアントユーザー	クライアントワークステーション
DES	Yes	Yes
LOCAL	Yes	No

表 6–2　資格のタイプ　続く

NIS+ の承認とアクセス - 紹介

NIS+ の承認の基本的目的は、各 NIS+ 主体が各 NIS+ オブジェクトとサービスに対して持っているアクセス権を特定することにあります。

一度主体の NIS+ 要求が認証されると、NIS+ は承認クラスを与えます。NIS+ オブジェクトに対して行うことのできる動作を指定するアクセス権は各クラスに与えられます。クラスごとに別のアクセス権が与えられる場合、各承認クラスはそれぞれアクセス権を持つということです。

- 「承認クラス」には次の 4 つがあります。所有者、グループ、その他、未認証 (詳細は、次の122ページの「承認クラス」を参照)。
- 「アクセス権」には次の 4 つがあります。作成 (CREATE)、削除 (DESTROY)、変更 (MODIFY)、読み取り (READ) (詳細は、126ページの「NIS+ のアクセス権について」を参照)。

承認クラス

NIS+ オブジェクトは NIS+ 主体に直接アクセス権を与えずに、主体の 4 つの「クラス」にアクセス権を与えます。

- 「所有者」

  オブジェクトの所有者になった主体が所有者クラスの権限を与えられます。

- 「グループ」

  各 NIS+ オブジェクトグループは関連した 1 つのグループを持ちます。オブジェクトグループのメンバーは NIS+ 管理機能が指定します。オブジェクトグループクラスに属している主体がグループクラスの権限を与えられます。この場合の「グループ」とは NIS+ グループを表します。UNIX のグループやネットグループのことではありません。

- 「その他」

  その他クラスは、サーバーが認証できるすべての NIS+ 主体を包含するクラスです。所有者とグループクラスを除く、すべての認証されたものを意味します。

- 「未認証」

  認証されていないものすべてを意味します。

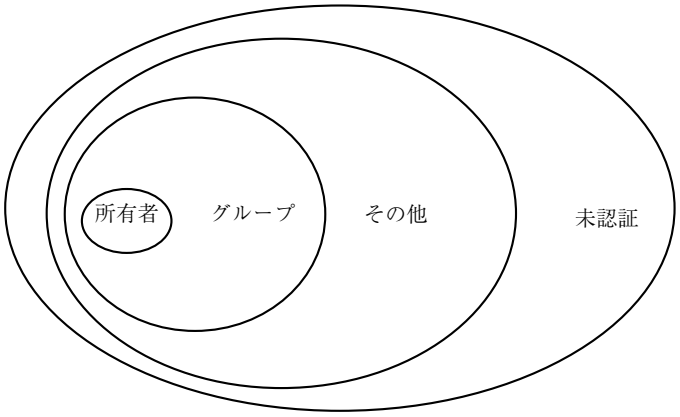


図 6–4　承認クラス

NIS+ 要求のすべてについて、システムは要求主体が属しているクラスと、そのクラスが所持しているアクセス権を判定します。

オブジェクトに対するアクセス権は、各クラスで任意の組み合わせが可能です。しかしながら、通常は上位クラスが下位クラスのすべての権限に追加権限を持つように割り当てられます。

たとえば、未認証クラスとその他クラスに読み取り権が与えられていれば、読み取り権と変更権の両方をグループクラスに、そして読み取り権、変更権、作成権、および削除権を所有者クラスに与えます。

4 つのクラスについては、以下に詳述します。

所有者クラス

所有者は 1 つの NIS+ 主体です。

NIS+ オブジェクトへのアクセスを要求する主体は、所有者アクセス権を得る前に認証され (有効な DES 資格を提示し) なければなりません。

デフォルトでは、オブジェクト所有者はオブジェクトを作成できる主体です。しかしながら、オブジェクト所有者は 2 つの方法で別の主体に所有権を譲ることができます。

- オブジェクトが作成されたときに、主体が別の所有者を指定する方法 (190ページの「コマンドによるアクセス権の指定」を参照)

- オブジェクトの作成後に、主体がオブジェクトの所有権を変更する方法 (202ページの「オブジェクトとエントリの所有権の変更」を参照)

一度主体が所有権を放棄すると、その主体はオブジェクトに対する所有者のアクセス権すべてを放棄することになり、グループ、その他、未認証のいずれかの所有権を持つことになります。

グループクラス

オブジェクトグループは 1 つの NIS+ グループです。この場合の「グループ」とは、UNIX のグループやネットグループのことではありません。

NIS+ オブジェクトにアクセスを要求する主体は、認証され (有効な DES 資格を提示し) かつグループアクセス権を与えられる前にそのグループに属していなければなりません。

1 つの NIS+ グループは NIS+ 主体の集合であり、名前空間へのアクセス権を与えるのに都合の良いようにまとめられたものです。NIS+ グループに与えられるアクセス権はそのグループのメンバーである主体すべてに適用されます。オブジェクトの所有者が、そのオブジェクトのグループに属している必要はありません。

オブジェクトが作成された時、そのオブジェクトはデフォルトグループに割り当てられます。オブジェクトが作成された時でもその後でも、そのオブジェクトをデフォルト以外のグループに割り当てることができます。オブジェクトグループはいつでも変更できます。

注 - NIS+ グループに関する情報は NIS+ `group` テーブルに格納されるのではないことに注意してください。NIS+ `group` テーブルには UNIX グループに関する情報が格納されます。NIS+ グループに関する情報は、適切な `groups_dir` ディレクトリのオブジェクトに格納されます。

NIS+ グループの情報は、各 NIS+ ドメインの `groups_dir` サブディレクトリにある、NIS+ グループ「オブジェクト」に格納されます。

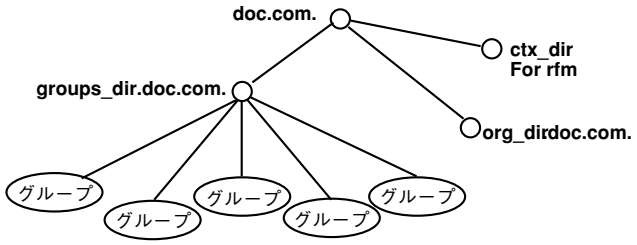


図 6–5　NIS+のディレクトリ構造

NIS+ グループの管理については、第 12 章を参照してください。

その他クラス

その他クラスは、NIS+ によって認証されたすべての NIS+ 主体のクラスです。すなわち、所有者およびグループクラスのすべてと有効な DES 資格を提示したものを加えたものです。

その他に与えられたアクセス権は、認証されたすべての主体に適用されます。

未認証クラス

未認証クラスは、正しく認証されなかったものすべてで構成されます。すなわち、有効な DES 資格を提示しなかったものです。

承認クラスと NIS+ オブジェクトの階層

NIS+ オブジェクトと承認クラスには各レベルに適用される階層があります。標準的な NIS+ ディレクトリ階層のデフォルトは次のようになります。

- 「ディレクトリレベル」

  各 NIS+ ドメインには `groups_dir` と `org_dir` という 2 つの NIS+ ディレクトリオブジェクトがあります。`groups_dir` ディレクトリオブジェクトはさまざまなグループで構成されます。各 `org_dir` ディレクトリオブジェクトには種々のテーブルが含まれます。

- 「グループレベルまたはテーブルレベル」

  グループは個々のエントリやグループで構成されます。テーブルには列 と個々のエントリが含まれます。

- 「列レベル」

テーブルは 1 つ以上の列で構成されます。

- 「エントリ (行) レベル」

  グループもしくはテーブルは 1 つ以上のエントリを持ちます。

各レベルには 4 つの承認クラスが適用されます。ディレクトリオブジェクトはそれ自身の所有者とグループを持ちます。ディレクトリ内の個々のテーブルは、ディレクトリの所有者およびグループと異なる所有者およびグループを持ちます。テーブル内では、1 エントリ (行) が個々の所有者もしくはグループを持ちます。この所有者もしくはグループは、テーブル全体の所有者やグループとは異なり、またオブジェクト全体の所有者やグループとも異なります。テーブル内の個々の列は、テーブル全体と同じ所有者やグループを持ちます。

NIS+ のアクセス権について

NIS+ オブジェクトはオブジェクト定義の一部としてそのアクセス権を指定します。187ページの「NIS+ オブジェクトのアクセス権の読み取り」に `niscat –o` コマンドを使用した場合の説明があります。

NIS+ オブジェクトは、UNIX ファイルが UNIX ユーザーに対するアクセス権を指定するのと同じ方法で、NIS+ 主体に対するアクセス権を指定します。アクセス権は、NIS+ 主体が NIS+ オブジェクトについて実行することが許されている動作の種類を表します。

NIS+ の動作はオブジェクトの種類によって異なりますが、読み取り、変更、作成、および削除の 4 つのクラスに分類されます。

- 「読み取り」

  オブジェクトの読み取り権を持った主体はオブジェクトの内容を読み取ることができます。

- 「変更」

  オブジェクトの変更権を持った主体はオブジェクトの内容を変更することができます

- 「削除」

  オブジェクトの削除権を持った主体はオブジェクトの内容を削除することができます。

- 「作成」

上位レベルのオブジェクトの作成権を持った主体はそのレベル内で新しいオブジェクトを作成できます。すなわち、NIS+ ディレクトリオブジェクトの作成権を持っていれば、そのディレクトリ内に新しいテーブルを作成できます。NIS+ テーブルの作成権を持っていれば、そのテーブル内に新しい列とエントリを作成できます。

NIS+ クライアントから NIS+ サーバーへのすべての通信は、実際には特定の NIS+ オブジェクトに対してこれらの動作のうちの 1 つを実行してほしいという要求です。たとえば、NIS+ 主体が他のワークステーションの IP アドレスを要求した場合、これは実際には、この種の情報を格納している「hosts」テーブルオブジェクトへの読み取り権を要求しています。主体が NIS+ 名前空間にディレクトリを追加するようサーバーに要求した場合、これは実際には、そのディレクトリの親オブジェクトに対して変更権を要求しています。

これらの権限は、ディレクトリからテーブルへ、さらにテーブルの列およびエントリへと展開することを覚えておいてください。たとえば、新しいテーブルを作成するには、テーブルを格納するディレクトリオブジェクトに対して作成権を持っていなければなりません。テーブルを作成する場合は、ユーザーはそのテーブルのデフォルト所有者になります。所有者として、テーブルに新しいエントリを作成できる作成権を自分自身に割り当てることができます。テーブルに新しいエントリを作成する場合は、ユーザーはそのエントリのデフォルト所有者になります。所有者として、他のユーザーにテーブルレベルの作成権を与えることもできます。たとえば、テーブルのグループクラスにテーブルレベルの作成権を与えることができます。その場合、そのテーブルグループのすべてのメンバーがテーブルに新しいエントリを作成できます。新規のテーブルエントリを作成する個々のグループメンバーはそのエントリのデフォルト所有者になります。

NIS+ 管理者

NIS+ オブジェクトの「管理権限」を持つものは誰でも NIS+ の管理者になります。説明を分かりやすくするために、管理権限が作成権、削除権、そしていくつかのオブジェクトに対しては変更権を持つと定義します。NIS+ のアクセス権については、126ページの「NIS+ のアクセス権について」を参照してください。

NIS+ オブジェクトを作成するものは誰でもそのオブジェクトに対する初期アクセス権を持つと設定します。もし作成者が管理権限をオブジェクトの所有者に限った場合 (初期状態では作成者) は、所有者だけがそのオブジェクトに対する管理権限を持

ちます。一方、作成者が管理権限をオブジェクトのグループに与えた場合は、グループの全員がそのオブジェクトに対して管理権限を持つことになります。

あるオブジェクトに対して管理権限を持つものは誰でもそのオブジェクトの NIS+ 管理者とみなされます。

すなわち、NIS+ ソフトウェアは NIS+ 管理者を 1 人にしようとするものではないということです。

理論的には、管理権限をその他クラスに与えることも、未認証クラスに与えることもでき、ソフトウェア上では可能です。しかし、管理権限をグループクラス以上に広げて与えてしまうと、NIS+ のセキュリティを実質的に無効にしてしまいます。もし管理権限をその他クラスや未認証クラスに与えた場合、NIS+ のセキュリティは保証されません。

NIS+ のパスワード、資格、およびコマンド

管理者のパスワード、資格、および鍵には次のコマンドを使用します。各コマンドの説明についてはマニュアルを参照してください。

- `chkey` - 主体の Secure RPC 鍵の組み合わせを変更します。必要のない場合は `chkey` を使わずに、`passwd` を使ってください。詳細は、159ページの「NIS+ 主体の鍵の変更」 を参照してください。
- `keylogin` - `keyserv` を使って主体の非公開鍵を復号化し格納します。
- `keylogout` - `keyserv` から非公開鍵を削除します。
- `keyserv` - サーバーに非公開暗号鍵を格納させます。詳細は、158ページの「キーログイン」を参照してください。
- `newkey` - 公開鍵データベースに新規の鍵の組み合わせを作成します。
- `nisaddcred` - NIS+ 主体に資格 (credential) を作成します。詳細は、146ページの「資格情報の作成」および、155ページの「NIS+ 資格情報の管理」を参照してください。
- `nisauthconf` - Diffie - Hellman 鍵の長さの表示あるいは設定を行います。
- `nisupdkeys` - ディレクトリオブジェクト内の公開鍵を更新します。詳細は、164ページの「公開鍵の更新」を参照してください。
- `passwd` - 主体のパスワードを変更し管理します。詳細は、第 11 章を参照してください。

パート III　NIS+ の管理

パート III では NIS+ の名前空間の管理方法を説明します。

- 第 7 章
- 第 8 章
- 第 9 章
- 第 10 章
- 第 11 章
- 第 12 章
- 第 13 章
- 第 14 章
- 第 15 章
- 第 16 章
- 第 17 章

第 7 章

NIS+ 資格の管理

この章では、NIS+ 資格とその管理方法について説明します。

注 - NIS+ セキュリティタスクの中には、可能であれば Solstice AdminSuite ツールを使用した方が簡単に実行できるものがあります。

NIS+ 資格について

NIS+ 資格は、NIS+ のユーザーを識別するために使用します。この章では、NIS+ セキュリティシステム全体と、特にシステムで資格が果たす役割について、十分理解しているユーザーを想定しています。

これらタスクを実行するのに使われるコマンドの構文、オプションなどの詳細は、NIS+ のマニュアルを参照してください。

注 - この章の DES 資格の説明は、192 ビット Diffie - Hellmaun DES 資格に適用することができます。これらは似てはいますが、その他の鍵の長さを使用した認証は細部では異なっています。コマンド行インタフェースを使用して鍵を操作するときは、その違いはユーザーにもシステム管理者にも意識されることはありません。指定した鍵の長さの表示や設定は `nisauthconf(1M)` を使って行います。

資格の機能

注 - この章では、NIS+ のコマンド一式を使用してクライアント情報を作成する方法について説明します。

資格および認証システムは、誰かが本人以外のユーザーになりすますのを防ぐシステムです。あるマシンの root 権限で `su` コマンドを使って、ログインしていないもしくは別のマシンにログインしている第 2 のユーザーになりすまし、第 2 のユーザーの NIS+ アクセス権を使用して NIS+ オブジェクトにアクセスすることを防ぎます。

注意 - NIS+ には、他人のユーザーログインパスワードを知っている者が、そのユーザーになりすましてユーザー ID と NIS+ アクセス権を使用することを妨げることはできません。また root 権限を持つユーザーが同一マシンからログインしている別のユーザーになりすますのを防ぐこともできません。

NIS+ 名前空間のセキュリティ保全のために、NIS+ 資格および認証機能がどのように承認およびアクセス権を使用するかについては、第 6 章を参照してください。

資格と資格情報

DES 資格をどのように作成しそれがどのように機能するのかを理解するには、実際の資格と資格を作成しチェックするのに使われる情報とを区別する必要があります。

- 「資格情報」

  DES 資格を作成するのに使われるデータであり、サーバーが当該資格をチェックするのに使われる

- 「DES 資格」

  主体を認証するために、主体からサーバーへ渡される一連の数字。主体が NIS+ 要求を行うたびに作成されチェックされる。実際の DES 資格については、138 ページの「DES 資格の詳細」を参照

認証

資格および認証プロセスを機能させるには、次の要素が必要です。

- 「主体の DES 資格情報」

  この情報は各主体に NIS+ 管理者が最初に作成します。この情報は主体のホームドメインの `cred` テーブルに格納されます。主体の DES 資格情報は次のもので構成されます。

  - 「主体名」

    ユーザーの完全ログイン ID もしくはマシンの完全ホスト名
  - 「主体の Secure RPC ネット名」

    各主体はユニークな Secure RPC ネット名を持っています (Secure RPCネット名の詳細は、138ページの「DES 資格の Secure RPC ネット名」を参照)
  - 「主体の公開鍵」
  - 「主体の非公開暗号鍵」

- 「主体の LOCAL 資格」
- 「サーバーの公開鍵」

  各ディレクトリオブジェクトには、当該ドメインのすべてのサーバーの公開鍵の複製が格納されています。`cred` テーブルには各サーバーの DES 資格も格納されていることに注意してください。

- 「主体の公開鍵のキーサーバーの複製」

キーサーバーは、その時点でログインしている主体 (ユーザーまたはマシン) の公開鍵の複製を持っています。

主体の認証方法

承認プロセスには 3 つのフェーズがあります。

- 「準備フェーズ」

  NIS+ 管理機能がユーザーのログインに先立って行う設定動作。たとえば、ユーザーのための資格情報を作成

- 「ログインフェーズ」

  ユーザーがログインした時にシステムが行う動作

- 「要求フェーズ」

  NIS+ 主体が、NIS+ サービスもしくは NIS+ オブジェクトに要求を発する時にソフトウェアが行う動作

これら 3 つのフェーズの詳細は、次の項目を参照してください。

資格準備フェーズ

NIS+ 管理者がユーザーに資格情報を作成する最も容易な方法は、『*Solaris* ネーミングの設定と構成』に述べられている、`nisclient` スクリプトを使う方法です。この節では、NIS+ のコマンド一式を使用してクライアント情報を作成する方法について説明します。

NIS+ 主体がログインする前に、NIS+ 管理者は当該主体 (ユーザーまたはマシン) のための資格情報を作成する必要があります。管理者は次のようにする必要があります。

- 各主体の公開鍵と非公開暗号鍵を作成します。これらの鍵は主体のホームドメインの `cred` テーブルに格納されます。これは `nisaddcred` コマンドを使って行われます (150ページの「NIS+ 主体の資格情報を作成する方法」を参照)。
- サーバーの公開鍵を作成します (164ページの「公開鍵の更新」を参照)。

ログインフェーズ - 詳細

主体がシステムにログインする時、次のステップが自動的に実行されます。

1. 主体のために keylogin プログラムが起動します。keylogin プログラムは cred テーブルから主体の非公開暗号鍵を取得し、主体のログインパスワードを使って復号化します。

注 - 主体のログインパスワードが Secure RPC パスワードと異なる場合、keylogin コマンドはパスワードを復号化できず、「cannot decrypt」などのエラーとなるか、メッセージなしでコマンドが失敗します。この問題の解説は、141ページの「Secure RPC パスワードとログインパスワードの問題」を参照してください。

---

2. 復号化された主体の非公開暗号鍵は、要求フェーズでの利用に備えてキーサーバーに渡され格納されます。

注 - 復号化された非公開暗号鍵はユーザーが keylogout コマンドを実行するまでキーサーバーに格納されます。ユーザーが keylogout を実行せずにログアウトした (またはログアウトせずに帰宅してしまった) 場合には、復号化された非公開鍵がサーバーに格納されたままになっています。もしユーザーマシンの root 権限を持った誰かがユーザーログイン ID に su コマンドを使った場合、その人間はユーザーの復号化された非公開鍵を使用でき、ユーザーのアクセス承認を使って NIS+ オブジェクトにアクセスできることになります。このような事態を避けるため、ユーザーが業務を終了する時は確実に keylogout コマンドを使ってログアウトするように注意する必要があります。同様にもしユーザーがログオフする場合は、業務に戻る時はログバックするだけで良いのです。もしユーザーが確実にログオフしなかった場合、業務に戻る時に確実に keylogin を実行する必要があります。

要求フェーズ - 詳細説明

NIS+ 主体の要求が NIS+ オブジェクトにアクセスするつど、その主体を認証するために NIS+ ソフトウェアは複数段階のプロセスを実行します。

1. **NIS+** はオブジェクトドメインの cred テーブルをチェックします。
   - もし主体が LOCAL 資格情報を持っていた場合、LOCAL 資格にあるドメイン情報を使い、主体のホームドメイン cred テーブルを見つけます。ここでは必要な情報を獲得します。

- もし主体が資格情報を持っていなかった場合、残りのプロセスは実行されず、主体の承認アクセスクラスには未認証クラスが与えられます。

2. ユーザーのホームドメインの `cred` テーブルから、**NIS+** がユーザーの **DES** 資格を取得します。ユーザーパスワードを使って非公開暗号鍵が復号化され、キーサーバーに格納されます。

3. **NIS+** が **NIS+** ディレクトリオブジェクトからサーバーの公開鍵を獲得します。

4. キーサーバーが、主体の復号化された非公開鍵とオブジェクトのサーバー **(**オブジェクトが格納されているサーバー**)** の公開鍵を使って、「共通鍵」を作成します。

5. 共通鍵を使って暗号化された「**DES** 鍵」が作成されます。これは、**Secure RPC** が乱数を発生させ、これを共通鍵を使って暗号化することで行われます。このため、**DES** 鍵は「乱数鍵」もしくは「乱数 **DES** 鍵」として参照される場合があります。

6. **NIS+** が主体のサーバーの時刻情報に基づいてタイムスタンプ **(DES** 鍵を使って暗号化される**)** を作成します。

7. **NIS+** が **15** 秒のタイムウィンドウ **(DES** 鍵を使って暗号化される**)** を作成します。この「ウィンドウ」はタイムスタンプとサーバーの内部クロックとの間で許容される最長時間になります。

8. **NIS+** が主体の **DES** 資格を作成します。これは次のもので構成されます。
   - 主体の cred テーブルに基づく Secure RPC ネット名 (unix.*identifier@domain*)
   - キーサーバーから得た暗号化された主体の DES 鍵
   - 暗号化されたタイムスタンプ
   - 暗号化されたタイムウィンドウ

9. **NIS+** が **NIS+** オブジェクトの格納されているサーバーに次の情報を渡します。
   - アクセス要求
   - 主体の DES 資格

- ウィンドウ判定子 (暗号化済)。ウィンドウ判定子は暗号化されたウィンドウに 1 を加えたもの

10. オブジェクトのサーバーがこの情報を受信します。

11. オブジェクトのサーバーが、資格の中の **Secure RPC** ネット名の一部を使って、主体のホームドメインにある `cred` テーブル内にある主体の公開鍵をチェックします

12. サーバーが主体の公開鍵とサーバーの非公開鍵を使ってもう一度共通鍵を作成します。この共通鍵は主体の非公開鍵とサーバーの公開鍵を使って作成されている共通鍵と一致する必要があります。

13. 共通鍵を使って、主体の資格の一部として受信している **DES** 鍵を暗号化します。

14. サーバーが、新しく復号化した **DES** 鍵を使って主体のタイムスタンプを復号化し、ウィンドウ判定子を使ってそれをチェックします。

15. サーバーが、復号化しチェックしたタイムスタンプと自分の内部時計とを比較し、次のプロセスを実行します。

a. サーバーとの時間差がウィンドウ許容値を越えていた場合、要求は拒否され、プロセスが中止されて、エラーメッセージが表示されます。たとえば、タイムスタンプが **9:00 am** であり、ウィンドウが **1** 分であったとします。もしその要求をサーバーが受信して復号化したのが **9:01 am** であれば、その要求は拒否されます。

b. タイムスタンプがウィンドウ許容値内である場合、サーバーは主体から前に受信した要求のタイムスタンプよりも大きいかチェックします。これによって、**NIS+** 要求を正しい順序で処理しているか確認できます。

- 順序誤りの要求はエラーメッセージとともに拒否されます。たとえば、タイムスタンプが 9:00 am でありその主体から最後に受信した要求のタイムスタンプが 9:02 am だった場合、その要求は拒否されます。
- 最後に受信した要求と同じタイムスタンプであった場合もエラーメッセージが表示されて拒否されます。これによって、要求を 2 度処理することはなくなります。たとえば、タイムスタンプが 9:00 am であり最後にその主

体から受信した要求のタイムスタンプも 9:00 am だった場合、この要求は拒否されます。

16. タイムスタンプがウィンドウ許容値内であり、その主体からの最後の要求よりも大きかった場合に、サーバーはその要求を受け入れます。

17. サーバーが、要求をコンパイルし、タイムスタンプをその主体から受信した最後のものとして格納し、要求に基づいて動作します。

18. 要求に対する応答としてサーバーから受信した情報が信頼できるサーバーからのものであるかを主体が確認できるようにするため、サーバーは主体の **DES** 鍵を使ってタイムスタンプを暗号化し、データとともに主体に送り返します。

19. 主体側では、主体の **DES** 鍵を使って送り返されたタイムスタンプを復号化します。
    - 復号化が成功した場合は、サーバーからの情報を要求者へ戻します。
    - 何らかの理由で復号化に失敗した場合は、エラーメッセージが表示されます。

DES 資格の詳細

DES 資格は次の内容で構成されます。

- 主体の「Secure RPC ネット名」

  138ページの「DES 資格の Secure RPC ネット名」を参照

- 「確認 (verification)」フィールド

  139ページの「DES 資格確認フィールド」を参照

DES 資格の Secure RPC ネット名

- 「Secure RPC ネット名」

  資格のこの部分は NIS+ の主体を特定するのに使用します。Secure RPC ネット名はすべて次の 3 つの要素で構成されます。

- 「接頭辞」

  接頭辞は常に unix になります。
- 「識別子」

  主体がクライアントユーザーの場合、ID フィールドはユーザーの UID です。主体がクライアントワークステーションの場合、ID フィールドはワークステーションのホスト名になります。
- 「ドメイン名」

  主体の DES 資格を含んでいるドメイン名がドメイン名になります (主体のホームドメイン)。

注 - NIS+ 主体名は常にドット (.) で終わりますが、Secure RPC ネット名はドット (.) で終わらないことに注意してください。

表 7–1 Secure RPC ネット名のフォーマット

主体	接頭辞	識別子	ドメイン	例
ユーザー	unix	UID	ユーザーのパスワードエントリと DES 資格そのもののあるドメイン	unix.24601@sales.doc.com
ワークステーション	unix	ホスト名	ドメイン名はワークステーションが domainname コマンドを実行すると返される	unix.machine7@sales.doc.com

DES 資格確認フィールド

確認フィールドは資格を確かめるために使用するフィールドです。cred テーブルに格納された資格情報に基づいて作成されます。

確認フィールドの構成は次のとおりです。

- 主体の非公開鍵と NIS+ サーバーの公開鍵に基づく、暗号化された主体の DES 鍵 (135ページの「要求フェーズ - 詳細説明」参照)
- 暗号化されたタイムスタンプ

■ タイムウィンドウ

DES 資格の作成方法

図 7–2 を参照してください。DES 資格を作成するには、keylogin コマンドを実行する必要がありますが、これは主体が資格を作成する前に実行します。keylogin コマンド (単に「keylogin」とする場合が多い) は、NIS+ 主体がログインする時に自動的に実行されます。

注 - 主体のログインパスワードが Secure RPC パスワードと異なる場合は、keylogin コマンドは成功しないことに注意してください。詳しくは、141ページの「Secure RPC パスワードとログインパスワードの問題」を参照してください。

---

keylogin コマンドの目的は、主体が自身の非公開鍵にアクセスできるようにすることにあります。keylogin コマンドを実行すると、cred テーブルから主体の非公開鍵を取り込み (fetch)、主体の Secure RPC パスワード (非公開鍵は主体の「Secure RPC パスワード」を使って最初に暗号化される) を使ってそれを復号化し、将来の NIS+ 要求に備えてキーサーバーに格納します。

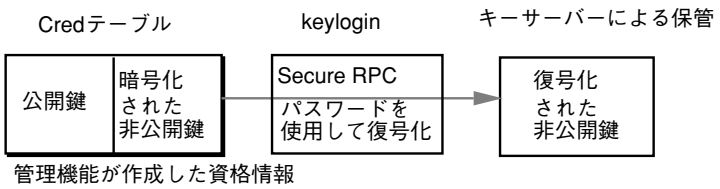


図 7–1　keylogin コマンドで主体の非公開鍵を作成

主体が DES 資格を作成するには、主体が要求を送る相手サーバーの公開鍵が必要です。この情報は主体のディレクトリオブジェクトに格納されています。主体がこの情報を獲得すれば資格の確認フィールドを作成できます。

まず、主体が種々の資格情報を暗号化するためにランダム DES 鍵を作成します。主体は自分の非公開鍵 (キーサーバーに格納されている) とサーバーの公開鍵を使って、ランダム DES 鍵を作成し暗号化するための共通鍵を作成します。次に DES 鍵で暗号化したタイムスタンプを作成し、それを確認フィールドで他の資格関連情報の中に入れます。

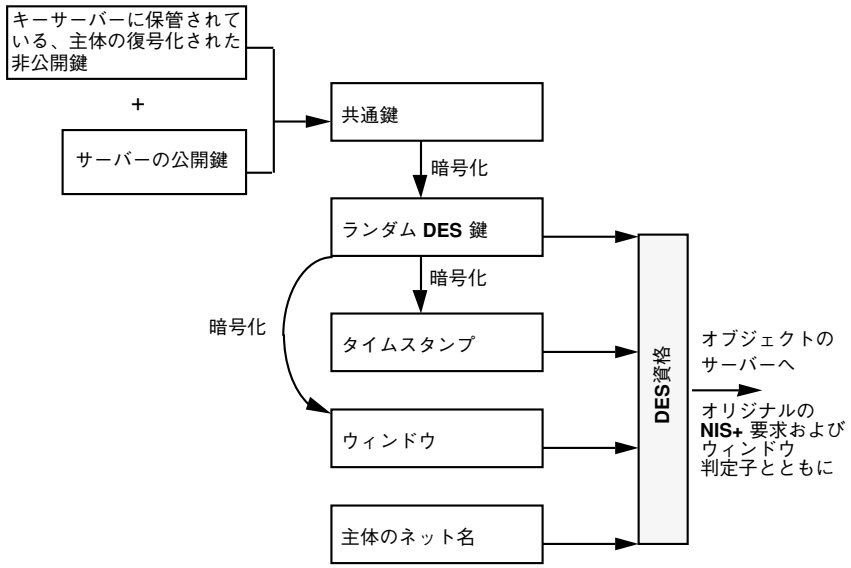


図 7–2　DES 資格の作成

Secure RPC パスワードとログインパスワードの問題

主体のログインパスワードがユーザーの Secure RPC パスワードと異なる場合、`keylogin` コマンドは主体のログインパスワードを使って復号化し、主体の Secure RPC パスワードを使って非公開鍵を暗号化するのがデフォルト動作なので、ログインした時に主体のログインパスワードを復号化できません。

この場合でも主体がシステムにログインすることは可能ですが、キーサーバーがそのユーザーのための復号化された非公開鍵を持っていませんから、未認証クラスが与えられることになります。ほとんどの NIS+ 環境において、未認証クラスにはほとんどの NIS+ オブジェクトに対する作成権、削除権、変更権を与えないように設定されますから、結果としてユーザーが NIS+ オブジェクトにアクセスしても「アクセス権拒否」などのエラーになります。

注 - 「ネットワークパスワード」を「Secure RPC パスワード」の同意語として使用する場合があります。ネットワークパスワードの入力プロンプトが表示された場合は、Secure RPC パスワードを入力してください。

別の承認クラスを得るには、この場合 `keylogin` プログラムを実行し、パスワード入力のプロンプトに主体の Secure RPC パスワードを入力する必要があります (158 ページの「キーログイン」の項を参照)。

しかし、keylogin コマンドを実行するのは一時的な解決であり、有効なのはそのセッション限りです。こうしてキーサーバーはユーザーの復号化された非公開鍵を入手しますが、ユーザーの cred テーブルにある非公開鍵は、ログインパスワードではなく Secure RPC パスワードを使って暗号化されています。次にログインした時には同じ問題が発生します。この問題を永久的に解決するには、ユーザーの Secure RPC パスワードではなくユーザーログイン ID を使って、cred テーブルにある非公開鍵を再度暗号化する必要があります。このためにはユーザーが chkey –p コマンドを実行します (159ページの「NIS+ 主体の鍵の変更」の項を参照)。

Secure RPC パスワードがログインパスワードと異なる問題を永久的に解決するには、ユーザー (もしくはユーザーに対応している管理者) が次の手順を実行してください。

1. ログインパスワードを使ってログインします。
2. キーサーバーにある復号化された非公開鍵を一時的に獲得するために、keylogin プログラムを実行し、**NIS+** アクセス権を得ます。
3. chkey –p を実行し、cred テーブル内の暗号化された非公開鍵を、ユーザーログインパスワードに基づいたものに永久的に変更します。
4. ログインセッションの終了の準備ができたら、システムをログオフする前に keylogout を実行します。
5. logout コマンドを使って、システムからログオフします。

キャッシュに保存された非公開鍵の問題

正しい資格を作成し、正しいアクセス権を与えられたのに、主体の要求が拒否される場合があります。この原因のほとんどは、サーバーの公開鍵が古いバージョンだった時に作られた、古いオブジェクトが残っていることにあります。この問題は通常次の方法で解決します。

- アクセスしようとしているドメインで、nisupdkeys を実行します (この種の問題解決の情報については、164ページの「nisupdkeys コマンド」と 641ページの「資格に関する情報が古くなっている」の項を参照)。
- マシンの nis_cachemgr を終了します。次に /var/nis/NIS_SHARED_DIRCACHE を削除してから、nis_cachemgr を再起動します。

資格関連情報の格納場所

この節では、NIS+ 名前空間のどこに資格関連情報が格納されるのかを説明します。

公開鍵などの資格に関連する情報は、名前空間内のあちこちに格納されています。NIS+ は、情報を格納しているオブジェクトの生存期間に応じてこの情報を定期的に更新しますが、場合によっては、更新の間に同期が失われ、正常に動作しなくなることがあります。資格関連の情報を格納するすべてのオブジェクト、テーブル、ファイル、および再設定方法を、表 7–2 に示します。

表 7–2 資格に関連する情報が格納されている場所

項目	格納している情報	リセットまたは変更の方法
cred テーブル	NIS+ 主体の秘密非公開鍵と公開鍵。これらの鍵のマスターコピーとなっている	nisaddcred を使用して、新しい資格を作成する。既存の資格を更新する。chkey も使用可
ディレクトリオブジェクト	サポートする各サーバーの公開鍵のコピー	ディレクトリオブジェクトに対して /usr/lib/nis/nisupdkeys コマンドを実行する
キーサーバー	現在ログインしている NIS+ 主体の秘密鍵	クライアントユーザーの場合は keylogin を、クライアントワークステーションの場合は keylogin –r を実行する
NIS+ デーモン	ディレクトリオブジェクトのコピー。これはサーバーの公開鍵のコピーを持つ	rpc.nisd デーモンとキャッシュマネージャを終了し、/var/nis から NIS_SHARED_DIRCACHE を削除する。その後両方を再起動する
ディレクトリキャッシュ	ディレクトリオブジェクトのコピー。これはサーバーの公開鍵のコピーを持つ	NIS+ キャッシュマネージャのプロセスを終了し、nis_cachemgr –i コマンドで再起動する。–i オプションは、コールドスタートファイルからディレクトリキャッシュを再設定し、キャッシュマネージャを再起動する

表 7–2　資格に関連する情報が格納されている場所　続く

項目	格納している情報	リセットまたは変更の方法
コールドスタートファイル	ディレクトリオブジェクトのコピー。これは、サーバーの公開鍵のコピーを持つ	ルートマスターでは、NIS+ デーモンのプロセスを終了してから再起動する。デーモンは、既存の NIS_COLD_START ファイルに新しい情報を再ロードする。クライアントでは、まず /var/nis から NIS_COLD_START と NIS_SHARED_DIRCACHE ファイル削除し、次にキャッシュマネージャのプロセスを終了する。その後、nisinit –c でクライアントを再び初期設定する。クライアントの信頼できるサーバーは、クライアントワークステーションの NIS_COLD_START ファイルに新しい情報を再ロードする
passwd テーブル	ユーザーのパスワード	passwd –r nisplus コマンドを使用する。このコマンドは、NIS+ passwd テーブル内のパスワードを変更し、cred テーブル内のパスワードを更新する
passwd ファイル	ユーザーのパスワード、またはワークステーションのスーパーユーザーパスワード	passwd –r nisplus コマンドを使用する。スーパーユーザーとしてログインしていても、自分のユーザー名でログインしていてもよい
passwd マップ (NIS)	ユーザーのパスワード	passwd –r nisplus コマンドを使用する

cred テーブルの詳細

主体の資格情報は「cred テーブル」に格納されています。cred テーブルは NIS+ が標準で持つ 16 のテーブルの 1 つです。各ドメインに cred テーブルが 1 つあり、そのドメインに属しているクライアントワークステーションとログインを許されたクライアントユーザー (そのドメインの主体) の資格情報が格納されています。cred テーブルはそれらドメインの org_dir サブディレクトリにあります。

注意 - cred テーブルをリンクしないでください。各 org_dir ディレクトリはそれ自身の cred テーブルを持つ必要があります。他の org_dir の cred テーブルとのリンクも使用しません。

ドメイン内のすべてのマシンにログインできるユーザーすべての LOCAL 資格情報が cred テーブルに格納されています。cred テーブルにはまた、ホームドメインとしてとしてドメインを持っているユーザーの DES 資格情報も格納されています。

cred テーブルの内容は niscat コマンドを使って見ることができます。第 14 章を参照してください。

表 7–3 に示すように、cred テーブルには 5 つの列があります。

表 7–3 cred テーブルの資格情報

	NIS+主体名	認証の種類	認証名	公開データ	非公開データ
列名	cname	auth_type	auth_name	public_data	private_data
ユーザー	完全主体名	LOCAL	UID	GID list	
マシン	完全主体名	DES	Secure RPCネット名	公開鍵	暗号化された非公開鍵

2 列目の認証の種類で他の 4 列の値の種類を判定します。

- 「LOCAL」

  認証種類が LOCAL の場合、他の列は主体のユーザー名、UID、および GID となり、最後の列は空白になります。

- 「DES」

  認証種類が DES の場合、他の列は主体の Secure RPC ネット名、公開鍵、および暗号化された非公開鍵となります。これらの鍵は、他の情報と共に DES 資格を暗号化したり、復号化するのに使われます。

資格情報の作成

資格情報を作成したり管理したりするには次の方法があります。

- Solstice AdminSuite ツールを使用します。このツールを使用すると容易に資格を管理できます。個々の資格を管理する場合はこのツールの使用を推奨します。
- `nisclient` スクリプトを使用します。1 つの主体の資格を容易に作成し変更できます。便利な方法なので、個々の資格を管理する方法として推奨します。『*Solaris* ネーミングの設定と構成』のパート I「ネーミングの紹介」に `nisclient` スクリプトを使用して資格情報を作成する方法を示しています。
- `nispopulate` スクリプトを使用します。これはすでに NIS+ マップもしくは `/etc` ファイルに資格情報を格納してある 1 つ以上の主体の資格を作成変更する便利な方法です。これは、NIS+ 主体グループの資格を管理する方法として推奨します。『*Solaris* ネーミングの設定と構成』のパート I「ネーミングの紹介」に `nispopulate` スクリプトを使用して資格情報を作成する方法を示しています。
- `nisaddcred` コマンドを使用します。次の節では、`nisaddcred` コマンドを使って資格と資格情報を作成する方法を説明します。

nisaddcred コマンド

`nisaddcred` コマンドは資格情報を作成するコマンドです。

注 - `nispopulate` と `nisclient` スクリプトを使って資格情報を作成できます。これらスクリプトは `nisaddcred` コマンドよりも容易に使えて効果的なものです。ネットワークが特別な機能を必要とするのでなければ、スクリプトを使用してください。

`nisaddcred` コマンドは、LOCAL 資格と DES 資格情報の作成、更新、および削除を行います。資格情報を作成するには、適切なドメインの `cred` テーブルに対する作成権が必要です。資格を更新するには、`cred` テーブル、または少なくとも `cred` テーブルの特定エントリに対する変更権が必要です。資格を削除するには、`cred` テーブル、または `cred` テーブルのエントリに対する削除権が必要です。

- 別の NIS+ 主体資格を作成または更新する場合

  LOCAL 資格

```
nisaddcred -p uid -P     principal-name local
```

DES 資格

```
nisaddcred -p rpc-netname -P principal-name des
```

- 自分の資格を更新する場合

  LOCAL 資格

```
nisaddcred –local
```

  DES 資格

```
nisaddcred des
```

- 資格を削除する場合

```
nisaddcred –r principal-name
```

関連コマンド

この章で説明する nisaddcred コマンドに加えて、資格に関して有用な情報を提供するコマンドが 2 つ用意されています。

表 7–4 その他の資格関連コマンド

コマンド	説明	参照ページ
niscat –o	ディレクトリの属性を一覧表示する。ディレクトリのサーバーの非公開鍵フィールドをチェックすることで、ディレクトリオブジェクトに非公開鍵が格納されているかわかる	254ページの「ディレクトリのオブジェクト属性を表示する」
nismatch–	cred テーブル上で実行すると主体の資格情報を表示する	301ページの「nismatch と nisgrep コマンド」

nisaddcred コマンドを使って資格情報を作成する方法

LOCAL 資格情報

LOCAL 資格情報を作成するために nisaddcred コマンドを使用すると、nisaddcred コマンドは主体のログインレコードから主体ユーザーの UID (および GID) を抽出し、ドメインの cred テーブルに置きます。

DES 資格情報

DES 資格情報を作成するのに使用した場合、nisaddcred は 2 つのプロセスを実行します。

1. 主体の Secure RPC ネット名を作成します。Secure RPC ネット名は、パスワードレコードから取得した主体のユーザー ID 番号とドメイン名 (たとえば、unix.1050@doc.com) を結合して作成されます。
2. 主体の非公開鍵と公開鍵を生成します。

nisaddcred が非公開鍵を暗号化するには、Secure RPC パスワードが必要です。nisaddcred コマンドを引数 –des で呼び出すと、主体の Secure RPC パスワードの入力を要求されます。通常このパスワードは主体のログインパスワードと同じです。(異なる場合は、141ページの「Secure RPC パスワードとログインパスワードの問題」に示す手順に従ってさらに操作が必要となります。)

nisaddcred コマンドは 1 対の乱数 (Difie-Hellman 方式を使った 192 ビットの認証鍵) を作成します。この鍵は Difie-Hellman の鍵ペア (key-pair) または省略して単に「鍵ペア」と呼びます。

この鍵ペアの一方が「非公開鍵」であり、もう一方が「公開鍵」になります。公開鍵は cred テーブルの公開データフィールドに置かれます。非公開鍵は主体の Secure RPC パスワードで暗号化された後に非公開データに置かれます。

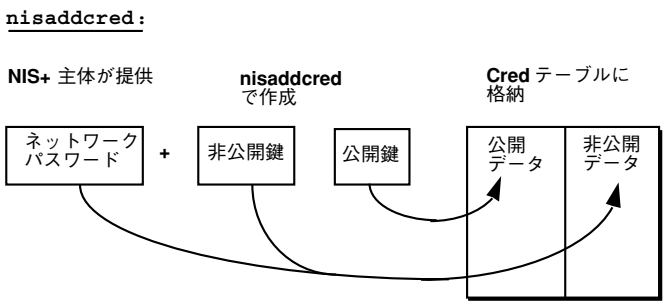


図 7–3　nisaddcred による主体キーの作成方法

デフォルトでは、cred テーブルをすべての NIS+ 主体 (未認証主体さえも) が読み取れるので、セキュリティ上の予防措置として、主体の非公開鍵を暗号化しています。

Secure RPC ネット名と NIS+ 主体名

資格情報を作成する際、何度も主体の「RPC ネット名 (rpc-netname)」と「主体名 (principal-name)」を入力しなければなりません。どちらも独自の構文が必要です。

- 「Secure RPC ネット名」

  Secure RPC ネット名は Secure RPC プロトコルで判定されますから、NIS+ のネーミング規則には従いません。

  - ユーザーの場合：unix.*uid@domain*
  - マシンの場合：unix.*hostname@domain*

  Secure RPC ネット名がユーザーを識別する場合、ユーザーの UID が必要です。ワークステーションを識別する場合は、ワークステーションのホスト名が必要です (nisaddcred コマンドとともに使用する場合は、常に –p (小文字) フラグで始まります)。

  Secure RPC ネット名は常に unix (すべて小文字) で始まりドメイン名で終わります。Secure RPC プロトコルに従いますから、ドメイン名にはドットを付けません。

- 「主体名」

  NIS+ 主体名は通常の NIS+ ネーミング規則に従いますが、常に完全指定名でなければなりません。構文は、*principal.domain* になります。

識別する対象がクライアントユーザーであっても、クライアントワークステーションであっても、主体名で始まり、その後にドットと完全なドメイン名が続き、最後

はドットで終わります。資格の作成に使用する場合、常に先頭は –P (大文字) フラグで始まります。資格の削除に使用する場合、–P フラグは使用しません。

管理者のために資格情報を作成する方法

名前空間を最初に設定する場合、ドメインをサポートする管理者の資格情報を最初に作成します。一度管理者の資格情報を作成すると、他の管理者、クライアントワークステーション、およびクライアントユーザーの資格情報を作成できます。

自分の資格情報を作成しようとすると、堂々めぐりに陥ります。つまり、ドメインの cred テーブルに対する作成権がなければ自分の資格情報を作成できず、もし NIS+ 環境が適切に設定されていれば、資格を持つまではそのような権限を持つこともできません。この循環から抜け出さなくてはなりません。これには、次の 2 つの方法があります。

- ドメインのマスターサーバーにスーパーユーザーとしてログインしているうちに、資格情報を作成する
- ダミーパスワードを使用して、他の管理者に自分の資格を作成してもらってから chkey コマンドを使用してパスワードを変更する

いずれの場合も、別の NIS+ 主体に資格情報を作成してもらうことになります。自分の資格情報を作成するには、150ページの「NIS+ 主体の資格情報を作成する方法」の項を参照してください。

NIS+ 主体の資格情報を作成する方法

NIS+ の資格情報はドメインの設定後いつでも作成できます。すなわち、cred テーブルがあれば作成できます。

NIS+ 主体の資格情報を作成する

- 主体のホームドメインの cred テーブルに対する作成権が必要
- サーバーは、主体の存在を認識している必要がある。この場合「サーバーが認識している」とは以下のことを意味する
  - 主体がユーザーの場合、ドメインの NIS+ passwd テーブルかサーバーの /etc/passwd ファイルのどちらかにエントリが必要
  - 主体がワークステーションン場合、ドメインの NIS+ host テーブルかサーバーの /etc/hosts ファイルのどちらかにエントリが必要

これらの条件を満たせば、–p と –P の両方のオプション付きで nisaddcred コマンドを実行できます。

LOCAL 資格

```
nisaddcred -p uid -P principal-name local
```

DES 資格

```
nisaddcred -p rpc.netname -P principal-name des
```

次の原則を銘記してください。

- 主体ユーザーの場合、LOCAL 資格と DES 資格の両方の情報を作成できます。
- 主体ワークステーションの場合、DES 資格情報しか作成できません。
- 主体のホームドメイン (ユーザーまたはマシン) でだけ DES 資格情報を作成できます。
- ユーザーの LOCAL 資格情報はユーザーのホームドメインでも他のドメインでも作成できます。

ユーザー主体 - 例

UID が 11177 の morena という名前の NIS+ ユーザーの LOCAL および DES 資格情報を作成する例を次に示します。彼女の所属するドメインは doc.com. で、この例ではドメインの主体マシンから彼女の資格情報を入力します。

```
client# nisaddcred -p 11177 -P morena.doc.com. local
client# nisaddcred -p unix.11177@sales.doc.com \
   -P morena.doc.com. des
Adding key pair for unix.11177@sales.doc.com
   (morena.doc.com.).
Enter login password:
```

Enter login password: プロンプトに対する正しい応答は morena のログインパスワードです。彼女のログインパスワードを知らない場合は、ダミーパスワードを使用します。ダミーパスワードは、次の例のように、後で chkey コマンドを使って変更できます。

ダミーパスワードと **chkey** の使い方 **-** 例

ユーザーのログインパスワードを知らない場合、次に説明するようにダミーパスワードを使うことができます。

表 7–5 は、ダミーパスワードを使って資格情報を作成した管理者が、chkey コマンドを使ってパスワードを変更する方法を示します。この例では、UID が 119 の eiji という名前の管理者の資格情報を作成します。eiji はルートドメインに属しているので、rootmaster という名前のルートマスターサーバーから彼の資格情報を入力します。

表 7–5 管理者の資格情報を作成する作業

作業	コマンド
eiji の LOCAL 資格情報を作成する	rootmaster# **nisaddcred -p 119 -P eiji.doc.com. local**
eiji の DES 資格情報を作成する	rootmaster#**nisaddcred -p unix.119@doc.com -P eiji.doc.com. des** Adding key pair for unix.119@doc.com (eiji.doc.com.).
eiji のダミーパスワードを入力する	Enter eiji's login password: nisaddcred: WARNING: password differs from login passwd
ダミーパスワードを再度入力する	Retype password:
使用したダミーパスワードを eiji に伝える	
eiji がルートマスターにログイン	rootmaster% login: **eiji**
eiji が本当のログインパスワードを入力	Password:
eiji はエラーメッセージを受けたが、ログインは許可される	Password does not decrypt secret key for unix.119@doc.com.
eiji がログインを実行	rootmaster% **keylogin**

表 7–5 管理者の資格情報を作成する作業 続く

作業	コマンド
eiji がダミーパスワードを入力	Password: *dummy-password*
eiji が chkey を実行	rootmaster%**chkey -p** Updating nisplus publickey database Generating new key for'unix.119@doc.com'.
eiji が本当のログインパスワードを入力	Enter login password:
eiji が本当のログインパスワードを再入力	Retype password: Done.

まず、通常の方法で eiji の資格情報を作成しますが、ダミーのログインパスワードを使用します。NIS+ は警告を発して、再入力を要求します。再入力を行うと、この操作は完了します。ドメインの cred テーブルには、ダミーパスワードに基づく eiji の資格情報が入っています。しかし、ドメインの passwd テーブル (あるいは /etc/passwd ファイル) には、まだ彼のパスワードエントリが残っているので、彼はシステムにログオンできます。

次に、eiji は本当のログインパスワードを入力して、ドメインのマスターサーバーにログインします (ログイン手順では、passwd テーブルまたは /etc/passwd ファイルのパスワードをチェックするため)。そこから eiji は、まずダミーパスワードを使用して keylogin を実行し (cred テーブルをチェックするため)、chkey –p コマンドを使用して cred エントリを本当のパスワードに変更します。

別のドメインでの作成 - 例

これらの 2 つの例では、主体ユーザーのホームドメインのマスターサーバーにログインしている間に、主体ユーザーの資格情報を作成しました。しかし、適切なアクセス権がある場合、別のドメインに資格を作成できます。次の構文でドメイン名を付加するだけです。

LOCAL 資格情報

```
nisaddcred -p uid -P      principal-name local domain-name
```

DES 資格情報

```
nisaddcred -p rpc-netname -P      principal-name des domain-name
```

次の例では、まず chou という名前のシステム管理者の LOCAL および DES 資格情報を、そのシステム管理者のホームドメイン (これは、たまたまルートドメイン) に作成し、次にその LOCAL 資格を doc.com ドメインに追加します。chou の UID は 11155 です。このコマンドは、ルートマスターサーバーから入力します。簡単にするため、chou の正しいログインパスワードを入力しているものとします。

```
rootmaster# nisaddcred -p 11155 -P chou.doc.com. local
rootmaster# nisaddcred -p unix.11155@doc.com -P chou.doc.com. des
Adding key pair for unix.11155@doc.com (chou.doc.com.).
Enter login password:
rootmaster#  nisaddcred -p 11155 -P chou.doc.com. local doc.com.
```

LOCAL 資格情報は、UID を NIS+ 主体名にマップします。クライアントユーザーである NIS+ 主体は、さまざまなドメインにさまざまな UID を持てますが、NIS+ 主体名は 1 つしか持てません。したがって、chou などの NIS+ 主体が自分のホームドメイン以外のドメインからログインする場合、そのドメインにパスワードエントリを持つだけではなく、そのドメインの cred テーブルに LOCAL 資格も持っていなければなりません。

ワークステーションでの例

主体「ワークステーション」の資格情報を作成する例を次に示します。このホスト名は starshine1 で、ルートドメインに所属しています。したがって、この資格はルートマスターサーバーから作成されます。この例では、ルートマスターにルートとしてログインしている間に資格情報を作成します。しかし、有効な資格情報と適切なアクセス権をすでに持っている場合、自分のユーザー名でログインしているときに、資格を作成できます。

```
rootmaster# nisaddcred -p unix.starshine1@doc.com -P \
starshine1.doc.com. des
Adding key pair for unix.starshine1@doc.com
(starshine1.doc.com.).
Enter starshine1.doc.com.'s root login password:
Retype password:
```

パスワードプロンプトに対しては、クライアントワークステーションのスーパーユーザーパスワードを入力してください。もちろん、ダミーパスワードを使用することもできます。このダミーパスワードは、その主体ワークステーションにスーパーユーザーとしてログインすれば、後で変更できます。

NIS+ 資格情報の管理

以下のいくつかのセクションでは、nisaddcred コマンドで資格情報を管理する方法について説明します。nisaddcred で資格情報を管理するには、cred テーブルに対する作成権、変更権、読み取り権、削除権が必要です。

自分の資格情報を更新する方法

自分の資格情報を更新することは、資格を作成することよりはるかに簡単です。自分のユーザー名でログインしているときに、次のような nisaddcred コマンドを入力するだけです。

```
# nisaddcred des
# nisaddcred local
```

別のユーザーの資格情報を更新する場合は、そのユーザーの資格情報を作成するのと同じ操作を行います。

資格情報を削除する方法

nisaddcred コマンドは、主体の資格情報を削除しますが、コマンドを実行しているローカルドメインだけから削除できます。

システム全体から完全に主体を削除するには、主体のホームドメインおよび LOCAL 資格情報のあるすべてのドメインから主体の資格情報を明示的に削除しなければなりません。

資格情報を削除するには、ローカルドメインの cred テーブルへの変更権が必要です。–r オプションを使い、完全 NIS+ 主体名で主体を指定します。

```
# nisaddcred -r principal-name
```

次の 2 つの例では、システム管理者 morena.doc.com. の LOCAL と DES の資格情報を削除します。最初の例では、システム管理者のホームドメイン (doc.com.) から両方の資格情報を削除し、2 番目の例では、sales.doc.com. ドメインから LOCAL 資格情報を削除します。該当するドメインのマスターサーバーから、それぞれがどう入力されるかに注意してください。

```
rootmaster# nisaddcred -r morena.doc.com.
salesmaster#  nisaddcred -r morena.doc.com.
```

資格が本当に削除されたことを確かめるには、次に示すように、cred テーブルに対して nismatch を実行します。nismatch の詳細は、第 14 章を参照してください。

```
rootmaster# nismatch morena.doc.com. cred.org_dir
salesmaster# nismatch morena.doc.com. cred.org_dir
```

第 8 章

NIS+ 鍵の管理

この章では、NIS+ の鍵と、鍵を管理する方法について説明します。

注 - Solstice AdminSuite ツールを利用した方が容易に NIS+ セキュリティタスクを実行できます。

NIS+ 鍵について

NIS+ 鍵は NIS+ の関連情報を暗号化するために使用します。

この章では、NIS+ セキュリティシステム全体と、特にシステムで鍵が果たす役割について、十分理解しているユーザーを想定しています。(詳しくは、第 6 章を参照してください。)

NIS+ 鍵に関するコマンドとその構文やオプションについての詳しい説明は、nis+(1) のマニュアルページを参照してください。また、`nisaddcred` コマンドも鍵に関連

のある働きをします。詳細は、146ページの「nisaddcred コマンド」を参照してください。

キーログイン

主体がログインする時、パスワードの入力を求めるプロンプトが現れます。このパスワードはユーザーがセキュリティゲートをパスし、ネットワークにアクセスするために必要なものです。ログインプロセスは、ユーザーのホームドメインの cred テーブルにあるユーザーの非公開鍵を復号化しキーサーバーへ渡します。キーサーバーはその復号化された非公開鍵を使って、ユーザーが NIS+ オブジェクトにアクセスするつど、ユーザーの認証を行います。

通常は、この場合だけ主体にパスワードが要求されます。しかし、cred テーブルにある主体の非公開鍵がユーザーのログインパスワードと異なるパスワードを使用して暗号化されていると、ログインプロセスはそれを login の際にログインパスワードを使って復号化できないので、キーサーバーに復号化した非公開鍵を提供できません。(これは cred テーブル内にあるユーザーの非公開鍵が、ログインパスワードと異なる Secure-RPC パスワードを使用して暗号化された場合に多く起こります。)

注 - 「ネットワークパスワード」と「Secure-RPC パスワード」を同意語として使用する場合があります。

この問題を一時的に解決するには、ログインの後には必ず主体が keylogin コマンドを使用して、キーログインを実行する必要があります。-r フラグを使って、スーパーユーザー主体にログインし、ホストの /etc/.rootkey にスーパーユーザーの鍵を格納します。

主体ユーザーの場合

```
keylogin
```

主体マシンの場合 (一度だけ)

```
keylogin -r
```

しかしながら、オリジナルのパスワードを使用して確実にキーログインを実行しても、そのログインセッションにおいて一時的に問題が解決するだけです。cred テーブルの非公開鍵はユーザーのログインパスワードと異なるパスワードを用いて暗号化されたままであり、次にユーザーがログイン した時に同じ問題が起こります。こ

の問題を永久的に解決するには、chkey コマンドを使用して、非公開鍵の暗号化に使ったパスワードをユーザーのログインパスワードに変更します (159ページの「NIS+ 主体の鍵の変更」を参照)。

NIS+ 主体の鍵の変更

chkey コマンドは cred テーブルに格納されている NIS+ 主体の公開および非公開鍵を変更します。このコマンドは passwd テーブル内もしくは /etc/passwd ファイル内のいずれのエントリにも影響を与えません。

chkey コマンドについて

- 新規に鍵を作成し、パスワードを用いて非公開鍵を暗号化します。–p オプションをつけて実行した場合、chkey コマンドは既存の非公開鍵を新規のパスワードで再暗号化します。
- 新規に Diffie-Hellman の鍵ペアを作成し、提供されたパスワードを使用して非公開鍵を暗号化します (各主体に複数の Diffie-Hellman の鍵のペアが存在することは可能) 。しかし、ほとんどの場合ユーザーは新規の鍵ペアを求めず、既存の非公開鍵を新規のパスワードを使って再暗号化しようとします。これを行うには、–p オプション付きで chkey を実行します。

これらのテーマの詳細は、マニュアルを参照してください。

注 - NIS+ 環境においては、どんな管理ツールまたは passwd (もしくは nispasswd) コマンドを用いてログインパスワードを変更しても、cred テーブル内の非公開鍵は新規のパスワードを使用して自動的に再暗号化されます。従って、ログインパスワードの変更後に chkey コマンドを実行する必要はありません。

---

chkey コマンドはキーサーバー、cred テーブル、および passwd テーブルとの関連で動作します。chkey を実行するには次のようにします。

- ホームドメインの passwd テーブルにエントリが必要です。この条件を満たさなければ、エラーメッセージが返ります。
- キーサーバーに復号化された非公開鍵のあることを確認するために、keylogin を実行する必要があります。
- cred テーブルの変更権を持っていなければなりません。この権限がない場合は「permission denied」などのエラーメッセージが返ります。

- cred テーブル内の非公開鍵の暗号化に使われた、オリジナルのパスワードを知らなくてはなりません (ほとんどの場合、これは Secure-RPC パスワード)。

ログインパスワードを使って非公開鍵を再暗号化するために chkey コマンドを使用するには、最初にオリジナルのパスワードを用いて keylogin を実行し、次に chkey –p を実行します。表 8–1 では、主体ユーザーの keylogin と chkey の実行方法を示します。

表 8–1 非公開鍵の暗号化：コマンドの説明

作業	コマンド
ログインする	`Sirius% login` *Login-name*
ログインパスワードを入力する	`Password:`
ログインパスワードと Secure RPC パスワードが異なっている場合、keylogin を実行する	`Sirius% keylogin`
非公開鍵の暗号化に使用したオリジナルパスワードを入力する	`Password:` *Secure RPC password*

chkey を実行する	`Sirius% chkey -p`
	`Updating nisplus publickey database`
	`Updating new key for 'unix.1199@Doc.com.'`

ログインパスワードを入力する	`Enter login password:` *login-password*
ログインパスワードを再入力する	`Retype password:`

鍵の変更

次の節では、NIS+ 主体の鍵の変更方法を説明します。

注 - サーバーの鍵を変更するときは、常にドメイン内の全クライアントの鍵情報も更新する必要があります。その方法は、167ページの「クライアントの鍵情報を更新する」で説明します。

ルートからルート鍵を変更する

ルートマスター (root として) からルートマスターサーバーの鍵を変更するには、表 8–2 の手順を実行します。

表 8–2 ルートマスターからルートマスター鍵を変更する

作業	コマンド
新規 DES 資格を作成	rootmaster# **nisaddcred des**
rpc.nisd のプロセス ID を発見	rootmaster# **ps -e | grep rpc.nisd**
NIS+ デーモンを終了	rootmaster# **kill** *pid*
セキュリティなしで NIS+ デーモンを再起動	rootmaster# **rpc.nisd -S0**
keylogout を実行 (以前の keylogin はタイムアウト)	rootmaster# **keylogout -f**
マスターがディレクトリに保管していた鍵を更新	rootmaster# **nisupdkeys** *dirs*
rpc.nisd のプロセス ID を発見	rootmaster# **ps -e | grep rpc.nisd**
NIS+ デーモンを終了	rootmaster# **kill** *pid*
デフォルトセキュリティで NIS+ デーモンを再起動	rootmaster# **rpc.nisd**
keylogin を実行	rootmaster# **keylogin**

- *pid* は ps -e | grep rpc.nisd コマンドで通知されるプロセス ID 番号
- *dirs* は更新するディレクトリオブジェクト (rootmaster によって保管されたディレクトリオブジェクト)

表 8–2 に示すプロセスの最初の手順では、nisaddcred がルートマスターの cred テーブルを更新し、/etc/.rootkey を更新し、ルートマスターのキーログインを実行します。この時点では、マスターに保管されたディレクトリオブジェクトが更新されておらず、その資格情報とルートマスターとは同期がとれていません。表 8–2 のその後の手順は、すべてのオブジェクトを更新するのに必要です。

注 - サーバーの鍵を変更するときは、常にドメイン内の全クライアントの鍵情報も更新する必要があります。その方法は、167ページの「クライアントの鍵情報を更新する」で説明します。

別のマシンからルート鍵を変更する手順

別のマシンからルートマスターの鍵を変更する場合、それに必要な NIS+ 資格とそれを行う承認を得ていなくてはなりません。

表 8–3 異なるマシンによるルートマスターの鍵の変更 - コマンド一覧

作業	コマンド
新規の DES 資格を作成	othermachine% **nisaddcred -p** *principal* **-P** *nisprincipal* **des**
ディレクトリオブジェクトを更新	othermachine% **nisupdkeys** *dirs*
/etc.roootkey を更新	othermachine% **keylogin -r**
クライアントとして再度初期化	othermachine% **nisinit -cH**

- *principal* はルートマシンの Secure RPC ネット名。たとえば、unix.rootmaster@doc.com (ドットで終わらない)
- *nis-principal* はルートマシンの NIS+ 主体名。たとえば、rootmaster.doc.com. (ドットで終わる)
- *dirs* は更新するディレクトリオブジェクト (rootmaster で保管されたディレクトリオブジェクト)

nisupdkeys を実行する場合、関連したディレクトリオブジェクトすべてを同時に更新するように注意します。すなわち、すべてを 1 つのコマンドで行います。分割して更新すると、認証エラーになります。

注 - サーバーの鍵を変更するときは、常にドメイン内の全クライアントの鍵情報も更新する必要があります。その方法は 167ページの「クライアントの鍵情報を更新する」で説明します。

複製からルート複製の鍵を変更する手順

複製からルート複製の鍵を変更する手順を次に示します。

```
replica# nisaddcred des
replica# nisupdkeys dirs
```

- *dirs* は更新するディレクトリオブジェクト (replica で保管されたディレクトリオブジェクト)

nisupdkeys を実行する場合、関連したディレクトリオブジェクトのすべてを同時に更新するように注意します。すなわち、すべてを 1 つのコマンドで行います。分割して更新すると、認証エラーになります。

注 - サーバーの鍵を変更するときは、常にドメイン内の全クライアントの鍵情報も更新する必要があります。その方法は、167ページの「クライアントの鍵情報を更新する」で説明します。

ルート以外のサーバーの鍵の変更手順

サーバーからルート以外のサーバー (マスターまたは複製) の鍵を変更する手順を以下に示します。

```
subreplica# nisaddcred des
subreplica# nisupdkeys parentdir dirs
```

- *parentdir* はルート以外のサーバーの親ディレクトリ (subreplica の NIS+ サーバーを持つディレクトリ)
- *dirs* は更新しようとするディレクトリオブジェクト (subreplica で保管されたディレクトリオブジェクト)

nisupdkeys を実行する場合、関連したディレクトリオブジェクトのすべてを同時に更新するように注意します。すなわち、すべてを 1 つのコマンドで行います。分割して更新すると、認証エラーになります。

注 - サーバーの鍵を変更するときは、常にドメイン内の全クライアントの鍵情報も更新する必要があります。その方法は、167ページの「クライアントの鍵情報を更新する」で説明します。

公開鍵の更新

NIS+ サーバーの公開鍵は名前空間のあちこちに格納されています。サーバーに新規の資格情報を作成する場合、新規の鍵ペアが作成され cred テーブルに格納されます。しかし、名前空間ディレクトリオブジェクトには、まだサーバーの古い公開鍵のコピーが残っています。nisupdkeys コマンドを使用して、これらのディレクトリオブジェクトのコピーを更新します。

nisupdkeys コマンド

古い鍵ペアの保全が危うくなったり、非公開鍵の暗号化に使ったパスワードを忘れてしまったりして新規の鍵ペアを作成する場合、nisupdkeys を使用してディレクトリオブジェクト内の古い公開鍵を更新できます。

- 1 台の特定サーバーの鍵を更新する
- NIS+ ディレクトリオブジェクトをサポートしているサーバーすべての鍵を更新する
- サーバーの公開鍵をディレクトリオブジェクトから削除する
- サーバーの IP アドレスが変更された場合にそれを更新する

しかしながら、nisupdkeys は主体ワークステーション上の NIS_COLD_START ファイルを更新できません。サーバーの鍵のコピーを更新するには、NIS+ クライアントが nisclient コマンドを実行しなければなりません。もしくは、NIS+ キャッ

シュマネージャを実行中でかつコールドスタートファイル内で 1 つ以上のサーバーを利用できる場合、主体はディレクトリオブジェクト上で生存期間がタイムアウトするまで待つことができます。タイムアウトが発生すると、キャッシュマネージャはコールドスタートファイルを自動的に更新します。生存期間のデフォルトは 12 時間です。

nisupdkeys を使うには、NIS+ ディレクトリオブジェクトへの変更権が必要です。

公開鍵の引数の更新と例

nisupdkeys コマンドは /usr/lib/nis にあります。nisupdkeys コマンドは次の引数を使います。nisupdkeys コマンドの詳細と引数すべてのリストは、nisupdkeys(1M) のマニュアルページを参照してください。

表 8–4 nisupdkeys の引数

引数	**Effect**
(引数なし)	カレントドメインのサーバーの鍵をすべて更新する
ディレクトリ名	ディレクトリ名で指定したディレクトリオブジェクトの鍵を更新する
–H サーバー名	カレントドメインディレクトリオブジェクト内のサーバー名で指定したサーバーの鍵を更新する。他のドメインにあるサーバーの鍵を更新する場合は、完全ホスト名を使用する
–s –H サーバー名	サーバー名で指定したサーバーで保管されたディレクトリオブジェクトすべての鍵を更新する
–C	鍵をクリアする

表 8–5 で公開鍵の更新手順の例を示します。

表 8–5　公開鍵の更新: コマンド例

タスク	コマンド
カレントドメイン (doc.com) のすべてのサーバーのすべての鍵を更新	rootmaster# **/usr/lib/nis/nisupdkeys** Fetch Public key for server rootmaster.doc.com. netname='unix.rootmaster@doc.com' Updating rootmaster.doc.com.'s public key. Public key: *public-key*
sales.doc.com ドメインのディレクトリオブジェクトをサポートしているすべてのサーバーの鍵を更新	salesmaster# **nisupdkeys sales.doc.com** (画面上には何も表示されません。)
すべてのディレクトリ内のサーバー名が master7 であるサーバーの鍵を更新	rootmaster# **nisupdkeys -H master7**
sales.doc.com ディレクトリオブジェクトで保管された鍵をクリア	rootmaster# **nisupdkeys -C sales.doc.com**
カレントドメインのディレクトリオブジェクトのサーバー名が master7 であるサーバーの鍵をクリア	rootmaster# **nisupdkeys -C -H master7**

IP アドレスの更新

サーバーの IP アドレスを変更するか、またはアドレスを追加する場合、NIS+ アドレス情報を更新するために nisupdkeys を実行する必要があります。

1 つ以上のサーバーの IP アドレスを更新するには、–a オプション付きで nisupdkeys を使用します。

指定されたドメインのサーバーの IP アドレスを更新。

```
rootmaster# nisupdkeys -a domain
```

特定サーバーの IP アドレスを更新。

```
rootmaster# nisupdkeys -a -H server
```

クライアントの鍵情報を更新する

サーバーの鍵を変更するときは、常に全クライアントの鍵情報も更新する必要があります。NIS+ のサーバーは NIS+ のクライアントでもあります。そのため、あるサーバーの鍵を更新した場合は、それが NIS+ のサーバーであるか通常のクライアントであるかにかかわらず、ドメイン内にあるほかのすべてのマシンの鍵情報を更新する必要があります。

クライアントの鍵情報を更新するには、以下の 3 通りの方法があります。

- 最も簡単に個々のクライアントの鍵情報を更新するには、クライアント側で `nisclient` スクリプトを実行します。『*Solaris* ネーミングの設定と構成』を参照してください。
- 別の方法で個々のクライアントの鍵情報を更新するには、クライアント側で `nisinit` コマンドを実行します。267ページの「クライアントを初期設定する」を参照してください。
- ドメイン内にある全てのマシンのクライアントの鍵情報を一括で更新するには、ドメインのディレクトリオブジェクトの生存期間の値を短くします。167ページの「クライアントの鍵情報を一括で更新する」を参照してください。

クライアントの鍵情報を一括で更新する

サーバーの鍵を変更したあとで、ドメイン内にあるすべてのマシンのクライアントの鍵情報を一括更新できます。

1. `nischttl` コマンドを使用して、ドメインのディレクトリオブジェクトの生存期間 **(TTL)** を、すぐに満了するような小さな値にしてください。

   たとえば、`sales.doc.com.` ドメイン内のサーバーの鍵を変更した場合、ディレクトリの TTL 値を 1 分にするには、以下のように入力します。

   ```
   client% nischttl 60 sales.doc.com.
   ```

2. ディレクトリの **TTL** 値が満了すると、キャッシュマネージャはエントリを終了し、クライアントのために新しく更新された情報を入手します。

3. ディレクトリオブジェクトの **TTL** 値が満了したあとで、ディレクトリオブジェクトの **TTL** 値をデフォルト値に戻します。

たとえば、sales.doc.com. ドメインのディレクトリオブジェクトの TTL 値を 12 時間に戻すには、以下のように入力します。

```
client% nischttl 12h sales.doc.com.
```

TTL 値の使用の詳細は、275ページの「nischttl コマンド」を参照してください。

第 9 章

拡張セキュリティ資格の管理

Diffie-Hellman 拡張鍵

NIS+ は、RPCSEC_GSS RPC(3N) セキュリティフレイバーを使用して、192 ビット Diffie-Hellman [RPC(3N) セキュリティフレイバー AUTH_DES] を超える RPC(3N) 層におけるより厳格なセキュリティを提供します。システム上で使用可能なセキュリティメカニズムのリストについては、nisauthconf(1M) コマンドを参照してください。また、これらのセキュリティメカニズムは、より厳格な暗号と各 NIS+ トランザクションの完全性を提供します。すなわち、各 NIS+ トランザクションのデータが変更されていないことが検証されます。

システム管理者は、後述のガイドラインに従って、NIS+ サーバー環境を構築する前または後に nisauthconf(1M) を実行することによって、より厳格なセキュリティメカニズムの利点を活用することができます。

新しい公開鍵ベースのセキュリティメカニズムへの移行

Diffie-Hellman 640 ビット (dh640-0) のような公開鍵暗号ファミリーのより厳格なセキュリティメカニズムを使用するには、既存の cred テーブルに各主体の新しい資格を追加することが必要です。後述の手順は、現在 Diffie-Hellman 192 ビット (RPC セキュリティフレイバー AUTH_DES) セキュリティを実行しているシステムを、Diffie-Hellman 640 ビット (RPC セキュリティフレイバー RPCSEC_GSS) セキュリティを実行するように変換する場合のものです。この移行手順は、最もよくある場合を説明したものですが、公開鍵暗号ファミリーのどれか 1 つのセキュリティメカニズムタイプから、公開鍵暗号ファミリーの別のセキュリティメカニズムに変換する場合、原理は同じです。

注 - 以下の例では、$PATH に /usr/lib/nis があることを前提にしています。

NIS+ セキュリティメカニズムの構成

NIS+ セキュリティの構成は、nisauthconf(1M) コマンドを使用して行います。nisauthconf はセキュリティメカニズムのリストを設定した順に取ります。セキュリティメカニズムは、secure_rpc(3N) で指定された 1 つまたは複数の認証フレイバーを使用することができます。**des** が唯一の指定されたメカニズムである場合には、NIS+ は他の NIS+ クライアントおよびサーバーの認証に AUTH_DES だけを使用します。NIS+ は、nisaddcred(1M) の場合を除いて、des より後のメカニズムは無視します。

```
nisauthconf [-v] [mechanism, ...]
```

この場合の mechanism は、システム上で使用可能な RPC セキュリティメカニズムです。使用可能なメカニズムのリストについては、nisauthconf(1M) を参照して

ください。メカニズムを指定しないと、現在構成されているメカニズムのリストが出力されます。

新しいセキュリティメカニズム資格の作成

NIS+ ユーザーおよびホスト主体ごとに、新しいメカニズムの資格情報を作成する必要があります。これを行うには、システムが現在のメカニズムによる認証を継続している間に、NIS+ を実行しているマシンの 1 つで `nisauthconf(1M)` コマンドを実行して新しい資格を作成しなければなりません。資格作成の基本の詳細は、150 ページの「NIS+ 主体の資格情報を作成する方法」も参照してください。

新しいセキュリティメカニズム資格 - 例

des から dh640-0 へ変換します。`nisauthconf` がスーパーユーザーとして実行されており、`nisaddcred` が、ホームディレクトリの中で作成権を持っているどれかの主体として実行されていることが必要です。サーバー名は `server1`、ユーザーの主体名は `morena` です。ユーザー `morena` の UID は 11177 です。

```
client# nisauthconf des dh640-0
client% nisaddcred -P server1.doc.com. -p unix.server1@doc.com dh640-0
   (画面上には何も表示されない)
client% nisaddcred -P morena.doc.com. -p unix.11177@doc.com -ldummy-password dh640-0
   (画面上には何も表示されない)
```

NIS+ ディレクトリオブジェクトへの新しい鍵の追加

すべてのサーバーの新しい資格が生成されたら、`nisupdkeys(1M)` を実行して、これらのサーバーが提供するすべてのディレクトリオブジェクトに新しい公開鍵を追加します。`nisupdkeys(1M)` コマンドを使用するには、その NIS+ ディレクトリオブジェクトに対する変更権を持っていなければなりません。詳細は、164ページの「公開鍵の更新」を参照してください。

注意 - これらの NIS+ ディレクトリを提供するすべてのサーバー、およびこれらのディレクトリにアクセスするすべてのクライアントは、Solaris 7 以降を実行していなければなりません。

NIS+ ディレクトリオブジェクトへの新しい公開鍵の追加 - 例

この例では、新しい公開鍵を使用してサーバーが提供しているディレクトリは、doc.com、org_dir.doc.com.、および groups_dir.doc.com. です。更新は、マスターサーバー主体として行われます。新しいメカニズムを実行する前に、nisauthconf(1M) を使用して nisupdkeys を構成することが必要です。この例では、現在の認証メカニズムは des で、新しいメカニズムは dh640-0 です。

```
masterserver# nisauthconf dh640-0 des
masterserver# nisupdkeys doc.com.
   (画面上には何も表示されない)
masterserver# nisupdkeys org_dir.doc.com.
   (画面上には何も表示されない)
masterserver# nisupdkeys groups_dir.doc.com.
   (画面上には何も表示されない)
```

新しいセキュリティメカニズム資格を受け入れるようにする NIS+ サーバーの構成

各サーバー上で NIS+ 認証を構成して、新旧両方の資格を受け入れるようにします。そのためには、nisauthconf(1M) および keylogin(1) を実行し、keyserv(1M) を再起動することが必要です。keylogin(1) コマンドは、各メカニズムの鍵を /etc/.rootkey に格納します。keylogin の基本の詳細は、158ページの「キーログイン」を参照してください。

新しいセキュリティメカニズム資格を受け入れるようにする NIS+ サーバーの構成 - 例

この例では、現在の認証メカニズムは des で、新しいメカニズムは dh640-0 です。ここでは順序が重要です。クライアントおよびサーバーの NIS+ 認証では des より後のエントリは無視されます。

```
server# nisauthconf dh640-0 des
server# keylogin -r
   (画面上には何も表示されない)
server# /etc/reboot
```

新しいセキュリティメカニズム資格を使用するようにするワークステーションの構成

新しい資格を受け入れられるようになったので、ワークステーションを新しい資格によって認証するように変換することができます。そのためには、nisauthconf(1M) および keylogin(1) をスーパーユーザーとして実行し、リブートします。

新しいセキュリティメカニズム資格を使用するようにするワークステーションの構成 - 例

この例では、新しいメカニズムは dh640-0 ですが、システムは、dh640-0 資格が使用可能でないか、あるいは dh640-0 による認証に成功しなかった場合には、des 資格による認証も試みます。

```
workstation# nisauthconf dh640-0 des
workstation#  keylogin -r
   (画面上には何も表示されない)
workstation# /etc/reboot
```

次の例では、新しいメカニズムは dh640-0 で、このメカニズムによる認証だけが行われます。システムを新しいメカニズムだけで認証するように構成する前に、

キャッシュに書き込まれているディレクトリオブジェクトが新しいメカニズムの鍵を含むように、リフレッシュされることが必要です。これは、nisshowcache(1M) によって検証することができます。キャッシュに書き込まれているディレクトリオブジェクトがタイムアウトしてリフレッシュされるのを待つ代わりとして、次の方法があります。nis_cachemgr(1M) を終了し、続いて nisinit(1M) を使用して新しい NIS_COLD_START を構築し、続いて niscachemgr(1M) を再起動します。

ディレクトリオブジェクトの手動によるリフレッシュ - 例

ディレクトリオブジェクトを手動でリフレッシュするには、次のようにします。

```
# pkill -u 0 nis_cachemgr
# nisinit -cH masterserver
# niscachemgr -i
```

注意 - ワークステーション主体およびこのワークステーションのすべてのユーザーが cred テーブルの中に dh640-0 資格を持っていなければ、dh640-0 だけで認証を行うようにシステムを構成することはできません。

```
workstation# nisauthconf dh640-0
workstation#  keylogin -r
   (画面上には何も表示されない)
workstation# /etc/reboot
```

新しい資格を保護するパスワードの変更

ダミーパスワードを持った新しい資格を与えられた各ユーザーは、chkey(1) を実行してログインパスワードに変換することが必要です。詳細については、159ページの「NIS+ 主体の鍵の変更」を参照してください。

新しい資格を保護するパスワードの変更 - 例

chkey(1) を実行してログインパスワードに変換します。

```
# chkey -p
    (画面上には何も表示されない)
```

新しいセキュリティメカニズム資格だけを受け入れるようにするサーバーの構成

低グレードのセキュリティメカニズムから高グレードのものに変換する場合には、新しい高グレードのセキュリティメカニズムタイプの資格だけを受け入れるように NIS+ サーバーを構成することによって、最大限のセキュリティが達成されます。新旧両方のメカニズムによって認証を行うようにサーバーを構成した後で、変換します。

新しいメカニズムだけを使用して認証を行うようにシステムを構成する前に、キャッシュに書き込まれているディレクトリオブジェクトをリフレッシュして、新しいメカニズムの鍵を含むようにし、nisshowcache(1M) を使用してこれを検証します。

新しいセキュリティメカニズム資格だけを受け入れるようにするサーバーの構成 - 例

各 NIS+ サーバー上で nisauthconf(1M) を実行し、リブートします。この例では、NIS+ サーバーは dh640-0 資格の認証だけを受け入れるように構成されます。

```
server#  nisauthconf dh640-0
server# /etc/reboot
```

この段階で、オプションで、ディレクトリオブジェクトを更新して、古い公開鍵を削除できます。これはマスターサーバーから行う必要があり、新しいセキュリティメカニズムだけを使用して認証を行うサーバーが管理するディレクトリごとに一度、nisupdkeys(1M) を実行しなければなりません。この例では、更新されるディ

レクトリは、doc.com、org_dir.doc.com.、および groups_dir.doc.com. です。

```
masterserver# nisupdkeys doc.com.
   (画面上には何も表示されない)
masterserver#  nisupdkeys org_dir.doc.com.
   (画面上には何も表示されない)
masterserver# nisupdkeys groups_dir.doc.com.
```

cred テーブルからの古い資格の削除

必要に応じて、cred テーブルから古いセキュリティメカニズムの資格を削除できます。そのためには、ローカルドメインの cred テーブルに対する変更権が必要です。詳細は、「資格情報の削除」の項を参照してください。

cred テーブルからの古い資格の削除 - 例

この例では、主体 morena.doc.com が自分の des 資格を cred テーブルから削除します。

```
master# nisaddcred -r morena.doc.com. dh192-0
```

第 10 章

NIS+ のアクセス権の管理

この章では、NIS+ アクセス権とその管理方法について説明します。

注 - NIS+ セキュリティタスクには、Solstice AdminSuite ツールがあればより簡単に実行できるものがあります。

NIS+ アクセス権

NIS+ アクセス権とは、どの NIS+ ユーザーがどの操作を行うことができるか、どの情報にアクセスすることができるかを決めたものです。ここでの説明は、読者が NIS+ セキュリティシステム全般に関する知識を有し、NIS+ セキュリティシステムにおけるアクセス権の役割を理解していることを前提としています (詳細は第 6 章を参照)。

NIS+ アクセス関連のコマンドとその構文、オプションについては、`nis+(1)` のマニュアルページを参照してください。

承認およびアクセス権について

承認とアクセス権が NIS+ 資格および認証とどのように関連して NIS+ 名前空間のセキュリティを維持しているかについては、122ページの「NIS+ の承認とアクセス - 紹介」および第 6 章を参照してください。

承認クラス - 復習

122ページの「承認クラス」に説明のあるように、NIS+ のアクセス権は各クラスに与えられるものです。4 つの NIS+ クラスが用意されています。

- 「所有者」

  所有者クラスは 1 つの NIS+ 主体です。デフォルトではオブジェクトの所有者は、オブジェクトを作成した主体になります。しかし、オブジェクトの所有者は所有権を別の主体に譲ることで所有者を変更できます。

- 「グループ」

  グループクラスは 1 つ以上の NIS+ 主体の集まりです。1 つの NIS+ オブジェクトは 1 つだけの NIS+ グループを持つことができます。

- 「その他」

  その他クラスには、NIS+ に認証された NIS+ 主体のすべて (すべての所有者クラスとグループクラス、および有効な DES 資格を提示したものすべて) が含まれます。

- 「未認証」

  未認証クラスは、すべての適切な認証を受けられなかったもので構成されます。すなわち、有効な DES 資格を提示できなかったすべてのものです。

アクセス権 - 復習

126ページの「NIS+ のアクセス権について」で詳述したように、NIS+ には 4 種類のアクセス権があります。

- 「読み取り」

  オブジェクトに対する読み取り権を持った主体がオブジェクトの内容を読み取れます。

- 「変更」

  オブジェクトに対する変更権を持った主体がオブジェクトの内容を変更できます。

- 「削除」

  オブジェクトに対する削除権を持った主体がオブジェクトを削除できます。

- 「作成」

  上位レベルのオブジェクトに対する作成権を持った主体が、そのレベルの下位に新規のオブジェクトを作成できます。つまり、NIS+ ディレクトリオブジェクトに対して作成権を持っていれば、そのディレクトリ内に新規のテーブルを作成できます。NIS+ テーブルに対する作成権があれば、そのテーブル内に新規の列とエントリを作成できます。

これらの権限は論理的には、ディレクトリ、テーブル、列とエントリのように下位に展開するものであることを銘記してください。たとえば、新規にテーブルを作成するには、そのテーブルを格納する NIS+ ディレクトリオブジェクトに対する作成権が必要です。そのテーブルを作成した場合、そのデフォルト所有者になります。所有者として、自分自身にそのテーブルに対する作成権を与え、テーブルに新規のエントリを作成できます。テーブル内に新規のエントリを作成した場合、それらのエントリの所有者になります。テーブルの所有者として、テーブルレベルの作成権を他の人に与えることもできます。たとえば、自身のテーブルのグループクラスのテーブルレベルの作成権を与えることができます。その場合、テーブルのグループのすべてのメンバーがテーブル内に新規のエントリを作成できます。新規のテーブルエントリを作成した、グループの個々のメンバーはそのエントリのデフォルト所有者になります。

アクセス権の連鎖

承認クラスは連鎖の関係にあります。つまり、上位クラスは通常下位クラスにも属しており、自動的に下位クラスの権限を得ることになります。次のように機能します。

- 「所有者クラス」

  オブジェクトの所有者はそのオブジェクトのグループに所属していても、いなくてもかまいません。所有者があるグループに属していると、そのグループに与えられている権限をすべて得ることになります。オブジェクトの所有者が自動的に、その他と未認証のクラスにも属することになり、自動的にこれら 2 つのクラスに与えられている権限を獲得することになります。

- 「グループクラス」

  オブジェクトのグループメンバーは自動的にその他と未認証クラスに所属します。したがって、グループメンバーは自動的にその他クラスと未認証クラスのメンバーがそのオブジェクトに持っている権限を獲得します。

- 「その他クラス」

  その他クラスは自動的に、未認証クラスがオブジェクトに対して持っている権限と同じ権限を持つことになります。

- 「未認証クラス」

  未認証クラスは、オブジェクトが未認証クラスに与えている権限を持つだけです。

この基本原則は、下位クラスのアクセス権は自動的に上位クラスに与えられるということです。つまり、上位クラスは下位クラスよりも多くの権限を持つことができ、権限が少なくなることはないということです (この原則の例外は、もし所有者がグループのメンバーでなければ、所有者の持っていない権限をグループクラスに与えることが可能になる場合)。

アクセス権の割り当てと変更方法

NIS+ オブジェクトを作成した時、NIS+ はそのオブジェクトに所有者クラスとグループクラスのアクセス権のデフォルトセットを与えます。デフォルトでは、所有者はそのオブジェクトを作成した NIS+ 主体です。デフォルトのグループは `NIS_GROUP` の環境変数で指定されたグループになります (詳細は、188ページの「デフォルトのアクセス権」を参照)。

異なるデフォルト権限の指定

NIS+ は NIS+ オブジェクトが作成された時に自動的に付与されたデフォルト権限を変更する 2 つの異なった方法を提供しています。

- NIS_DEFAULTS 環境変数。NIS_DEFAULTS はセキュリティに関するデフォルト値を保管し、その 1 つはアクセス権です。このデフォルトアクセス権は、オブジェクトが作成された時にオブジェクトに自動的に付与されるものです。(詳細は、194ページの「NIS+ デフォルトの表示 - nisdefaults コマンド」を参照してください。)

  NIS_DEFAULTS 環境変数の値を変更すると、変更後に作成されたオブジェクトに新規の値が与えられます。しかし、以前に作成されたオブジェクトは影響を受けません。

- –D オプションはいくつかの NIS+ コマンドに用いられます。NIS+ オブジェクトを作成するコマンドに –D オプションを使用すると、NIS_DEFAULTS 環境変数が指定したデフォルト権限を上書きします。(詳細は、198ページの「デフォルトを無効にする」を参照してください。

既存オブジェクトへのアクセス権を変更する

NIS+ オブジェクトを作成する場合、既存のデフォルトアクセス権 (NIS_DEFAULTS 環境変数か –D オプションの指定かのいずれかによる) に対処する必要があります。デフォルト権限を変更するコマンドは次のとおりです。

- nischmod コマンド
- テーブルの列の場合は nistbladm コマンド

テーブル、列、およびエントリのセキュリティ

NIS+ テーブルに対するアクセス権を指定する方法には、次の 3 通りあります。

- 「テーブル」全体を対象にアクセス権を指定する
- 「エントリ」(行) 単位でアクセス権を指定する
- 「列」単位でアクセス権を指定する

列とエントリ (行) が交差する部分をフィールドといいます。データ値はすべてこの交差領域、つまりフィールドに入力します。

列とエントリレベルのアクセス権を持っていると、テーブルレベルのアクセス権の制限があっても個々の行と列にアクセスできますが、テーブル全体へのアクセス権以上に列とエントリレベルの権限を制限することはできません。

- 「テーブル」

  テーブルレベルは基本的なレベルです。テーブルレベルに付与されたアクセス権は、列ごとまたはエントリごとに特に変更された場合を除き、テーブル内のすべての部分に適用されます。テーブルレベルの権限は最も厳格であるべきです。

注 - 承認クラスは連結しているということに注意してください。上位クラスは、下位クラスに割り当てられた権利を取得していることになります。180ページの「アクセス権の連鎖」を参照してください。

- 「列」

  列レベルの権限を持っていると、列ごとに追加アクセス権を持つことになります。たとえば、その他クラスと未認証クラスにテーブルレベルの権限が何も付与されていなかったとします。この場合、この 2 つのクラスはテーブル内のデータに対して、読み取り権、変更権、作成権、削除権を持ちません。列レベルの権限を持てば、テーブルレベルの権限の制限を超えて、その他クラスのメンバーであっても特定の列のデータを読み取ることができます。

  一方、所有者クラスとグループクラスにテーブル全体の読み取り権が付与されている場合、列レベルの権限を使ってグループクラスの列の読み取り権を妨げることはできません。

  列のグループはテーブルのグループやエントリのグループと同じにはなりません。これらはまったく異なるグループです。

- 「エントリ (行)」

  エントリレベルの権限があれば、行ごとに追加アクセス権を持つことになります。たとえば、個々のユーザーに指定したエントリに限って変更する権限を与えることができるのです。

  エントリのグループはテーブルのグループとは同じである必要はなく、別のグループにできます。そうすることによって、特定のグループのメンバーに、他のグループに属するエントリに影響を与えないで、あるエントリのセットにアクセスする権限を付与できます。

テーブル、列、エントリの例

列またはエントリレベルのアクセス権は、追加アクセスを次の 2 つの方法で提供できます。1 つは権限を持つ主体を増やす方法で、もう 1 つは同じ主体に権限を追加する方法です。もちろん、これらを組み合わせることも可能です。以下にその例を示します。

テーブルオブジェクトが、そのテーブルの所有者に対して読み取り権を与えたとします。

表 10–1 テーブル、列、エントリの例 1

	未認証	所有者	グループ	その他
テーブルのアクセス権	—-	r—	—-	—-

このことは、テーブルの所有者だけがテーブル全体の内容を読み取れることを意味します。所有者でない主体は、アクセスできません。テーブルのエントリ 2 にグループクラスに対して読み取り権を与えることができます。

表 10–2 テーブル、列、エントリの例 2

	未認証	所有者	グループ	その他
テーブルのアクセス権	----	r---	----	—-
エントリ 2 のアクセス権	----	----	r---	—-

テーブルの内容をすべて読み取れるのは所有者だけですが、このテーブルのグループのメンバーであれば、この特定エントリの内容を読み取ることができます。次に、特定の列がその他クラスに読み取り権を与えたとします。

表 **10–3** テーブル、列、エントリの例 3

	未認証	所有者	グループ	その他
テーブルのアクセス権	`----`	`r---`	`----`	`----`
エントリ 2 のアクセス権	`----`	`----`	`r---`	`----`
列 1 のアクセス権	`----`	`----`	`----`	`r---`

その他のクラスのメンバーは列 1 のすべてのエントリを読み取ることができます (表 10–4 の薄い影の部分)。グループクラスのメンバーは (その他クラスにも属しているので) 列 1 のすべてとエントリ 2 の全列を読み取ることができます (表 10–4 の濃い影の部分)。`*NP*` になっている列は、「グループ」、「その他」いずれのクラスも読み取りができません (どちらにもアクセス権がない)。

表 **10–4** テーブル、列、エントリの例 4

	列 **1**	列 **2**	列 **2**
エントリ 1	読み取り	`*NP*`	`*NP*`
エントリ 2	読み取り	読み取り	読み取り
エントリ 3	読み取り	`*NP*`	`*NP*`
エントリ 4	読み取り	`*NP*`	`*NP*`
エントリ 5	読み取り	`*NP*`	`*NP*`

異なるレベルの権限

この節では 4 つの権限 (読み取り、作成、変更、削除) が 4 つのアクセスレベル (ディレクトリ、テーブル、列、エントリ) とどのようにかかわるのかを説明します。

種々の権限とレベルに関係した機能を次の表 10–5 にまとめます。

表 10–5 アクセス権、アクセスレベル、およびその機能

	ディレクトリ	テーブル	列	エントリ
読み取り	ディレクトリ内容のリスト	テーブル内容の読み取り	列内容の読み取り	エントリ (行) 内容の読み取り
作成	新規ディレクトリまたはテーブルオブジェクトの作成	新規エントリ (行) の追加	列に新規のデータを入力	エントリ (行) に新規のデータを入力
変更	オブジェクトの移動とオブジェクト名の変更	テーブル内の任意のデータを変更	列内のデータを変更	エントリ (行) 内のデータを変更
削除	テーブル等のディレクトリオブジェクトの削除	エントリ (行) の削除	列内のデータの削除	エントリ (行) 内のデータの削除

読み取り権

- 「ディレクトリ」

  ディレクトリの読み取り権があれば、ディレクトリの内容を表示できます。

- 「テーブル」

  テーブルの読み取り権があれば、テーブル内のすべてのデータを読み取れます。

- 「列」

  列の読み取り権があれば、その列のすべてのデータを読み取れます。

- 「エントリ」

  エントリの読み取り権があれば、そのエントリのすべてのデータを読み取れます。

作成権

- 「ディレクトリ」

  ディレクトリレベルの作成権があれば、テーブル等の新規オブジェクトをディレクトリ内に作成できます。

- 「テーブル」

  テーブルレベルの作成権があれば、テーブル内に新規のエントリを作成できますが、列は作成できません。テーブルレベルの作成権だけでは、テーブルに新規のエントリを追加できますが、新規の列を追加することはできません。

- 「列」

  列の作成権があれば、列内のフィールドに新規のデータを入力できます。新規の列を作成できません。

- 「エントリ」

  エントリの作成権があれば、その行のフィールドに新規のデータを入力できます。(エントリレベルの作成権では新規の行を作成することはできません。)

変更権

- 「ディレクトリ」

  ディレクトリレベルの変更権があれば、ディレクトリオブジェクトの移動と名前の変更ができます。

- 「テーブル」

  テーブルレベルの変更権があれば、テーブル内のデータをすべて変更できます。新規の行を作成 (追加) できますが、新規の列は作成できません。空白フィールドにデータを入力することも可能です。

- 「列」

  列の変更権があれば、その列の任意のフィールドのデータを変更できます。

- 「エントリ」

  エントリの変更権があれば、その行の任意のフィールドのデータを変更できます。

削除権

- 「ディレクトリ」

  ディレクトリレベルの削除権があれば、テーブル等のディレクトリ内の既存オブジェクトを削除できます。

- 「テーブル」

テーブルレベルの削除権があれば、テーブル内の既存のエントリ (行) を削除できますが、列は削除できません。削除できるのは既存のエントリだけで、既存の列は削除できません。

- 「列」

  列の削除権があれば、その列の任意のフィールドのデータを削除できます。

- 「エントリ」

  エントリの削除権があれば、その行の任意のフィールドのデータを削除できます。

アクセス権の格納場所

オブジェクトのアクセス権は、そのオブジェクトの定義の一部として指定され、格納されます。この情報は NIS+ テーブルには格納されません。

NIS+ オブジェクトのアクセス権の読み取り

アクセス権を読み取るには `niscat` コマンドを使用します。

```
niscat -o objectname
```

アクセス権を読み取るオブジェクト名を指定します。

このコマンドは、NIS+ オブジェクトに関する次の情報を返します。

- 「所有者」

  所有権を持っている NIS+ 主体。通常はオブジェクトを作成した人ですが、元の所有者が所有権を渡した場合もある

- 「グループ」

  オブジェクトの NIS+ 主体

- 「未認証クラスのアクセス権」

  認証された (有効な DES 資格を提示した) か否かにかかわらず、全員が持つ権限

- 「所有者クラスのアクセス権」

  オブジェクトの所有者に付与されたアクセス権

- 「グループクラスのアクセス権」

オブジェクトのグループに付与されたアクセス権

- 「その他クラスのアクセス権」

  認証された NIS+ 主体全てに付与されたアクセス権

4 つの承認クラスのアクセス権は、次のように 16 文字の文字列で表示されます。

```
r---rmcdr---r---
```

各文字がアクセス権の種類を表します。

- r は読み取り権
- m は変更権
- d は削除権
- c は作成権
- - はアクセス権のないことを表す

先頭の 4 文字は未認証に、次の 4 文字は所有者に、その次の 4 文字はグループに、そして最後の 4 文字はその他に、それぞれ与えられたアクセス権を表します。

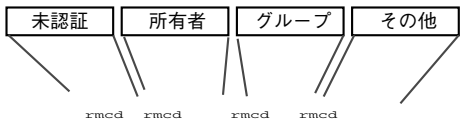


図 10–1　アクセス権の表示

注 - UNIX ファイルシステムとは異なり、先頭のアクセス権は未認証用であり、所有者用ではありません。

デフォルトのアクセス権

オブジェクトを作成すると、NIS+ はそのオブジェクトにデフォルトの所有者、グループ、および アクセス権を割り当てます。デフォルトの所有者は、そのオブジェクトを作成する NIS+ 主体です。デフォルトのグループは、環境変数 NIS_GROUP の中で名前をつけられたグループです。デフォルトのアクセス権は次のようになります。

表 10–6 デフォルトのアクセス権

未認証	所有者	グループ	その他
-	読み取り	読み取り	読み取り
-	変更	-	-
-	作成	-	-
-	削除	-	-

環境変数 NIS_DEFAULTS のセットがある場合、NIS_DEFAULTS 内の値が新規のオブジェクトに適用されるデフォルト値を決定します。コマンド行でオブジェクトを作成した場合は、–D フラグを使ってデフォルト値以外を設定できます。

テーブルに対するアクセス権をサーバーが割り当てる方法

この節では、読み取り、変更、削除、作成の操作が行われる際、テーブルオブジェクト、エントリ、列に対するアクセス権をサーバーが各クラスにどのように割り当てるかということについて説明します。

注 - セキュリティレベル 0 では、サーバーはアクセス権を実行しないため、すべてのクライアントがテーブルオブジェクトに対する完全なアクセス権を付与されます。セキュリティレベル 0 は管理者だけがテストの目的で使用します。通常の業務にはレベル 0 を使用しないでください。

サーバーがアクセスを許可するか否かを決定する 4 つの要素があります。

- 主体が要求する処理の種類
- 主体がアクセスしようとしているテーブル、エントリ、または列
- その特定のオブジェクトに対して、主体が所属する承認クラス
- テーブル、エントリ、または列がその主体の承認クラスに割り当てたアクセス権

  認証後に主体は、主体が有効な DES 資格を所持しているかを確認することで要求を行い、NIS+ サーバーは処理の種類と要求のオブジェクトを決定します。
- 「ディレクトリ」

オブジェクトがディレクトリかグループの場合、サーバーは 4 つのクラスに付与されている権限を知るためにオブジェクトの定義をチェックし、どのクラスに主体が所属しているか判定し、主体のクラスとそのクラスに付与された権限に基づいて、その要求を受け入れるかまたは拒否します。

- 「テーブル」

  オブジェクトがテーブルの場合、サーバーは 4 つのクラスに付与されているテーブルレベルの権限を知るためにテーブルの定義をチェックし、どのクラスに主体が所属しているか判定します。所属しているクラスが要求処理を行うテーブルレベルの権限を持っていない場合は、サーバーは次にどの行または列にかかわる処理かを判定し、要求処理に必要な該当する行または列レベルのアクセス権があるかを決定します。

コマンドによるアクセス権の指定

ここでの説明では、NIS+ 環境のセキュリティレベルが 2 (デフォルト) であるものと想定しています。

この節では、この章で説明するコマンドを使用するときにアクセス権や所有者、グループ所有者、オブジェクトを指定する方法を説明します。

アクセス権の構文

この節では、承認とアクセス権に関係する NIS+ コマンドに使われるアクセス権の構文について説明します。

クラス、演算子、および権限の構文

アクセス権は、環境変数で指定する場合もコマンドで指定する場合も、「クラス (class)」、「演算子 (operator)」、「権利 (right)」という 3 種類の引数で区別されます。

- 「クラス」

  クラスは、「権利」が適用される NIS+ 主体のカテゴリを意味します。

表 10–7 アクセス権の構文 - クラス

クラス	説明
n	未認証: すべての未認証要求
o	オブジェクトまたはテーブルエントリの所有者
g	オブジェクトまたはテーブルエントリのグループ所有者
w	その他: すべての認証済み主体
a	すべて: 所有者、グループ、およびその他の省略形。これはデフォルト

- 「演算子」

  演算子は、「権限」について行われる操作を示します。

表 10–8 アクセス権の構文 - 演算子

演算子	説明
+	「権利」によって指定されたアクセス権を追加する
-	「権利」によって指定されたアクセス権を取り消す
=	この演算子権利によって指定されたアクセス権に変更する。つまり、既存の「権利」をすべて取り消し、新しいアクセス権に置き換える

- 「権限」

  アクセス権そのものです。使用可能な値は次のとおりです。

表 10–9 アクセス権の構文 - 権限

権利	説明
r	オブジェクト定義またはテーブルエントリを読み取る
m	オブジェクト定義またはテーブルエントリを変更する

表 10–9　アクセス権の構文 - 権限　続く

権利	説明
c	テーブルエントリまたは列を作成する
d	テーブルエントリまたは列を削除する

コンマ (,) で区切ることで、複数のコマンドを 1 つのコマンド行にまとめることができます。

表 10–10　クラス、演算子、権限の構文 - 例

操作	構文
読み取りアクセス権を「所有者」クラスに追加する	o+r
変更アクセス権を所有者、グループ、およびその他のクラスに追加する	a=m
読み取りと変更の権限をその他と未認証クラスに追加する	wn+m
グループ、その他、および未認証クラスから 4 つの権限をすべて削除する	gwn-rmcd
所有者クラスに作成と削除の権限を追加し、その他と未認証クラスに読み取り権と変更権を追加する	o+cd,wn+rm

所有者とグループの構文

- 「所有者」

  NIS+ グループを指定するには、NIS+ グループ名にドメイン名を付けて使います。

- 「グループ」

  NIS+ グループを指定するには、NIS+ グループ名にドメイン名を付けて使います。

  主体名は完全指定されていることに注意してください。

(*principalname.domainname*)

所有者

principalname

グループ

groupname.domainname

オブジェクトとテーブルエントリの構文

オブジェクトとテーブルエントリは異なる構文を使います。

- オブジェクトは単純なオブジェクト名を使います。
- テーブルエントリはインデックス付きの名前を使います。

オブジェクト

objectname

テーブルエントリ

[*columnname=value*] , *tablename*

注 - この場合、角括弧 ([ ])は構文の一部であり、オプション記号ではありません。

インデックス付きの名前では、列と値のペアを複数指定できます。その場合、操作はすべての列と値のペアに一致するエントリにだけ適用されます。列と値のペアが増えると、検索条件が厳しくなります。

以下に例を示します。

表 10–11 オブジェクトとエントリの構文

型	例
オブジェクト	`hosts.org_dir.sales.doc.com.`
テーブルエントリ	`'[uid=33555],passwd.org_dir.Eng.doc.com.'`
2 つの値のテーブルエントリ	`'[name=sales,gid=2],group.org_dir.doc.com.'`

列はインデックス付きの名前の特殊な形式になっています。列の操作は `nistbladm` コマンドでだけ可能なため、詳細は、280ページの「nistbladm コマンド」を参照してください。

NIS+ デフォルトの表示 - nisdefaults コマンド

`nisdefaults` コマンドは、名前空間内で現在有効な 7 つのデフォルトを表示します。これらのデフォルトは次のいずれかです。

- NIS+ ソフトウェアが提供するプリセット値
- 変数 `NIS_DEFAULTS` 値で指定 されるデフォルト (変数 `NIS_DEFAULTS` のセットがある 場合)

オブジェクトを作成する時に `–D` オプション付きのコマンドを使って上書きしなければ、このマシン上でオブジェクトを作成すると自動的にデフォルト値を獲得することになります。

表 10–12 7 つの NIS+ デフォルト値と `nisdefaults` オプション

デフォルト	オプション	元データ	説明
ドメイン	`–d`	`/etc/defaultdomain`	コマンドを入力したワークステーションのホームドメインを表示する
グループ	`–g`	環境変数 `NIS_GROUP`	このシェルが作成する次のオブジェクトに付与されるグループを表示する

表 10–12 7 つの NIS+ デフォルト値と nisdefaults オプション 続く

デフォルト	オプション	元データ	説明
ホスト	–h	uname –n	ワークステーションのホスト名を表示する
主体	–p	gethostbyname()	nisdefaults コマンドを入力した NIS+ 主体の完全指定ユーザー 名またはホスト名を表示する
アクセス権	–r	NIS_DEFAULTS 環境変数	このシェルが作成する次のオブジェクトまたはエントリに付与されるアクセス権を表示する。 フォーマット：----rmcdr---r---
検索パス	–s	環境変数 NIS_PATH	検索パスの構文を表示する。これは NIS+ が情報を検索する時のドメインを示す。もし設定してあれば、環境変数 NIS_PATH の値を表示する
生存期間	–t	環境変数 NIS_DEFAULTS	このシェルが作成する次のオブジェクトに付与される生存期間を表示する。デフォルトは 12 時間
全部 (簡潔)	–a		7 つのデフォルトすべてを簡潔フォーマットで表示する
冗長	–v		指定した値を冗長モードで表示する

これらのオプションを使用して、すべてのデフォルト値もしくはそのサブセットを表示できます。

- すべての値を冗長フォーマットで表示するには、引数なしで nisdefaults コマンドを実行します。以下に例を示します。

```
master% nisdefaults
Principal Name : topadmin.doc.com.
Domain Name : doc.com.
Host Name : rootmaster.doc.com.
Group Name : salesboss
Access Rights : ----rmcdr---r---
Time to live : 12:00:00:00:00
Search Path : doc.com.
```

- すべての値を簡潔フォーマットで表示するには、–a オプションを付けます。

- 値のサブセットを表示するには、適切なオプションを使用します。その値は簡潔モードで表示されます。権限と検索パスのデフォルトを簡潔モードで表示する例を次に示します。

```
rootmaster% nisdefaults -rs
----rmcdr---r---
doc.com.
```

- 値のサブセットを冗長モードで表示するには、–v フラグを使用します。

デフォルトセキュリティ値の設定

この節では、nisdefaults コマンド、環境変数 NIS_DEFAULTS、および –D オプションに関連したタスクを実行する方法を説明します。環境変数 NIS_DEFAULTS は次のデフォルト値を指定します。

- 所有者
- グループ
- アクセス権
- 生存期間

環境変数 NIS_DEFAULTS に設定した値はデフォルトとなり、そのシェルを使用して作成したすべての NIS+ オブジェクトに適用されます (–D オプション付きでコマンドを実行してデフォルト値に上書きした場合を除く)。

環境変数 NIS_DEFAULTS を指定することで、デフォルト値 (所有者、グループ、アクセス権、および生存期間) を指定できます。一度 NIS_DEFAULTS の値を設定するとそのシェルから作成したすべてのオブジェクトは、–D オプション付きでコマンドを実行して上書きした場合を除きそのデフォルトに設定されます。

NIS_DEFAULTS の値を表示する

echo コマンドを使って、環境変数の値をチェックできます。以下にその例を示します。

```
client% echo $NIS_DEFAULTS
owner=butler:group=gamblers:access=o+rmcd
```

nisdefaults コマンドを使用して、名前空間でアクティブな NIS+ デフォルトの一般的リストを表示することも可能です。194ページの「NIS+ デフォルトの表示 - nisdefaults コマンド」を参照してください。

デフォルトを変更する

環境変数 NIS_DEFAULTS の値を変更することで、アクセス権、所有者、およびグループのデフォルトを変更できます。ユーザーのシェルに適切な環境コマンド (csh には setenv、sh と ksh には $NIS_DEFAULTS=, export) を次の引数を付けて使用します。

- access=アクセス権 (190ページの「コマンドによるアクセス権の指定」で説明されたフォーマットを使って)
- owner=所有者名
- group=グループ名

複数の引数をまとめる場合は、コロン (:) で区切ります。

owner= 主体名 :group= グループ名

表 10–13 に例をいくつか示します。

表 10–13 デフォルトの変更例

作業	例
所有者のデフォルトアクセス権に読み取り権を設定	client% **setenv NIS_DEFAULTS access=o+r**
デフォルトの所有者をホームドメインが doc.com. である abe に設定	client% **setenv NIS_DEFAULTS owner=abe.doc.com.**
2 つのコード行をまとめる	client% **setenv NIS_DEFAULTS access=o+r:owner=abe.doc.com.**

デフォルトを変更したシェルから作成されるすべてのオブジェクトとエントリは、指定した新規の値になります。テーブルの列またはエントリに対してはデフォルトを指定できません。列とエントリはデフォルトをそのまま継承します。

NIS_DEFAULTS の値を再設定する

変数 NIS_DEFAULTS をオリジナルの値に再設定するには、ユーザーのシェルに適したフォーマットを使って、引数なしで変数の名前を入力します。

C シェルの場合

```
client# unsetenv NIS_DEFAULTS
```

Bourne シェルまたは Korn シェルの場合

```
client$ NIS_DEFAULTS=; export NIS_DEFAULTS
```

デフォルトを無効にする

次の NIS+ コマンドのどれかを使えば、NIS+ オブジェクトまたはテーブルエントリを作成する時に、いつでもデフォルトのアクセス権、所有者、およびグループを無効にできます。

- nismkdir - NIS+ ディレクトリオブジェクトを作成
- nisaddent - エントリを NIS+ テーブルに転送
- nistbladm - NIS+ テーブル内にエントリを作成

デフォルト値を無効にするには、190ページの「コマンドによるアクセス権の指定」の説明のように、コマンドの構文に –D オプションを挿入します。

デフォルトを設定する時と同様、複数の引数を 1 行のコマンド行で指定できます。なお、列とエントリの所有者とグループは常に同じであるため、これらを無効にすることはできません。

nismkdir コマンドを使用して、sales.doc.com ディレクトリを作成し、デフォルトアクセス権を無効にして所有者にだけ読み取り権を付与するには、次のように入力します。

```
client% nismkdir -D access=o+r sales.doc.com
```

オブジェクトとエントリのアクセス権を変更する

nischmod コマンドは、NIS+ オブジェクトまたはテーブルエントリのアクセス権を変更します。テーブル列のアクセス権は操作しません。列の場合、nistbladm コマンドに –D オプションを付けて実行してください。nischmod コマンドを使用するには、そのオブジェクトかエントリに対する変更権が必要です。

nischmod コマンドを使用して権限を追加する

オブジェクトまたはエントリに権限を追加する例を次に示します。

オブジェクト

```
nischmod class+right object-name
```

テーブルエントリ

```
nischmod class+right [column-name=value],table-name
```

sales.doc.com. ディレクトリオブジェクトのグループに読み取り権と変更権を追加する場合は、次のように入力します。

```
client% nischmod g+rm sales.doc.com.
```

hosts.org_dir.doc.com. テーブル内の name=abe エントリのグループに読み取り権と変更権を追加する場合は、次のように入力します。

```
client% nischmod g+rm '[name=abe],hosts.org_dir.doc.com.'
```

nischmod を使用して権限を削除する

オブジェクトまたはエントリの権限を削除するには、次のように入力します。

オブジェクト

```
nischmod class-right object-name
```

エントリ

```
nischmod class-right [column-name=value], table-name
```

sales.doc.com. ディレクトリオブジェクトのグループから作成権と削除権を削除するには次のように入力します。

```
client% nischmod g-cd sales.doc.com.
```

たとえば、hosts.org_dir.doc.com. テーブル内の name=abe エントリのグループから削除権を削除する場合は、次のようにします。

```
client% nischmod g-d '[name=abe],hosts.org_dir.doc.com.'
```

列アクセス権を指定する

nistbladm コマンドは NIS+ テーブルに種々の操作を行うことができます。これについては 280ページの「nistbladm コマンド」にほとんどの説明があります。このコマンドは –c と –u の 2 つのオプションを使用してセキュリティ関連のタスクを実行できます。

- –c オプション。nistbladm コマンドを使用して テーブルを作成する際に、–c オプションを使用すると最初の列アクセス権を指定できます。
- –u オプション。–u オプションを nistbladm コマンドとともに使用すると、列アクセス権を変更できます。

テーブル作成時の列権限設定

テーブルが作成された時、列にはテーブルオブジェクトと同じ権限が付与されます。このテーブルレベルの権限は環境変数 NIS_DEFAULTS で指定されるか、またはテーブルを作成したコマンドの一部で指定されます。nistbladm コマンドでテーブルを作成する際に –c オプションを使用すれば、最初の列アクセス権を指定できます。このオプションを使用するには、テーブルを作成しようとするディレクトリの作成権が必要です。テーブル作成時に列権限を指定するには、次のように入力します。

```
nistbladm -c type ' columname=[flags] [, access]... tablename'
```

- *type* はテーブルの種類を示す文字列です。テーブルの *type* は何にでもできます。
- *columnname* は列名です。
- *flags* には、列の種類を指定します。使用できるフラグには次のものがあります。

- S (検索可能)
- I (大文字と小文字を区別しない)
- C (暗号化)
- B (2 進データ)
- X (XDR 符号化データ)

- *access* は、190ページの「コマンドによるアクセス権の指定」で説明した構文を使用したこの列のアクセス権です。
- ... はそれぞれの型と権限のセットを持った複数の列を指定できることを示しています。
- *tablename* は完全指定名で表した作成しようとするテーブル名です。

テーブルを作成する際、その列にはテーブルオブジェクトと同じ権限が付与されます。列に独自の権限を付与するには、列の型とコンマの後の各列の等号記号にアクセス権を追加し、列をスペースで区切ります。

```
column=type,    rights column= type, rights column =    type, rights
```

以下の例では、div 特性で depts という名前のテーブルが doc.com ディレクトリに作成されます。列は Name、Site および Manager の 3 つです。第 2、第 3 列のグループには、変更権が追加されます。

```
rootmaster% nistbladm -c div Name=S Site=S,g+m Manager=S,g+m depts.doc.com.
```

nistbladm –c オプションの詳細は、第 14 章を参照してください。

既存のテーブル列に権限を追加する

既存の NIS+ テーブルの列にアクセス権を追加するには、nistbladm –u オプション付きの nistbladm コマンドを使用します。このオプションを使用する場合、テーブル列の変更権が必要です。入力例を次に示します。

```
nistbladm -u [column=    access,...],tablename
```

- *column* は列名です。
- *access* は、190ページの「コマンドによるアクセス権の指定」で説明した構文を使用したこの列のアクセス権です。

- ... はそれぞれの type と権限のセットを持った複数の列を指定できることを示しています。
- *tablename* は完全指定名で表した作成しようとするテーブル名です。

権限を更新する列ごとに *column=access* のペアを使用します。複数の列を更新するにはコンマ (,) で区切り全体を角括弧 ([]) で囲みます。

```
[column=access,  column=access,  column=access]
```

(このオプションの完全な構文については、280ページの「nistbladm コマンド」を参照してください。)

次の例は、hosts.org_dir.doc.com. テーブルの name および addr 列のグループに、作成権と変更権を追加するものです。

```
client% nistbladm -u '[name=g+rm,addr=g+rm],hosts.org_dir..doc.com.'
```

テーブル列から権限を削除する

NIS+ テーブル列のアクセス権を削除するには、上記 201ページの「既存のテーブル列に権限を追加する」で説明したように –u オプションを使用します。ただし + 記号を使って権限を追加するのではなく、− 記号を使って権限を削除します。

次の例は、hosts.org_dir.doc.com. テーブルの hostname 列のグループの読み取り権と変更権を削除するものです。

```
client% nistbladm -u 'name=g-rm,hosts.org_dir.doc.com.'
```

オブジェクトとエントリの所有権の変更

nischown コマンドは、1 つ以上のオブジェクトまたはエントリの所有者を変更します。このコマンドを使用するには、オブジェクトまたはエントリに対する変更権が必要です。テーブルの列の所有者はテーブルの所有者であるため、nischown コマンドでは列の所有者を変更できません。列の所有者を変更するには、テーブルの所有者を変更する必要があります。

nischown コマンドを使用してオブジェクトの所有者を変更する

オブジェクトの所有者を変更するには、次のように入力します。

```
nischown new-owner object
```

- *new-owner* はそのオブジェクトの新規の所有者のユーザー ID で完全指定します。
- *object* はそのオブジェクトの完全指定名です。

オブジェクト名と新規所有者名にドメイン名を必ず追加します。

次の例は、doc.com. ドメイン内の hosts テーブルの所有者を、ホームドメインが doc.com. でユーザー名 lincoln であるユーザーに変更するものです。

```
client% nischown lincoln.doc.com. hosts.org_dir.doc.com.
```

nischown コマンドを使用してテーブルエントリの所有者を変更する

テーブルエントリの所有者を変更する構文は、エントリを特定するのにインデックス付きエントリを使います。次に例を示します。この構文の詳細は、193ページの「オブジェクトとテーブルエントリの構文」を参照してください。

```
nischown new-owner [ column=value,...], tablename
```

- *new-owner* はそのオブジェクトの新規の所有者のユーザー ID で完全指定します。
- *column* は、所有者を変更するエントリ (行) を特定する値を持った列名です。
- *value* は、所有者を変更するエントリ (行) を特定するデータ値です。
- ... は所有者を変更する複数のエントリを示します。
- *tablename* は、所有者を変更するエントリを含むテーブルの完全指定名です。

所有者名とテーブル名にドメイン名を必ず追加します。

次の例は、doc.com. ドメインのホストテーブル内のエントリの所有者を、ホームドメインが doc.com. であるユーザー takeda に変更するものです。そのエントリの name 列の値は virginia です。

```
client% nischown takeda.doc.com. '[name=virginia],hosts.org_dir.doc.com.'
```

オブジェクトまたはエントリグループの変更

nischgrp コマンドは、1 つ以上のオブジェクトまたはテーブルエントリのグループ所有者を変更します。このコマンドを使用するには、オブジェクトまたはエントリに対する変更権が必要です。テーブルの列に割り当てられたグループは、テーブルに割り当てられたグループと同じであるため、nischgrp コマンドは、列のグループを変更できません。列のグループ所有者を変更するには、テーブル所有者を変更する必要があります。

nischgrp コマンドを使用してオブジェクトのグループを変更する

オブジェクトのグループを変更するには、次の構文を使用します。

```
nischgrp group object
```

ここで、

- *group* はオブジェクトの新規のグループの完全指定名です
- *object* はオブジェクトの完全指定名です

オブジェクト名と新規のグループ名にはドメイン名を必ず追加します。

次の例は、doc.com. ドメイン内の hosts テーブルのグループを admins.doc.com. に変更するものです。

```
client% nischgrp admins.doc.com. hosts.org_dir.doc.com.
```

nischgrp コマンドを使用してテーブルエントリのグループを変更する

テーブルエントリのグループを変更する構文は、エントリを識別するためにインデックス付きのエントリを使用します。(この構文については、193ページの「オブジェクトとテーブルエントリの構文」に詳細説明があります。) 以下に構文を示します。

```
nischgrp new-group [ column=value,...], tablename
```

- *new-group* はオブジェクトの新規グループの完全指定名です。
- *column* は変更される特定のエントリ (行) を識別する値を持った列名です。
- *value* は変更されるグループの特定の エントリ (行) を識別するデータ値です。
- *tablename* は変更ことになるグループのエントリを含むテーブルの完全指定名です。
- ... は複数のエントリで変更するグループを指定できることを示します。

新規グループ名とテーブル名にはドメイン名を必ず追加します。

次の例は、doc.com. ドメインのホストテーブル内のエントリのグループを sales.doc.com. に変更するものです。そのエントリの name 列の値は virginia です。

```
client% nischgrp sales.doc.com. '[name=virginia],hosts.org_dir.doc.com.'
```

第 11 章

パスワードの管理

この章では、一般ユーザー (NIS+ 主体) の観点から見た passwd コマンドの使用方法と、NIS+ 管理者によるパスワードシステムの管理方法を説明します。

- 229ページの「パスワードの強制的な変更」
- 230ページの「パスワードの変更禁止期間の設定」
- 229ページの「パスワードの有効期間の設定」
- 231ページの「警告期間の設定」
- 232ページの「パスワードの使用期間に関する設定の解除」
- 233ページの「パスワード使用権の有効期限」
- 235ページの「ログインの間隔の最大値の指定」
- 237ページの「パスワードの使用規則の設定 (およびそのデフォルト) 」
- 237ページの「/etc/defaults/passwd ファイル」
- 239ページの「パスワード変更時の制限」

注 - NIS+ セキュリティタスクには、Solstice AdminSuite ツールが利用可能であればもっと簡単に実行できるものがあります。

パスワードの使用

マシンへのログインの際には、ユーザー名 (「ログイン ID」とも呼ばれる)、およびパスワードを入力する必要があります。ログイン ID は公開の情報ですが、パスワードを知っているのは所有者だけです。

ログイン

システムへのログインは以下の手順で行います。

1. `Login:` プロンプトで、ログイン **ID** を入力します。
2. `Password:` プロンプトで、パスワードを入力します。

   (秘密を守るため、入力してもパスワードは画面に表示されません)

   ログインに成功すると、本日のメッセージ (ない場合もある)、続いてコマンド行プロンプト、ウィンドウシステム、通常のアプリケーションなどが表示されます。

Login incorrect メッセージ

「Login incorrect」というメッセージは以下のことを意味します。

- 「誤ったログイン ID あるいはパスワードを入力した」

  最も一般的な理由です。スペルを確認してもう一度やり直します。誤入力の回数は、ほとんどのシステムで「5」に制限されているので注意してください。

  - 誤入力の回数が制限を超えると、「Too many failures - try later」というメッセージが表示され、一定の時間が経過するまで再試行ができなくなります。
  - 指定された時間内にログインが成功しないと、「Too many tries; try again later」というメッセージが表示され、一定の時間が経過するまで再試行できなくなります。

- 「管理者がパスワードをロックしている」

  ログイン ID、パスワードともに正しく入力しているにもかかわらず「Login incorrect」メッセージが表示される場合は、システム管理者に問い合わせてください。

- 「パスワード使用権がシステム管理者によって取り消された」

  ログイン ID、パスワードともに正しく入力しているにもかかわらず「Login incorrect」メッセージが表示される場合は、システム管理者に問い合わせてください。

password expired メッセージ

このメッセージは、「パスワードが有効期限を過ぎている」ということを意味します。つまり、パスワードを作成してから時間が経ちすぎているので、すぐに作成し直す必要があるということです (新しいパスワードを作成する場合の必要条件については、212ページの「パスワードの選択」を参照してください)。

この場合、新しいパスワードの作成は以下の手順で行います。

1. Enter login password **(多少異なる場合がある)** プロンプトで、従来のパスワードを入力します。

   キー入力の内容は画面には表示されません。

2. Enter new password プロンプトで、新しいパスワードを入力します。

   キー入力の内容は画面には表示されません。

3. Re-enter new password プロンプトで、新しいパスワードをもう一度入力します。

   キー入力の内容は画面には表示されません。

will expire メッセージ

このメッセージ (あるいは「Your password will expire within 24 hours」というメッセージ) は、「パスワードが、*N* 日あるいは 24 時間以内に有効期限に達する」ということを意味します。

このメッセージが表示されたら、パスワードをすぐに変更する必要があります (210 ページの「パスワードの変更」を参照)。

Permission denied メッセージ

ログイン ID およびパスワードを入力したあと、このメッセージが表示されて login: プロンプトに戻った場合は、「管理者によってパスワードがロックされた、アカウントが取り消された、パスワード使用権の有効期限が過ぎたなどの理由で、ログインが正しく行われなかった」ということを意味します。このような場合、管理者がパスワードロックを解除するか、アカウントを復旧するまではログインができません。システム管理者に問い合わせてください。

パスワードの変更

セキュリティを確保するため、パスワードは定期的に変更してください (新しいパスワードを作成する場合の必要条件、および基準については、212ページの「パスワードの選択」を参照)。

注 - 現在の passwd コマンドでは、以前 nispasswd で行なっていた操作がすべて行えます。NIS+ の名前空間に特有な操作を行うには、passwd –r nisplus を使用します。

パスワードの変更は以下の手順で行います。

1. システムプロンプトから passwd コマンドを実行します。

2. Enter login password **(多少異なる場合がある)** プロンプトで、従来のパスワードを入力します。キー入力の内容は画面には表示されません。

- 「Sorry: less than *N* days since the last change」というメッセージが表示されたら、「現在使用中のパスワードは、作成されてからまだ十分な時間が経過していないため変更できない」ということを意味します。この場合はシステムプロンプトに戻ります。システム管理者に「パスワードは、作成後何日経過すれば変更できるのか」を問い合わせてください。
- 「You may not change this password」というメッセージが表示された場合は、変更がネットワーク管理者によって禁止されているということを意味します。

3. Enter new password プロンプトで、新しいパスワードを入力します

   キー入力の内容は画面には表示されません。

   この時点で、新しいパスワードが必要条件を満たしているかどうかがシステムによって確認されます。

   - 満たしている場合は再入力を求められます。
   - 満たしていない場合は、その旨を知らせるメッセージが表示されます。このときは、必要条件を満たす別のパスワードを入力する必要があります。

   パスワードの必要条件については、212ページの「パスワードの必要条件」を参照してください。

4. Re-enter new password プロンプトで、新しいパスワードを再入力します。

   キー入力の内容は画面には表示されません。

   1 回目と 2 回目で入力したパスワードが異なっていた場合は、手順 1 から繰り返すようにプロンプトが表示されます。

注 - root のパスワードを変更した場合は、その直後に必ず chkey –p を実行する必要があります(詳細は、161ページの「ルートからルート鍵を変更する」、および、162ページの「別のマシンからルート鍵を変更する手順」を参照してください)。root のパスワードを変更したあと chkey –p を実行しないと、root で正常にログインできなくなります。

パスワード変更の失敗

システムの中には、パスワード変更の試行回数、所要時間に制限を設けているものもあります (他の人が試行錯誤によって勝手にパスワードを変更してしまうのを防ぐため)。

パスワード変更、ログインの試行回数、または所要時間が指定の範囲を超えた場合は、「Too many failures - try later」、または
「Too many tries: try again later」といったメッセージが表示されます。この種のメッセージが表示されると、一定の時間 (システム管理者が指定) が経過するまでログイン、パスワードの変更は行えなくなります。

パスワードの選択

コンピュータのセキュリティの侵害の例としては、「他のユーザーのパスワードを推測によって盗む」というものが多くみられます。passwd コマンドには、パスワードを推測しにくいものにするための基準がいくつか設けられていますが、ユーザーに関していくつかの情報を得るだけでパスワードがわかってしまう人もいるのです。したがって、「自分にとって覚えやすく他人にとって推測しにくい」というのが良いパスワードです。逆に悪いパスワードとは、「自分にとって覚えにくく (メモしないと覚えられない)、自分を知る他人にとっては推測しやすい」というものです。

パスワードの必要条件

パスワードの必要条件は以下のとおりです。

- 「長さ」は 6 文字以上にします (デフォルト)。また意味を持つのは最初の 8 文字だけです (つまり、「8 文字より長いパスワードを作成することはできるが、システムは最初の 8 文字のみをチェックする」ということです)。パスワードの長さの最小値 (文字数) はシステム管理者によって変更される可能性があり、システムによっては 6 文字ではない場合があります。
- 「英文字」2 つ以上 (大文字、小文字どちらでも良い) と数字または記号 (@、#、% など) 1 つ以上で構成します。つまり、dog#food、dog2food といったパスワードは使用できますが、dogfood というパスワードは使用できません。
- 「ログイン ID と同じにならないようにします」。またログイン ID の文字を並べ替えたものも使用できません (この場合、大文字と小文字は同じものとして考える)。ログイン ID が Claire2 の場合、たとえば e2clair をパスワードとして使用できません。
- 「古いパスワードと同じにならないようにします」。古いパスワードを、少なくとも 3 文字以上入れ替える必要があります (この場合、大文字と小文字は同じものとして考える)。古いパスワードが Dog#fooD なら、新しいパスワードにたとえば dog#Meat は使用できますが、daT#Food は使用できません。

悪いパスワードの例

悪いパスワードの例としては以下のようなものが考えられます。

- 自分の名前をもとにしたもの
- 家族やペットの名前
- 運転免許証の番号
- 電話番号
- 社会保険の番号
- 従業員番号
- 趣味に関係のある名前
- 季節に関係のある名前 (たとえば 12 月に `Santa` を使うなど)
- 普通の辞書に載っている単語

普通に使われる言葉をそのままパスワードに使用しないようにします。辞書に載っていたら使えないものと思ってください。

良いパスワードの例

良いパスワードの例としては以下のようなものが考えられます。

- フレーズに数字や記号を加えたもの (例：`beam#meup`)
- フレーズを構成する単語の頭文字に番号や記号を加えた意味のない言葉 (例：SomeWhere Over The RainBow の頭文字に 7 を加えた `swotrb7`)
- 単語の一部を数字や記号に置き換えたもの (例：`snoopy` ではなく、`sn00py` にする)

パスワードの管理

ここでは、NIS+ 名前空間の中でパスワードを管理する方法について説明します。この説明は、読者が NIS+ セキュリティシステム全般に関する知識を有し、NIS+ セキュリティシステムにおけるログインパスワードの役割を理解していることを前提としています (詳細は、第 6 章を参照)。

注 - 現在の passwd コマンドでは、以前 nispasswd で行なっていた操作がすべて行えます。NIS+ の名前空間に特有な操作を行うには、passwd –r nisplus を使用します。

nsswitch.conf ファイルの必要条件

passwd コマンドの使用や、パスワードの使用期間に関する設定を正しく行うには、nsswitch.conf ファイル中の passwd エントリがすべてのマシンにおいて正しくなければなりません。passwd コマンドが「パスワード情報をどこに要求するか」および「パスワード情報をどこで更新するか」は、このエントリによって決定されます。

passwd エントリの設定として考えられるのは、以下の 5 種類だけです。

- passwd: files
- passwd: files nis
- passwd: files nisplus
- passwd: compat
- passwd: compat

  passwd_compat: nisplus

注意 - ネットワーク上のすべてのワークステーションの nsswitch.conf ファイルで、上記の passwd エントリの設定のうちいずれかを使用する必要があります。上記の 5 つ以外の設定をすると、ログインができなくなります。

nispasswd コマンド

現在の passwd コマンドでは、以前 nispasswd で行なっていた操作がすべて行えます。コマンド行で実行するときは、nispasswd ではなく passwd を使用します。

旧バージョンとの互換性を確保するため、nispasswd も完全な形で残っている点に注意してください。

yppasswd コマンド

現在の passwd コマンドでは、以前 yppasswd で行なっていた操作がすべて行えます。コマンド行で実行するときは、yppasswd ではなく passwd を使用します。

旧バージョンとの互換性を確保するため、yppasswd も完全な形で残っている点に注意してください。

passwd コマンド

passwd コマンドでは、パスワードに関する様々な操作が行えます。現在の passwd コマンドは、nispasswd コマンドの代わりとして使用できます。従来 nispasswd で行なっていた操作にも、passwd コマンドを使用するようにしてください。passwd コマンドのフラグ、オプション、引数の詳細は、マニュアルページを参照してください。

一般ユーザーが passwd コマンドで行える操作には以下のものがあります。

- 自らのパスワードの変更
- 自らのパスワード情報の表示

管理者が passwd コマンドで行える操作には以下のものがあります。

- 他のユーザーにログイン時のパスワード変更を強制する
- 他のユーザーのパスワードをロック (使用不可能に) する
- 「パスワード作成後、どのくらいの期間変更禁止にするか」を指定する
- 「パスワードが間もなく無効になる」という警告を出すタイミングを指定する
- パスワードの有効期間を指定する

passwd コマンドと nsswitch.conf ファイル

passwd などのコマンドがパスワード情報をどこから得て、どこに保存するのかということは、nsswitch.conf ファイルで設定します。nsswitch.conf ファイルの passwd エントリの設定に使用する文字列はそれぞれ以下のことを意味します。

- nisplus

  パスワード情報の獲得、変更、保存は、該当するドメインの passwd テーブルおよび cred テーブルで行う

- nis

パスワード情報の獲得、変更、保存は、passwd マップで行う

- files

  パスワード情報の獲得、変更、保存は、/etc/passwd ファイルおよび /etc/shadow ファイルで行う

passwd -r オプション

passwd コマンドの –r nisplus、–r nis、–r files といった引数は、nsswitch.conf ファイルの設定よりも優先されます。これらの引数をつけて passwd コマンドを実行すると、「nsswitch.conf ファイルより優先される」ということを知らせる警告メッセージが表示されます。警告メッセージ表示後も操作を続行すると、nsswitch.conf ファイルのシーケンスは無視され、–r によって指定された場所でパスワード情報の更新が行われます。

たとえば、nsswitch.conf ファイルにおいて passwd エントリが以下のように指定されている場合を考えてみましょう。

```
passwd: files nisplus
```

この場合、passwd コマンドを –r を使用しないで実行すると、パスワード情報のソース (情報の獲得、変更、保存が行われる場所) は /etc/passwd ファイルになります。しかし –r nisplus オプションを使用して passwd コマンドを実行すると、パスワード情報のソースは /etc/passwd ファイルから NIS+ の passwd テーブルに変更されます。

–r オプションは、「検索シーケンスが誤っていて nsswitch.conf ファイルが使用できない」という場合にのみ使用するようにします。たとえば、2 カ所に格納されているパスワード情報を更新する必要がある場合、1 つ目については nsswitch.conf ファイルで指定されているソースを使用できますが、2 つ目については別のソースを使用する必要があります。

```
Your specified repository is not defined in the nsswitch file!
```

このメッセージは、「パスワード情報の更新は、–r オプションで指定された場所において行われるが、nsswitch.conf ファイルでその場所がソースとして指定されるまでは更新の影響がまったく現れない」ということを意味します。たとえば、nsswitch.conf ファイルにおいて passwd: files nis という指定が行われているときに、–r nisplus オプションでパスワード使用期間の設定を行なったとします (つまり設定は NIS+ の passwd テーブルにおいて行われることになる)。パ

スワード情報のソースが nsswitch.conf ファイルにおいて NIS+ passwd テーブル以外の場所に指定されているため、この設定による影響はまったく現れません。

passwd コマンドと NIS+ 環境

この章で「NIS+ 環境とは」、「nsswitch.conf ファイルでパスワード情報のソースが nisplus に設定されている」、「passwd コマンドが –r nisplus という引数をつけて実行されている」という状況を指します。

passwd コマンドと資格

passwd コマンドは NIS+ 環境 (前節参照) で実行された場合、ユーザーに資格があってもなくても機能するよう設計されています。ただし資格のないユーザーが passwd コマンドで行えるのは、自らのパスワードの変更だけです。他のパスワード操作は、資格のある (認証された)、必要なアクセス権を持った (承認された) ユーザーだけが行えます。

passwd コマンドとアクセス権

承認およびアクセス権については、ユーザーがすべて適切な資格を持っているという前提で説明をします。

通常の NIS+ 環境では、passwd テーブルの所有者はいつでも制約なしにパスワード情報の変更ができます (デフォルトの場合)。つまり passwd テーブルの所有者は、読み取り、作成、変更、削除に関して完全に承認されている (アクセス権を与えられている) ということになります。また所有者は以下のことも行えます。

- nischown コマンドを使用して、テーブルの所有権を別のユーザーに与える
- テーブルのグループ、その他、未認証といったクラスに、読み取り権、作成権、変更権、削除権などを与える (この種の権利をその他クラス、未認証クラスに与えると NIS+ のセキュリティが弱くなる)
- 任意のクラスに与えられたアクセス権を、nisdefaults、nischmod、nistbladm などのコマンドを使用して変更する

注 - 与えられているアクセス権には関わりなく、その他クラス、未認証クラスのユーザーはすべてパスワードの使用期間の制約に従います。つまり、自分のパスワードであろうと他のユーザーのパスワードであろうと、作成されてから一定の時間が経過するまでは変更ができないということです。また有効期間の過ぎたパスワードを変更しなければならないという点も、グループ、その他、未認証といったクラスのメンバーすべてに共通です。しかしパスワード使用期間に関する上記のような制約は、passwd テーブルの所有者には適用されません。

NIS+ 環境で passwd コマンドを使用する場合、行おうとする操作に関する承認 (アクセス権) が必要です。

表 11–1 passwd コマンドに関するアクセス権

操作の種類	必要な権利	アクセス対象となるオブジェクト
情報を表示する	読み取り権	passwd テーブルのエントリ
情報を更新する	変更権	passwd テーブルのエントリ
情報を追加する	変更権	passwd テーブル

passwd コマンドと鍵

NIS+ 環境で passwd コマンドを使用して主体のパスワードを変更しようとすると、主体の非公開鍵が cred テーブル中で更新されます。

- cred テーブルの DES エントリに対してユーザーが変更権を持っていて、ログインパスワードと Secure RPC パスワードが同じであれば、passwd コマンドによって cred テーブルの非公開鍵が更新されます。
- cred テーブルの DES エントリに対する変更権がユーザーになく、ログインパスワードと Secure RPC パスワードが異なっている場合は、passwd コマンドによってパスワードだけが変更され、非公開鍵は変更されません。

  つまり cred テーブルの非公開鍵が、passwd テーブルに格納されているものとは異なったパスワードで作られることになります。この場合は、chkey コマンド

を実行する、ログイン後に毎回 keylogin を実行するといった方法で、鍵を変更する必要があります。

passwd コマンドと他のドメイン

他のドメインの passwd テーブルに対して操作をするには、passwd コマンドを以下のように使用します。

```
passwd [options] -D      domainname
```

nistbladm コマンド

nistbladm は、passwd テーブルをはじめとする NIS+ テーブルに関する情報を作成、変更、表示するのに使用するコマンドです。

注意 - nistbladm コマンドを使用してパスワード操作をするには、nistbladm を passwd テーブルのシャドウ列に適用する必要があります。nistbladm をシャドウ列に適用するのは複雑で微妙な作業になります。passwd コマンド、admintool、または Solstice AdminSuite ツールで容易に行える操作には、nistbladm コマンドを使用しない方が良いでしょう。

つまり以下のような操作には、nistbladm ではなく passwd コマンドや Solstice AdminSuite ツールを使用してください。

- パスワードの変更
- パスワードの使用期間の設定
- パスワード作成後から変更が可能になるまでの時間の設定
- 「パスワードが間もなく無効になる」という警告が表示されるタイミングの設定
- パスワードの使用期間に関する設定の解除

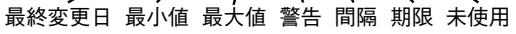
nistbladm コマンドには以下の機能があります。

- 新しい passwd テーブルエントリの作成
- 既存のエントリの削除
- passwd テーブル中の UID フィールド、GID フィールドの更新
- passwd テーブルの、アクセス権などセキュリティに関する属性の更新
- ユーザーアカウントに関して「どれだけの期間使用できるか」、「どれだけの期間使用されなければ無効になるか」を設定する (233ページの「パスワード使用権の有効期限」、235ページの「ログインの間隔の最大値の指定」を参照)

nistbladm と シャドウ列のフィールド

nistbladm コマンドでは、シャドウ列の様々なフィールドの値を指定することによってパスワードパラメータの設定を行います。シャドウ列のフィールドの値は、以下のような形式で設定します。

```
nistbladm -m shadow=n1:n2:n3:n4:n5:n6:n7 [name=login],passwd.orgd_ir
```



個々のフィールドの内容は以下のとおりです。

- 「n1 最終変更日」は、パスワードが最後に変更されたときの日付です。1970 年 1 月 1 日からの日数で表されます。このフィールドの値は、パスワードが変更されると自動的に更新されます (日数に関する詳細な情報は、222ページの「nistbladm と日数」に示されています)。このフィールドが空白であったり、指定されている値が 0 の場合は、過去に一度もパスワードの変更が行われていないことを意味します。

  このフィールドの値は、残りのフィールドの設定の基礎となることに注意してください。このフィールドの値が不当に変更されると、残りのフィールドの設定も不当なものになります。

- 「n2 最小値」は、パスワードが変更されてから、次の変更が可能になるまでの期間です。たとえば、パスワード作成時の最終変更日フィールドの値が 9201 (1970 年 1 月 1 日から 9201 日経過したことを示す) で、最小値フィールドの値が 8 だとすると、9209 日目まではパスワードの変更ができません (詳細は、230ページの「パスワードの変更禁止期間の設定」を参照)。

  この値としては以下の 3 とおりが考えられます。

  - 「0」は、変更禁止期間がないことを意味します。
  - 「0 より大きい数字」はこの数字がパスワードの変更禁止期間 (単位：日) になります。
  - 「最大値 (n3) より大きい値」はこのフィールド値が最大フィールドの値より大きい場合、パスワードを変更できません。変更しようとすると、`You may not change this password` というメッセージが表示されます。

- 「n3 最大値」は、パスワードが作成されてから、使用できなくなるまでの日数を示します。この日数を過ぎると、ログインの際新しいパスワードの設定が強制されます。たとえば、パスワード作成時の最終変更日の値が 9201 の最大値が 30

である場合、9231 (9201+30) 日目が経過した後は、ログイン時に新しいパスワードの設定が強制されます (詳細は、229ページの「パスワードの有効期間の設定」を参照)。

この値としては以下の 3 とおりが考えられます。

- 「0」は「次回のログイン時にパスワードの変更を強制し、その後は有効期間を設定する機能を停止する」ということを意味します。
- 「0 より大きい値」は、この数字がパスワードの有効期間 (単位：日) になります。
- 「-1」は、有効期間を設定する機能を停止するということを意味します。つまり **passwd -x -1** *username* というコマンドを実行すると、以前に行われたパスワードの有効期間の設定が解除されることになります。このフィールドを空白にしておくと、-1 として扱われます。

- 「n4 警告」は、「パスワードが間もなく無効になる」という警告メッセージが表示されるタイミングを示します。パスワードが作成されてからの日数で指定します。たとえば、最終変更日が 9201、最大値が 30、警告が 5 である場合、9226 (9201+30-5) 日が経過した後は、ログインの度に「パスワードが間もなく無効になる」という警告メッセージが表示されます (詳細は、231ページの「警告期間の設定」を参照)。

  この値としては以下の 2 とおりが考えられます。

  - 「0」は、警告メッセージが表示される期間がないことを示します。
  - 「0 より大きい値」は、この数字が、パスワードが作成されてから警告メッセージが表示されるまでの期間 (日数) になります。

- 「n5 間隔」は、ログインとログインの間隔 (日数) の最大値です。ここで指定された日数を超える期間ログインを行わないと、ログインができなくなります。たとえば、間隔に 6 を指定した場合、6 日間ログインを行わないと 7 日目にはログインができなくなります (詳細は、235ページの「ログインの間隔の最大値の指定」を参照)。

  この値としては以下の 2 とおりが考えられます。

  - 「-1」は、デフォルト設定です。機能を停止します。この場合、ログインとログインとの間隔が何日開いてもログイン特権が失われることはありません。
  - 「0 より大きい値」は、この数字が、ログインとログインの間隔 (日数) の最大値になります。

- 「n6 期限」は、パスワードの有効期限を示します。1970 年 1 月 1 日からの日数で表されます。この日を過ぎると、ログインができなくなります。たとえば、期限が 9739 (1995 年 9 月 1 日) に設定されている場合、1995 年 9 月 2 日 (GMT) になるとログインができなくなります。このときログインを試行すると「Login incorrect」というメッセージが表示されます (詳細は、233ページの「パスワード使用権の有効期限」を参照)。

  この値としては以下の 2 とおりが考えられます。

  - 「-1」は、機能を停止します。パスワードが有効期限を過ぎている場合でも、-1 を設定すると再び使えるようになります。有効期限を設定したくないときは、-1 を指定するようにしてください。
  - 「0 より大きい値」は、この数字が有効期限の日付になります (1970 年 1 月 1 日から数えたもの)。現在以前の日付を設定すると、パスワードはすぐに無効になります。

- 「n7 未使用」は、このフィールドは現在使用されていません。値を設定しても無視されます。
- *Login* はユーザのログイン ID

注意 - nistbladm を使用して passwd テーブルのシャドウ列を設定する場合、すべてのフィールドに適切な値を指定する必要があります。空白のまま残したり、0 を入力したりしても「変更なし」という意味にはなりません。

ユーザー amy が、パスワードの最終変更日が 1995 年 5 月 1 日 (1970 年 1 月 1 日から数えて 9246 日目)、パスワード作成後の変更禁止期間が 7 日、パスワードの有効期間が 30 日、「パスワードが間もなく無効になる」という警告が表示されるのがパスワード作成から 26 日目以降、ログインとログインの間隔の最大値が 15 日、アカウントの有効期限が (1970 年 1 月 1 日から) 9255 日目という設定にするには、以下のように入力します。

```
master# nistbladm -m shadow=9246:7:30:5:15:9255:0 [name=amy], passwd.ord_dir
```

nistbladm と日数

パスワードの使用期間に関するパラメータは、日数で表されるものがほとんどです。日数を指定する際には以下の規則を守る必要があります。

- 1970 年 1 月 1 日を 0 とします。つまり 1970 年 1 月 2 日は 1 になります。

- NIS+ では、日付の計算、カウントに GMT (Greenwich Mean Time) が使用されます。つまり日付が変更されるのは、GMT の午前 0 時です。
- 日数の指定には整数を使用します。分数は使用できません。
- パスワードのロックなどの動作は、指定された日付に行われます。たとえば、パスワード使用権の有効期限を1995 年 1 月 2 日 (1970 年 1 月 1 日から 9125日目) に設定した場合、この日は「ユーザーがパスワードを使用できる最後の日」となります。パスワードが使用できなくなるのは、次の日からです。

最終変更日、期限のどちらのフィールドも、入力するのは 1970 年 1 月 1 日から数えた日数です。この日数と実際の日付との関係を以下の表に示します。

表 11–2 1970 年 1 月 1 日からの日数

日付	日数
1970 年 1 月 1 日	0
1970 年 1 月 2 日	1
1971 年 1 月 2 日	365
1997 年 1 月 1 日	9863

関連コマンド

`passwd` および `nistbladm` コマンドと類似の機能を持つコマンドは他にもあります。それぞれについて表 11–3 にまとめてあります。

表 11–3 関連コマンド

コマンド	説明
`yppasswd`	現在は `passwd` コマンドにリンク。`yppasswd` を起動すると `passwd` コマンドが起動される
`nispasswd`	現在は `passwd` コマンドにリンク。`nispasswd` を起動すると `passwd` コマンドが起動される
`niscat`	テーブルの内容を表示するのに使用

表 11–3　関連コマンド　続く

パスワード情報の表示

ドメイン中のユーザーのパスワード情報を表示するには、passwd コマンドを使用します。情報は、全ユーザーについて同時に表示することも、ユーザーごとに表示することもできます。

自分のパスワード情報

```
passwd -s
```

ドメインの全ユーザーの情報

```
passwd -s -a
```

他のユーザーの情報

```
passwd -s username
```

表示できるのは、エントリおよび列のうち読み取り権を持っているものだけです。エントリの表示形式は以下のとおりです。

- 使用期間に関する情報が省略されたもの：*username status*
- 使用期間に関する情報が付加されたもの：
  username status mm/dd/yy min max warn expire inactive

表 11–4　NIS+ パスワード情報表示の形式

フィールド	説明	参照箇所
username	ユーザーのログイン名	
status	パスワードの状態。PS (アカウントにパスワードが存在する)、LK (パスワードがロックされている)、NP (アカウントにパスワードが存在しない) の 3 種類がある	227ページの「パスワードのロック」参照
mm/dd/yy	パスワードが最後に変更された日付 (GMT で表す)	
min	パスワード作成後の変更禁止期間 (日数)	230ページの「パスワードの変更禁止期間の設定」参照

表 11–4 NIS+ パスワード情報表示の形式 続く

フィールド	説明	参照箇所
max	パスワードの有効期間 (日数)	229ページの「パスワードの有効期間の設定」参照
warn	「パスワードが間もなく無効になる」という警告が表示されるまでの日数	231ページの「警告期間の設定」参照
expire	パスワードの有効期限 (日付)	233ページの「パスワード使用権の有効期限」参照
inactive	ログインとログインの間隔の最大値 (日数)。ログインとログインの間隔がここで指定した日数を超えると、ログインができなくなる	235ページの「ログインの間隔の最大値の指定」参照

別のドメインの passwd テーブルのエントリを表示するには、–D オプションを使用します。

ドメイン中の全ユーザー

```
passwd -s -a -D domainname
```

個々のユーザー

```
passwd -s -D domainname username
```

パスワードの変更

新しいパスワードを作成するときは、212ページの「パスワードの必要条件」に示された必要条件を満たすようにします。

自分のパスワードの変更

自分のパスワードを変更するには、以下のように入力します。

```
station1% passwd
```

「古いパスワードの入力」、「新しいパスワードの入力」、「新しいパスワードの再入力 (確認のため)」という順にプロンプトが表示されるので、それに従って作業します。

他人のパスワードの変更

他人のパスワードを変更するには、以下のように入力します。

同じドメイン内の他のユーザー

```
passwd username
```

他のドメインのユーザー

```
passwd -D domainname username
```

NIS+ 環境 (217ページの「passwd コマンドと NIS+ 環境」参照) で passwd コマンドを使用して他人のパスワードを変更する場合、passwd テーブルにおける該当ユーザーエントリへの変更権が必要になります。つまり、該当する passwd テーブルに対して変更権を持つグループのメンバーになる必要があります。このとき、該当ユーザーの古いパスワードや自分のパスワードを入力する必要はありません。確認のため新しいパスワードの入力を求めるプロンプトが 2 回表示されます。一致しない場合は、さらに 2 回入力する必要があります。

root のパスワードの変更

passwd コマンドで root のパスワードを変更した場合は、その直後に chkey –p を実行する必要があります。chkey –p を実行しないと、root で正しくログインできなくなります。

root のパスワードの変更手順は以下のとおりです。

1. **root** でログインします。
2. passwd コマンドで **root** のパスワードを変更します。nispasswd は使用しないでください。
3. chkey –p を実行します。
   必ず –p オプションを使用します。

パスワードのロック

NIS+ 環境 (217ページの「passwd コマンドと NIS+ 環境」を参照) で passwd コマンドを使用してパスワードのロックができるのは、該当ユーザーの passwd テーブル中のエントリに対する変更権を持っている管理者 (グループのメンバー) です。パスワードがロックされると、そのアカウントは使用できなくなります。また、パスワードがロックされているユーザー名を使ってログインをしようとすると、「Login incorrect」というメッセージが表示されます。

該当ユーザーがすでにログインしている場合、パスワードをロックすることによる影響は現れないという点に注意してください。ただ、パスワードの入力が必要になる login、rlogin、ftp、telnet などの機能は使用できなくなります。

すでにログインしているユーザーのパスワードをロックしても、そのユーザーが passwd コマンドでパスワードを変更するとロックは解除されます。

以下のようなことが有効です。

- 休暇などで不在になっているユーザーのパスワードを一時的にロックする。他の人に不正に使用されることを防止できる
- セキュリティ上の問題が発生している可能性があれば、すぐに一部のユーザーのパスワードをロックする
- 解雇になったユーザーのパスワードをすぐにロックする。ユーザーのアカウントを削除するより速く容易に行える上、そのアカウントに格納されているデータをそのまま容易に保管できる
- UNIX プロセスにパスワードを設定している場合には、この種のパスワードもロックできる。パスワードがロックされてもプロセスの実行はできるが、プロセスを使用してログインすることは (たとえパスワードを知っていても) できなくなる。多くの場合、プロセスは NIS+ 主体の形では設定されない。しかしプロセスのパスワード情報は、/etc ディレクトリのファイルに保存される。このとき /etc に格納されたパスワードをロックするにはファイルモードで passwd コマンドを実行する必要がある。

パスワードのロックは以下のように行います。

```
passwd -l username
```

パスワードロックの解除

パスワードのロックは、パスワードを変更すれば解除されます。この場合、「変更」とは必ずしもパスワードを新しいものに変えるという意味ではなく、パスワー

ド変更の手順を踏むということです (元とまったく同じパスワードに「変更する」ということもある)。

たとえば、jody というユーザーのパスワードのロックを解除するには、以下のように入力します。

```
station1% passwd jody
```

パスワードの使用期間に関する設定

設定により、パスワードの定期的な変更をユーザーに強制できます。

たとえば、以下のような設定が可能です。

- 次回のログインの際に強制的にパスワードを変更させる (詳細は、229ページの「パスワードの強制的な変更」を参照)
- パスワードの有効期間 (日数) を設定する (詳細は、229ページの「パスワードの有効期間の設定」を参照)
- パスワード作成後の変更禁止期間を設定する (詳細は、230ページの「パスワードの変更禁止期間の設定」を参照)
- パスワードの有効期間が終わる前に「パスワードが間もなく無効になる」という警告メッセージが (ログイン時に) 表示されるタイミングを設定する (詳細は、231ページの「警告期間の設定」を参照)
- ログインとログインの間隔の最大値 (日数) を指定する。指定された日数を超えてログインが行われないと、パスワードがロックされる (詳細は、235ページの「ログインの間隔の最大値の指定」を参照)
- パスワードの有効期限 (日付) を設定する。この日を過ぎると、該当ユーザはシステムにログインできなくなる (詳細は、233ページの「パスワード使用権の有効期限」を参照)

上記の警告期間や有効期間などに達したのがログインの行われた後であった場合、影響は現れません。そのまま普通に動作し続けます。

影響が現れるのは、次回のログイン時か、ログインを必要とする機能 (以下に例を示す) を使用した時です。

- login
- rlogin
- telnet
- ftp

パスワードの使用期間に関する設定は、ユーザーごとに行います。パスワードの使用期間に関する必要条件はユーザーによって違っている可能性があります。ユーザー一般にデフォルト設定を適用することもできます。詳細は、237ページの「/etc/defaults/passwd ファイル」を参照してください。

パスワードの強制的な変更

次回ログイン時のパスワード変更をユーザーに強制するには、以下の 2 種類の方法があります。

使用期間に関する設定を引き続き有効にする場合

```
passwd -f username
```

使用期間に関する設定を解除する場合

```
passwd -x 0 username
```

パスワードの有効期間の設定

パスワードの有効期間を設定するには、passwd コマンドの引数 *max* を使用します。有効期間は日数で指定します。有効期間が終了した後は、新しいパスワードをユーザーが、作成しなければなりません。有効期間終了後に同じパスワードでログインしようとしても、
「Your password has been expired for too long」というメッセージが表示され、新しいパスワードを作成しない限りログインを完了できません。

引数 *max* は以下の形式で使用します。

```
passwd -x max username
```

引数の意味はそれぞれ以下のとおりです。

- *username* は、ユーザーのログイン ID
- *max* は、以下のうちいずれかになります。
  - 「0 より大きい値」は、パスワードの有効期間の日数

- 「0」は、次回ログイン時のパスワード変更を強制する。使用期間に関する設定は解除される
- 「-1」は、パスワードの使用期間に関する設定を解除します。つまり **passwd -x -1** *username* と入力すると、使用期間に関して該当ユーザーに行われていた設定はすべて解除されます。

たとえば、schweik というユーザーに 45 日ごとのパスワード変更を強制するには、以下のように入力します。

```
station1% passwd -x 45 schweik
```

パスワードの変更禁止期間の設定

パスワード作成後の変更禁止期間を設定するには、引数 *min* を使用します。変更禁止期間が終了しないうちにパスワードを変更しようとすると、「Sorry less than N days since the last change」というメッセージが表示されます。

引数 *min* は以下の形式で使用します。

```
passwd -x max  -n min  username
```

引数の意味はそれぞれ以下のとおりです。

- *username* は、ユーザーのログイン ID
- *max* は、パスワードの有効期間 (前のセクションを参照)
- *min* は、パスワード作成後の変更禁止期間

たとえば、ユーザー eponine のパスワードの、有効期間を 45 日間、変更禁止期間を 7 日間に設定する場合は、以下のように入力します。

```
station1% passwd -x 45 -n 7 eponine
```

min 引数には以下の規則があります。

- *min* 引数は必ずしも使用しなくて良い (つまり、パスワード変更禁止期間は必ずしも指定しなくて良い)
- *min* 引数は、必ず –max 引数と組み合わせて使用する。つまり、変更禁止期間を設定するには、有効期間も設定する必要がある
- *min* に *max* より大きい値を設定すると (例： passwd -x 7 -n 8)、ユーザーはパスワードを変更できなくなる。変更しようとすると、

「You may not change this password」というメッセージが表示される。*min* に *max* より大きな値を設定すると、次の 2 つの効果が得られる

- ユーザーによるパスワードの変更を禁止する。つまり、管理特権を有する一部の人物以外は、パスワードを変更できなくなる。たとえば、複数のユーザーに共通のグループパスワードを割り当て、そのパスワードに対して *min* 値より大きな *max* 値を設定すると、グループ内のユーザーは誰一人としてパスワードを変更できなくなる
- パスワードは *max* 値で設定される期限内のみ有効となる。しかも、*min* 値が *max* 値より大きいため、ユーザーはパスワードを変更できない。つまり、*max* 値の示す有効期限が満了する前にパスワードの期限延長を図る手だては、ユーザーには一切与えられない。実際、有効期限が満了した後は、管理者の介在なしにユーザーはログインできなくなる

警告期間の設定

パスワードの有効期限以前の一定期間、ログイン時に「Your password will expire in N days」(*N* は日数) というメッセージが表示されるようにするには、引数 *warn* を使用します。

たとえば、パスワードの有効期限が 30 日間 (引数 –max で指定する) で、warn が 7 に設定されている場合、パスワード作成後 24 日目のログイン時に「Your password will expire in 7 days」という警告メッセージが表示されます。また翌日 (パスワード作成後 25 日目) には、警告メッセージは「Your password will expire in 6 days」となります。

警告メッセージは、電子メールで送信されるものでも、ユーザーのコンソールウィンドウに表示されるものでもないという点に注意してください。表示されるのは、ユーザーのログイン時のみです。つまり指定の期間ユーザーがログインしなければ警告メッセージは表示されません。

warn の値は、*max* の値との「関連によって機能する」ものであるという点に注意してください。つまり、「*max* によって設定される有効期限から逆算されるもの」ということです。*warn* の値が 14 であれば、「Your password will expire in N days」というメッセージの表示が開始されるのは、有効期限の 2 週間前です。

warn の値は *max* の値との関連によって機能するため、*max* の指定がない場合は意味がなくなります。

引数 *warn* は以下の形式で使用します。

```
passwd -x max  -w   warn  username
```

引数の意味はそれぞれ以下のとおりです。

- *username* は、ユーザーのログイン ID
- *max* は、パスワードの有効期限 (日数) (229ページの「パスワードの有効期間の設定」を参照)
- *warn* は、「有効期限の何日前から警告メッセージの表示を開始するか」を指定する

たとえば、nilovna というユーザーの、パスワードの有効期間を 45 日間とし、警告メッセージの表示を有効期限の 5 日前から開始する場合、以下のように入力します。

```
station1% passwd -x 45 -w 5 nilovna
```

warn 引数には以下の規則があります。

- *warn* 引数は必ずしも使用しなくて良い。*warn* の値を設定しなければ、警告メッセージは表示されない
- *warn* 引数は、必ず *max* 引数と組み合わせて使用する。つまり、警告メッセージの表示期間を指定するには、有効期間の指定が必要になる

注 - Solstice AdminSuite を使用して、*warm* 値をユーザーのパスワード用に設定することもできます。

パスワードの使用期間に関する設定の解除

ユーザーのパスワードの有効期間を解除するには、以下の 2 つの方法があります。

有効期間を取り消す。パスワードはそのまま使用できる

```
passwd -x -1 username
```

次回ログイン時にパスワードを強制的に変更させた後、有効期間を取り消す

```
passwd -x 0 username
```

上記の方法では、*max* の値を 0 か -1 に設定する (詳細は、229ページの「パスワードの有効期間の設定」を参照)

たとえば、ユーザー mendez のパスワードを次回ログイン時に強制的に変更させ、その後パスワードの有効期間を解除するには、以下のように入力します。

```
station% passwd -x 0 mendez
```

注 - Solstice AdminSuite を使用して、このパラメータをユーザーのパスワード用に設定できます。

この値を設定するには、nistbladm コマンドを使用できます。たとえば、ユーザー otsu のパスワードの有効期間を取り消し、変更の必要がないようにするには、以下のように入力します。

```
station1% nistbladm -m 'shadow=0:0:-1:0:0:0:0' [name=otsu],passwd.org_dir
```

nistbladm コマンドの使用方法の詳細は、219ページの「nistbladm コマンド」を参照してください。

パスワード使用権の有効期限

引数 *expire* には「ユーザーのパスワード使用権がいつ無効になるか」を日付で指定します。該当ユーザーのログインをこの日以降禁止し、システムから締め出します。

たとえば、ユーザー pete の有効期限を 1997 年 12 月 31 日に設定すると、どんなパスワードを使用しても 1998 年 1 月 1 日以降 pete という ID ではログインができなくなります。ログインをしようとしても、「Login incorrect」というメッセージが表示されます。

「有効期間」と「有効期限」の違い

パスワード使用権の有効期限は、パスワードの有効期間とは異なっています。

- 「パスワードの有効期間」

  厳密に区別をする必要のない場合には、有効期間が終了した後も変更されないパスワードのことを「有効期限の過ぎたパスワード」と呼ぶことがあります。しかしこのパスワードは、もう一度だけログインに使用できます。ただしその際に変更を強制されます。

- 「パスワード使用権の有効期限」

  パスワード使用権が「有効期限」を過ぎたユーザーは、どんなパスワードを使用してもログインできません。ネットワークへのログインアクセス権が無効になったということです。

有効期限の設定

パスワード使用権が無効になっても、その影響が現れるのは、該当ユーザーがログインをしようとするときだけです。該当ユーザーがすでにログインしていた場合には、パスワード使用権が無効になったことによる影響は現れません。ただし、`rlogin`、`telnet` などログインを必要とする機能は使用できなくなり、いったんログアウトするとログインできなくなります。以上のことから、パスワード使用権に有効期限を設ける場合、毎日のワークセッション終了時には必ずログアウトするようユーザー全員に指示してください。

注 - Solstice AdminSuite ツールを使用して有効期限の設定ができるときは、`nistbladm` を使用しないでください。Solstice AdminSuite ツールの方が使いやすく、設定を誤る可能性が低くなります。

`nistbladm` コマンドでパスワード使用権の有効期限を指定するには、以下のように入力します。

```
nistbladm -m 'shadow=n:n:n:n:n:n6:n' [name=login],passwd.org_dir
```

引数の意味はそれぞれ以下のようになります。

- *login* は、ユーザーのログイン ID です。
- *n* は、シャドウ列の (n6 を除くフィールドの) 値です。
- *n6* は、ユーザーのパスワード使用権の有効期限 (日付) を設定します。「1970 年 1 月 1 日から数えて何日目か」で表します (表 11–2 を参照)。このフィールドに指定する値には、以下の 2 種類があります。

  - 「-1」は、有効期限の設定を解除します。ユーザーのパスワードがすでに有効期限を過ぎていても、-1 を設定することによって再び使用できるようになります。有効期限を設定したくないときは、初めから -1 を指定しておきます。
  - 「0 より大きい値」は、有効期限の日付 (「1970 年 1 月 1 日から数えて何日目か」) です。現在以前の日付を指定すると、ユーザーのパスワードはすぐに無効になります。

たとえば、ユーザー pete の有効期限を 1995 年 12 月 31 日に設定する場合は、以下のように入力します。

```
station1% nistbladm -m 'shadow   =n:n:n:n:n:9493:n'  [name=pete],passwd.org_dir
```

注意 - 空白のフィールドや、値の正しくないフィールドがあると機能しません。

パスワード使用権の有効期限の解除

有効期限を過ぎたパスワード使用権を再び使用できる状態にするには、nistbladm コマンドを使用して該当フィールド (n6) に -1 を指定します。例を以下に示します。

```
station1% nistbladm -m 'shadow=   n:n:n:n:n:-1:n'   [name=huck],passwd.org_dir
```

また、nistbladm で「n6」フィールドを将来の日付に設定し直すという方法も使用できます。

ログインの間隔の最大値の指定

引数 *inactive* には、「ログインからログインまでの間隔を最大何日あけることができるか」を指定します。この引数に指定された日数を超えてログインが行われなかった場合、該当ユーザーはマシンにログインできなくなります。ログインしようとすると、「Login incorrect」というメッセージが表示されます。

この設定はネットワーク全体ではなく、個々のマシンに対して行われます。したがって、NIS+ 環境においてログイン間隔の最大値を設定するには、該当ユーザーのエントリをホームドメインの passwd テーブルに指定します。この設定はネットワーク上のすべてのマシンで該当ユーザーに対して適用されます。しかし、最後にログインをした日付はマシンごとに異なるため、個別に (各マシンの /var/adm/utmp ファイルに) 記録されます。

たとえば、ユーザー sam の最大ログイン間隔を 10 日間に設定した場合を考えてみましょう。sam は 1 月 1 日にマシン A とマシン B にログインをした後、ログアウトしました。続いて 3 日後の 1 月 4 日、sam はマシン B にログインした後ログアウトしました。さらに 9 日後の 1 月13日になると、sam はマシン B にはログインできますがマシン A にはログインできません。マシン B においては最終ログインから 9 日間しか経過していませんが、マシン A においては 13 日間が経過しているからです。

最大ログイン間隔の設定は、まだ一度もログインをしていないマシンには適用されないという点に注意してください。一度もログインをしていないマシンには、引数 *inactive* の設定および他のマシンへのログイン実績がどのようになっていようとログインが可能です。

注意 - 「毎日の作業終了時には必ずログアウトをせよ」という指示がユーザーに対して行われていない場合、最大ログイン間隔の設定はしないでください。この設定はログインにだけ関係しており、他の形でシステムを使用する場合のことは考慮されていません。仮にユーザーが作業終了後もログインしたままの状態でシステムを放置しておくと、ログインが発生しないため指定のログイン間隔をすぐ超えてしまいます。指定のログイン間隔を超えた状態でリブートやログアウトをすると、再びログインできなくなります。

注 - Solstice AdminSuite ツールを使用して最大ログイン間隔の設定ができるときは、`nistbladm` を使用しないでください。Solstice AdminSuite ツールの方が使いやすく、設定を誤る可能性が低くなります。

最大ログイン間隔の設定を `nistbladm` コマンドで行う場合は、以下の形式を使用します。

```
nistbladm -m 'shadow= n:n:n:n:n5:n:n' [name=login], passwd.org_dir
```

引数の意味はそれぞれ以下のとおりです。

- *login* は、ユーザーのログイン ID です。
- *n* は、シャドウ列の (n5 以外の) フィールドの値です。
- *n5* は、最大ログイン間隔 (日数) です。指定される値には、以下の 2 種類があります。
  - 「-1」は、デフォルト設定です。最大ログイン間隔の設定を解除します。この設定の場合、ログインされない期間が何日続いてもパスワード使用権が失われることはありません。
  - 「0 より大きい値」は、最大ログイン間隔 (日数) を指定します。

たとえば、ユーザー `sam` の最大ログイン間隔を 7 日間に指定する場合は、以下のように入力します。

```
station1% nistbladm -m 'shadow=  n:n:n:n:n:7:n:n' [name=sam],passwd.org_dir
```

最大ログイン間隔の設定を解除する (ログインできなくなったユーザーを再びログインできるようにする) には、nistbladm で *inactive* を -1 に指定します。

パスワードの使用規則の設定 (およびそのデフォルト)

パスワードの使用規則に関する設定およびそのデフォルトについて説明します。

/etc/defaults/passwd ファイル

パスワード情報の獲得先が nsswitch.conf ファイルで files に指定されているすべてのユーザーのパスワードについて、4 つのデフォルト設定を行うためのファイルです。/etc/defaults/passwd ファイルで行われたデフォルト設定は、/etc ディレクトリのファイルからパスワード情報を獲得しているユーザーにだけ適用されます。パスワード情報の獲得先が NIS マップあるいは NIS+ テーブルであるようなユーザーには適用されません。NIS+ サーバー上の /etc/defaults/passwd ファイルは、パスワード情報をローカルファイルから獲得しているローカルユーザーにだけ影響を与えます。NIS+ 環境または、nsswitch.conf ファイルでパスワード情報の獲得先が nis、nisplus に設定されているユーザーには影響を与えません。

/etc/defaults/passwd ファイルで行われる「パスワードに関する 4 つのデフォルト設定」とは、具体的には以下のものを指します。

- 有効期間 (週単位)
- 変更禁止期間 (週単位)
- 「パスワードが間もなく無効になる」という警告メッセージの表示される期間 (週単位)
- 最低文字数

/etc/defaults/passwd ファイルのデフォルト設定には、以下の規則があります。

- パスワード情報をローカルの /etc ファイルから得ているユーザーの場合、passwd コマンド、Solstice AdminSuite、admintool で個々に行われた設定の方が、/etc/defaults/passwd のデフォルト設定よりも優先します。つまり /etc/defaults/passwd ファイルのデフォルト設定は、passwd テーブル中のエントリで (/etc/defaults/passwd の設定に対応した) 個別の設定が行われていないユーザーにだけ適用されるということです。
- パスワードの長さを除き、すべての設定は週単位で行われます (個々のパスワードの使用期間についての設定は、日単位で行われていた点に注意)

- MAXWEEKS、MINWEEKS、WARNWEEKS は、パスワードの最終変更日を起点として設定します (nistbladm などでは *warn* の起点を *max* としていた点に注意)。

/etc/defaults/passwd ファイルには、デフォルトでは以下のエントリがすでに含まれています。

```
MAXWEEKS=
MINWEEKS=
PASSLENGTH=
```

設定に必要な作業は、= の後に適切な数字を入力することだけです。= の後に数字が入っていないエントリは無効です。たとえば、MAXWEEKS に 4 を指定するには、以下のように入力します。

```
MAXWEEKS=4
MINWEEKS=
PASSLENGTH=
```

有効期限 (週単位)

ユーザーのパスワードの有効期間を週単位で指定するためのエントリです。以下に示すとおり、"=" の後に適切な数字を入力するだけで指定ができます。

```
MAXWEEKS=N
```

N は週の数です。たとえば、MAXWEEKS=9 というようにします。

変更禁止期間 (週単位)

パスワードの変更禁止期間を週単位で指定するためのエントリです。以下に示すとおり、"=" の後に適切な数字を入力するだけで指定ができます。

```
MINWEEKS=N
```

N は週の数です。たとえば、MINWEEKS=2 というようにします。

警告期間 (週単位)

注 - 同時に MAXWEEKS のデフォルトを設定していない限り、WARNWEEKS のデフォルトを設定しても問題はありません。

パスワードが無効になる前に、「パスワードが間もなく無効になる」という警告メッセージの表示が開始されるタイミングを指定するエントリです。たとえば、MAXWEEKS の値を 9 に指定していて、パスワードが無効になる 2 週間前に警告メッセージの表示を開始したいときは、WARNWEEKS を 7 に設定します。

WARNWEEKS は、MAXWEEKS に指定された有効期限ではなく、パスワードの最終変更日を起点と考えて設定することに注意してください。したがって WARNWEEKS に、MAXWEEKS 以上の値を設定することはできません。

注 - WARNWEEKS のデフォルトは、MAXWEEKS のデフォルトも一緒に設定されていない限り無効です。

WARNWEEKS は、以下に示すとおり "=" の後に適切な数字を入力するだけで指定ができます。

```
WARNWEEKS=N
```

N は週の数です。たとえば、WARNWEEKS=1 というようにします。

最低文字数

passwd コマンドには、デフォルトで「パスワードの長さは 6 文字以上にする」という規則があります。しかし、/etc/defaults/passwd ファイルの PASSLENGTH を使用することによってこの規則を変更することが可能です。

パスワードの最低文字数を 6 以外の値にするには、以下に示すとおり PASSLENGTH= の後に適切な値を入力します。

```
PASSLENGTH=N
```

N は文字数です。たとえば、PASSLENGTH=7 というようにします。

パスワード変更時の制限

パスワード変更の際の入力試行回数と所要時間には、制限を設定できます。設定は rpc.nispasswdd デーモン起動時の引数で行います。

入力試行回数を制限したり、タイムウィンドウを設定したりすることにより、「権利のない人が、パスワードを試行錯誤で知ることのできるものに変えてしまう」という危険をある程度 (完璧ではない) 回避できます。

試行回数の制限

パスワード変更時の誤入力試行回数を制限するには、rpc.nispasswdd に –a *number* という引数 (*number* は試行回数) を指定します。rpc.nispasswdd を起動するには、NIS+ マスターサーバー上にスーパーユーザー特権が必要です。

誤入力試行回数を 4 に制限する (デフォルトは 3) 場合、以下のように入力します。

```
station1# rpc.nispasswdd -a 4
```

この場合、4 回目のパスワード入力が失敗すると、
「Too many failures - try later」というメッセージが表示され、一定の時間が経過するまで該当ユーザーのパスワード変更はできなくなります。

所要時間の制限

パスワード変更時の所要時間 (ログインの所要時間) を制限するには、rpc.nispasswdd に –c *minutes* という引数 (*minutes* には時間を分単位で指定する) を指定します。rpc.nispasswdd を起動するには、NIS+ マスターサーバー上にスーパーユーザー特権が必要です。

たとえば、ユーザーが 2 分以内にログインしなければならないように設定するには、以下のように入力します。

```
station1# rpc.nispasswdd -c 2
```

この場合、2 分以内にパスワードの変更に成功しないとエラーメッセージが表示され、一定の時間が経過するまで該当ユーザーのパスワード変更はできなくなります。

第 12 章

NIS+ グループの管理

この章では、NIS+ グループとその管理方法について説明します。

注 - NIS+ セキュリティグループのタスクには、Solstice AdminSuite ツールを利用するともっと簡単に実行できるものもあります。

Solaris グループ

Solaris/NIS+ 環境には、UNIX グループ、ネットグループ、NIS+ グループの 3 種類のグループがあります。

- 「UNIX グループ」

  UNIX グループとは、特別な UNIX アクセス権を与えられたユーザーの集まりのことです。NIS+ 名前空間では、UNIX グループに関する情報は `org_dir` ディレ

クトリオブジェクト (group.org_dir) 内のグループテーブルに格納されます。UNIXグループメンバーの追加、修正、削除の方法については、第 14 章を、グループテーブルの詳細は、753ページの「group テーブル 」をそれぞれ参照してください。

- 「ネットグループ」

  ネットグループとは、ワークステーションとユーザー (他のワークステーション上でリモート操作を実行する権限を与えられている) の集まりのことです。NIS+ 名前空間では、ネットグループに関する情報はorg_dir ディレクトリオブジェクト (netgroup.org_dir) 内のグループテーブルに格納されます。ネットグループメンバーの追加、修正、削除の方法については第 14 章を、ネットグループテーブルの詳細は、755ページの「netgroup テーブル」をそれぞれ参照してください。

- 「NIS+ グループ」

  NIS+ グループとは、NIS+ オブジェクトに対する特別なアクセス権 (ネームスペースの管理を許されることが多い) を与えられている NIS+ ユーザーの集まりのことをいいます。NIS+ グループに関する情報は groups_dir ディレクトリオブジェクト内のテーブルに格納されます。

NIS+ グループ

NIS+ グループは、NIS+ オブジェクトに対するアクセス権を NIS+ の主体に割り当てるために考えられた概念です。(これらのアクセス権については、第 6 章を参照してください。) NIS+ グループに関する情報は、NIS+ groups_dir ディレクトリオブジェクト内のテーブルに格納されます。各グループは個別のテーブルを持ち、グループ名とテーブル名はそれぞれ対応します。たとえば、admin グループに関する情報は admin.groups_dir テーブルに格納されます。

NIS+ グループを作成したら、どれか 1 つには admin という名前を付けることをお勧めします。そして、この admin グループには、NIS+ アクセス権を持たせるユーザーを割り当ててください。必ずしも admin という名前にしなければならないわけでもないのですが、NIS+ マニュアルでは、NIS+ 管理権限を持たせるユーザーのグループを admin と呼んでいます。異なるユーザーと異なる権限を組み合わせて複数の NIS+ グループを作成することもできます。

注 - NIS+ グループのメンバー (ユーザー) の管理には、nisgrpadm コマンドを使います。グループテーブルの管理には、nisls コマンドと nischgrp コマンドを使います。グループテーブルに対して nistbladm コマンドを使うことはできないので注意してください。

NIS+ グループ関連のコマンドとその構文、オプションの詳細は、nis+(1) のマニュアルページを参照してください。

関連するコマンド

nisgrpadm コマンドを使ってほとんどのグループ管理作業を実行できますが、グループ管理に関連するコマンドには次のものがあります。

表 12–1 グループに関連するコマンド

コマンド	説明	参照する項目
nissetup	ドメインのグループが格納されるディレクトリである groups_dir を作成する	307ページの「nissetup コマンド」
nisls	groups_dir ディレクトリの内容、つまり、ドメイン内の全グループを表示する。各グループは groups_dir に個別のテーブルを持ち、グループ名とテーブル名はそれぞれ対応する	255ページの「nisls コマンドでディレクトリを表示する」
nischgrp	グループを任意の NIS+ オブジェクトに割り当てる	204ページの「オブジェクトまたはエントリグループの変更」
niscat	NIS+ グループのオブジェクト属性とメンバーを表示する	246ページの「NIS+ グループについて niscat を使用する」
nisdefaults	新しい NIS+ オブジェクトに割り当てられるグループを表示する	194ページの「NIS+ デフォルトの表示 - nisdefaults コマンド」

以上のコマンドの詳細 (構文、オプションなど) は、nis+(1) のマニュアルページを参照してください。

注 - NIS+ グループテーブルに対して nistbladm コマンドを使うことはできません。

NIS+ グループメンバーのタイプ

NIS+ グループには明示的 (explicit)、暗黙的 (implicit)、および再帰的 (recursive) という 3 タイプのメンバーがあります。またメンバー以外の主体にも、同様の 3 つのタイプがあります。メンバーのタイプは、メンバーを追加、削除する際に使用されます (247ページの「nisgrpadm コマンド」を参照)。

メンバーのタイプ

- 「明示的なもの」

  個々の NIS+ 主体です。これらは、すべてのグループ管理コマンドで、主体名によって識別されます。主体名は、デフォルトドメインから入力される場合、完全指定名を使用する必要はありません。

- 「暗黙的なもの」

  NIS+ ドメインに所属するすべての NIS+ 主体です。これらは、* 記号とドットで始まるドメイン名によって識別されます。選択した処理は、グループ内のすべてのメンバーに適用されます。

- 「再帰的なもの」

  他の NIS+ グループのメンバーの、すべての NIS+ 主体です。これらは、@ 記号で始まる NIS+ グループ名によって識別されます。選択した処理は、グループ内のすべてのメンバーに適用されます。

NIS+ グループは、明示的、暗黙的、および再帰的の 3 つすべてのカテゴリでのメンバー以外も受け付けます。メンバー以外の主体とは、「メンバーになることができるが、指定によってグループから排除されているもの」を指します。

メンバー以外の主体のタイプ

メンバー以外の主体はマイナス記号で始まり、メンバー主体と区別されます。

- 「明示的なもの」- 名前の先頭に「-」だけをつけます。
- 「暗黙的なもの」-「-」の後に「*」、「.」をつけます。
- 「再帰的なもの」-「-」の後に「@」をつけます。

グループの構文

同じ主体について「グループに含まれる」という指定と「グループに含まれない」という指定があった場合、指定の順序に関係なく「含まれない」という指定が優先されます。たとえば、同じ主体について「グループに含まれる暗黙的なドメインのメンバーである」という指定と「グループに含まれない再帰的なグループのメンバーである」という指定の両方がある場合、後者の方が優先されます。

`nisgrpadm` コマンドを使用する場合、主体のタイプは表 12–2 のように指定します。

表 12–2 主体のタイプの指定方法

主体のタイプ	構文
明示的なメンバー	*username.domain*
暗黙的なメンバー	**.domain*
再帰的なメンバー	*@groupname.domain*
メンバー以外 (明示的なもの)	*-username.domain*
メンバー以外 (暗黙的なもの)	*-*.domain*
メンバー以外 (再帰的なもの)	*@groupname.domain*

NIS+ グループについて niscat を使用する

niscat –o コマンドを使用して、NIS+ グループのオブジェクト属性を表示できます。

グループのオブジェクト属性を表示する方法

グループのオブジェクト属性を表示するには、そのグループが格納されている groups_dir ディレクトリへの読み取り権が必要です。ここでは、niscat –o とグループの完全指定名を使用します。この完全指定名には、次に示すようにその groups_dir サブディレクトリを含んでいなければなりません。

```
niscat -o group-name .groups_dir.domain-name
```

次に例を示します。

```
rootmaster# niscat -o sales.groups_dir.doc.com.
Object Name : sales
Owner : rootmaster.doc.com.
Group : sales.doc.com.
Domain : groups_dir.doc.com.
Access Rights : ----rmcdr---r---
Time to Live : 1:0:0
Object Type : GROUP
Group Flags :
Group Members : rootmaster.doc.com.
  topadmin.doc.com.
  @.admin.doc.com.
  *.sales.doc.com.
```

注 - nisgrpadm -1 コマンドを使うと、メンバーリストはさらに整理されて表示されます。

グループ属性のいくつかは、環境変数 NIS_DEFAULTS から継承されます。ただし、このグループの作成時に環境変数が無効になっている場合を除きます。Group Flags フィールドは、現在使用されていません。グループメンバーのリストでは、* 記号はメンバーのドメインを、@ 記号はメンバーのグループをそれぞれ示します。

nisgrpadm コマンド

nisgrpadm コマンドは、NIS+ グループを作成したり、削除したり、さまざまな管理作業を実行します。nisgrpadm を使用するには、処理に適したアクセス権が必要です。

表 12–3 nisgrpadm に必要なアクセス権

処理	必要なアクセス権	処理の対象
グループの作成	作成権	groups_dir ディレクトリ
グループの削除	削除権	groups_dir ディレクトリ
メンバーの表示	読み取り権	グループオブジェクト
メンバーの追加	変更権	グループオブジェクト
メンバーの削除	変更権	グループオブジェクト

nisgrpadm には、グループ作業用とグループメンバー作業用に 2 つの形式があります。

グループの作成、削除、あるいはメンバーの表示

```
nisgrpadm -c group-name.domain-name
nisgrpadm -d group-name
nisgrpadm -l group-name
```

メンバーの追加または削除、あるいはメンバーがグループに所属するかどうかの判定 (*member...* には、表 12–2 に示した、6 種類のメンバーのどのような組み合わせでも指定できます)。

```
nisgrpadm -a group-name  member...
nisgrpadm -r group-name  member...
nisgrpadm -t group-name  member...
```

作成 (–c) 以外のすべての処理には、部分指定名 *group-name* を使用できます。ただし、–c オプションの場合でも、nisgrpadm では *group-name* 引数に groups_dir を使用する必要はありません。実際これは受け付けられません。

NIS+ グループを作成する

NIS+ グループを作成するには、グループのドメインの groups_dir ディレクトリに対する作成権が必要です。–c オプションと完全指定グループ名を使用します。

```
nisgrpadm -c group-name.domainname
```

グループを作成すると、指定した名前の NIS+ グループテーブルが groups_dir に作成されます。nisls コマンドを使うと、対応するテーブルが groups_dir に作成されているかどうかを確認できます。また、niscat コマンドを使うと、そのテーブル中のグループメンバーを一覧表示できます。

新しく作成したグループにはメンバーがありません。どのメンバーをどのグループに追加するかについては、250ページの「NIS+ グループにメンバーを追加する」を参照してください。

次の例では、admin という名前の 3 つのグループを作成します。最初のグループは doc.com. ドメインに、次のグループは sales.doc.com. に、3 番目のグループは manf.doc.com. に作成されます。これらはすべて、それぞれのドメインのマスターサーバーから作成されます。

```
rootmaster# nisgrpadm -c admin.doc.com.
Group admin.doc.com. created.
salesmaster# nisgrpadm -c admin.sales.doc.com.
Group admin.sales.doc.com. created.
manfmaster# nisgrpadm -c admin.manf.doc.com.
Group admin.manf.doc.com. created.
```

作成されるグループは、変数 NIS_DEFAULTS で指定されたオブジェクト属性をすべて継承します。つまり、その所有者、所有グループ、アクセス権、生存期間、および検索パスです。これらのデフォルトを表示するには、nisdefaults コマンドを使用します (第 10 章を参照)。オプションなしで使用すると、次のように出力されます。

```
rootmaster# nisdefaults
Principal Name : rootmaster.doc.com.
Domain Name : doc.com.
Host Name : rootmaster.doc.com.
Group Name :
Access Rights : ----rmcdr---r---
Time to live : 12:0:0
Search Path : doc.com.
```

所有者は Principal Name: フィールドに表示されます。所有者グループは、環境変数 NIS_GROUP を設定した場合にだけ表示されます。たとえば、C シェルを想定して NIS_GROUP を fns_admins.doc.com に設定する場合は、次のように入力します。

```
rootmaster# setenv NIS_GROUP fns_admins.doc.com
```

もちろん、グループの作成時に –D オプションを使用することによって、これらのデフォルトはどれでも変更できます。

```
salesmaster# nisgrpadm -D group=special.sales.doc.com.-c admin.sales.doc.com.
Group admin.sales.doc.com. created.
```

NIS+ グループを削除する

NIS+ グループを削除するには、グループのドメイン内の groups_dir ディレクトリに対する削除権が必要です。–d オプションを使用します。

```
nisgrpadm -d group-name
```

デフォルトドメインが正しく設定されている場合、完全指定グループ名を使用する必要はありません。ただし、別のドメイン内のグループを誤って削除しないように、まず (nisdefaults を使用して) チェックしなければなりません。次の例では test.sales.doc.com. グループを削除します。

```
salesmaster% nisgrpadm -d test.sales.doc.com.
Group test.sales.doc.com. destroyed.
```

NIS+ グループにメンバーを追加する

NIS+ グループにメンバーを追加するには、グループオブジェクトに対する変更権が必要です。–a オプションを使用します。

```
nisgrpadm -a group-name members. . .
```

244ページの「NIS+ グループメンバーのタイプ」の記述どおり、主体 (明示的なメンバー)、ドメイン (暗黙的なメンバー)、およびグループ (再帰的なメンバー) を追加できます。デフォルトドメインに所属するメンバー名またはグループ名は、完全指定する必要がありません。次の例では、デフォルトドメイン sales.doc.com. からの NIS+ 主体 panza と valjean、および manf.doc.com. ドメインからの主体 makeba を、グループ Ateam.sales.doc.com. に追加します。

```
client% nisgrpadm -a Ateam panza valjean makeba.manf.doc.com.
Added panza.sales.doc.com to group Ateam.sales.doc.com
Added valjean.sales.doc.com to group Ateam.sales.doc.com
Added makeba.manf.doc.com to group Ateam.sales.doc.com
```

この動作を確認するには、nisgrpadm –l オプションを使用します。「明示的なメンバー」のカテゴリにあるメンバーを探してください。

次の例では、doc.com. ドメイン内のすべての NIS+ 主体を staff.doc.com. グループに追加します。これは doc.com. ドメイン内のクライアントから入力します。ドメイン名の前にある * 記号とドットに注意してください。

```
client% nisgrpadm -a Staff *.doc.com.
Added *.doc.com. to group Staff.manf.doc.com.
```

次の例では、NIS+ グループ admin.doc.com. を admin.manf.doc.com. グループに追加します。これは manf.doc.com. ドメインのクライアントから入力します。グループ名の前にある @ 記号に注意してください。

```
client% nisgrpadm -a admin @admin.doc.com.
Added @admin.doc.com. to group admin.manf.doc.com.
```

NIS+ グループのメンバーを表示する

NIS+ グループのメンバーを表示するには、グループオブジェクトに対する読み取り権が必要です。–l オプションを使用します。

```
nisgrpadm -l group-name
```

次の例では、admin.manf.doc.com. グループのメンバーを表示します。これは manf.doc.com. グループ内のクライアントから入力します。

```
client% nisgrpadm -l admin
Group entry for admin.manf.doc.com. group:
  No explicit members
  No implicit members:
  Recursive members:
  @admin.doc.com.
  No explicit nonmembers
  No implicit nonmembers
  No recursive nonmembers
```

NIS+ グループからメンバーを削除する

NIS+ グループからメンバーを削除するには、グループオブジェクトに対する変更権が必要です。–r オプションを使用します。

```
nisgrpadm -r group-name members. . .
```

次の例では、Ateam.sales.doc.com グループから NIS+ 主体 allende と hugo.manf.doc.com. を削除します。これは sales.doc.com.domain ドメイン内のクライアントから入力します。

```
client% nisgrpadm -r Ateam allende hugo.manf.doc.com.
Removed allende.sales.doc.com. from group Ateam.sales.doc.com.
Removed hugo.manf.doc.com. from group Ateam.sales.doc.com.
```

次の例では、admin.manf.doc.com. グループから admin.doc.com. グループを削除します。これは manf.doc.com. ドメイン内のクライアントから入力します。

```
client% nisgrpadm -r admin @admin.doc.com.
Removed @admin.doc.com. from group admin.manf.doc.com.
```

NIS+ グループのメンバーかどうかを調べる

ある NIS+ 主体が特定の NIS+ グループのメンバーであるどうかを調べるには、そのグループオブジェクトに対する読み取り権が必要です。–t オプションを使用します。

```
nisgrpadm -t group-name members. . .
```

次の例では、NIS+ 主体 topadmin が admin.doc.com. グループに所属するかどうかを調べます。これは doc.com. ドメイン内のクライアントから入力します。

```
client% nisgrpadm -t admin topadmin
topadmin.doc.com. is a member of group admin.doc.com.
```

次の例では、sales.doc.com. ドメインの NIS+ 主体 jo が admin.sales.doc.com. グループに所属するかどうかを調べます。これは doc.com. ドメイン内のクライアントから入力します。

```
client% nisgrpadm -t admin.sales.doc.com. jo.sales.doc.com.
jo.sales.doc.com. is a member of group admin.sales.doc.com.
```

第 13 章

NIS+ ディレクトリの管理

この章では、NIS+ ディレクトリのオブジェクトとその管理方法を説明します。

NIS+ ディレクトリ

NIS+ ディレクトリオブジェクトは、NIS+ ドメインに関する情報を格納するために使われます。NIS+ ドメインごとに個別の NIS+ ディレクトリ構造があります。NIS+ ディレクトリの詳細は、第 4 章を参照してください。

NIS+ ディレクトリ関連のコマンドとその構文、オプションについては、`nis+(1)` のマニュアルページを参照してください。

ディレクトリについて niscat コマンドを使用する

`niscat –o` コマンドを使えば、NIS+ ディレクトリのオブジェクト属性を表示できます。このコマンドを使うには、ディレクトリオブジェクトの読み取り権が必要です。

ディレクトリのオブジェクト属性を表示する

ディレクトリのオブジェクト属性を表示するには、`niscat –o` とディレクトリ名を使います。

```
niscat -o directory-name
```

次に例を示します。

```
rootmaster# niscat -o doc.com.
Object Name : doc
Owner : rootmaster.doc.com.
Group :
Domain : Com.
Access Rights : r---rmcdr---r---
Time to Live : 24:0:0
Object Type : DIRECTORY
.
```

(続く)

.

nisls コマンドでディレクトリを表示する

nisls コマンドは、NIS+ ディレクトリの内容を表示します。このコマンドを使うには、ディレクトリオブジェクトに対する読み取り権が必要です。

簡潔形式での表示

```
nisls [-dgLmMR] directory-name
```

詳細形式での表示

```
nisls -l [-gm] [-dLMR] directory-name
```

表 13–1 nisls コマンドのオプション

オプション	目的
–d	ディレクトリオブジェクト。ディレクトリの内容を表示するのではなく、別のオブジェクトのように扱う
–L	リンク。ディレクトリ名が実際にはリンクである場合、コマンドはリンクをたどり、リンクされたディレクトリの情報を表示する
–M	マスター。マスターサーバーだけから情報を取得する。これによって最新の情報が得られるが、マスターサーバーが使用中の場合は長時間を要することがある
–R	再帰的 (recursive)。ディレクトリを再帰的に表示する。つまり、ディレクトリ中に他のディレクトリがある場合、それらの内容も表示する
–l	長形式。情報を長形式で表示する。長形式では、オブジェクトのタイプ、作成時間、所有者、およびアクセス権を表示する

表 13–1　nisls コマンドのオプション　続く

オプション	目的
–g	グループ。情報を長形式で表示するとき、ディレクトリ所有者ではなく、そのグループ所有者を表示する
–m	変更時間。情報を長形式で表示するとき、ディレクトリ作成時間ではなく、その変更時間を表示する

ディレクトリの内容を表示する (簡潔形式)

ディレクトリ内容をデフォルトの短形式で表示するには、次に示すオプションの内の 1 つまたは複数と、ディレクトリ名を使います。ディレクトリ名を指定しない場合、NIS+ はデフォルトのディレクトリを使います。

```
nisls [-dLMR] directory-name
```

または

```
nisls [-dLMR]
```

たとえば、次の nisls の場合は、ルートドメイン doc.com. のルートマスターサーバーから入力します。

```
rootmaster% nisls doc.com.:
org_dir
groups_dir
```

次に、ルートマスターサーバーから入力した例をもう 1 つ示します。

```
rootmaster% nisls -R sales.doc.com.
sales.doc.com.:
org_dir
groups_dir
groups_dir.sales.doc.com.:
admin
org_dir.sales.doc.com.:
auto_master
auto_home
bootparams
cred
```

(続く)

.

ディレクトリの内容を表示する (詳細形式)

ディレクトリの内容を詳細形式で表示するには –l オプション、および次に示すオプションのうちの 1 つまたは複数を使います –g と –m のオプションは、表示される属性を変更します。ディレクトリ名を指定しない場合、NIS+ はデフォルトのディレクトリを使います。

```
nisls -l [-gm] [-dLMR] directory-name
```

または

```
nisls -l [-gm] [-dLMR]
```

次に示すのは、ルートドメイン doc.com. のマスターサーバーから入力した例です。

```
rootmaster% nisls -l
doc.com.
D r---rmcdr---r--- rootmaster.doc.com. date org_dir
D r---rmcdr---r--- rootmaster.doc.com. date groups_dir
```

nismkdir コマンド

注 - この節では、nismkdir コマンドを使用して既存のドメインに非ルートサーバーを追加する方法を説明します。nisserver スクリプトを使うともっと簡単に非ルートサーバーを追加できますが、その方法については『*Solaris* ネーミングの設定と構成』で説明します。

nismkdir コマンドは、ルート以外の NIS+ ディレクトリを作成し、これをマスターサーバーに関連付けます。ルートディレクトリを作成するには、266ページの「nisinit コマンド」で説明する nisinit –r コマンドを使います。nismkdir コマンドを使って、複製サーバーを既存のディレクトリに追加できます。

NIS+ ディレクトリを作成するには、いくつかの関連作業のほかに、いくつかの前提条件があります。詳細は、『*Solaris* ネーミングの設定と構成*S*』を参照してください。

ディレクトリを作成するには、以下のように入力します。

```
nismkdir [-m master-server]  directory-name
```

複製を既存のディレクトリに追加するには、以下のように入力します。

```
nismkdir -s replica-server  directory-name
nismkdir -s replica-server  org_dir.directory-name
nismkdir -s replica-server  groups_dir.directory-name
```

ディレクトリを作成する

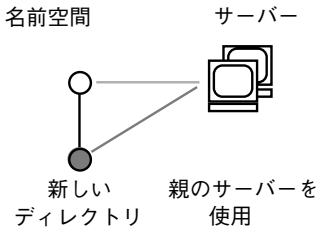
ディレクトリを作成するには、(ドメインのマスターサーバー上の) 親ディレクトリに対する作成権が必要です。まず –m オプションを使ってマスターサーバーを定義し、次に –s オプションを使って複製を定義します。

```
nismkdir -m master  directory
nismkdir -s replica  directory
```

注意 - nismkdir は必ず (複製サーバーではなく) マスターサーバー上で実行してください。複製サーバーで実行すると、マスターサーバーと複製サーバーの間で通信上の問題が発生します。

次の例では、sales.doc.com. ディレクトリを作成し、そのマスターサーバーである smaster.doc.com. とその複製サーバーである rep1.doc.com を指定します。これはルートマスターサーバーから入力します。

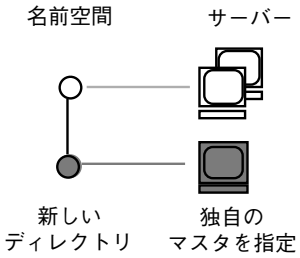
```
rootmaster% nismkdir -m smaster.doc.com. sales.doc.com.
rootmaster% nismkdir -m smaster.doc.com. org_dir.sales.doc.com.
rootmaster% nismkdir -m smaster.doc.com. groups_dir.sales.doc.com.
rootmaster% nismkdir -s rep1.doc.com. sales.doc.com.
rootmaster% nismkdir -s rep1.doc.com. org_dir.sales.doc.com.
rootmaster% nismkdir -s rep1.doc.com. groups_dir.sales.doc.com.
```



小規模またはテスト用の名前空間でないかぎりは推奨しませんが、nismkdir コマンドを使えば、独自のディレクトリを指定する代わりに、新しいディレクトリとして親ディレクトリのサーバーを使えます。次に 2 つの例を示します。

- 最初の例では、sales.doc.com. ディレクトリを作成し、これをその親ディレクトリのマスターと複製サーバーに関連付けます。

```
rootmaster% nismkdir sales.doc.com
```



2 番目の例では、sales.doc.com. ディレクトリを作成し、独自のマスターサーバーである smaster.doc.com. を指定します。

```
rootmaster% nismkdir -m smaster.doc.com. sales.doc.com.
```

複製サーバーは指定されないため、nismkdir を再び使用して複製を割り当てるまでは、新しいディレクトリにはマスターサーバーしかありません。sales.doc.com. ドメインがすでに存在する場合、上記の nismkdir コマンドは、salesmaster.doc.com. をその新しいマスターサーバーとし、その古いマスターサーバーを複製サーバーに格下げします。

複製サーバーを既存のディレクトリに追加する

この章では nismkdir コマンドを使用して複製サーバーを既存のシステムに追加する方法を説明します。簡単な方法としては、『*Solaris* ネーミングの設定と構成』で説明している nisserver スクリプトを使う方法があります。

以下の点に注意してください。

- ルートドメインサーバーはルートドメイン (またはその一部) に置く
- サブドメインサーバーは、サブドメインの 1 つ上の階層の親ドメイン (またはその一部) に置く。たとえば、名前空間に prime という名前のルートドメインと sub1 という名前のサブドメインがある場合は、次のようになる
  - prime ドメインにサービスを提供するマスターサーバーと複製サーバーは、prime がルートドメインであるため、prime ドメインの一部になる
  - sub1 サブドメインにサービスを提供するマスターサーバーと複製サーバーも、prime が sub1 の親ドメインであるため、prime ドメインの一部になる
- マスターサーバーまたは複製サーバーから複数のドメインにサービスを提供することもできるが、そのような措置は避けた方がよい

新しい複製サーバーを既存のディレクトリに割り当てるには、–s オプションと、既存のディレクトリ名を使います。

```
nismkdir -s replica-server existing-directory-name
nismkdir -s replica-server org_dir.existing-directory-name
nismkdir -s replica-server groups_dir.existing-directory-name
```

nismkdir コマンドは、ディレクトリがすでに存在することを知っているため、再び作成しません。単に、追加の複製サーバーを割り当てるだけです。たとえば、新しい複製サーバーマシン名が rep1 である場合は、以下のように入力します。

```
rootmaster% nismkdir -s rep1.doc.com. doc.com.
rootmaster% nismkdir -s rep1.doc.com. org_dir.doc.com.
rootmaster% nismkdir -s rep1.doc.com. groups_dir.doc.com.
```

注意 - nismkdir は必ず (複製サーバーではなく) マスターサーバー上で実行してください。複製サーバーで実行すると、マスターサーバーと複製サーバーの間で通信上の問題が発生します。

先述のように nismkdir を 3 回繰り返して実行した後は、以下の 3 つのディレクトリでマスターサーバーから nisping を実行する必要があります。

```
rootmaster# nisping doc.com.
rootmaster# nisping org_dir.doc.com.
rootmaster# nisping group_dir.doc.com.
```

この結果は以下のようになります。

```
rootmaster# nisping doc.com.
Pinging replicas serving directory doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at Wed Nov 18 19:54:38 1995
Replica server is rep1.doc.com.
 Last update seen was Wed Nov 18 11:24:32 1995
 Pinging ... rep1.doc.com
```

マスターサーバーの cron ファイルに、nisping コマンドが少なくとも 24 時間に一度 (この 3 つのディレクトリに対して) 実行されるよう設定しておくことをお勧めします。

nisrmdir コマンド

nisrmdir コマンドでは、ディレクトリを削除したり、ディレクトリと複製サーバーを切り離すことができます。ディレクトリを削除、またはディレクトリと複製サーバーを切り離すと、その NIS+ ドメインに対しては、マシンは NIS+ 複製サーバーとしては機能しなくなります。

ディレクトリの削除では、まずマスターサーバーと複製サーバーをディレクトリから切り離し、次にそのディレクトリを削除します。

- ディレクトリを削除するには、その親ディレクトリに対する削除権が必要です。

- ディレクトリから複製サーバーを切り離すには、そのディレクトリに対する変更権が必要です。

nisrmdir コマンドの実行で問題が生じる場合は、629ページの「複製の失敗からのNIS+ ディレクトリの削除または分離」を参照してください。

ディレクトリを削除する

ディレクトリ全体を削除し、そのマスターサーバーと複製サーバーを切り離すには、nisrmdir コマンドをオプションなしで実行します。

```
nisrmdir directory-name
nisping  domain
```

次の例では、doc.com. ディレクトリの下の manf.doc.com. ディレクトリを削除します。

```
rootmaster% nisrmdir manf.doc.com.
rootmaster% nisping doc.com.
```

複製サーバーをディレクトリから切り離す

複製サーバーをディレクトリから切り離すには、まず、ディレクトリの org_dir と groups_dir サブディレクトリを削除します。その時は、nisrmdir コマンドに –s オプションを付けて実行します。サブディレクトリが削除されたら、親ドメインに戻って nisping を実行してください。

```
nisrmdir -s replicaname org_dir   domain
nisrmdir -s replicaname groups_dir .domain
nisrmdir -s replicaname  domain
nisping domain
```

次の例では、manfreplica1 サーバーを manf.doc.com. ディレクトリから切り離します。

```
rootmaster% nisrmdir -s manfreplica1 org_dir.manf.doc.com.
rootmaster% nisrmdir -s manfreplica1 groups_dir.manf.doc.com.
rootmaster% nisrmdir -s manfreplica1 manf.doc.com.
rootmaster% nisping manf.doc.com.
```

nisrmdir –s コマンドを実行したが、切り離し対象の複製サーバーがダウンしていた、または通信が途絶していたという場合、「Cannot remove replica *name*: attempt to remove a non-empty table」というエラーメッセージが出されます。このような場合は、マスターサーバー上で nisrmdir –f –s *replicaname* コマンドを実行すれば、強制的に切り離すことができます。ただし、nisrmdir –f –s コマンドでダウン状態の複製サーバーを切り離した場合、その複製サーバーがオンラインに復帰したらただちに nisrmdir –f –s コマンドを再実行して複製サーバーの /var/nis ファイルシステムの中を整理する必要があります。この nisrmdir –f –s *replicaname* コマンドの再実行を怠ると、複製サーバーに古い情報が残り、問題が生じる原因にもなりかねません。

nisrm コマンド

nisrm コマンドは、標準の rm システムコマンドと似ています。このコマンドは、ディレクトリと空でないテーブルを除いて、名前空間からすべての NIS+ オブジェクトを削除します。nisrm コマンドを使うには、オブジェクトに対する削除権が必要です。しかし、この権利がない場合、–f オプションを使うことができます。このオプションは、アクセス権がなくても動作を強行しようと試みます。

nisgrpadm –d コマンド (249ページの「NIS+ グループを削除する」を参照) を使えばグループオブジェクトを削除でき、nistbladm –r または nistbladm –R (289ページの「テーブルを削除する」を参照) を使えばテーブルを空にできます。

```
nisrm [-if] object-name
```

表 13–2　nisrm 構文オプション

オプション	目的
-i	照会。オブジェクトを削除する前に確認を要求する。指定されたオブジェクト名が完全指定されていない場合、このオプションが自動的に使われる
-f	強制。たとえ適切なアクセス権がない場合でも、削除の強行を試みる。nischmod コマンドを使ってアクセス権の変更を試み、再度オブジェクトの削除を試みる

ディレクトリ以外のオブジェクトを削除する

ディレクトリ以外のオブジェクトを削除するには、nisrm コマンドを使い、オブジェクト名を指定します。

```
nisrm object-name...
```

次の例では、名前空間からグループとテーブルを削除します。

```
rootmaster% nisrm -i admins.doc.com. groups.org_dir.doc.com.
Remove admins.doc.com.? y
Remove groups.org_dir.doc.com.? y
```

rpc.nisd コマンド

rpc.nisd コマンドは、NIS+ デーモンを起動します。このデーモンは NIS 互換モードで実行でき、NIS クライアントからの要求にも応答できます。NIS+ デーモンを起動するにはアクセス権は不要ですが、そのすべての前提条件と関連作業を知っておく必要があります。これらについては、『*Solaris* ネーミングの設定と構成』を参照してください。

デフォルトでは、NIS+ デーモンはセキュリティレベル 2 で起動します。

デーモンを起動するには、以下のように入力します。

```
rpc.nisd
```

デーモンを NIS 互換モードで起動するには、以下のように入力します。

```
rpc.nisd -Y [-B]
```

DNS 転送機能によって NIS 互換デーモンを起動するには、以下のように入力します。

```
rpc.nisd -Y -B
```

表 13–3 rpc.nisd 構文オプション

オプション	目的
–S *security-level*	セキュリティレベルを指定する。0 は「NIS+ セキュリティを使用しない」、2 は「最高レベルの NIS+ セキュリティを使用する」という意味
–F	デーモンによって提供されるディレクトリのチェックポイント設定を強行する。この動作は、ディレクトリのトランザクションログを空にして、ディスク空間を解放することになる

DNS 転送機能によって NIS+ 互換デーモンを起動する

```
rpc.nisd
```

ルートマスター以外の任意のサーバーで NIS+ デーモンを起動するには、このコマンドをオプションなしで実行します。

デーモンは、デフォルトであるセキュリティレベル 2 で起動します。セキュリティレベル 0 でデーモンを起動するには、–S フラグを使います。

```
rpc.nisd -S 0
```

NIS 互換の NIS+ デーモンを起動する

ルートマスターを含む任意のサーバーで、NIS+ デーモンを NIS 互換モードで起動できます。–Y (大文字) オプションを使います。

```
rpc.nisd -Y
```

サーバーが再起動された場合にデーモンを NIS 互換モードで再起動させるには、サーバーの /etc/init.d/rpc ファイル内で EMULYP=Y を含む行をコメント解除しなければなりません。

DNS 転送 NIS 互換デーモンを起動する

NIS 互換モードで実行している NIS+ デーモンに DNS 転送機能を追加するには、rpc.nisd に –B オプションを追加します。

```
rpc.nisd -Y -B
```

サーバーが再起動された場合に、デーモンを DNS 転送 NIS 互換モードで再起動させるには、サーバーの /etc/init.d/rpc ファイル内の EMULYP=-Y を含む行をコメント解除し、次のように変更しなければなりません。

```
EMULYP -Y -B
```

NIS+ デーモンを停止する

NIS+ デーモンを停止するには、その実行モードが通常であろうと NIS 互換であろうと、他のデーモンと同じようにプロセスを終了させます。最初にそのプロセス ID を見つけ、次にプロセスを終了させます。次に例を示します。

```
rootmaster# ps -e | grep rpc.nisd
root 1081 1 61 16:43:33 ? 0:01 rpc.nisd -S 0
root 1087 1004 11 16:44:09 pts/1 0:00 grep rpc.nisd
rootmaster# kill 1081
```

nisinit コマンド

この節では、nisinit コマンドを使用してワークステーションクライアントを初期設定する方法について説明します。クライアントの初期設定は、nisclient スクリプトを使用するとさらに容易に行えます (『*Solaris* ネーミングの設定と構成』を参照)。

nisinit コマンドは、NIS+ クライアントとなるワークステーションを初期設定します。rpc.nisd コマンドと同様、nisinit コマンドを使うにはアクセス権は不要ですが、その前提条件と関連作業を知っておく必要があります。これらについては、『*Solaris* ネーミングの設定と構成』を参照してください。

クライアントを初期設定する

クライアントを初期設定するには、次の 3 つの方法があります。

- ホスト名による方法
- ブロードキャストによる方法
- コールドスタートファイルによる方法

前提条件と関連作業は、方法によってそれぞれ異なります。たとえば、ホスト名によってクライアントを初期設定するには、その前にクライアントの /etc/hosts ファイルまたは /etc/inet/ipnodesファイルに、使用するホスト名を登録しなければなりません。また nsswitch.conf ファイルの hosts の最初の選択肢として、files を指定する必要があります。IPv6 アドレスには、hosts の最初の選択肢としてipnodes を指定してください。 前提条件や関連作業を含め、各方法の詳細は、『*Solaris* ネーミングの設定と構成』を参照してください。nisinit コマンドを使う手順を次にまとめてみます。

ホスト名によってクライアントを初期設定するには、–c と –H のオプションを使い、クライアントがそのコールドスタートファイルを取得するサーバー名を指定します。

```
nisinit -c -H hostname
```

コールドスタートファイルによってクライアントを初期設定するには、–c と –C のオプションを使い、コールドスタートファイル名を指定します。

```
nisinit -c -C filename
```

ブロードキャストによってクライアントを初期設定するには、–c と –B のオプションを使います。

```
nisinit -c -B
```

ルートマスターサーバーを初期設定する

ルートマスターサーバーを初期設定するには、nisinit –r コマンドを使います。

```
nisinit -r
```

ここで次の情報が必要になります。

- ワークステーション (ルートマスターサーバーにする) のスーパーユーザーのパスワード

- 新しいルートドメイン名。ルートドメイン名は少なくとも 2 つの要素 (ラベル) で構成され、その末尾にドット (ピリオド) が打たれていなければならない (例: *something*.com)。最後の要素はインターネットの組織ドメイン (表 13–4 を参照) か、2〜3 文字の地域識別子 (日本であれば .jp) のどちらかにするように決められている

表 13–4 インターネットの組織ドメイン

ドメイン	種類
com	営利団体
edu	教育機関
gov	行政機関
mil	軍事組織
net	ネットワークサポートセンター
org	非営利団体
int	国際組織

nis_cachemgr コマンド

nis_cachemgr コマンドは、NIS+ キャッシュマネージャプログラムを起動します。このプログラムは、すべての NIS+ クライアント上で動作します。キャッシュマネージャは、名前空間で最もよく使われるディレクトリをサポートする NIS+ サーバーの位置情報キャッシュを管理します。位置情報はトランスポートアドレス、認証情報、生存期間値などです。

キャッシュマネージャが起動すると、クライアントのコールドスタートファイルから初期情報を取得し、それを /var/nis/NIS_SHARED_DIRCACHE ファイルにダウンロードします。

キャッシュマネージャは、クライアントワークステーションとして要求を行います。クライアントワークステーションには必ず適切な資格を持たせてください。そうしないと、性能が向上するどころか、キャッシュマネージャが性能を低下させます。

キャッシュマネージャの起動と停止

キャッシュマネージャを起動するには、nis_cachemgr コマンドを入力します。

```
client% nis_cachemgr
client% nis_cachemgr -i
```

–i オプションがない場合、キャッシュマネージャは再起動されますが、情報を /var/nis/NIS_SHARED_DIRCACHE ファイルに入れておきます。コールドスタートファイル内の情報は、このファイル内の既存情報に追加されます。–i オプションは、キャッシュファイルをクリアし、クライアントのコールドスタートファイルの内容からこれを再び初期設定します。

キャッシュマネージャを停止させるには、他のプロセスと同様にプロセスを終了します。

nisshowcache コマンド

nisshowcache コマンドは、クライアントのディレクトリキャッシュの内容を表示します。

NIS+ キャッシュの内容を表示する

nisshowcache コマンドは、/usr/lib/nis 内にあります。このコマンドはキャッシュヘッダーとディレクトリ名だけを表示します。ルートマスターサーバーから入力した例を次に示します。

```
rootmaster# /usr/lib/nis/nisshowcache -v
Cold Start directory:
Name : doc.com.
Type : NIS
Master Server :
```

```
 Name : rootmaster.doc.com.
 Public Key : Diffie-Hellman (192 bits)
 Universal addresses (3)
 . .
Replicate:
 Name : rootreplica1.doc.com.
 Public Key : Diffie-Hellman (192 bits)
 Universal addresses (3)
 .
 .
 .
 Time to live : 12:0:0
 Default Access Rights :
```

ping とチェックポイントを実行する

NIS+ データセットに変更を加えると、その変更は、当該 NIS+ ドメイン (またはサブドメイン) のマスターサーバーのメモリに格納されます。変更の記録はマスターサーバーのトランザクションログ (/var/nis/data/trans.log) にも残されます。

通常、NIS+ データセットに変更が加えられると、その 120 秒 (2 分) 後に、マスターサーバーから当該ドメインの複製サーバーにその変更の内容が転送されます。この転送プロセスのことを「ping」といいます。マスターサーバーから複製サーバーへの ping が実行されると、通知された変更内容に従って複製サーバーのデータセットが更新されます。これにより、変更された NIS+ データがマスターサーバーと複製サーバーの両方のメモリに格納されることになります。

自動 ping プロセスが不調で複製サーバーのデータセットが更新されないこともあります。その場合は、272ページの「ping を強制的に実行する」の説明に従って ping を強制的に実行する必要があります。複製サーバーが最新の NIS+ データどおりに正しく更新されているかどうか不安な場合は、271ページの「最新更新時間を表示する」の説明に従って複製サーバーが最後に更新されたのはいつであるかを確認してください。

NIS+ データセットに対する変更は、サーバーのメモリーに格納され、トランザクションログに記録されたのち、ディスク上の NIS+ テーブルに書き込まれなければなりません。この NIS+ テーブルを更新することを「チェックポイントを実行する」といいます。

チェックポイントの実行は自動的には行われません。272ページの「ディレクトリにチェックポイントを実行する」の説明に従ってチェックポイントコマンドを実行する必要があります。

nisping コマンド

nisping コマンドには次の用途があります。

- 複製サーバーが最後に ping されたのはいつであるかを表示する (詳細は、271ページの「最新更新時間を表示する」を参照)
- 自動 ping サイクルが不調に終わった場合、マスターサーバーから複製サーバーへの ping を強制的に実行する (詳細は、272ページの「ping を強制的に実行する」を参照)
- サーバーにチェックポイントを実行する (詳細は、272ページの「ディレクトリにチェックポイントを実行する」を参照)

最新更新時間を表示する

–u オプションを指定して nisping コマンドを実行すると、ローカルドメインのマスターサーバーと複製サーバーが更新された時間が表示されます。

```
/usr/lib/nis/nisping -u [domain]
```

他のドメインで最後に更新が行われた時間を表示するには、コマンド行にそのドメイン名を指定します (–u オプションを指定して nisping コマンドを実行した場合、複製サーバーに対する ping は一切実行されない点に注意)。

たとえば、ローカルドメイン doc.com. で複製サーバーが最後に更新された時間を表示するには、次のように入力します。

```
rootmaster# /usr/lib/nisping -u
Last updates for directory doc.com.:
Master server is rootmaster.doc.com.
 Last update occurred at Wed Nov 25 10:53:37 1992
Replica server is rootreplica1.doc.com.
 Last update seen was Wed Nov 25 10:53:37 1992
```

ping を強制的に実行する

–u オプションを指定して nisping コマンドを実行した結果、複製サーバーが正しく更新されていないことがわかったとします。そのような場合は、nisping コマンドを実行することにより、マスターサーバーからドメイン内のすべての複製サーバーに対して、または特定の複製サーバーに対して、ping を強制的に実行できます。

すべての複製サーバーを ping の対象とする場合、次に示すように、nisping コマンドをオプションなしで実行します。

```
/usr/lib/nis/nisping
```

これにより、マスターサーバーからドメイン内のすべての複製サーバーへの ping が強制的に実行されます。ローカルドメイン doc.com. のすべての複製サーバー に対する ping を強制的に実行するには、次のように入力します。

```
rootmaster# /usr/lib/nis/nisping
Pinging replicas serving directory doc.com.:
Master server is rootmaster.doc.com.
 Last update occurred at Wed Nov 25 10:53:37 1992
Replica server is rootreplica1.doc.com.
 Last update seen was Wed Nov 18 11:24:32 1992
 Pinging ... rootreplica1.doc.com.
```

ローカルドメインでないドメイン内のすべての複製サーバーに対し ping を実行するには、ドメイン名を指定します。

```
/usr/lib/nis/nisping domainname
```

特定の 1 台のホスト上のすべてのディレクトリにあるすべてのテーブルに対し ping を実行することもできます。この場合、–a オプションを使用します。

```
/usr/lib/nis/nisping -a hostname
```

ディレクトリにチェックポイントを実行する

スワップの頻度やディスク領域との関連でトランザクションログが大きくなりすぎる場合、各ドメインおよびサブドメインに対しては、少なくとも 24 時間に 1 回はチェックポイントを実行する必要があります。

注 - 大きなドメイン、または大きなトランザクションログを持つドメインに対してチェックポイントを実行すると、終了するまでにかなりの時間がかかり、その間 NIS+ サーバーが拘束され、NIS+ サービスの速度が低下します。サーバーは、チェックポイントを実行している間もサービス要求に応答しますが、更新できません。できるだけ、システムが混んでいない時間帯を選んでチェックポイントを実行することをお勧めします。チェックポイントの実行スケジュールは cron ファイルで調整できます。

チェックポイントを実行するには、当該ドメインのマスターサーバー上で nisping –C コマンドを実行します。先にすべての複製サーバーに対して ping を実行してから、チェックポイントを実行することをお勧めします。その場合には、複製サーバーに対して常に最新のデータでチェックポイントを実行できます。

- 特定のディレクトリに対してチェックポイントを実行するには、次に示すように、–C オプションを指定して nisping コマンドを実行します。

```
rootmaster# /usr/lib/nis/nisping
rootmaster# /usr/lib/nis/nisping -C org_dir
```

- ローカルドメイン内のすべてのディレクトリに対してチェックポイントを実行するには、次に示すように、–C –a オプションを指定して nisping コマンドを実行します。

```
rootmaster# /usr/lib/nis/nisping
rootmaster# /usr/lib/nis/nisping -C -a
```

サーバーによってそのトランザクションログの情報が NIS+ テーブルに転送されると、ログファイル内のトランザクションは、ディスク領域の節約のため消去されます。

doc.com. ドメイン内のすべてのディレクトリに対してチェックポイントを実行するには、次のように入力します。

```
rootmaster# /usr/lib/nis/nisping -C -a
Checkpointing replicas serving directory doc.com. :
Master server is rootmaster.doc.com.
 Last update occurred at Wed May 25 10:53:37 1995
Master server is rootmaster.doc.com.
checkpoint has been scheduled with rootmaster.doc.com.
Replica server is rootreplica1.doc.com.
 Last update seen was Wed May 25 10:53:37 1995
Replica server is rootreplica1.doc.com.
checkpoint has been scheduled with rootmaster.doc.com.
```

nislog コマンド

nislog コマンドは、トランザクションログの内容を表示します。

```
/usr/sbin/nislog
/usr/sbin/nislog -h [number]
/usr/sbin/nislog -t [number ]
```

表 13–5 nislog コマンドのオプション

オプション	機能
–h [*num*]	ログの先頭からトランザクションを表示する。数字を指定しなければ、最初のトランザクションから表示が開始される。0 を指定すると、ログヘッダーだけが表示される
–t [*num*]	ログの末尾からトランザクションを表示する。数字を指定しなければ、最後のトランザクションから表示が開始される。0 を指定すると、ログヘッダーだけが表示される
–v	冗長モード

トランザクションログの内容を表示する

各トランザクションは、トランザクションの明細部とオブジェクト定義のコピー部の 2 つの部分から構成されます。

doc.com. ディレクトリが最初に作成されたときに作られたトランザクションログエントリを次の例に示します。「XID」はトランザクション ID を意味します。

```
rootmaster# /usr/sbin/nislog -h 1
NIS Log printing facility.
NIS Log dump:
 Log state : STABLE
Number of updates : 48
Current XID : 39
Size of log in bytes : 18432
***UPDATES***
@@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@@@@
#00000, XID : 1
Time : Wed Nov 25 10:50:59 1992
Directory : doc.com.
Entry type : ADD Name
Entry timestamp : Wed Nov 25 10:50:59 1992
Principal : rootmaster.doc.com.
Object name : org_dir.doc.com.
..................Object.....................
Object Name : org_dir
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : doc.com.
Access Rights : r---rmcdr---r---
Time to Live : 24:0:0
Object Type : DIRECTORY
Name : 'org_dir.doc.com.'
Type: NIS
Master Server : rootmaster.doc.com.
.
.
.............................................
@@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@@@@
#00000, XID : 2
```

nischttl コマンド

nischttl コマンドは、名前空間内のオブジェクトまたはエントリの生存期間値を変更します。キャッシュマネージャは、この生存期間値を使って、キャッシュエントリをいつ期限切れにするかを決めます。生存期間を指定するには、合計秒数か、日、時間、分、秒の組み合わせのいずれかを使います。

オブジェクトまたはエントリに割り当てる生存期間値は、オブジェクトの安定性に左右されます。よく変化するオブジェクトの場合、生存期間値を低くします。安定したオブジェクトの場合、高い値を指定します。高い生存期間値であれば、1 週間

などを指定し、低い値であれば 1 分未満を指定します。パスワードエントリの生存期間値は約 12 時間とし、1 日に 1 回パスワード変更ができるようにする必要があります。RPC テーブル内のエントリなど、あまり変化しないテーブルのエントリには、数週間の値を設定できます。

オブジェクトの生存期間を変更するには、そのオブジェクトに対する変更権が必要です。テーブルエントリの生存期間を変更するには、テーブルに対する変更権が必要であり、テーブルがなければエントリに対して、エントリもなければ変更したい列に対して変更権が必要です。

オブジェクトまたはテーブルエントリの現在の生存期間値を表示するには、第 10 章で説明する `nisdefaults –t` コマンドを使います。

オブジェクトの生存期間値を変更するには、以下のいずれかのように入力します。

```
nischttl time-to-live object-name
```

```
nischttl [-L] time-to-live object-name
```

エントリの生存期間値を変更するには、以下のいずれかのように入力します。

```
nischttl time-to-live \
  [column=value,...], \
  table-name
```

```
nischttl [-ALP] time-to-live \
  [column=value,...], \
  table-name
```

time-to-live には以下のものを指定します。

- 「秒」

  数字だけで文字を指定しなければ、単位が秒であると解釈される (例 : `1234` は 1234 秒)。数字の後に `s` をつけると、単位が秒であると解釈される (例 : `987s` は 987 秒)。日、時間、分などとともに指定する場合は、秒であることを明らかにするため `s` をつける

- 「分」

  数字の後に `m` を指定すると、単位が分であると解釈される (例： `90m` は 90 分)

- 「時間」

  数字の後に h をつけると、単位が時間であると解釈される (例： 9h は 9 時間)

- 「日」

  数字の後に d をつけると、単位が日であると解釈される (例：7d は 7 日)

以上の値は組み合わせて使うことができます。たとえば「 4d3h2m1s 」は、4 日 3 時間 2 分 1 秒を意味します。

nischttl コマンドでは以下のフラグを使用できます。

表 13–6 nischttl 構文のオプション

オプション	目的
–A	すべて。[*column=value*] 指定に合致する全エントリに変更を適用する
–L	リンク。リンクをたどり、リンク自体ではなく、リンクされたオブジェクトまたはエントリを変更する
–P	パス。条件を満足するエントリが 1 つ見つかるまで、パスをたどる

オブジェクトの生存期間を変更する

オブジェクトの生存期間を変更するには、生存期間の値とオブジェクト名を指定して nischttl コマンドを入力します。–L コマンドを追加すれば、リンクされたオブジェクトにまで変更を拡張できます。

```
nischttl -L time-to-live    object-name
```

生存期間は秒か、日、時間、分、秒の組み合わせかで指定できます。前者の場合、秒数だけを指定します。後者の場合、日数、時間数、分数と秒数に「 s、m、h、d」をつけてください。同じ動作をする 2 つの例を次に示します。

```
client% nischttl 86400 sales.doc.com.
client% nischttl 24h sales.doc.com.
client% nischttl 2d1h1m1s sales.doc.com.
```

最初の 2 つでは、sales.doc.com. ディレクトリの生存期間が 86,400 秒、つまり 24 時間に変更されます。3 つ目では、hosts テーブル内の全エントリの生存期間が 176,461 秒、つまり 2 日と 1 時間 1 分 1 秒に変更されます。

テーブルエントリの生存期間を変更する

エントリの生存期間を変更するには、インデックス付きのエントリフォーマットを使います。–A、–L、または –P のどのオプションでも使えます。

```
nischttl [-ALP] time-to-live \
   [column=value,...], \
   table-name
```

注 - Cシェル を使用している場合、[ ] がメタキャラクタとして解釈されないように引用符で囲みます。

次に示す例は上記の例と似ていますが、オブジェクトではなく、テーブルエントリの値を変更します。

```
client% nischttl 86400 '[uid=99],passwd.org_dir.doc.com.'
client% nischttl 24h `[uid=99],passwd.org_dir.doc.com.'
client% nischttl 2d1h1m1s `[name=fred],hosts.org_dir.doc.com'
```

第 14 章

NIS+ テーブルの管理

この章では、NIS+ テーブルとその管理方法について説明します。(デフォルトの NIS+ テーブルの詳細は、付録 C を参照してください。)

NIS+ テーブル

NIS+ で使われる情報は NIS+ テーブルに格納されます。Solaris 8 リリースに付属のデフォルトの NIS+ システムテーブルについては、付録 C を参照してください。

これらのコマンドの構文やオプションなどの詳細は、`nis+(1)` のマニュアルページを参照してください。

nistbladm コマンド

注 - NIS+ テーブルに関連した作業のうちいくつかは、 Solstice AdminSuite™ ツールを使用すると容易に行えます。

`nistbladm` コマンドは NIS+ テーブル管理コマンドの中でも最も重要なコマンドであり、NIS+ ディレクトリオブジェクトに格納されている NIS+ テーブル上で使います。`nistbladm` コマンドを使うと、NIS+ テーブルまたはそのエントリを作成、修正、削除できます。ただし、ディレクトリのないところにテーブルを作成はできません。同様に、テーブルと列が定義されていないところにエントリを追加できません。

テーブルを作成するには、そのテーブルが所属するディレクトリに対する作成権が必要です。テーブルを削除するには、そのディレクトリに対する削除権が必要で

す。テーブルの内容を変更 (エントリの追加、変更、または削除) するには、そのテーブルまたはエントリの変更権が必要です。

nistbladm 構文

nistbladm コマンドの一般的な構文は次のとおりです。

```
nistbladm -a column=" value" \
  column=" value" \
  column=" value" \
  ... tablename
nistbladm -a indexedname
```

```
nistbladm options \
   [columspec | columnvalue] \
   [tablename | indexedname ]
```

- *columnspec* には、テーブル内に作成する列を指定する (具体的な指定方法については、287ページの「テーブルの列を指定する」 を参照)
- *columnvalue* には、*tablename* によって表されるテーブルの中の特定のセルを指定する (具体的な指定方法については、282ページの「nistbladm と列の値」を参照)
- *tablename* には、テーブル名 (hosts.org_dir.doc.com. など) を指定する
- *indexedname* には、特定のテーブルの中の特定のセル値を指定する (具体的な指定方法については、282ページの「nistbladm と列の値」を参照) 。*indexedname* は本質的に、*columnvalue* と *tablename* の役割を兼ね備えている

表 14–1 nistbladm コマンドのオプション

オプション	説明
–a | –A	既存の NIS+ テーブルにエントリを追加する。–a を指定した場合、nistbladm コマンドを実行すると既存のエントリが上書きされる場合には、エラーが返される。–A を指定した場合、nistbladm コマンドが既存エントリを上書きする場合でも強制的に実行される (289ページの「エントリをテーブルに追加する」を参照)
–D *defaults*	別のデフォルト特性を使ってオブジェクトを作成する (詳細は、デフォルトの nistbladm(1) のマニュアルページを参照)

表 14–1 nistbladm コマンドのオプション 続く

オプション	説明
–d	テーブルを破棄する (289ページの「テーブルを削除する」を参照)
–c	テーブルを作成する (286ページの「テーブルを作成する」を参照)
–r | –R	既存の NIS+ テーブルからエントリ (1 つまたは複数) を削除する。–r を指定した場合、複数のエントリの削除につながる nistbladm コマンドは実行されず、エラーが返される。–R を指定した場合、複数のエントリの削除につながる nistbladm コマンドであっても強制的に実行される (296ページの「テーブルからエントリを削除する」を参照)
–m	テーブルエントリを修正するためのオプション。旧リリースとの互換性を維持するためにだけ残されている。エントリを修正するのであれば、–e オプションまたは –E オプションを使うほうが望ましい
–e | –E	既存の NIS+ テーブルのエントリを修正する。–e を指定した場合、複数のエントリの変更につながる nistbladm コマンドは実行されず、エラーが返される。–E を指定した場合、複数のエントリの変更につながる nistbladm コマンドであっても強制的に実行される (293ページの「エントリを修正する」を参照)

nistbladm と列の値

列の値は、テーブル内の個々のエントリを識別するために使われます。列の値の書式は次のとおりです。

```
columname="value", \
 columnname="value ", ...
```

- *columname* はテーブルの列名を示す
- *value* は列内の特定のセルの内容 (値) を示す。この値により、テーブルの行を識別することができる (*column=value* を指定してテーブルデータを作成または修正する場合は、必ず *value* を引用符で囲むこと)

たとえば、マシン名と IP アドレスを登録した hosts という名前のテーブルがあるとします。

表 14–2　サンプルテーブル hosts

IP アドレス	名前	別名
129.146.168.4	altair	
129.146.168.119	deneb	mail
129.146.168.120	regulus	dnsmaster
129.146.168.121	regulus	dnsmaster
129.146.168.11	sirius	

このサンプルテーブルの altair エントリ (行) を識別するには、次の 3 通りの *column=value* の指定方法が考えられます。

- name=altair
- address=129.146.168.4
- name=altair,address=129.146.168.4.

上記のテーブルで注目すべき点としては、regulus という名前のマルチホームマシンに 2 つの IP アドレスが割り当てられていることです。この場合、ホストマシン regulus の *column=value* は 2 つの行を示します。そこで、最初の regulus 行だけを識別したいという場合は次のいずれかを入力します。

- address=129.146.168.120
- address=129.146.168.120.,name=regulus,dnsmaster

注 - nistbladm コマンドの一部のオプションには、テーブル内のすべての列に *column=value* を指定しなければならないものがあります。

nistbladm、検索可能列、キー、列の値

NIS+ テーブルを作成する際は、s フラグまたは i フラグを指定し、1 つまたは複数の列が検索可能になるようにします (287ページの「テーブルの列を指定する」を参照)。なお、niscat –o *tablename* と入力すれば、テーブルの列とその特性を一覧表示できます。

テーブル内で行の「識別」に使われるのが検索可能列です。検索可能列の値 (値の組み合わせ) はすべて一意でなければなりません。したがって、検索可能列が 1 つしかないテーブルの場合、各行の検索可能列の値はすべて一意でなければならず、重複は一切許されません。

ここで、`city` という名前の検索可能列と `country` という名前の検索不可列からなるテーブルを作成する場合を想定します。次に示すのは、正しい (`City` 列に重複がない) テーブルの例です。

City	**Country**
San Francisco	United States
Santa Fe	United States
Santiago	Chile

次は正しくない (`City` 列に重複がある) テーブルの例です。

City	**Country**
London	Canada
London	England

1 つのテーブルに検索可能列が複数ある場合は、検索可能列どうしの値の組み合わせが一意であればかまいません。ここでは、`Lastname` および `Firstname` という 2 つの検索可能列と、`city` という検索不可列からなるテーブルを作成する場合を想定します。次に示すのは、正しい (検索可能列の値の組み合わせに重複がない) テーブルの例です。

Lastname	**Firstname**	**City**
Kuznetsov	Sergei	Odessa
Kuznetsov	Rima	Odessa
Sergei	Alex	Odessa

次は正しくない (検索可能列の値の組み合わせに重複がある) テーブルの例です。

Lastname	Firstname	City
Kuznetsov	Rima	Odessa
Kuznetsov	Rima	Chelm

NIS+ のコマンドはいずれも、検索可能列の値に基づいて特定の行を識別します。

nistbladm とインデックス名

NIS+ には、テーブル名と列の値の検索基準の組み合わせ (これを「インデックス名」という) によって特定のテーブルの特定のエントリを識別する、というテーブル管理の方法もあります。インデックス名の書式は次のとおりです。

```
[search_criteria], tablename.directory
```

search_criteria には検索基準を指定しますが、このとき角カッコ ([]) で囲むのを忘れないでください。書式は次のとおりです。

```
columname=value, \
  columname=value ,...
```

columname=value にはテーブルの検索可能列の値を指定します (282ページの「nistbladm と列の値」を参照)。

たとえば、表 14–2 の `altair` エントリを識別するには、次のようにインデックス名を指定します。

```
[addr=129.146.168.4,cname=altair],hosts.org_dir.doc.com.
```

`nistbladm -R` コマンドを使用すると、間になにも入れない角カッコ `[ ]`をすべてのテーブル列を指定するワイルドカードとして使用し、テーブル中のすべてのエントリを一度に削除できます。

nistbladm とグループ

Solaris の NIS+ 環境には 3 種類のグループがあります。

- UNIXグループ - UNIX グループに関する情報は `groups.org_dir` テーブルに格納される。UNIX グループ情報の管理には `nistbladm` コマンドを使う

- ネットグループ - ネットグループに関する情報は `netgroups.org_dir` テーブルに格納される。ネットグループ情報の管理には `nistbladm` コマンドを使う
- NIS+ グループ - NIS+ グループに関する情報は `groups_dir` ディレクトリオブジェクトのテーブル (1 つまたは複数) に格納される。NIS+ グループ情報の管理には `nisgrpadm` コマンドを使う

注 - NIS+ グループの管理に `nistbladm` を使うことはできません。

その他のグループの詳細は、241ページの「Solaris グループ」を参照してください。

テーブルを作成する

NIS+ テーブルには、少なくとも 1 つの列が必要で、その列のうち少なくとも 1 つが検索可能でなければなりません。NIS+ テーブルを作成するには、`nistbladm` コマンドに `–c` オプションを付けて使います。

```
nistbladm -c tabletype columnspec \
  ... tablename
```

- *tabletype* は、単にテーブルをあるテーブルクラスに所属するものとして識別する文字列です。
- *columnspec* 引数には、各列の名前と特性を指定します。1 つの *columnspec* には、新規テーブルに含める、それぞれの列を入力してください。

```
nistbladm -c tabletype columnspec columnspec  \
columnspec tablename
```

columnspec の書式については、287ページの「テーブルの列を指定する」を参照してください。

テーブルの列を指定する

各 *columnspec* エントリは、以下の形式のように、2 つから 4 つの要素から成っています。

```
name=type,rights:
```

表 14–3 テーブルの列の構成要素

構成要素	説明
name	列の名前
=	等号記号 (必須)
type	(オプション) S、I、または C で列の種類を指定する。表 14–4 参照。*type* を指定しないと、その列はデフォルトとなる
rights	(オプション) アクセス権を指定する。ここに指定したアクセス権は、テーブル全体や特定エントリに付与したアクセス権より優先される。*access* を指定しないと、その列のアクセス権は、テーブル全体や特定エントリに付与したアクセス権になる。アクセス権の構文については、190ページの「コマンドによるアクセス権の指定」を参照

列には、次に挙げる種類のうち 1 つを指定できます。

表 14–4 列の種類

種類	説明
	等号 (=) のみ。列の種類は指定しない。その列は検索可能にもならなければ、暗号化されることもない
S	検索可能
I	大文字と小文字の区別をしない。NIS+ コマンドで列を検索する場合、大文字と小文字を区別しない
C	暗号化する

NIS+ の各コマンドは、列全体を調べ、検索可能列の値に基づいて個々の行を識別します。検索可能列は `S` フラグまたは `I` フラグで指定します (データベースの分野では、検索可能列のことをキー列といいます)。テーブルの最初の列は必ず検索可能にします。その他の列は検索可能にする必要はありません。検索可能列はそのテーブル内の行の識別に使われるため、検索可能列を複数にする場合は、最初の列とその隣の列を検索可能にします。検索可能列の間に通常の列を置くことはできません。したがって、常に先頭の列を検索可能とし、検索可能列の数を 1 つ増やすたびにその隣の列を検索可能にしていきます (検索可能列の詳細は、283ページの「nistbladm、検索可能列、キー、列の値」を参照)。

アクセス権だけを指定する場合、コンマを使用する必要はありません。`–S`、`–I`、`–C` フラグの 1 つか複数を指定する場合は、アクセス権の前にコンマを追加します。

以下の例では、上と同じテーブルが作成されますが、最初の 2 列にはその列に固有なアクセス権が付与されます。

```
master% nistbladm -c depts Name=I,w+m Site=w+m Name=C \
divs.mydir.doc.com.
```

テーブル作成時の列のアクセス権の指定については、200ページの「テーブル作成時の列権限設定」を参照してください。

注 - NIS+ では、すべての列エントリが NULL で終了するものと仮定しています。NIS+ テーブルに情報を書き込むアプリケーションやルーチンは、列エントリをすべて NULL で終了するように構成しなければなりません。

自動マウントテーブルを追加作成する

自動マウントテーブルには 2 つの列しか作ることができません。1 番目の列には `key` という名前を付け、2 番目の列には `value` という名前を付けます。たとえば、`auto1` という名前の自動マウントテーブルを作成するには、次のように入力します。

```
master% nistbladm -c key-value key=S value= auto1.org_dir.doc.com.
```

テーブルを削除する

テーブルを削除するには、–d オプションを使ってテーブル名を入力します。

```
nistbladm -d tablename
```

テーブルを削除するには、その前にテーブルが空になっていなければなりません (296ページの「テーブルからエントリを削除する」を参照)。次の例では、doc.com. ディレクトリから divs テーブルを削除します。

```
rootmaster% nistbladm -d divs.doc.com.
```

エントリをテーブルに追加する

エントリ (行) をテーブルに追加するには、nistbladm コマンドに –a オプションまたは –A オプションを指定し、続けて column=value のペア (1 つまたは複数)、テーブル名の順に指定します。あるいは、nistbladm コマンドに –a オプションまたは –A オプションを指定し、続けてインデックス名を指定します (285ページの「nistbladm とインデックス名」を参照)。

```
nistbladm [-a | -A] indexedname
nistbladm [-a | -A] column="value" \
column="value" \
  ... tablename
```

–a オプションまたは –A オプションを指定して既存のテーブルに新しいエントリ (行) を追加する場合は、次の点に注意してください。

- *value* は必ず引用符で囲む。たとえば、新たに追加するエントリの cname 列の値を deneb にする場合は、*column=value* ペアを cname="deneb" と指定する
- テーブル内のすべての列の値を指定する
- 追加するエントリ (行) の列を空白にする場合は、*column*=" " と指定する。つまり、*value* には、引用符で囲んだ半角スペースを指定する

注 - NIS+ はネームサービスであり、設計上、そのテーブルにはオブジェクト本体ではなく、オブジェクトのリファレンスが格納されます。NIS+ は、最適化された状態では 10,000 におよぶオブジェクトをサポートし、すべてのテーブルのサイズを合計しても 10M バイトを超えることはありません。NIS+ では、1 つの列のフィールドサイズの合計が 7k を超えるようなテーブルはサポートされません。テーブルが大きすぎると、`rpc.nisd` は失敗します。

-a オプションを指定してエントリを追加する

`nistbladm` コマンドに –a オプションを指定しておくと、追加するエントリと同じエントリがすでに存在する場合は、エントリの上書きは行われずにエラーが返されます。あるエントリの検索可能列の値が、追加するエントリの検索可能列の値とまったく同じである場合、そのエントリは「すでに存在しているもの」と判断されます。(このとき、検索不可列の値は一切考慮されません。)

–a オプションを指定する場合、次のように、テーブル内のすべての列の値を指定する必要があります。

```
nistbladm -a column=" value" \
   column=" value" \
   ... tablename
nistbladm -a indexedname
```

(テーブルのすべての列名とその特性を表示する場合は、`niscat –o` *tablename* コマンドを実行します。)

たとえば、`depts` テーブルに新しい行を追加する場合は、次に示すように、列の数だけ *column=value* ペアを指定します。

```
rootmaster% nistbladm -a Name='R&D' Site='SanFran'  \
 Name='vattel' depts.doc.com.
```

なお、これと同じエントリをインデックス名を指定して追加する場合は、次のように入力します。

```
rootmaster% nistbladm -a [Name='R&D',Site='SanFran' \
Name='vattel'],depts.doc.com.
```

いずれの場合も、次のようなテーブルとなります。

Dept	Site	Name
R&D	SanFran	vattel

C シェルを使っている場合は、角カッコを含む数式を引用符でオフセットすることもできます。

nistbladm コマンドでは 1 回につき 1 つのエントリしか追加できません。つまり、追加する行の数だけ nistbladm コマンドを実行する必要があります。

追加するエントリ (行) の各列と同じ値を持つ行がすでに存在する場合、nistbladm –a はエラーを返します。1 つのテーブルの中に同じ行が存在することはできません。検索可能列の値が同一である場合、それらの行は同一であるとみなされます。このとき、検索不可列の値は考慮されません。

たとえば、Dept 列と Site 列が検索可能であって、Name 列は検索可能ではないテーブルに対して nistbladm を実行すると、次の 2 つの行は同一であるとみなされます。

Dept (検索可能)	Site (検索可能)	Name (検索不可)
Sales	Vancouver	Hosteen
Sales	Vancouver	Lincoln

この場合、nistbladm –a を実行して Sales Vancouver Lincoln という行を作成できません。

しかし、複数の検索可能列のどちらか一方の値が重複するだけであれば、nistbladm –a を実行して新しい行を作成できます。たとえば、以下の 2 つのコマンドを実行すると、一部の値が異なるだけの 2 つの列を作成できます。

```
rootmaster% nistbladm -a Dept='Sales' \
Site='Vancouver' Name='hosteen' staff.doc.com.
rootmaster% nistbladm -a Dept='Sales' \
Site='SanFran' Name='lincoln' staff.doc.com.
```

これらのコマンドを実行すると、検索可能列の一部が同じで、すべてが同じではない 2 つの行が作成されます。

Dept	Site	Name
Sales	Vancouver	hosteen
Sales	SanFran	lincoln

-A オプションを指定してエントリを追加する

–A オプションは、nistbladm コマンドで既存のエントリを上書きするアプリケーションに使用します。–a と同様に、–A もテーブルにエントリを追加するためのオプションです。しかし、追加するエントリがすでに存在する場合、エラーを返して終了するのではなく、既存のエントリを上書きします。

–A オプションを指定する場合、エントリ内のすべての列の値を指定する必要があります。

ここで、Dept 列と Site 列が検索可能である次のテーブルを想定します。

Dept (検索可能)	Site (検索可能)	Name
Sales	SanFran	Lincoln

このテーブルに対して、次のコマンドを実行します。

```
rootmaster% nistbladm -A Name=Sales Site=SanFran \
Name=Tsosulu depts.doc.com.
```

Name=Sales Site=SanFran はすでに存在するので、オプションが –a であれば、上記のコマンドはエラーを返します。しかし、上記のコマンドでは –A が指定されているので、既存の行が上書きされます。

Dept	Site	Name
Sales	SanFran	Tsosulu

エントリを修正する

既存のエントリを修正 (編集) する場合は、–e オプションまたは –E オプションを指定します。Solaris 8 リリースでは、旧リリースとの互換性を維持するため、–m オプションも残されています。新しいアプリケーションまたはコマンド行での操作には、–e オプションまたは –E オプションを使うことをお勧めします。

テーブル内の既存のエントリ (行) を編集するには、nistbladm コマンドに –e オプションまたは –E オプションを指定し、それに続けて、値を変更した *column=value* のペア (1 つまたは複数) とテーブル内の特定の行を示すインデックス名を指定します (285ページの「nistbladm とインデックス名」を参照)。

```
nistbladm [-e | -E] column=" value" \
 column=" value" \
 ... indexedname
```

–e オプションまたは –E オプションを指定して既存エントリ (行) を修正する場合は、次の点に注意してください。

- *value* は必ず引用符で囲む。たとえば、cname 列の値を deneb に変更する場合は、*column=value* ペアを cname="deneb" と指定する
- 検索可能列の値は、1 行 (エントリ) 単位でしか変更できない
- エントリ (行) の列を空白にする場合は、*column*=" " と指定する。つまり、*value* には、引用符で囲んだ半角スペースを指定する

-e オプションを指定してエントリを編集する

–e オプションを指定した場合、複数のエントリの検索可能列値を変更することになる nistbladm コマンドは実行されず、エラーが返されます。このとき、検索不可列の値は考慮されません。

```
nistbladm column=" value" \
 column="  value" \
 ... indexedname
```

–e オプションを指定した場合、変更する列の値を指定するだけです。

ここで次のテーブルについて考えます。

Dept	Site	Name
Sales	SanFran	Tsosulu

Name 列の値を Chandar に変更するには、次のように入力します。

```
master% nistbladm -e Name="Chandar" [Dept='Sales',Site='SanFran'], \
 depts.doc.com.
```

修正後のテーブルは次のようになります。

Dept	Site	Name
Sales	SanFran	Chandar

(上記の例では、インデックス名に Name 列が含まれていません。これは、Name 列が検索可能ではないためです。)

C シェルを使っている場合は、角カッコを含む数式を引用符でオフセットすることもできます。

–e オプションを指定して検索可能列の値を編集するには、新たに指定した値が (インデックス名によって識別される) 1 行にだけ適用されることが条件になります。したがって、Dept 列の値を Manf に変更するには、次のように入力します。

```
master% nistbladm -e Dept="Manf" [Dept='Sales',Site='SanFran'], \
 depts.doc.com.
```

Dept (検索可能)	Site (検索可能)	Name
Manf	SanFran	Chandar

しかし、検索可能列に Manf と SanFran という値を持つエントリが既に存在する場合は、nistbladm –e はエラーを返します。

同一のエントリが対象である場合は、複数の列の変更を指定できます。たとえば、Dept 列と Name 列の値を変更するには、次のように入力します。

```
master% nistbladm -e Dept="Manf" Name=''Thi'' \
 [Dept='Sales',Site='SanFran'],depts.doc.com.
```

Dept (検索可能)	Site (検索可能)	Name
Manf	SanFran	Thi

-E オプションを指定してエントリを編集する

–E オプションは、nistbladm コマンドで既存のエントリを上書きする場合のアプリケーションに使用します。

ここで次のテーブルを想定します。

Dept (検索可能)	Site (検索可能)	Name
Sales	SanFran	Chandar
Sales	Alameda	Achmed

このテーブルに対して次のコマンドを実行します。

```
master% nistbladm -E Site="Alameda'' Mgr="Chu"  \
[Div='Sales',Site='SanFran'],depts.doc.com.
```

すると、Sales SanFran Chandar 行が Sales Alameda Chu に変わります。しかも、キーの値 Sales Alameda によって識別される Sales Alameda Achmed 行までもが変更の対象になります。その結果、コマンド実行前には 2 つあったはずの行が 1 つになります。

Dept (検索可能**)**	**Site (**検索可能**)**	**Name**
Sales	Alameda	Chu

これは複数の行を編集することになるので、オプションが –e であれば、上記のコマンドはエラーを返します。しかし、実際には –E が指定されているので、複数のエントリ (行) が編集されます。

テーブルからエントリを削除する

- テーブルから 1 つのエントリを削除する場合は、–r オプションを指定します (296ページの「1 つのエントリを削除する」を参照)。
- テーブルから複数のエントリを削除する場合は、–R オプションを指定します (297ページの「複数のエントリを削除する」を参照)。

1 つのエントリを削除する

テーブルから 1 つのエントリを削除するには、次に示すように –r オプションを指定します。

```
nistbladm -r indexed-name
```

次に示すコマンドでは、depts テーブルから Manf-1 エントリを削除します。

```
rootmaster% nistbladm -r [Dept=Manf-1,Site=Emeryville,Name=hosteen], \
depts.doc.com.
```

指定する列の値の数は最小限に減らすことができます。NIS+ では、列の値の重複が見つかると、行は 1 つも削除されることなく、エラーが返されます。次のテーブルから Manf-1 エントリを削除するには、次に示すように Site 列の値を指定するだけで済みます。

```
rootmaster% nistbladm -r [Site=Emeryville],depts.doc.com.
```

しかし、R&D エントリも Sales エントリと同じ値を持っているため、Site 列の値 (SanFran) を指定するだけでは Sales エントリを削除できません。

Dept	Site	Name
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

複数のエントリを削除する

テーブルから複数のエントリを削除するには、次に示すように –R オプションを指定します。

```
nistbladm -R indexedname
```

オプションが –r であれば、最小限必要な列値を指定するだけで済みます。しかし、上記のコマンドでは –R が指定されているので、NIS+ が重複を見つけると、それに該当するすべてのエントリが削除されます。テーブル内の列の名前を確認したい場合は、niscat –o コマンドを実行します。次のコマンドを実行すると、Site 列の値が SanFran であるすべてのエントリが削除されます。

```
rootmaster% nistbladm -R [Site=SanFran],depts.doc.com.
```

Dept	Site	Name
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln

–R オプションを指定すると、次に示すように、列の値を指定しなければテーブル内のすべてのエントリを削除できます。

```
rootmaster% nistbladm -R [],depts.doc.com.
```

nistbladm -R コマンドと共に使用すると、一対の空の角カッコはすべてのテーブル列を指定するワイルドカードとして解釈されます。

niscat コマンド

niscat コマンドは NIS+ テーブルの内容を表示します。このコマンドを使って、テーブルのオブジェクト属性を表示できます。表示するテーブル、エントリ、または列に対する読み取り権が必要です。

構文

テーブルの内容を表示するには、以下のように入力します。

```
niscat [-hM] tablename
```

テーブルのオブジェクト属性を表示するには、以下のように入力します。

```
niscat -o tablename
niscat -o entry
```

表 14–5 niscat 構文のオプション

オプション	目的
–h	ヘッダー。テーブルエントリの上にあるヘッダー行を表示し、各列の名前を表示する
–M	マスター。マスターサーバーに収められているテーブルのエントリだけを表示する。これによって最新情報を取得できる。デバッグにだけ使用する
–o	オブジェクト。列名、属性、サーバーなどの、テーブルについてのオブジェクト情報を表示する

テーブルの内容を表示する

テーブルの内容を表示するには、niscat にテーブル名を付けて実行します。

```
niscat tablename
```

次の例では、depts というテーブルの内容を表示します。

```
rootmaster% niscat -h depts.doc.com.
#  Name:Site:Name
R&D:SanFran:kuznetsov
Sales:SanFran:jhill
Manf-1:Emeryville:hosteen
Manf-2:Sausalito:lincoln
```

注 - *NP* は、ユーザーにそのエントリを表示するためのアクセス権がないことを示します。アクセス権は、テーブル、列、エントリ (行) ごとに与えられます。アクセス権の詳細は、第 10 章を参照してください。

テーブルまたはエントリのオブジェクト属性を表示する方法

テーブルのオブジェクト属性を表示するには、niscat –o にテーブル名を付けて実行します。

```
niscat -o tablename.org_dir
```

テーブルエントリのオブジェクト属性を表示するには、niscat –o を使い、インデックス付きの名前でエントリを指定します。

```
entry ::=column=value    ... tablename | \
  [column=valu   ,...], tablename
```

次に、テーブルの例とテーブルエントリの例を示します。

「テーブル」

```
rootmaster# niscat -o hosts.org_dir.doc.com.
Object Name : hosts
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : org_dir.doc.com.
Access Rights : ----rmcdr---r---
Time to Live : 12:0:0
Object Type :
TABLE Table Type : hosts_tbl
Number of Columns : 4
Character Separator :
Search Path :
Columns :
 [0] Name : cname
 Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INS
 Access Rights: ----------------
 [1] Name : name
 Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INS
 Access Rights: ----------------
 [2] Name : addr
 Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INS
 Access Rights: ----------------
 [3] Name : comment
 Attributes : (TEXTUAL DATA)
 Access Rights: ----------------
```

「テーブルエントリ」

```
rootmaster# niscat -o [name=rootmaster],hosts.org_dir.doc.com.
Object Name : hosts
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : org_dir.doc.com.
Access Rights : ----rmcdr---r---
Time to Live : 12:0:0
Object Type : ENTRY
Entry data of type hosts_tbl
Entry has 4 columns.
  .
#
```

nismatch と nisgrep コマンド

nismatch コマンドと nisgrep コマンドは、NIS+ テーブルを検索して、それぞれ特定の文字列または正規表現に一致するエントリを探します。これらのコマンドは、エントリ自体、またはエントリの検索できた回数のどちらかを表示します。nismatch コマンドと nisgrep コマンドの相違を表 14–6 に示します。

表 14–6 nismatch と nisgrep の 比較

機能	nismatch	nisgrep
検索指定	テキストのみ指定可能	正規表現が指定可能
速度	高速	低速
検索対象	検索可能列のみ	すべての列 (検索可能かどうかとは無関係)
検索構文	*column=string ... tablename*[*column=string,...*], *tablename*	*column=exp ... tablename*

この節の例では、両方のコマンドの構文を説明します。

どちらのコマンドを使用する場合にも、検索するテーブルに対する読み取りアクセス権が必要です。

この節の例は、次に示す `depts.doc.com.` というテーブルの値をベースにしています。最初の 2 つの列だけが検索可能です。

Name (S)	Site (S)	Name
R&D	SanFran	kuznetsov
Sales	SanFran	jhill
Manf-1	Emeryville	hosteen
Manf-2	Sausalito	lincoln
Shipping-1	Emeryville	tsosulu
Shipping-2	Sausalito	katabami
Service	Sparks	franklin

正規表現について

正規表現により、テキストと記号を組み合わせて使用し、特定の構成の列の値を検索できます。たとえば、正規表現 `'Hello'` は、`Hello` で始まる値を検索します。正規表現をコマンド行で使用するときは、必ず引用符で囲んでください。その理由は、正規表現記号の多くが Bourne シェルと C シェルでは特殊な意味をもつからです。たとえば、次のように入力します。

```
rootmaster% nisgrep -h greeting='Hello' phrases.doc.com.
```

正規表現の記号を表 14–7 にまとめます。

表 14–7 正規表現記号

記号	意味
^*string*	*string* で始まる値を見つける
string $	*string* で終わる値を見つける
.	ピリオドの数と等しい数の文字をもつ値を見つける
[*chars*]	角かっこ内の文字のどれかを含む値を見つける

表 14–7 正規表現記号 続く

記号	意味
**expr*	*expr* についてゼロ回以上一致する値を見つける
+	1 回以上現われるものを見つける
?	任意の値を見つける
\'*s-char*'	? や $ などの特殊文字を見つける
x | y	x または y の値を見つける

構文

最初の列を検索するには、以下のように入力します。

```
nismatch string tablename
nisgrep  reg-exp tablename
```

特定の列を検索するには、以下のように入力します。

```
nismatch column=string tablename
nisgrep column= reg-exp tablename
```

複数の列を検索するには、以下のように入力します。

```
nismatch column=string tablename ...\
nismatch [column=string,...], tablename
nisgrep column= reg-exp ... \
  tablename
```

表 14–8　nismatch 構文と nisgrep 構文のオプション

オプション	目的
–c	カウント。エントリ自体ではなく、検索指定に一致したエントリのカウントを表示する
–h	ヘッダー。エントリの上のヘッダー行を表示し、各列名を表示する
–M	マスター。マスターサーバーに収められているテーブルのエントリだけを表示する。これによって、最新情報を取得できる。デバッグにだけ使うようにする

最初の列を検索する

テーブルの最初の列で特定の値を検索するには、最初の列の値と「*tablename*」を入力します。nismatch では、この値は文字列でなければなりません。nisgrep では、この値は正規表現でなければなりません。

```
nismatch [-h] string tablename
nisgrep [-h] reg-expression tablename
```

次の例では、depts テーブルを検索して、最初の列の値が R&D となっているすべてのエントリを探します。

```
rootmaster% nismatch -h 'R&D' depts.doc.com.
rootmaster% nisgrep -h 'R&D' depts.doc.com.
```

注 - & がシェルによってメタキャラクタとして解釈されないように、R&D が引用符で囲まれています。

特定の列を検索する

最初の列以外の特定の列を検索するには、次の構文を使います。

```
nismatch column=string tablename
nisgrep column=reg- expression tablename
```

次の例では、depts テーブルを検索して、第 2 列の値が SanFran となっているすべてのエントリを探します。

```
rootmaster% nismatch -h Site=SanFran depts.doc.com.
rootmaster% nisgrep -h Site=SanFran depts.doc.com.
```

複数の列を検索する

2 つ以上の列で一致するエントリを検索するには、次の構文を使います。

```
nismatch [-h] [column= string, ... \
 column= string,...], tablename
nisgrep [-h] column=reg-exp ... \
 tablename
```

以下の例では、第 2 列の値が SanFran で、第 3 列の値が jhill のエントリを検索します。

```
rootmaster% nismatch -h [Site=SanFran,Name=jhill], depts.doc.com.
rootmaster% nisgrep -h Site=SanFran Name=jhill depts.doc.com.
```

nisln コマンド

nisln コマンドは、NIS+ オブジェクトとテーブルエントリの間でシンボリックリンクを作成します。すべての NIS+ 管理コマンドで、NIS+ オブジェクト間のリンクをたどるように指示する –L フラグを使用できます。

注 - あるテーブルから他のテーブルへのリンクはできますが、あるテーブルのエントリから他のテーブルのエントリへのリンクはできません。

他のオブジェクトまたはエントリへのリンクを作成するには、ソースオブジェクトまたはエントリ、つまり他のオブジェクトまたはエントリを指すものに対する変更権が必要です。

注意 - cred テーブルをリンクしないでください。org_dir ディレクトリには、それぞれ専用の cred テーブルが必要です。org_dir cred テーブルもリンクしないでください。

構文

リンクを作成するには次のように入力します。

```
nisln source target
```

表 14–9 nisln 構文のオプション

オプション	目的
–L	リンクをたどる。ソース (*source*) 自体がリンクである場合、新しいリンクはこれにはリンクされず、そのリンク元のソースにリンクされる
–D	デフォルト。リンクされたオブジェクトに対して別のデフォルトセットを指定する。デフォルトについては、198ページの「デフォルトを 無効にする」を参照

リンクを作成する

オブジェクト間のリンク (テーブルとディレクトリの間のリンクなど) を作成するには、最初に「ソース (*source*)」、次に「リンク先 (*target*)」の順で、両方のオブジェクト名を指定します。

```
nisln source-object target-object
```

nissetup コマンド

nissetup コマンドは、org_dir ディレクトリと groups_dir ディレクトリ、それにフルセットの NIS+ テーブルを作成することによって、既存の NIS+ ディレクトリオブジェクトをドメインに展開します。しかし、データでテーブルを生成することはありません。これを行うには、308ページの「nisaddent コマンド」で説明する nisaddent コマンドが必要です。ドメインにディレクトリを展開することは、ドメインの設定作業の一部です。前提条件および必要な動作の詳細は、パート II「NIS+ の紹介と概要」を参照してください。

注 - 新しい NIS+ ドメインを設定するのであれば、nissetup コマンドを使うより、nisserver スクリプトを使った方が簡単です。nisserver スクリプトの具体的な使い方については、『*Solaris* ネーミングの設定と構成』を参照してください。

---

nissetup コマンドは、NIS クライアントをサポートするドメインにもディレクトリを展開できます。

nissetup を使用するには、テーブルを格納するディレクトリに対する変更権が必要です。

ディレクトリを NIS+ ドメインに展開する

nissetup コマンドは、ディレクトリ名を付けても付けなくても使えます。ディレクトリ名を付けない場合、このコマンドはデフォルトのディレクトリを使います。追加される各オブジェクトは、出力に表示されます。

```
rootmaster# /usr/lib/nis/nissetup doc.com.
org_dir.doc.com. created
groups_dir.doc.com. created
a uto_master.org_dir.doc.com. created
auto_home.org_dir.doc.com. created
bootparams.org_dir.doc.com. created
cred.org_dir.doc.com. created
ethers.org_dir.doc.com. created
group.org_dir.doc.com. created
hosts.org_dir.doc.com. created
mail_aliases.org_dir.doc.com. created
sendmailvars.org_dir.doc.com. created
netmasks.org_dir.doc.com. created
netgroup.org_dir.doc.com. created
networks.org_dir.doc.com. created
```

(続く)

```
passwd.org_dir.doc.com. created
protocols.org_dir.doc.com. created
rpc.org_dir.doc.com. created
services.org_dir.doc.com. created
timezone.org_dir.doc.com. created
```

ディレクトリを NIS 互換ドメインに展開する方法

NIS+ と NIS のクライアント要求をサポートするドメインにディレクトリを展開するには、–Y フラグを使います。作成するテーブルは、NIS のクライアント要求がアクセスできるように、未認証カテゴリに対する読み取り権が与えられます。

```
rootmaster# /usr/lib/nis/nissetup -Y Test.doc.com.
```

nisaddent コマンド

nisaddent コマンドは、テキストファイルまたは NIS マップからの情報を NIS+ テーブルにロードします。また、NIS+ テーブルの内容をテキストファイルに逆にダンプできます。NIS+ テーブルを初めて生成 (populate) する場合、『*Solaris* ネーミングの設定と構成』を参照してください。すべての前提条件と関連作業が説明してあります。

nisaddent を使用して、情報をある NIS+ テーブルから別のテーブルに (たとえば、別のドメインの同じ種類のテーブルに) 転送できますが、直接には転送できません。まず、テーブルの内容をファイルにダンプし、次にそのファイルを他のテーブルにロードする必要があります。ただし、ファイル内の情報は正しくフォーマットされていなければなりません。各テーブルに必要なフォーマットを付録 C で説明します。

情報をテーブルにロードするとき、置換 (replace)、追加 (append)、またはマージ (merge) の 3 つのオプションを自由に使用できます。追加オプションは、ソースエントリを NIS+ テーブルに単純に追加します。置換オプションの場合、NIS+ は、まずテーブル内のすべての既存エントリを削除し、次にソースからエントリを追加します。大規模なテーブルでは、これによって多くのエントリセット (1 セットは既存エントリの削除用、他のセットは新エントリの追加用) がテーブルの .log ファイ

ルに追加され、/var/nis 内の領域を占有し、複製への伝達に長時間を要することになります。

マージオプションでは、置換オプションと同じ結果が得られますが、異なるプロセスを使っており、複製に送信する動作数が大幅に減少します。マージオプションの場合、NIS+ は 3 種類のエントリを別に処理します。

- ソース内にだけ存在するエントリは、テーブルに「追加」される
- ソースとテーブルの両方に存在するエントリは、テーブル内で「更新」される
- NIS+ テーブルにだけ存在するエントリは、テーブルから「削除」される

大きなテーブルを更新する場合で、内容の変更があまりないときは、マージオプションを使用するとサーバーは多くの動作を節約できます。ソース内で重複していないエントリだけを削除する (置換オプションは全エントリを無差別に削除する) ため、重複エントリごとに 1 回の削除と 1 回の追加動作を省略できます。

初めてテーブルに情報をロードする場合、テーブルオブジェクトに対する作成権が必要です。テーブル内の既存情報を上書きする場合、テーブルに対する変更権が必要です。

構文

テキストファイルから情報をロードするには、以下のように入力します。

```
/usr/lib/nis/nisaddent -f filename table-type [domain]
/usr/lib/nis/nisaddent -f filename -t tablename table-type [domain]
```

NIS マップから情報をロードするには、以下のように入力します。

```
/usr/lib/nis/nisaddent -y NISdomain table-type [domain]
/usr/lib/nis/nisaddent -y NISdomain -t tablename table-type [domain]
/usr/lib/nis/nisaddent -Y map table-type [domain]
/usr/lib/nis/nisaddent -Y map -t tablename table-type [domain]
```

NIS+ テーブルから情報をファイルにダンプするには、以下のように入力します。

```
/usr/lib/nis/nisaddent -d [-t tablename tabletype ] > filename
```

ファイルから情報をロードする

ファイルの内容を NIS+ テーブルに転送するには、いくつかの方法があります。

■ –f を単独で指定すると、*table-type* の内容が *filename* の内容に置き換えられます。

```
nisaddent -f filename table-type
```

■ –f と –a を同時に指定すると、*filename* の内容が *table-type* に「追加」されます。

```
nisaddent -a -f filename table-type
```

■ –f と –m を同時に指定すると、*filename* の内容が *table-type* の内容に「マージ」されます。

```
nisaddent -m -f filename table-type
```

次に示す 2 つの例では、テキストファイル /etc/passwd.xfr の内容が NIS+ passwd テーブルにロードされます。最初の例ではローカルドメインに、2 番目の例では別のドメインのテーブルにロードされます。

```
rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow
rootmaster# /usr/lib/nis/nisaddent -f /etc/passwd.xfr passwd sales.doc.com.
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow sales.doc.com.
```

注 - /etc ディレクトリのファイルから NIS+ passwd テーブルを作成する場合は、nisaddent を 2 回 (/etc/passwd ファイルと /etc/shadow ファイルに対して 1 回ずつ) 実行する必要があります。

もう 1 つの方法では、stdin をソースとして使います。しかし、stdin では –m オプションを使えません。次に例を示します。リダイレクト (>) やパイプ (|) を使用することは可能です。ただし、別のドメインに出力が行われるような形でパイプを使用することはできません。

作業	コマンド
リダイレクト	cat *filename* > nisaddent *table-type*
リダイレクト処理後、追加する	cat *filename* > nisaddent -a *table-type*
リダイレクト処理後、別のドメインに追加する	cat *filename* > nisaddent -a *table-type NIS+ domain*

作業	コマンド
パイプ	`cat` *filename* | `nisaddent` *table-type*
パイプ処理後、追加する	`cat` *filename* | `nisaddent -a` *table-type*

NIS+ テーブルが、オートマウンタテーブルの 1 つであるか標準以外のテーブルである場合、`–t` オプションと NIS+ テーブルの完全な名前を追加します。

```
master# nisaddent -f /etc/auto_home.xfr \
-t auto_home.org_dir.doc.com.key-value
master# nisaddent -f /etc/auto_home.xfr \
-t auto_home.org_dir.doc.com. key-value sales.doc.com.
```

NIS マップからデータをロードする

NIS マップから情報を転送する方法は 2 つあり、NIS ドメインを指定するか、または実際の NIS マップを指定します。ドメインを指定した場合、NIS+ は、*table-type* に基づいて、`/var/yp/`*nisdomain* 内のどのマップファイルをソースとして使うかを判断します。`/var/yp/`*nisdomain* は「ローカル」ファイルでなければなりません。

NIS+ テーブル形式	**NIS** マップ名
`Hosts`	`hosts.byaddr`
`Nodes`	`ipnodes.byaddr`
`Passwd`	`passwd.byname`
`Group`	`group.byaddr`
`Ethers`	`ethers.byname`
`Netmasks`	`netmasks.byaddr`
`Networks`	`networks.byname`
`Protocols`	`protocols.byname`

NIS+ テーブル形式	NIS マップ名
RPC	rpc.bynumber
Services	services.byname

NIS ドメインの指定によって転送するには、–y (小文字) オプションを使い、NIS+ テーブル形式に加えて NIS ドメインを指定します。

「テーブルの置換」

```
nisaddent -y nisdomain table-type
```

「テーブルの追加」

```
nisaddent -a -y nisdomain table-type
```

「テーブルのマージ」

```
nisaddent -m -y nisdomain table-type
```

デフォルトでは、nisaddent は NIS+ テーブルの内容を NIS マップの内容に置き換えます。追加またはマージを行うには、–a または –m のオプションを使います。次の例では、対応する NIS マップ (passwd.byname) からの NIS+ passwd テーブルを old-doc ドメインにロードします。

```
rootmaster# /usr/lib/nis/nisaddent -y old-doc passwd
```

2 番目の例でも同じことを行いますが、ローカルドメイン doc.com. の代わりに、sales.doc.com. ドメインに対して行います。

```
rootmaster# /usr/lib/nis/nisaddent -y old-doc passwd sales.doc.com.
```

NIS+ テーブルが、オートマウンタテーブルの 1 つであるか非標準テーブルである場合、そのソースがファイルであるかのように、–t オプションと NIS テーブルの完全な名前を追加します。

```
rootmaster# nisaddent -y old-doc \
-t auto_home.org_dir.doc.com. key-value
rootmaster# nisaddent -y old-doc \
```

(続く)

```
-t auto_home.org_dir.doc.com.  key-value sales.doc.com.
```

マップファイルをドメイン用に使うのではなく、特定の NIS マップを指定したい場合、–Y (大文字) オプションを使い、マップ名を指定します。オプションを見つけやすいように、次の例では太字にしています。

```
rootmaster# nisaddent -Y hosts.byname hosts
rootmaster#  nisaddent -Y hosts.byname hosts sales.doc.com.
```

NIS マップがオートマウンタマップの 1 つであるか非標準マップである場合、–Y オプションと –t オプションを組み合わせます。

```
rootmaster# nisaddent -Y auto_home \
-t auto_home.org_dir.doc.com. key-value
rootmaster# nisaddent -Y auto_home \
-t auto_home.org_dir.doc.com.  key-value sales.doc.com.
```

NIS+ テーブルの内容をファイルにダンプする

NIS+ テーブルの内容をファイルにダンプするには、–d と –t のオプションを使います。–d オプションは、ダンプするようコマンドに指示します。–t オプションは、NIS+ テーブルを指定します。

```
rootmaster# nisaddent -Y auto_home\
-t auto_home.org_dir.doc.com. key-value
rootmaster# nisaddent -Y auto_home \
-t auto_home.org_dir.doc.com. key-value
 sales.doc.com.
```

第 15 章

サーバー使用のカスタマイズ

この章では、NIS+ クライアントが使用するサーバーのカスタマイズおよび制御方法について説明します。

NIS+ サーバーと NIS+ クライアント

クライアントマシン、ユーザー、アプリケーション、あるいはプロセスは、アクティブな NIS+ サーバー (マスターまたは複製) を検索して、そこから必要な情報を取り込みます。巨大なネットワーク、サブネットを多く持ったネットワーク、あるいは広域リンクにまたがったネットワークでは、サーバーの使用法をカスタマイズすることにより NIS+ の性能を強化できます。

デフォルトでのクライアントの検索動作

カスタマイズ前の状態で、`nisprefadm` コマンドによるサーバーの優先順位設定がなにも設定されていなければ、クライアントは、まず自分自身のローカルサブネット上の NIS+ サーバーから情報を入手しようとします。そのサブネットにアクティブなサーバーが見つかれば、応答のあった最初のローカルサーバーから必要な情報を入手します。ローカルサブネットにサーバーがない場合、次にクライアントは、ローカルサブネットの外部を検索し、応答のあった最初の遠隔サーバーから必要な NIS+ 情報を入手します。

ネットワークが大規模で、混雑している場合、上記のデフォルトの検索動作では NIS+ の性能を十分に発揮させることができないことがあります。それは次のような理由によります。

- サブネット上の複数のサーバーが多数のクライアントに情報の配布を行なっている場合、クライアントのデフォルト検索パターンがランダムなために、他にあまり使われていないサーバーがあるのに、いくつかのサーバーの負荷が高くなりすぎる可能性があります。
- ローカルサブネットの外で NIS+ サーバーを検索する際、クライアントは、オーバーワークしているサーバーであっても、低速な広域ネットワーク接続方式 (たとえば、モデム) や、すでにトラフィック多い専用回線によってリンクされているサーバーであっても、最初に応答したものから情報を入手します。

優先サーバーを指定する

Solaris 8 リリースには、新規機能 (サーバーの使用のカスタマイズ) が搭載されています。この機能を使用すると、クライアントが NIS+ サーバーを検索する順序をコ

ントロールできます。この新規機能では、次のような方法でサーバーの使用の度合いをバランスよくカスタマイズできます。

- クライアントが特定のサーバーを優先的に選択する (検索する) ように指定します。
- 使用できるローカルサーバーがない場合、クライアントが遠隔サーバーを使用してもよいかどうかを指定します。

指定した検索基準は、ドメイン内のすべてのクライアント、サブネット上のすべてのクライアント、またはマシンごとに独立したクライアントに適用できます。

注 - サーバー使用の優先順位を特定のマシンに設定すると、この優先順位は、そのマシンで実行中のユーザー、アプリケーション、処理、または他のクライアントすべてに適用されます。同じマシン上の別のクライアントに、異なるサーバー使用のパターンを設定できません。

広域ネットワーク (WAN) 上での NIS+

使用サーバーのカスタマイズは、多数のサブネットを持つ大規模ネットワークや、モデムまたは専用回線で接続された複数の地理的サイトにまたがるネットワークで使用すると特に効果があります。ネットワークの性能を最大にするには、サブネット間やより低速な接続によってリンクされたサイト間のトラフィックを最小限にする必要があります。これは、クライアントが使用できる NIS+ サーバーとその優先順位を指定することによって実現できます。このようにして、可能な限り NIS+ ネットワークトラフィックがローカルサブネットから出ないようにします。

サーバー使用の最適化 - 概要

この節では、サーバー使用のカスタマイズの概要について説明します。

nis_cachemgr が必要

使用サーバーをカスタマイズするには、クライアントで `nis_cachemgr` を実行している必要があります。クライアントマシンが `nis_cachemgr` を実行していない場

合は、使用サーバーのカスタマイズ機能は利用できません。クライアントマシン上で、nis_cachemgr を実行していない場合は、そのクライアントは、316ページの「デフォルトでのクライアントの検索動作」で説明した方法で識別された最初のサーバーを使用します。

グローバルテーブルまたはローカルファイル

nisprefadm コマンドは、次のように、その使用方法により、ローカルの client_info ファイルまたはドメインの client_info テーブルのどちらかを作成します。

- 「ファイル」

  nisprefadm を使用して、マシンの /var/nis ディレクトリに格納される、ローカルなマシン固有の client_info を作成できます。ローカルファイルは、サーバーの優先順位をそのマシンだけに指定するためのものです。マシンに、ローカルの /var/nis/client_info ファイルがある場合、そのマシンは、ドメインの client_info.org_dir テーブルに入っているサーバーの優先順位は無視します。ローカル client_info ファイルを作成するには、–L オプションを指定した nisprefadm を実行してください。

- 「テーブル」

  nisprefadm を使用して、各ドメインの NIS+ ディレクトリオブジェクト org_dir 内に格納される NIS+ client_info テーブルを作成できます。このテーブルは、次のものに対してサーバーの優先順位を指定できます。

  - 個別のマシン

    マシンにローカルの /var/nis/client_info ファイルがある場合、ドメインの client_info テーブルによって、そのマシンに対して指定された優先順位はすべて無視されます。

  - 1 つの特定のサブネット上にあるすべてのマシン

    サブネット上のマシンにローカルの /var/nis/client_info ファイルがあるか、またはテーブル内にそのマシン固有の優先順位が設定されている場合、そのマシンは、サブネットに指定された優先順位を無視します。

  1 つのサブネット上のすべてのマシンに適用されるグローバル client_info テーブルを作成するには、–G および –C オプションを指定して nisprefadm を実行してください。これについては、327ページの「グローバルサーバー優先順位を指定する」で説明しています。

マシンに、以下で説明するような、固有のローカル client_info ファイルがある場合、そのマシンに対してグローバル client_info テーブル内に設定されたサーバーの優先順位はすべて無視されるので注意してください。マシンに、ローカル client_info ファイルがあるか、またはグローバル client_info テーブルにそのマシン固有のエントリがある場合は、そのサブネットに設定された優先順位は無視されます。

注意 - client_info ファイルと client_info テーブルを変更する場合は、nistbladm コマンドだけを使用してください。nistbladm などの、他の NIS+ コマンドは絶対に使用しないでください。

client_info テーブルまたは client_info ファイルを処理する場合は、–G または –L オプションを使用して、グローバルテーブル (–G) またはコマンドを実行中のマシンのローカルファイル (–L) のどちらにそのコマンドを適用させるかを指定する必要があります。

優先順位の番号の指定

サーバーの優先順位は、各サーバーに「優先順位を表す番号」を指定することによって制御されます。クライアントは、数値で指定された優先順位にしたがって NIS+ サーバーを検索します。検索の順序は、まず、優先順位格付け番号が小さいサーバーを先に検索し、次に大きい数値の付いたサーバーを検索します。

つまり、クライアントは、まず初めにゼロの優先順位の付いた NIS+ サーバーから名前空間情報を入手しようとします。使用できる優先順位=0 のサーバーがない場合、クライアントは、次に優先順位=1 のサーバーに問い合わせを行います。1 のサーバーが使用できない場合は、2 のサーバーを検出しようと、次に 3 というように、この検索は必要な情報が入手できるか、または検索するサーバーがなくなるまで続きます。

優先順位格付け番号は、nisprefadm コマンドを使用してサーバーに割り当てます。これについては、327ページの「グローバルサーバー優先順位を指定する」で説明しています。

サーバーの優先順位番号は、client_info テーブルと client_info ファイル内に格納されます。マシンに、固有の /var/nis/client_info ファイルがある場合は、このファイル内に格納された優先順位番号が使用されます。マシンに、固有の client_info ファイルがない場合は、ドメインの client_info.org_dir テーブル内に格納された優先順位番号が使用されます。このような client_info テーブ

ルと client_info ファイルを、「優先サーバーのリスト」、または単に「サーバーリスト」と呼びます。

各クライアントのサーバーの優先順位を制御することで、サーバーの使用法をカスタマイズします。たとえば、ドメインに mailer という名前のクライアントマシンがあり、このマシンは名前空間情報を頻繁に利用していて、さらにこのドメインには、マスターサーバー (nismaster) と複製サーバー (replica1) の両方があるとします。ここで、mailer マシン用に nismaster に優先順位番号 1 を割り当て、replica1 に優先順位番号 0 を割り当てると、mailer マシンは、名前空間情報を、必ず最初に replica1 から入手しようとし、その後で nismaster に移ります。次に、このサブネット上にある他のすべてのマシンに対して、優先順位番号を nismaster サーバーにはゼロを、replica1 には 1 を割り当てます。この結果、他のマシンは、必ず最初に nismaster に問い合わせを行います。

同じ優先順位番号を、ドメイン内の複数のサーバーに割り当てることができます。たとえば、nismaster1 と replica2 の両方に優先順位番号 0 を割り当て、replica3、replica4、replica5 に優先順位番号 1 を割り当てることができます。

サーバーの優先順位のデフォルト

client_info ファイルまたはテーブルがない場合は、キャッシュ管理プログラムが自動的に、ローカルサブネット上のすべてのサーバーにデフォルトの優先順位番号ゼロ (0) を割り当て、ローカルサブネットの外部のすべてのサーバーに無限大の優先順位を割り当てます。nisprefadm は、このデフォルトの優先順位番号を自由に変更するためのものです。

効率とサーバーの優先順位番号

クライアントは、すべてのサーバーを指定された優先順位番号を使用して昇順に検索しなければなりません。クライアントが指定された優先順位番号をもつすべてのサーバーを検索するには、5 秒以上必要です。つまり、ドメイン内に 1 つのマスターサーバーと 4 つの複製サーバーがあり、それぞれのサーバーに 0 から 4 の異なる優先順位番号を指定した場合、クライアントがこれらの優先レベルをすべて実行するには、25 秒以上かかるということです。

性能を最大にするために、サーバーの優先レベルを 2 つまたは 3 つ以上使用しないでください。たとえば、上記のような場合は、5 つのサーバーのうちの 1 つに優先順位= 0 を割り当て、他すべてのサーバーには優先順位 1 を割り当てるか、または

5 つのサーバーのうち 2 つに優先順位 1 を、残りの 3 つのサーバーに 2 を割り当てるといった方法をとってください。

優先サーバー限定と全サーバー

サーバーリストには、クライアントが優先サーバーを検出できなかった場合の処置も指定できます。「優先サーバー」とは、優先順位にゼロが指定されたサーバー、または nisprefadm を使用して優先順位番号を割り当てたサーバーです。

デフォルトでは、クライアントは優先サーバーに到達できないと、検索モードを使用して、ネットワーク上のあらゆる場所を検索し、検出できるあらゆるサーバーを検索します。これについては、316ページの「デフォルトでのクライアントの検索動作」で説明しています。このデフォルト機能を、nisprefadm –o オプションを使用して変更し、クライアントが優先サーバーだけを使用し、使用できるサーバーがない場合でも、優先サーバー以外のサーバーには問い合わせしないように指定できます。詳細は、333ページの「優先サーバー限定指定」を参照してください。

注 - マシンのドメインに優先サーバーが全くない場合、このオプションは無視されます。

優先順位の表示

現在特定のクライアントマシンに対して有効なサーバーの優先順位を表示するには、–l オプションを指定した nisprefadm を実行してください。これについては、325ページの「現行のサーバー優先順位の表示」で説明しています。

サーバー名とクライアント名

サーバーまたはクライアントマシンを指定する場合、次の点に注意してください。

- サーバー名とクライアント名は、同じ NIS+ ドメイン内にあり、オブジェクトを個別に特定できれば、完全指定名である必要はありません。マシン名を単独で使用できます。
- サーバーまたはサブネットが別の NIS+ ドメインにある場合は、そのマシンを個別に特定できるだけのドメイン名を含める必要があります。たとえば、sales.doc.com ドメイン内にいて、manf.doc.com ドメイン内の

nismaster2 マシンを指定する必要がある場合、nismaster2.manf と入力します。

サーバーの優先順位

以下のマシンにサーバーの優先順位を指定する方法は、それぞれ次のとおりです。

- 「個別のクライアントマシン」にサーバーの優先順位を指定するには、–L オプションを使用すると、nisprefadm を実行中のマシンにローカル client_info ファイルを作成します。–G –C マシンオプションを使用すると、グローバル client_info テーブル内にマシン固有の優先順位を作成します。
- 「サブネット上のすべてのマシン」にサーバーの優先順位を指定するには、–G –C サブネット番号オプションを使用してください。
- マシン固有のまたはサブネット固有の優先順位を持たない、現在のドメイン内にあるすべてのマシンにサーバーの優先順位を指定するには、–G オプションを使用してください。

サーバーの優先順位が有効になるタイミング

マシンまたはサブネットのサーバー優先順位の変更内容は、通常、指定したマシンがその nis_cachemgr データを更新するまでは有効になりません。マシンの nis_cachemgr がそのサーバー使用情報を更新するタイミングは、マシンがサーバーの優先順位をグローバル client_info テーブルまたはローカル /var/nis/client_info ファイルのどちらから入手するかによって決まります(318ページの「グローバルテーブルまたはローカルファイル」を参照してください)。

- 「グローバルテーブル」から入手する場合

  サーバーの優先順位をグローバルテーブルから入手するマシンのキャッシュ管理プログラムは、マシンがブートされた時、または client_info テーブルの存続時間 (TTL) の値が満了した時に、マシン用のサーバーの優先順位を更新します。デフォルトでは、この TTL 値は 12 時間ですが、変更可能です。277ページの「オブジェクトの生存期間を変更する」を参照してください。

- 「ローカルファイル」

  ローカルファイルからサーバーの優先順位を入手するマシンのキャッシュ管理プログラムは、サーバーの優先順位を 12 時間ごとに更新するか、または nisprefadm を実行してサーバーの優先順位を変更した時に必ず更新します。マ

シンをリブートしても、キャッシュ管理プログラムのサーバーの優先順位情報は更新されません。

ただし、nisprefadm に –F オプションを指定して実行すると、サーバーの優先順位の変更内容を強制的にただちに有効にできます。–F オプションを使用すると、nis_cachemgr で、ただちにその情報が更新されます。詳細は、338ページの「優先順位の変更内容をただちに実現する方法」を参照してください。

nisprefadm コマンドの使用方法

ここからの節では、nisprefadm コマンドを使用して、サーバーの優先順位を設定、変更、削除する方法について説明します。

nisprefadm コマンドは、クライアントが優先的に選択するサーバーを指定するために使用します。

nisprefadm コマンドの構文は次のとおりです。

```
nisprefadm -a|-m|-r|-u|-x|-l -L|-G [-o type] \
  [-d domain] \
  [-C  machine] \
 servers
nisprefadm -F
```

表 15–1 nisprefadm コマンドのオプション

オプション	説明
–G	ドメインの org_dir ディレクトリ内に格納されるグローバル client_info テーブルを作成する。つまり、グローバルな優先されるサーバーリストを作成する。このオプションは、指定したサブネット上のすべてのマシンに対して優先順位を指定する場合は –C *subnet*、個別のマシンに優先順位を指定する場合は –C *machine* のどちらかと同時に使用しなければならない
–L	ローカルマシンの /var/nis ディレクトリに格納されるローカル client_info ファイルを作成する。つまり、コマンドを実行中のマシンにだけ適用される優先サーバーのリストを作成する

表 15–1 nisprefadm コマンドのオプション 続く

オプション	説明
–o *type*	オプションを指定する。有効なオプションには、クライアントが接続できる優先サーバーがない場合、優先サーバー以外のサーバーを使用できるように指定する pref_type=all と、指定した優先サーバーだけをクライアントが使用するように指定する pref_type=pref_only がある
–d *domain*	指定したドメインまたはサブドメインに、グローバルな優先サーバーの client_info テーブルを作成する
–C *subnet*	優先順位を適用するサブネットの番号
–C *machine*	クライアントマシン名
servers	1 つまたは複数の NIS+ サーバー。ここで指定されたサーバーは優先的に選択される
–a	サーバーリストに指定サーバーを追加する
–m	サーバーリストを変更する。たとえば、–m オプションを使用すると、1 つまたは複数のサーバーに指定された優先順位番号を変更できる
–r	サーバーリストから指定したサーバーを削除する
–u	サーバーリストを消去してから、指定したサーバーを追加する (つまり、現行のサーバーリストを優先サーバーの新規リストに置換する)
–x	サーバーリストを完全に削除する
–l	現行の優先サーバー情報を一覧表示 (表示) する
–F	優先サーバーのリストを強制的にただちに変更する

注 - –C *machine* オプションは、–L (ローカル) フラグとともに使用しても無効となるので、使用しないでください。たとえば、nisprefadm を altair マシン上で実行しているとします。ここで、–L フラグを使用して、指定した優先順位が altair のローカル client_info ファイルに書き込まれるように指定し、さらに、–C vega オプションを使用して、作成した優先順位が vega マシンに適用されるように指定します。すると、nisprefadm コマンドは、vega 用の優先順位を altair のファイルに書き込みますが、vega は、サーバーの優先順位を必ず自分のローカル client_info ファイルまたはドメインのグローバル client_info テーブルから入手するため、これらの情報を参照することはありません。そのため、–C オプションは、nisprefadm に –G (グローバル) フラグを指定して実行する場合にだけ意味をもちます。

現行のサーバー優先順位の表示

現行のサーバー優先順位を表示するには、nisprefadm に -l オプションを指定して実行してください。

マシンに指定された優先順位の表示方法

♦ そのマシン上で、nisprefadm に –L と –l を指定して実行します。

```
sirius# nisprefadm -L -l
```

このようにすると、マシンのローカル /var/nis/client_info ファイル内に定義されたサーバーの優先順位がすべて表示されます。ローカルファイルがない場合は、情報は表示されず、シェルプロンプトに戻ります。

1 台のマシンに設定されたグローバルな優先順位の表示方法

♦ nisprefadm に、–l、–G、–C ***machinename*** オプションを指定して実行します。

```
sirius# nisprefadm -G -l -C  machinename
```

machinename には、マシンの IP アドレス (番号) が入ります。

このようにすると、このマシン用にドメインのグローバル client_info テーブル内に設定された優先順位が表示されます。

サブネットに設定されたグローバル優先順位の表示方法

- nisprefadm に、–l、–G、–C ***subnet*** オプションを指定して実行します。

```
sirius# nisprefadm -G -l -C  subnet
```

subnet には、サブネットの IP アドレス (番号) が入ります。

このようにすると、このサブネット用にドメインのグローバル client_info テーブル内に設定された優先順位が表示されます。

優先順位格付け番号の指定方法

デフォルトでは、–a オプションの後ろにリストしたサーバーには、すべて優先順位番号ゼロが指定されます。別の優先順位番号を指定する場合は、次のように、サーバー名の直後に指定する番号をカッコで囲んで入れてください (–a *name*(*n*)) 。*name* にはサーバー名が、*n* には優先順位番号が入ります。

たとえば、replica2 サーバーに優先順位番号 3 を割り当てる場合は、次のように入力します。

```
# nisprefadm -G -a replica2(3)
```

注 - シェルの中には、次のように要素を 2 重引用符で囲まなければならないものがあります。"name(n)"

サーバーの優先順位格付け番号の基礎知識については、319ページの「優先順位の番号の指定」を参照してください。

グローバルサーバー優先順位を指定する

ローカルまたは遠隔ドメインに、グローバルサーバー優先順位を設定できます。優先順位は、個別のマシンと 1 つのサブネット上のすべてのマシンに設定できます。

この節では、NIS+ ドメインのマスターサーバーに存在するグローバル client_info テーブル内にサーバーの優先順位を設定する方法を説明します。マスターサーバー上にテーブルがある場合、NIS+ は、そのテーブルをドメインの既存の複製サーバー上に複写します。

- 個別のマシンにローカル client_info ファイルを作成する方法については、329 ページの「ローカルサーバー優先順位を指定する」 を参照してください。
- グローバル client_info テーブルとローカル client_info ファイルの違いについては、318ページの「グローバルテーブルまたはローカルファイル」を参照してください。

サーバーの優先順位番号を割り当てるには、次のどちらかを指定した nisprefadm を実行してください。

- 新規または別の優先サーバーを追加する場合は、–a オプションを指定します。
- 既存のサーバー優先順位を削除して、新規の優先順位を作成する場合は、–u オプションを指定します。

サブネットにグローバル優先順位を設定する方法

1 つのサブネット上のすべてのマシンのグローバルテーブルにサーバーの優先順位を割り当てるには、次のようにします。

◆ **nisprefadm に –G および –C *subnet* オプションを指定して実行します。**

```
# nisprefadm -G -a -C  subnet  servers  (preferences)
```

- –C *subnet* には、優先順位を適用するサブネットの IP 番号を指定します。使用するシェルによっては、マシンを引用符で囲む必要があります。
- *servers (preferences)* には、1 つまたは複数のサーバーが入ります。それらには優先順位格付け番号を指定できます。

たとえば、サブネット 123.123.123.123 が、nismaster および replica3 サーバーにはデフォルトの優先順位番号ゼロを付け、manf.replica6 サーバーには優先順位番号 1 を付けて使用するように指定する場合は、次のように入力してください。

```
polaris# nisprefadm -a -G -C 123.123.123.123 nismaster1 \
replica3 "manf.replica6(1)"
```

個別のマシンにグローバル優先順位を設定する方法

◆ –G と –C マシンオプションを指定した nisprefadm を実行します。

```
# nisprefadm -G -a -C  machine servers (preferences)
```

- –C マシンには、優先順位を適用するマシンを指定します。使用するシェルによっては、*machine* を引用符で囲む必要があります。
- *servers (preferences)* には、1 つまたは複数のサーバーが入ります。それらには優先順位格付け番号を指定できます。

  たとえば、マシン cygnus に指定された現行の優先順位を、replica7 と replica9 の両方にデフォルトの優先順位番号ゼロを指定したものに置き換える場合は、次のように入力します。

```
polaris# nisprefadm -u -G -C cygnus replica7 replica9
```

遠隔ドメインにグローバル優先順位を設定する方法

遠隔ドメイン内の個別のマシン、または遠隔ドメイン内の 1 つのサブネット上にあるすべてのマシンにサーバーの優先順位を割り当てるには、次のようにします。

◆ nisprefadm に –C、–G、–d オプションを指定して実行します。

```
# nisprefadm -a -G -C  name \
  -d  domain servers (preferences)
```

- *name* には、サブネットの IP 番号またはマシン名が入ります。このコマンドを使用して行なった変更は、ここに入力したサブネットまたはマシンに適用されます。
- *domainname* には、遠隔ドメイン名が入ります。
- *servers (preferences)* には、1 つまたは複数のサーバーが入ります。それらには優先順位格付け番号を指定できます。

たとえば、デフォルトの優先順位番号ゼロを指定した nismaster2 サーバーを、遠隔ドメイン sales.doc.com 内のサブネット 111.11.111.11 の優先サーバーのリストに追加する場合は、次のように入力します。

```
polaris# nisprefadm -a -G -C 111.11.111.11 -d sales.doc.com. nismaster2
```

ローカルサーバー優先順位を指定する

ここでは、ローカル client_info ファイルの作成および変更方法について説明します。ローカル client_info ファイルは、このファイルが存在するマシンにサーバーの優先順位を指定するためのものです。

マシンに /var/nis/client_info ファイルがある場合、このマシンは、NIS+ サーバー上のグローバル client_info テーブルではなく、そのマシンのローカルファイルからサーバーの優先順位を入手します。つまり、ローカルファイルはグローバルテーブルよりも優先されます。

- NIS+ サーバーのグローバル client_info テーブルを作成する方法については、327ページの「グローバルサーバー優先順位を指定する」を参照してください。
- グローバル client_info テーブルとローカル client_info ファイルの相違点については、318ページの「グローバルテーブルまたはローカルファイル」を参照してください。

サーバーの優先順位を割り当てるには、次のどちらかを指定した nisprefadm を実行してください。

- 新規または別の優先サーバーを追加する場合は、–a オプションを指定します。
- 既存のサーバー優先順位を削除して、新規の優先順位を作成する場合は、–u オプションを指定します。

ローカルマシン上に優先順位を設定する方法

nisprefadm コマンドを実行中のローカルマシンにサーバーの優先順位を割り当てるには、次のようにします。

- nisprefadm に –L オプションと –a または –u オプションを指定して実行します。

```
#nisprefadm -a -L servers （preferences）
```

ここで、*servers (preferences)* には、1 つまたは複数のサーバーが入ります。それらには優先順位格付け番号を指定できます。

たとえば、deneb マシンが NIS+ 情報を、まず初めにデフォルトの優先順位番号ゼロが指定された replica3 から検索し、次に manf.doc.com ドメイン内の (優先順位番号 1 が指定された) サーバー replica6 を検索するように指定する場合は、次のように入力します。

```
deneb# nisprefadm -a -L replica3 replica6.manf(1)
```

サーバーの優先順位を変更する

サーバーの優先順位番号は、変更することができ、また別のサーバーに優先順位番号の割り当てを移すこともできます

優先サーバーまたはサーバーに割り当てた優先順位番号を変更するには、nisprefadm に –m *oldserver–=newserver*（*n*）オプションを指定して実行してください。

サーバーの優先順位番号を変更する方法

- nisprefadm に –m ***server=server(new)*** オプションを指定して実行します。

```
#nisprefadm -L|-G -C name -m  oldserver=     newserver（n）
```

- L|G には、ローカルファイルとグローバルテーブルのどちらを修正するかを指定します。
- –C *name* には、サブネットの IP 番号またはマシン名が入ります。このオプションは、–G オプションを使用している場合にだけ使用します。このコマンドを使用して行なった変更は、ここに指定したサブネットまたはマシンに適用されます。
- –m は、サーバーリストを変更するオプションです。
- *old server* には、その優先順位番号を変更したいサーバー名が入ります。
- *new server(n)* には、サーバー名とその新しい優先順位番号が入ります。

たとえば、deneb マシン上で、deneb のローカル client_info ファイルの replica6.manf サーバーに指定された番号を 2 に変更する場合は、次にように入力します。

```
deneb# nisprefadm -L -m replica6.manf=replica6.manf(2)
```

優先順位リスト内で 1 つのサーバーを別のサーバーに置換する方法

優先順位リスト内で、1 つのサーバーを別のサーバーに変更するには、次のようにします。

◆ nisprefadm に –m ***oldserver=newserver*** オプションを指定して実行します。

```
# nisprefadm -L|-G -C   name  -m  oldserver=newserver  (  prefnumber)
```

- -L|-G には、ローカルまたはドメイン全体のどちらのサーバーリストを変更するのかを指定します。
- –C *name* には、サブネットの IP 番号またはマシン名が入ります。このオプションは、–G オプションを使用している場合にだけ使用します。このコマンドを使用して行なった変更は、ここに指定したサブネットまたはマシンに適用されます。
- –m は、サーバーリストを変更するオプションです。
- *oldserver* には、置換する対象となる旧サーバーが入ります。
- *newserver (prefnumber)* には、優先サーバーリスト内の旧サーバーの位置に入る新規サーバー (優先順位格付け番号を指定できる) を入力します。

–G オプションを使用して、グローバル client_info テーブル内のサーバーを置換する場合には、その置換内容は –C オプションで特定したサブネットまたはマシン

にだけ適用されるので注意してください。それ以外にリストされている置換対象 (旧) サーバーは影響を受けません。

たとえば、3 つのサブネットを持つドメインを所有していて、この中の 2 つのサブネット用に replica1 サーバーが優先サーバーのリストに入っている場合を考えます。これ以上 replica1 を使用しないので、使用サーバーから除外する場合は、nisprefadm –m を実行して、最初のサブネットの replica1 を新規サーバーに置換します。この時、2 番目のサブネットに対しても同様の処理を行うまでは、replica1 はそのサブネットの優先サーバーのリストに入っています。個別のマシンの優先サーバーでも同様です。

たとえば、ドメインのグローバル client_info テーブル内で、サブネット 123.12.123.12 用に指定された replica3 サーバーを replica6 サーバーに置換し、replica6 に優先順位番号 1 を割り当てる場合は、次のように入力します。

```
nismaster# nisprefadm -G -C 123.12.123.12 -m replica3 replica6(1)
```

優先順位リストからサーバーを削除する方法

優先順位リストから 1 つまたは複数のサーバーを削除するには、次のようにします。

♦ nisprefadm に –r オプションを指定して実行します。

```
# nisprefadm -L|-G -C  name -r  servers
```

- -L|-G には、ローカルまたはドメイン全体のサーバーリストのどちらを変更するのかを指定します。
- –C *name* には、サブネットの IP 番号またはマシン名が入ります。このオプションは、–G オプションを使用している場合にだけ使用します。このコマンドを使用した優先サーバーの削除は、ここに名前を指定したサブネットまたはマシンに適用されます。
- –r は、*servers* に名前を指定したサーバーをリストから削除するオプションです。

たとえば、ドメインのグローバル client_info テーブル内から、マシン polaris に指定された replica3 および replica6.manf サーバーを削除する場合は、次のように入力します。

```
polaris# nisprefadm -G -C polaris -r replica3 replica6.manf
```

優先サーバーリスト全体を置換する方法

グローバル client_info テーブルまたはマシンのローカル client_info ファイルどちらかの中にある、サブネットまたはマシンに指定された優先サーバーのリスト全体を置換するには、nisprefadm に –u オプションを指定して実行してください。

–u オプションは、マシンまたはサブネットに指定されたサーバーの優先順位を、まず初めにすべて削除してから指定した新規の優先順位を追加しますが、これ以外は、–a オプションと同じ処理を行います。既存の優先順位がある場合、–a オプションでは旧リストに新しい優先順位を追加します。

–u オプションの使用例については、328ページの「個別のマシンにグローバル優先順位を設定する方法」を参照してください。

優先サーバー限定指定

優先サーバーが使用できない場合に、クライアントがとる動作を指定できます。

デフォルトでは、クライアントは、優先サーバーに到達できない場合、検出できたサーバーであればどれでも使用してしまいます。優先サーバー限定オプションを設定すると、クライアントが優先サーバーだけを使用するように指定できます。優先サーバー限定オプションと全サーバーオプションの基礎知識については、優先サーバー限定とすべてのサーバーを参照してください。

優先サーバーが使用できない場合のクライアントの動作を指定するには、nisprefadm に –o *value* オプションを指定して実行してください。

優先サーバーだけを指定する方法

サーバーリストを使用するクライアントが、リスト内に記述されたサーバーだけから NIS+ 情報を入手するように指定するには、次のようにします。

- nisprefadm に –o pref_only オプションを指定して実行してください。

```
# nisprefadm -L|-G -o pref_only
```

- –L|–G には、ローカルまたはドメイン全体のサーバーリストのどちらを変更するかを指定します。
- –o pref_only により、クライアントがリスト内に記述されたサーバーだけから NIS+ 情報を入手するように指定されます。

注 - このオプションは、優先サーバーが全くないドメインでは無視されます。

たとえば、altair のローカル client_info ファイル内に、altair が必ず優先サーバーを使用し、altair の優先サーバーリストにないサーバーは使用しないように指定するには、次のように入力します。

```
altair# nisprefadm -L -o pref_only
```

優先サーバー以外のサーバーを使用する方法

サーバーリストを使用するクライアントが、優先サーバーが使用できない場合に、リストに記載されていないサーバーから NIS+ 情報を入手するように指定するには、次のようにします。

♦ nisprefadm に –o all オプションを指定して実行してください。

```
# nisprefadm -L|-G -o all
```

- –L|-G には、ローカルまたはドメイン全体のサーバーリストのどちらを変更するかを指定します。
- –o all は、クライアントが、優先サーバーが使用できない場合に、リストに記載されていないサーバーから NIS+ 情報を入手するように指定するオプションです。

注 - これは、デフォルトの動作です。そのため、–o all オプションは、以前に –o pref_only オプションを使用して優先サーバーだけを指定していた場合に限り使用する必要があります。

たとえば、altair が優先サーバーを使用できない場合には、altair のローカル client_info ファイル内に、優先サーバー以外のサーバーを使用できるように指定し直すには、次のように入力してください。

```
altair# nisprefadm -L -o all
```

サーバーの優先順位の使用の終了

使用サーバーのカスタマイズ機能の使用を終了し、316ページの「デフォルトでのクライアントの検索動作」で説明した NIS+ 情報の入手方法に戻すことができます。

サーバーの優先順位の使用を終了するには、nisprefadm に –x オプションを指定して実行してください。

注 - サーバーの優先順位の使用を終了する場合、クライアントは、322ページの「サーバーの優先順位が有効になるタイミング」で説明していることがらが通常どおり進行するまではサーバーの優先順位の使用を終了しません。また、サーバーの優先順位の使用を強制的にただちに終了させることもできます。これについては、337ページの「サーバーの優先順位の有効化」で説明しています。

グローバルサーバー優先順位を削除する方法

♦ nisprefadm に –G および –x オプションを指定して実行してください。

```
# nisprefadm -G -x
```

これにより、グローバルサーバーの優先順位が削除されます。

- ローカルサーバーの優先順位が指定されていないクライアントマシンは、316ページの「デフォルトでのクライアントの検索動作」で説明したように NIS+ 情報を入手します。
- ローカル /var/nis/client_info ファイルで設定されたサーバーの優先順位があるクライアントマシンは、引き続きそのファイルに指定されたサーバーを使用します。

ローカルサーバー優先順位を削除する方法

ローカル優先順位を終了するということは、次の異なる 3 つの事項のいずれか 1 つを意味すると考えられます。

- 特定のマシンで、サーバーの優先順位にローカル client_info ファイルの使用を終了し、かわってドメインのグローバル client_info テーブル内にそのサブネット用に設定された優先順位を使用したい場合
- このマシンで、サーバーの優先順位にローカル client_info ファイルの使用を終了し、かわってドメインのグローバル client_info テーブル内にそのマシン固有に設定された優先順位を使用したい場合
- 特定のマシンで、サーバーの優先順位をまったく使用したくない場合。マシンがサーバーの優先順位を使用しない場合、そのマシンは、316ページの「デフォルトでのクライアントの検索動作」に説明した方法で NIS+ 情報を入手する

ローカルからグローバルサブネットの優先順位に置換する方法

◆ マシンの /var/nis/client_info ファイルを削除します。

```
# rm /var/nis/client_info
```

この結果、マシンはドメインのグローバル client_info テーブル内でマシンのサブネットに指定された優先順位を使用するようになります。

ローカルからマシン固有のグローバル優先順位に置換する方法

1. マシンの /var/nis/client_info ファイルを削除します。

```
# rm /var/nis/client_info
```

2. –G と –C オプションを使用して、グローバルテーブル内にそのマシン用の優先順位を指定します。

   詳細は、328ページの「個別のマシンにグローバル優先順位を設定する方法」を参照してください。

マシンにサーバーの優先順位の使用を中止させる方法

1. マシンの /var/nis/client_info ファイルを削除します。

```
# rm /var/nis/client_info
```

マシンのドメインにグローバル client_info テーブルがない場合、必要な処理はこの手順だけです。ドメインに client_info テーブルがある場合は、337 ページの手順 2 に進んでください。

2. 空の /var/nis/client_info ファイルを作成します。

```
# touch /var/nis/client_info
```

マシンに固有の /var/nis/client_info ファイルがある場合、そのマシンは client_info テーブルのグローバル優先順位は使用しません。マシンに空の /var/nis/client_info ファイルがある場合、そのマシンはどの優先順位も使用せず、316ページの「デフォルトでのクライアントの検索動作」で説明した方法で NIS+ 情報を入手します。

サーバーの優先順位の有効化

使用サーバーの変更内容は、通常、クライアントマシンをリブートするか、またはマシンのキャッシュ管理プログラムを更新した時に有効になります。

ローカル client_infoファイル (–L オプション) を使用するローカルマシン上で、nisprefadm を使用してサーバーの優先順位の設定または変更した場合は、その変更内容はただちに有効になります。

グローバル client_info テーブル (–G オプション) からサーバーの優先順位を入手しているマシンの場合、nisprefadm に –F オプションを指定して実行すると、サーバーの優先順位に加えた変更を強制的にただちに有効にできます。

```
# nisprefadm -F
```

–F オプションとは、マシンのキャッシュ管理プログラムに、そのサーバーの優先順位情報をドメインのグローバル client_info テーブルを使って強制的にただちに更新させるオプションです。nisprefadm –F を実行するマシンが、/var/nis 内にそのマシン固有のローカル client_info ファイルを持つ場合、そのマシン上で nisprefadm –F を実行しても無効になります。

注 - –F オプションは、他の nisprefadm オプションと同時に使用できません。nisprefadm –F コマンドは、必ず、このコマンドを適用するマシン上で、単独で使用してください。–G オプションを使用して、ドメイン内にあるすべてのマシンのキャッシュ管理プログラムを更新できません。nisprefadm –F コマンドは、各マシン上でそれぞれ実行する必要があります。

優先順位の変更内容をただちに実現する方法

新しく作成、または変更したサーバーリストを、指定したマシン上で強制的にただちに有効にするためには、次のようにします。

- マシン上で、nisprefadm に –F オプションを指定して実行します。

```
# nisprefadm -F
```

たとえば、vega の優先サーバーリストへの変更内容を、強制的にただちに実現させるには (ローカルまたはグローバルのどちらの場合でも)、次のように入力してください。

```
vega# nisprefadm -F
```

第 16 章

NIS+ のバックアップと復元

この章では、NIS+ 名前空間のバックアップ方法と復元方法について説明します。

NIS+ のバックアップ機能と復元機能を使用すると、NIS+ 名前空間の保存と復元を素早く簡単に行うことができます。また、これらの機能を使用すると、新しい複製サーバーを簡単に作成でき、さらにそのサーバーをオンラインにするためにかかる時間を削減できます。これらのタスクは、次の 2 種類のコマンドを使用して実行します。

- `nisbackup` - NIS+ ディレクトリオブジェクトをバックアップする
- `nisrestore` - NIS+ ディレクトリオブジェクトを復元する

nisbackup を使用して名前空間をバックアップする

`nisbackup` コマンドは、1 つまたは複数の NIS+ ディレクトリオブジェクト、または 1 つの名前空間全体を、指定した UNIX ファイルシステムのディレクトリにバックアップします。

注 - nisbackup コマンドは、必ずマスターサーバー上で実行してください。複製サーバー上では絶対に実行しないでください。

nisbackup コマンドは、バックアップコマンドが動作するように設定された時点の NIS+ 名前空間をコピーします。この記録には、現行のすべての NIS+ データと、認証されたネットワーク管理者が NIS+ 名前空間に入力した変更が含まれます。ただし、まだ NIS+ テーブルにチェックポイントされていない (NIS+ テーブルに記入されていない) ものは除きます。このバックアップ処理は、NIS+ データのチェックや訂正は行いません。テーブル内のデータが破壊されると、破壊されたデータは、有効なデータと見分けがつかない状態にバックアップされます。

nisbackup コマンドは、マシンがそのマスターサーバーであるディレクトリオブジェクトだけをバックアップします。つまり、nisbackup は、マスターサーバー上でだけ使用でき、複製サーバー上では使用できません。

- マシンが、ドメインとサブドメイン両方の NIS+ ディレクトリオブジェクトのマスターサーバーである場合、nisbackup をこのマシン上で実行するとドメインとサブドメイン両方のディレクトリオブジェクトをバックアップできます。
- ただし、マシンが、1 つのディレクトリオブジェクトのマスターサーバーで、さらに別のディレクトリオブジェクトの複製サーバーである場合、nisbackup を実行してそのマシンがマスターサーバーであるディレクトリオブジェクトをバックアップできますが、複製サーバーのオブジェクトはバックアップされません。

バックアップ処理が、他の処理に割り込まれたり、またはその処理を正常に終了できない場合は、処理を停止し、転送先ディレクトリ内に格納された以前のバックアップファイルをすべて復元します。

nisbackup の構文

nisbackup コマンドで使用する構文は、次のとおりです。

```
nisbackup [-v][-a] backupdir objects
```

- *backupdir* には、バックアップファイルを格納する転送先ディレクトリが入ります。たとえば、/var/master1_bakup です。

- *objects* には、バックアップする NIS+ ディレクトリオブジェクトが入ります。たとえば、org_dir.doc.com です。ここには、複数の NIS+ ディレクトリオブジェクトをスペースで区切って入力できます。

nisbackup コマンドには、次のようなオプションを指定できます。

表 16–1 nisbackup コマンドのオプション

オプション	目的
–v	冗長モード。このモードは追加情報を出力する
–a	すべて。サーバーがマスターである NIS+ ディレクトリオブジェクトをすべてバックアップする。これには、このサーバーがマスターであるサブドメインのディレクトリオブジェクトも含まれる。ただし、他のマスターサーバーを持つサブドメインのディレクトリオブジェクトはバックアップされない

nisbackup コマンドは、バックアップする NIS+ ディレクトリオブジェクトのマスターサーバー上で実行する必要があります。

バックアップする NIS+ ディレクトリオブジェクトを指定する場合、そのディレクトリ名には完全指定名、または部分指定名を使用できます。

マルチレベルディレクトリをバックアップする場合、下位ディレクトリのバックアップファイルは、自動的にバックアップ転送先ディレクトリのサブディレクトリ内に配置されます。

nisbackup によるバックアップの対象

nisbackup を使用する場合、nisbackup はサーバー固有のコマンドであることに注意してください。–a オプションを使用するかどうかに関係なく、nisbackup は、このコマンドを実行中のサーバーがマスターサーバーであるディレクトリだけをバックアップします。他にマスターサーバーを持つ NIS+ ディレクトリオブジェクトは、バックアップされません。

たとえば、submaster1 は、sales.doc.com. ディレクトリオブジェクトのマスターサーバーで、さらに west.sales.doc.com. ディレクトリオブジェクトの複製サーバーでもあるとします。この場合、submaster1 上で nisbackup を実行すると、sales.doc.com. ディレクトリオブジェクトだけがバックアップされます。

このサーバー固有の原則には次のものがあります。

- 「全 NIS+ 名前空間」

  複数のドメインのすべての名前空間に、NIS+ バックアップを実行する場合で、ルートマスターサーバーが全サブドメインのマスターサーバーでもある場合は、ルートマスター上で nisbackup に –a オプションを指定して実行します。ただし、スーパーユーザーのマスターサーバーが、すべてのサブドメインのマスターサーバーでない場合は、すべての名前空間のバックアップを完全に実施するために、nisbackup を他のマスターサーバー上でも実行する必要があります。

- 「サブドメイン」

  1 つまたは複数のサブドメインの NIS+ バックアップを実行する場合、サブドメインのマスターサーバー上で nisbackup を実行する必要があります。ルートマスターなどの 1 つのマシンが、1 つまたは複数のサブドメインのマスターでもある場合、そのマシン上で nisbackup に –a オプションを指定して実行します。

- 「FNS ctx_dir」

  FNS を実行している場合は、nisbackup を ctx_dir のマスターサーバー上で実行し、ctx_dir をバックアップするように指定するか、または –a オプションを使用すると、ctx_dir ディレクトリだけがバックアップされます。通常は、ctx_dir と NIS+ ディレクトリオブジェクトが別々のマスターサーバーから提供されていますが、この場合は、すべてのディレクトリをバックアップするには、nisbackup を両方のマシン上で実行する必要があります。

バックアップ転送先ディレクトリ

バックアップ転送先ディレクトリは、バックアップ対象のサーバーが使用できるものでなければなりませんが、サーバー上に物理的にマウントしていない転送先ディレクトリを使用するのも良い方法です。この場合、サーバーがダメージを受けても、バックアップディレクトリは使用可能です。

独立した転送先ディレクトリは、バックアップ対象となるマスターサーバーごとに使用する必要があります。混乱を避けるために、マスターサーバーのマシン名を転送先ディレクトリ内に組み込むと良いでしょう。たとえば、master1 マシン上で実行した nisbackup の転送先ディレクトリは、/var/master1_bakup という名前にします。

注意 - 指定した 1 つの転送先ディレクトリに対して、複数のサーバーをバックアップしないでください。異なるマスターサーバーには、必ず、異なる転送先ディレクトリを使用してください。それは、指定した転送先ディレクトリに、1 つまたは複数の NIS+ ディレクトリオブジェクトをバックアップするたびに、このディレクト内のこれらの NIS+ ディレクトリオブジェクト用のそれ以前のバックアップファイルが上書きされるからです。

NIS+ のバックアップを日付順に保存する

バックアップファイルを日付順に保存するには、少なくとも次の 2 つの方法があります。

- 「別々の転送先ディレクトリとして保存」

  異なる転送先ディレクトリは、バックアップした日付ごとに保持できます。たとえば、/var/master1_bakup/July14、/var/master1_bakup/July15、など。この方法は簡単ですが、ディスク領域がかなり必要です。

- 「ファイルシステムのバックアップ」

  NIS+ バックアップを日付順に保存する最も一般的な方法は、使用する通常ファイルシステムのバックアップメソッドに、バックアップ転送先ディレクトリを単に組み込むだけの方法です。これを簡単に行うには、nisbackup コマンドを crontab ファイルから実行するか、または Solstice バックアップルーチン内から実行します。nisbackup のようなコマンドを、システムのバックアッププロシージャとして自動的に実行するように指定する方法については、Solstice の説明書を参照してください。

特定の NIS ディレクトリをバックアップする

特定の NIS+ ディレクトリオブジェクトをバックアップするには、これらのディレクトリをバックアップ転送先ディレクトリの後ろに入力します。

たとえば、ルート、sales ドメイン、manf ドメインの 3 つの org_dir ディレクトリオブジェクトを /master1_backup ディレクトリにバックアップするには、nisbackup を master1 マシン上で次のように実行します。

```
master1# nisbackup /var/master1_bakup org_dir org_dir.sales org_dir.manf
```

すべての NIS+ 名前空間をバックアップする

すべての NIS+ 名前空間をバックアップする場合は、ルートマスターサーバー上で、nisbackup コマンドに –a オプションを指定して実行します。

–a オプションを使用する場合は、バックアップする NIS+ ディレクトリオブジェクトは指定しません。サーバー上とそのサーバーの下にあるサブドメインのすべての NIS+ ディレクトリオブジェクトは、自動的にバックアップされます。

たとえば、doc.com. 名前空間を /master1_bakup ディレクトリにバックアップするには、ルートマスター上で、nisbackup を次のように実行してください。

```
rootmaster# nisbackup -a /var/master1_bakup
```

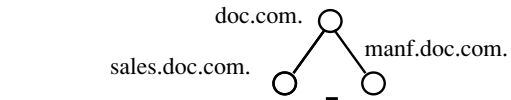
バックアップディレクトリの構造

ドメイン上でバックアップを実行すると、バックアップ転送先ディレクトリ内に、NIS+ ディレクトリオブジェクトごとにサブディレクトリが作成されます。これらのサブディレクトリ名は、完全指定の NIS+ ディレクトリオブジェクト名の末尾にピリオドが付いたものになります。

–a オプションを使用して、すべての NIS+ オブジェクトを完全にバックアップすると、3 つの関連ディレクトリオブジェクト (ドメイン、org_dir.domein、groups_dir.domein) すべてがバックアップされ、転送先サブディレクトリが 3 つ作成されます。複数のオブジェクトをバックアップすると、サブディレクトリはバックアップしたそれぞれのオブジェクトごとに作成されます。

複数の NIS+ ディレクトリオブジェクトのバックアップサブディレクトリは、それがサブドメインであるかどうかに関係なく、親バックアップ転送先ディレクトリのサブディレクトリになるので注意してください。つまり、nisbackup は、親バックアップ転送先ディレクトリの下にドメインの階層を複製しません。その代わりに、バックアップサブディレクトリはすべて、転送先ディレクトリの単純なサブディレクトリになります。

たとえば、ルート、sales、manf のそれぞれからディレクトリオブジェクト doc.com. を /var/master1_bakup ディレクトリにバックアップする場合、図 16–1 に示すように、/var/master1_bakup ディレクトリ内には 9 個のサブディレクトリが作成されます。



nisbackup

```
/var/master1_bakup
     /master1_bakup/doc.com./
     /master1_bakup/groups_dir.doc.com./
     /master1_bakup/org_dir.doc.com./
     /master1_bakup/sales.doc.com./
     /master1_bakup/groups_dir.sales.doc.com./
     /master1_bakup/org_dir.sales.doc.com./
     /master1_bakup/manf.doc.com./
     /master1_bakup/groups_dir.manf.doc.com./
     /master1_bakup/org_dir.manf.doc.com./
```

図 16–1　nisbackup によって作成されるディレクトリの例

バックアップファイル

バックアップ転送先ディレクトリには、この転送先ディレクトリにバックアップされた最新の NIS+ ディレクトリオブジェクトを表示する backup_list ファイルが入っています。

各サブディレクトリには、ファイルが 2 つと /data サブディレクトリが 1 つ組み込まれます。この 3 つのファイルを次に示します。

- data.dict

  このディレクトリにバックアップされた NIS+ ディレクトリオブジェクトの NIS+ データ辞書の入った XDR コード化ファイル

- last.upd

  このディレクトリにバックアップされた NIS+ ディレクトリオブジェクトに関する時刻スタンプ情報が入ったバイナリファイル

各 /data サブディレクトリには、1 つまたは複数の以下のファイルが入っています。

- root.object

  NIS+ ルートディレクトリオブジェクトの説明の入った XDR コード化ファイル。たとえば、/master1_bakup/doc.com/data/root.object

- root_dir

これらのオブジェクトのルートディレクトリとサーバー情報内に組み込まれた NIS+ オブジェクトの説明が入った XDR コード化ファイル。たとえば、/master1_bakup/doc.com/data/root_dir

- table.directory (テーブルディレクトリ)

  バックアップの実行時に、NIS+ テーブル内に表示されていたデータと関連するすべての NIS+ ログファイル内に含まれるデータがすべて入った XDR コード化ファイル。バックアップされた NIS+ ディレクトリオブジェクト内に NIS+ テーブルがある場合、対応する table.directory バックアップファイルが、そのディレクトリオブジェクトの /data サブディレクトリ内に作成される。

  たとえば、それぞれの NIS+ org_dir ディレクトリには、hosts テーブルが含まれるため、各 *target*/org_dir.*domain*/data サブディレクトリには、hosts.org_dir がある。たとえば、/master1_bakup/org_dir.doc.com./data/hosts.org_dir です。

  指定したディレクトリオブジェクト内に表示されたユーザー作成の NIS+ テーブルは、標準 NIS+ テーブルとして同じ方法でバックアップされます。

- groups_dir

  NIS+ グループ情報の入った XDR コード化ファイル。このファイルは、対応する NIS+ groups_dir 転送先ディレクトリに格納されます。

nisrestore を使用して NIS+ 名前空間を復元する

nisrestore コマンドによって、nisbackup を使用して作成したバックアップファイル内に格納されたデータと一致する NIS+ ディレクトリオブジェクトが再現されます。このコマンドを使用すると、NIS+ サーバーの復元、壊れたディレクトリオブジェクトの置換、または新しい NIS+ サーバーに NIS+ データを読み込めます。

nisrestore を実行するための前提条件

nisrestore を使用するためには、nisrestore から NIS+ データの受け取り先マシンが、NIS+ サーバーとして設定されている必要があります (NIS+ サーバーの設定の詳細は、『*Solaris* ネーミングの設定と構成』を参照)。つまり、次のような状態にしておく必要があります。

- マシンを、NIS+ クライアントとして初期化しておく必要があります。

- マシンを NIS 互換モードで実行し、ドメイン名システム (DNS) 転送をサポートする場合は、そのマシンには、適切に構成された /etc/resolv.conf ファイルが必要です。
- 名前空間内で他のサーバーが実行中の時、あるサーバーで nisrestore を使用する場合は、nisrestore は、他のサーバーを検証して、このサーバーがサーバーに復元するバックアップ NIS+ オブジェクトを配布するように構成されているかどうかを確認します。実行中のサーバーが他にない場合は、nisrestore に –f オプションを指定して実行する必要があります。つまり、nisrestore がチェックする他のサーバーがある場合は、–f オプションを使用する必要はありません。また、使用できるサーバーが他にない場合、たとえば、1 台のマスターサーバーを復元するときに、機能複製サーバーが他にない場合は、–f オプションを使用する必要があります。

注意 - 上記の 3 つの前提条件への追加条件として、マシン上で rpc.nisd デーモンを実行しないでください。rpc.nisd デーモンを実行する場合は、rpc.nisd を消去してから nisrestore を実行してください。

nisrestore の構文

nisrestore コマンドでは、次の構文を使用します。

```
nisrestore [-fv][-a][-t] backupdir [directory_objects]
```

- *backupdir* には、NIS+ オブジェクトの復元に使用するバックアップファイルの入ったディレクトリを入力します (/var/master1_bakup)。
- *directory_objects* には、復元する NIS+ ディレクトリオブジェクトを入力します (org_dir.doc.com)。複数の NIS+ ディレクトリオブジェクトを、スペースで区切って入れることができます。nisrestore に –a オプションを指定して実行する場合は、特定のディレクトリオブジェクトは指定しません。

nisrestore コマンドには、以下のオプションを指定できます。

表 16–2　nisrestore コマンドのオプション

オプション	目的
–a	すべて。バックアップディレクトリ内に入っている NIS+ ディレクトリオブジェクトをすべて復元する
–f	サーバーが、ディレクトリオブジェクトのサーバーリストに記載されているかどうかを検査せずに、強制的に復元を行う。このオプションは、ルートマスターサーバーを復元する時、または " オブジェクトを検出できません " といった種類のエラーを受け取った場合に使用する必要がある
–v	冗長モード。このモードは、追加情報を出力する
–t	このオプションを使用すると、バックアップディレクトリ内に格納された NIS+ ディレクトリオブジェクトがすべて表示される。オブジェクトの復元は行われない

nisrestore を使用する

NIS+ バックアップファイルから NIS+ データを復元するには、nisrestore コマンドを使用します。

たとえば、org_dir.doc.com. ディレクトリオブジェクトを replica1 サーバーに復元する場合は、スーパーユーザーになって replica1 にログインします。346 ページの「nisrestore を実行するための前提条件」 で説明した前提条件が満たされていることを確認してから、以下のように nisrestore を実行します。

```
replica1# nisrestore /var/master1_bakup org_dir.doc.com.
```

nisrestore には、以下の項目が適用されます。

- 「損傷した名前空間」

  損傷した、または破壊された NIS+ 名前空間を復元するには、復元する NIS+ ディレクトリオブジェクトのすべてのサーバー上で nisrestore コマンドを実行する必要があります。

- 「検出エラー」

  nisrestore が必要なデータを確認できないか、または検出できないというエラーメッセージを受け取った場合は、–f オプションを使用する必要があります。

たとえば、master1 という名前のルートマスターサーバー上に NIS+ データをロードし直す場合は、次のように入力します。

```
master1# nisrestore -f -a /var/master1_bakup
```

- 「ディレクトリ名」

  復元する NIS+ ディレクトリオブジェクトを指定する場合は、完全指定または部分指定ディレクトリ名を使用します。

バックアップと復元を使用して複製サーバーを設定する

NIS+ バックアップおよび復元機能を使用すると、NIS+ データを新しい複製サーバーに速く読み込むことができます。名前空間が広い場合は、この方法の方が、nisping を使用するよりもマスターサーバーからのデータを非常に速く入手できます。

新しい複製サーバーの設定に nisbackup と nisrestore を使用する方法の詳細は、『*Solaris* ネーミングの設定と構成』で説明しています。手順の簡単な説明を次に示します。

1. マスター上で nisserver を実行し、新しい複製サーバーを作成します。
2. 新しい複製サーバー上で rpc.nisd を消去します。

   この処理は、nisping コマンドを使用した名前空間データのマスターから複製への自動転送に割り込んで実行されます。
3. マスターサーバー上で、nisbackup を実行します。
4. 新しい複製サーバー上で nisrestore を実行し、**NIS+** データを読み込みます。
5. 新しい複製サーバー上で rpc.nisd を再起動します。

サーバーマシンを置換する

nisbackup と nisrestore を使用すると、サーバーとして使用中のマシンと別のマシンをすぐに置換できます。たとえば、旧サーバーを新しい高速のサーバーと交換すると、ネットワークのパフォーマンスを向上させることができます。

マシンを置換する場合の必要条件

NIS+ サーバーとして使用中のマシンを他のマシンに置き換える場合には、次の条件が必要です。

- 新しいマシンには、置換する旧マシンと同じ IP アドレスを割り当てます。
- 新しいマシンには、置換する旧マシンと同じマシン名を割り当てます。
- 新しいマシンは、置換する旧マシンと同じサブネットに接続します。

サーバーマシンの置換方法

サーバーマシンを置換する場合は、次の手順に従ってください。

1. 旧サーバーが管理するドメインのマスターサーバー上で nisbackup を実行します。

   詳細は、344ページの「すべての NIS+ 名前空間をバックアップする」を参照してください。置換する旧サーバーがマスターサーバーである場合もあるので注意してください。この場合は、この旧マスターサーバー上で nisbackup を実行します。

2. 旧サーバーの /var/nis/NIS_COLD_START ファイルをバックアップディレクトリにコピーします。

3. 旧サーバーの /etc/.rootkey ファイルをバックアップディレクトリにコピーします。

4. 旧サーバーをネットワークから切り離します。

5. 新しいサーバーをネットワークに接続します。

6. 新しいサーバーに旧サーバーと同じ **IP** アドレス **(**番号**)** を割り当てます。

7. 新しいサーバーに旧サーバーと同じマシン名を割り当てます。

8. 必要な場合は、新しいサーバー上で `rpc.nisd` を消去します。

9. 新しいサーバー上で `nisrestore` を実行し、**NIS+** データを読み込みます。
   詳細は、346ページの「nisrestore を使用して NIS+ 名前空間を復元する」を参照してください。

10. `.rootkey` ファイルを、バックアップディレクトリから新しいサーバーの `/etc` にコピーします。

11. `NIS_COLD_START` ファイルを、バックアップディレクトリから新しいサーバーの `/var/nis` にコピーします。

12. 新しいサーバーを再起動します。

第 17 章

NIS+ の削除

この章では、NIS+ ディレクトリ管理コマンドを使用して、クライアントまたはサーバーから NIS+ を削除する方法と、NIS+ 名前空間全体を削除する方法について説明します。

- 353ページの「クライアントマシンから NIS+ を削除する」
- 355ページの「サーバーから NIS+ を削除する」
- 356ページの「NIS+ 名前空間を削除する」

NIS+ 複製サーバーを、ディレクトリから分離し、そのドメインの複製サーバーとして機能しないようにする場合は、261ページの「nisrmdir コマンド」を参照してください。

クライアントマシンから NIS+ を削除する

この節では、クライアントマシンから NIS+ を削除する方法について説明します。ただし、クライアントマシンから NIS+ を削除しても、ネットワークから NIS+ ネームサービスを削除したことにはならないので注意してください。ネットワークから NIS+ ネームサービスを削除して、NIS または `/etc` ディレクトリのファイルをネームサービスとして使用する状態に戻す場合は、356ページの「NIS+ 名前空間を削除する」を参照してください。

nisclient を使用してインストールした NIS+ を削除する

『*Solaris* ネーミングの設定と構成』で説明されているように、nisclient –i スクリプトを使用して NIS+ クライアントとして設定したクライアントマシンから NIS+ を削除するには、以下のように –r オプションを指定して nisclient を実行します。

```
client# nisclient -r
```

nisclient –r では、nisclient –i 1 回分の処理が取り消されます。つまり、nisclient –i 実行以前にクライアントによって使用されていたネーミングシステム (NIS、あるいは /etc ディレクトリのファイルなど) が再び使用されるようになります。

NIS+ コマンドでインストールした NIS+ を削除する

nisaddcred、domainname、nisinit といったコマンドによって NIS+ クライアントとして設定したクライアントマシン (『*Solaris* ネーミングの設定と構成』を参照) から NIS+ を削除する手順は以下のとおりです。

1. ファイル .rootkey を削除します。

```
client# rm -f /etc/.rootkey
```

2. keyserv、nis_cachemgr、nscd のプロセス **ID** を確認して、終了します。

```
client# ps -ef | grep keyserv
 root 714 1 67 16:34:44 ? keyserv
client# kill -9 714
client# ps -ef | grep nis_cachemgr
 root 123 1 67 16:34:44 ? nis_cachemgr
client# kill -9 123
client# ps -ef | grep nscd
 root 707 1 67 16:34:44 ? nscd
client# kill -9 707
```

3. /var/nis ディレクトリとその下のファイルを削除します。

```
clientmachine# rm -rf /var/nis/*
```

サーバーから NIS+ を削除する

この節では、NIS+ サーバーから NIS+ を削除する方法を示します。

ただし、サーバーから NIS+ を削除しても、ネットワークから NIS+ ネームサービスを削除したことにはならないので注意してください。ネットワークから NIS+ ネームサービスを削除して、NIS または /etc ディレクトリのファイルをネームサービスとして使用する状態に戻す場合は、356ページの「NIS+ 名前空間を削除する」を参照してください。ルートマスターサーバーから NIS+ を削除します。

注 - NIS+ サーバーとして使用しているマシンを、別のマシンに置換できます。350ページの「サーバーマシンを置換する」を参照してください。

サーバーから NIS+ を削除する手順は以下のとおりです。

1. **クライアントから NIS+ を削除する作業を行います。**

   NIS+ サーバーも NIS+ クライアントの一種なので、まずクライアントに関連する部分を削除する必要があります。このためには nisclient –r (354ページの「nisclient を使用してインストールした NIS+ を削除する」を参照) か、NIS+ コマンド (354ページの「NIS+ コマンドでインストールした NIS+ を削除する」を参照) を使用します。

2. **サーバーの** groups_dir **ディレクトリと** org_dir **ディレクトリを削除する。**

```
server# nisrmdir -f groups_dir. domainname
server# nisrmdir -f org_dir. domainname
```

3. keyserv、rpc.nisd、nis_cachemgr、nscd **のプロセス ID を確認し、終了します。**

```
server# ps -ef | grep rpc.nisd
 root 137 1 67 16:34:44 ? rpc.nisd
server# kill -9 137
server# ps -ef | grep keyserv
 root 714 1 67 16:34:44 ? keyserv
server# kill -9 714
server# ps -ef | grep nis_cachemgr
 root 123 1 67 16:34:44 ? nis_cachemgr
server# kill -9 123
server# ps -ef | grep nscd
 root 707 1 67 16:34:44 ? nscd
server# kill -9 707
```

4. /var/nis ディレクトリとその下のファイルを削除します。

```
rootmaster# rm -rf /var/nis/*
```

NIS+ 名前空間を削除する

NIS+ 名前空間を削除し、NIS または /etc ディレクトリのファイルをネームサービスとして使用する状態に戻す手順は以下のとおりです。

1. ルートマスターから .rootkey ファイルを削除します。

```
rootmaster# rm -f /etc/.rootkey
```

2. ルートマスターのルートドメインから groups_dir サブディレクトリと org_dir サブディレクトリを削除します。

```
rootmaster# nisrmdir -f groups_dir. domainname
rootmaster# nisrmdir -f org_dir.domainname
```

domainname には、ルートドメイン名 (doc.com など) が入ります。

3. ルートドメインを削除します。

```
rootmaster# nisrmdir -f domainname
```

domainname には、ルートドメイン名 (doc.com など) が入ります。

4. keyserv、rpc.nisd、nis_cachemgr、nscd のプロセス **ID** を確認し、終了します。

```
rootmaster# ps -ef | grep rpc.nisd
 root 137 1 67 16:34:44 ? rpc.nisd
rootmaster# kill -9 137
rootmaster# ps -ef | grep keyserv
 root 714 1 67 16:34:44 ? keyserv
rootmaster# kill -9 714
rootmaster# ps -ef | grep nis_cachemgr
 root 123 1 67 16:34:44 ? nis_cachemgr
rootmaster# kill -9 123
rootmaster# ps -ef | grep nscd
 root 707 1 67 16:34:44 ? nscd
rootmaster# kill -9 707
```

5. 新しいドメインを作成します

```
rootmaster# domainname name
```

name には、新しいドメイン名 (NIS+ インストール前のドメイン名など) が入ります。

6. 既存の /etc/defaultdomain ファイルを削除します。

```
rootmaster# rm /etc/defaultdomain
```

7. /etc/defaultdomain ファイルを、新しいドメイン名を使用して作成し直します。

```
rootmaster# domainname > /etc/defaultdomain
```

8. nsswitch.conf ファイルを元のファイルに戻します。

   サーバーを nisserver –r を使用して設定した場合は、以下のコマンドを使用します。

```
rootmaster# cp /etc/nsswitch.conf.no_nisplus /etc/nsswitch.conf
```

   また、デフォルトスイッチテンプレートファイルの 1 つをコピーする方法もあります。NIS スイッチのデフォルトファイルテンプレートを使用する場合は、以下のコマンドを入力します。

```
rootmaster# cp /etc/nsswitch.nis etc/nsswitch.conf
```

   /etc ファイルのデフォルトスイッチファイルテンプレートを使用する場合は、以下のコマンドを入力します。

```
rootmaster# cp /etc/nsswitch.files etc/nsswitch.conf
```

9. keyserv プロセスを再起動します。

```
rootmaster# keyserv
```

10. /var/nis ディレクトリとその下のファイルを削除します。

```
rootmaster# rm -rf /var/nis/*
```

11. この状態で、別のネームサービス **(NIS** または /etc ファイル**)** を再起動できます。

パート IV　NIS の管理

パート IV では、「ネットワーク情報サービス」(NIS) とその管理方法について説明します。

- 第 18 章
- 第 19 章

第 18 章

ネットワーク情報サービス (NIS)

この章では、「ネットワーク情報サービス」(NIS) について説明します。

NIS の初期設定と構成の方法については、『*Solaris* ネーミングの設定と構成』を参照してください。

NIS の概要

NIS とは分散型ネームサービスであり、ネットワーク上のオブジェクトおよびリソースを識別し、探索するメカニズムです。NIS は、ネットワーク全体の情報に関する一様な記憶領域と検索方法を、転送プロトコルおよび媒体に依存しない形式で提供します。

システム管理者はネットワーク情報サービスを動作させることにより、「マップ」と呼ばれる管理データベースをさまざまなサーバー (「マスター」と「スレーブ」) に分散でき、またこれらの管理データベースを一元管理により自動的かつ確実な方法で更新できます。したがって、すべてのクライアントがネットワーク全体におい

て一貫した方法で同じネームサービス情報を共有できます。NIS の概要および背景の詳細は、46ページの「NIS とは」を参照してください。

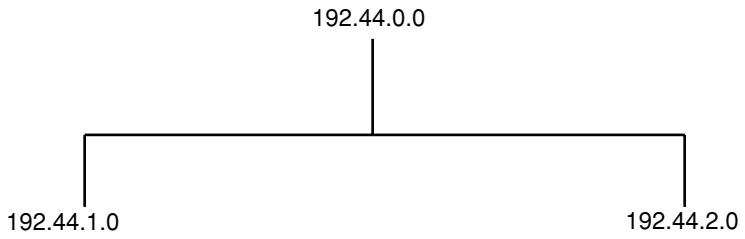
NIS (ネットワーク情報サービス) は DNS とは独立して開発され、目的はやや異なっています。DNS は数値 IP アドレスの代わりにマシン名を使うことによって、通信を簡略化することに焦点を当てているのに対して、NIS の場合は、多様なネットワーク情報を集中管理することによりネットワーク管理機能を高めることに焦点を絞っています。NIS には、マシン名とアドレスだけでなく、ユーザー、ネットワークそのもの、ネットワークサービスについての情報も格納されます。これらのネットワーク「情報」をまとめて NIS の「名前空間」と呼びます。

注 - 「マシン」名の代わりに「ホスト」名または「ワークステーション」名が使われることがあります。この解説では「マシン」名が使われていますが、一部の画面メッセージまたは NIS マップ名では「ホスト」名または「ワークステーション」名が使われています。

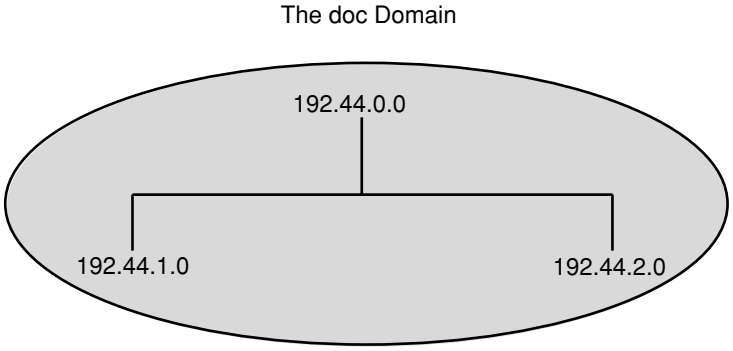
NIS アーキテクチャ

NIS はクライアントサーバー方式を使用します。NIS サーバーが NIS のクライアントへサービスを提供します。主サーバーは「マスター」サーバーと呼ばれ、信頼性を保証するためにバックアップつまり「スレーブ」サーバーを持っています。マスターサーバーとスレーブサーバーは、NIS の情報検索ソフトウェアを使い、NIS のマップを格納します。

NIS はドメインを使用して、マシン、ユーザー、およびネットワークを自分の名前空間に配置します。しかし、ドメイン階層を使用しないため、NIS の名前空間はフラットになっています。



したがって、上記のような物理ネットワークは、次のように 1 つの NIS ドメインに配置されます。



NIS だけを使って NIS ドメインをインターネットに直接接続できません。しかし、NIS を使用し、インターネットへの接続を希望する組織は、NIS と DNS を組み合せることができます。その場合、NIS を使用してすべてのローカル情報を管理し、DNS を使用してインターネットのホストを検索できます。NIS は、NIS マップで情報が見つからない場合にホスト検索の機能を DNS へ転送する転送サービス機能を持っています。Solaris 環境では、NIS 管理者からのホスト検索要求を DNS だけに転送したり、DNS に情報が見つからなければ次に NIS 、あるいは NIS で情報が見つからなければ次に DNS に転送する、という切り替えも `nsswitch.conf`ファイルの設定によって可能です (詳細は、第 2 章を参照)。

NIS と NIS+

NIS と NIS+ は、いくつかの同じ作業を行います。ただし NIS+では、NIS では提供されない階層ドメイン、名前空間セキュリティ、その他の機能が使用できます。NIS と NIS+ の相違点の詳細は、70ページの「NIS+ と NIS の違い」を参照してください。

NIS 管理者は、以下に示す原則および条件下で NIS を NIS+ と共に使用できます。

- 同一ドメインに NIS サーバーと NIS+ サーバーの両方が存在する

  NIS 管理者は同一ドメインで NIS サーバーと NIS+ サーバーの両方を動作させることは可能ですが、このような動作を長時間行わせることは望ましくありません。一般に、同一ドメイン内での NIS サーバーと NIS+ サーバーを両方使用するのは、NIS から NIS+ への短い移行期間だけに制限するべきです。クライアントから NIS サービスの要求があった場合に、NIS 管理者は、NIS+ を NIS 互換モー

ドで動作させることができます。詳細は、74ページの「Solaris 1.x と NIS 互換モード」を参照してください。

- サブドメイン

  NIS 管理者のルートドメインのマスターサーバーで NIS+ が動作している場合は、NIS 管理者は、すべてのサーバーで NIS が動作しているサブドメインを設定できます。NIS 管理者のルートドメインのマスターサーバーで NIS が動作している場合は、NIS 管理者はサブドメインの設定はできません。この操作は、NIS 管理者が NIS から NIS+ に切り換えるときに有用であるかもしれません。たとえば、互いに独立した複数の NISドメイン (ドメインは地理的に別々のサイトに置かれていることもある) を持った会社の中で、NIS 管理者がそれらのドメインをすべてリンクさせて NIS+ の下に1つの階層型マルチドメイン名前空間を構築するケースを考えてみます。この場合、NIS 管理者はまずルートドメインを NIS+ 下に設定し、次に従来の NISドメインをサブドメインとして指定できます。これらのサブドメインでは、NIS+ への切り換えが行われるまで NIS が継続して動作します。

- 同一ドメインに複数のマシンが存在する

  - 同一ドメイン内のサーバーで NIS+ が動作している場合は、NIS+、NIS、または `/etc` ファイルを使ってネームサービス情報を取得するように、ドメイン内の各マシンを設定できます。NIS+ サーバーが NIS クライアントのニーズに応えるには、この NIS+ サーバーは、『*Solaris* ネーミングの設定と構成』で説明されていように NIS 互換モードで動作していなければなりません。
  - 同一ドメイン内のサーバーで NIS が動作している場合は、NIS または `/etc` ファイルを使ってネームサービス情報を取得するように、このドメイン内の各マシンを設定できます (各マシンは NIS+ を使用することはできません) 。

さまざまなネームサービス情報を取得するためにマシンがどのサービスを使用するかは、そのマシンの `nsswitch.conf` ファイルで制御されます。このファイルは、「スイッチ」ファイルと呼ばれています。詳細は、第 2 章を参照してください。

NIS と FNS

ある特定の状況では、FNS コマンドを使用して NIS クライアントは、ファイルシステムやプリンタの自分に関係のあるネーム情報を更新します。詳細は、462ページの「NIS クライアントが SKI の実行中 FNS によってコンテキストを更新する」を参照してください。

NIS マシンのタイプ

NIS マシンには、以下の 3 つのタイプがあります。

- マスターサーバー
- スレーブサーバー
- NIS サーバーのクライアント

NIS クライアントにはどのマシンでもなれますが、NIS サーバー (マスターまたはスレーブ) となるのはディスクが装備されているマシンだけです。一般にサーバーは、クライアントでもあります。

NIS サーバー

NIS サーバーは、ネットワーク上のマシンおよびアプリケーションが使用可能な一群のマップを保存するマシンと定義されています。NIS サーバーは、FNS ファイルサーバーと同じマシンである必要はありません。

NIS サーバーには、マスターサーバーとスレーブサーバーがあり、マスターサーバーとして指定されているマシンには、NIS 管理者が必要に応じて作成、更新する一群のマップが保存されます。各 NIS ドメインには、マスターサーバーは 1 つだけ必要です。マスターサーバーは、パフォーマンスの低下を最小限におさえて NIS を更新できるマシンでなければなりません。

NIS 管理者は、ドメインに別の NIS サーバーをスレーブサーバーとして指定できます。各スレーブサーバーには、マスターサーバーの一群の NIS マップの完全なコピーが存在します。マスターサーバーの一群の NIS マップが更新されると、必ずこれらの更新がスレーブサーバーに反映されます。スレーブサーバーの存在によりシステム管理者は、NIS リクエストへの応答で発生する負荷を均等に分散できます。スレーブサーバーはまた、マスターサーバーが使用不可になったときの影響を最小限に抑えます。

通常、すべての NIS マップに対して 1 つのマスターサーバーを指定します。ただし、各 NIS マップ内にマスターサーバーのマシン名が符合化されているので NIS 管理者は、マスターサーバーおよびスレーブサーバーとして動作するように、異なる複数のマップに対して異なる複数のサーバーを指定することもできます。ただし、ある 1 つのサーバーをある 1 つのマップのマスターサーバーとして指定し、別のサーバーを別のマップのマスターサーバーとして指定するといったランダムな指

定は、管理上、非常に大きな混乱を発生させる可能性があります。したがって、1つのドメイン内に NIS 管理者が作成するすべてのマップに対して、1 つのサーバーをマスターサーバーとして指定することが最良です。この章の例では、1 つのサーバーがドメイン内のすべてのマップのマスターサーバーとなっています。

NIS クライアント

NIS クライアントでは、サーバー上のマップのデータを要求するプロセスが動作します。各 NIS サーバーに保存されている情報は同じであるはずなので、クライアントではマスターサーバーとスレーブサーバーの区別は行われません。

NIS サーバーはまた、多くの場合にクライアントともなっています。NIS クライアントの作成方法については、`ypbind(1M)` のマニュアルページを参照してください。

NIS の要素

NIS サービスは、以下の要素から構成されています。

- ドメイン (366ページの「NIS ドメイン」を参照)
- マップ (368ページの「NIS マップ」を参照)
- デーモン (367ページの「NIS デーモン」を参照)
- ユーティリティ (367ページの「NIS ユーティリティ」を参照)
- NIS コマンドセット (373ページの「NIS 関連コマンドについてのまとめ」を参照)

NIS ドメイン

NIS「ドメイン」は、共通な一群の NIS マップを共有するマシンの集合です。各ドメインはドメイン名を持っており、共通な一群の NIS マップを共有する各マシンは指定されたドメインに属します。ドメイン名は、大文字と小文字を区別します。

どのマシンも指定されたドメインに属することができます。ただしこれは、そのドメインのマップに対するサーバーが同一ネットワーク上に存在する場合に限ります。Solaris 2.x を動作しているマシンの場合はサーバーが同一サブネット上に存在する必要はありませんが、先行する NIS インプリメンテーションでは 1 つのサーバーが NIS が使用されている各サブネットに存在している必要がありました。NIS

クライアントマシンはドメイン名を取得し、NIS サーバーにブートプロセスの一部としてバインドされます。

NIS デーモン

NIS サービスは、表 18–1 に示されている 5 つのデーモンで提供されます。

表 18–1 NIS デーモン

デーモン	機能
`ypserv`	サーバープロセス
`ypbind`	バインドプロセス
`ypxfr`	高速マップ転送
`rpc.yppasswdd`	NIS パスワード更新デーモン
`rpc.ypupdated`	他のマップ (`publickey` など) を更新します

NIS ユーティリティ

NIS サービスは、表 18–2 に示されている 9 つのユーティリティでサポートされています。

表 18–2 NIS ユーティリティ

ユーティリティ	機能
`makedbm`	NIS マップの `dbm` ファイルを作成する
`ypcat`	マップのデータを一覧表示する
`ypinit`	NIS データベースの作成、インストール、および NIS クライアントの `ypservers` リストの初期化を行う
`yppmatch`	マップの特定エントリを検索する

表 18–2　NIS ユーティリティ　続く

ユーティリティ	機能
yppoll	サーバーからマップ順序番号を取得する
yppush	データを NIS マスターサーバーから NIS スレーブサーバーに反映させる
ypset	特定サーバーにバインドを設定する
ypwhich	NIS サーバー名およびニックネーム変換テーブルを表示する
ypxfr	NIS マスターサーバーから NIS スレーブサーバーにデータを転送する

NIS マップ

NIS は、マップと呼ばれている一群のファイルに情報を保存します。

NIS マップは、UNIX の /etc ファイルおよび他の構成ファイルを置換するように設計されているので、名前およびアドレスよりはるかに多くの情報を保存できます。NIS が動作しているネットワーク上では、各 NIS ドメインの NIS マスターサーバーは、照会されるドメイン内の他のマシンの一群の NIS マップを保持します。NIS スレーブサーバーは、NIS マスターサーバーのマップのコピーを保持します。NIS クライアントマシンは、マスターサーバーまたはスレーブサーバーから名前空間情報を取得できます。

NIS マップは、Solaris データベースのインプリメンテーションの 1 つのタイプです。他のタイプはこのタイプとは必ずしも重複しておらず、一般に /etc ディレクトリ、DNS リソースレコード (RR) 、NIS+ テーブルに存在するファイルです。

NIS マップの概要

NIS マップは、本質的には 2 列からなるテーブルです。1 つの列は「キー」であり、もう 1 つの列はキーに関連する情報値です。NIS は、キーを検索してクライアントに関する情報を見つけます。各マップでは異なるキーが使われるので、一部の情報はいくつかのマップに保存されます。たとえば、マシン名とアドレスは、hosts.byname と hosts.byaddr という 2 つのマップに保存されます。サーバーがマシンの名前を持っており、そのマシンのアドレスを見つける必要がある場

合は、サーバーは hosts.byname マップを調べます。サーバーがマシンのアドレスを持っており、そのマシンの名前を見つける必要がある場合は、サーバーは hosts.byaddr マップを調べます。

ドメインのマップは、各サーバーの /var/yp/*domainname* ディレクトリに存在します。たとえば、test.com ドメインに属しているマップは、各サーバーの /var/yp/test.com ディレクトリに存在します。

NIS Makefile は、インストール時に NIS サーバーとして指定されたマシンの /var/yp ディレクトリに保存されます。このディレクトリで make を実行すると、makedbm が入力ファイルからデフォルトの NIS マップを作成または更新します。このプロセスを使って NIS 名前空間を初期設定する方法については、『*Solaris* ネーミングの設定と構成』を参照してください。

注 - スレーブサーバー上でマップを作成しないでください。マップを作成する場合は、必ずマスターサーバー上で make を実行してください。

NIS マップの情報は、ndbm フォーマットで保存されます。マップファイルのフォーマットについては、ypfiles(4) および ndbm(3) のマニュアルページで説明されています。

デフォルトの NIS マップ

NIS 管理者には、デフォルトの一群の NIS マップが提供されます。NIS 管理者は、これらのマップをすべて使用することも、その一部だけを使用することもできます。NIS ではまた、他のソフトウェア製品のインストール時に NIS 管理者が作成または追加したマップはすべて使用できます。

表 18–3 には、デフォルトの NIS マップ、これらの NIS マップに存在する情報、および NIS 動作時にソフトウェアが対応する管理ファイルを調べているか否かが示されています。

表 **18–3** NIS マップに関する説明

マップ名	対応する **NIS** 管理ファイル	説明
bootparams	bootparams	ブート時にクライアントが必要とするファイルのパス名 (ルート、スワップ、その他) を含む
ethers.byaddr	ethers	マシン名と Ethernet アドレスを含む。マップのキーは、Ethernet アドレス
ethers.byname	ethers	ethers.byaddr と同じ。ただしキーは、Ethernet アドレスではなくマシン名
group.bygid	group	グループセキュリティ情報を含む。キーはグループ ID
group.byname	group	グループセキュリティ情報を含む。キーはグループ名
hosts.byaddr	hosts	マシン名と IP アドレスを含む。キーは IP アドレス
hosts.byname	hosts	マシン名と IP アドレスを含む。キーはマシン (ホスト) 名
mail.aliases	aliases	別名とメールアドレスを含む。キーは別名
mail.byaddr	aliases	メールアドレスと別名を含む。キーはメールアドレス
netgroup.byhost	netgroup	グループ名、ユーザー名、マシン名を含む。キーはマシン名
netgroup.byuser	netgroup	netgroup.byhost と同じ。ただし、キーはユーザー名
netgroup	netgroup	netgroup.byhost と同じ。ただし、キーはグループ名

表 18–3 NIS マップに関する説明 続く

マップ名	対応する **NIS** 管理ファイル	説明
netid.byname	passwd, hosts group	UNIX スタイルの認証に使用される。マシン名とメールアドレスを含む (ドメイン名も含む)。netid ファイルが使用可能な場合は、他のファイルの使用可能データーに加えて、netid ファイルも検索する
netmasks.byaddr	netmasks	IP 送出時に使用するネットワーを含む。キーはアドレス
networks.byaddr	networks	システムに認識されているネットワーク名、および IP アドレスを含む。キーは IP アドレス
networks.byname	networks	networks.byaddr と同じ。ただし、キーはネットワーク名
passwd.adjunct.byname	passwd、shadow	C2 クライアントに関する監査情報および隠されたパスワード情報を含む
passwd.byname	passwd、shadow	パスワード情報を含む。キーはユーザー名
passwd.byuid	passwd、shadow	passwd.byname と同じ。ただし、キーはユーザー ID
protocols.byname	protocols	システムに認識されているネットワークプロトコルを含む
protocols.bynumber	protocols	protocols.byname と同じ。ただし、キーはプロトコル番号
rpc.bynumber	rpc	システムに認識されている RPC のプログラム番号と名前を含む。キーは RPC のプログラム番号
services.byname	services	ネットワークに認識されているインターネットサービスを一覧表示する。キーはポートまたはプロトコル

表 18–3　NIS マップに関する説明　続く

マップ名	対応する NIS 管理ファイル	説明
services.byservice	services	ネットワークに認識されているインターネットサービスを一覧表示する。キーはサービス名
ypservers	N/A	ネットワークに認識されている NIS サーバーを一覧表示する

NIS マップの使用

NIS を使うと、/etc ファイルシステムを使った場合に比べ、ネットワークデータベースの更新がはるかに簡単になります。/etc ファイルシステムではネットワーク環境を更新するたびに各マシンの管理 /etc ファイルを変更する必要がありましたが、NISではこのような操作を行う必要はありません。

たとえば、NIS が動作しているネットワークに新しいマシンを追加する場合、NIS 管理者の作業は、マスターサーバーの入力ファイルを更新し、make を実行することだけです。これで、hosts.byname および hosts.byaddr マップが自動的に更新されます。次に、これらのマップはすべてのスレーブサーバーに転送され、ドメインのすべてのクライアントマシン、およびこれらのクライアントマシンのプログラムはこれらのマップを使用することが可能になります。クライアントマシンまたはアプリケーションがマシン名またはアドレスを要求すると、NIS サーバーは必要に応じて hosts.byname または hosts.byaddr マップを参照し、要求された情報をクライアントに送信します。

ypcat コマンドを使うと、マップの値を表示できます。ypcat の基本フォーマットは、次のとおりです。

```
% ypcat mapname
```

mapname は、調べたいマップ名またはその「ニックネーム」です。ypservers の場合のようにマップがキーだけで構成されている場合は、ypcat -k と入力してください。ypcat –k と入力しない場合は、空白行がプリントされます。他の ypcat オプションについては、ypcat(1) のマニュアルページで説明されています。

ypwhich コマンドを使うと、どのサーバーが特定マップのマスターサーバーなのかを判断できます。次のように入力してください。

```
% ypwhich -m mapname
```

mapname は、見つけたいマスターサーバーのマップ名またはニックネームです。*mapname* を入力すると、マスターサーバー名が表示されます。ypwhich の詳細は、ypwhich(1) のマニュアルページを参照してください。

NIS マップのニックネーム

ニックネームは、マップのフルネームの別名です。使用可能なマップのニックネーム (たとえば、passwd.byname の場合は passwd) を一覧表示するには、ypcat –x または ypwhich –x と入力してください。

ニックネームは、/var/yp/nicknames ファイルに保存されています。/var/yp/nicknames ファイルには、マップのニックネームとフルネームが 1 つの空白で区切られて入っています。ニックネームのリストは、追加または更新できます。ニックネーム数は現在、500 に制限されています。

NIS 関連コマンドについてのまとめ

NIS サービスには、特殊なデーモン、システムプログラム、コマンドが含まれています。これらのコマンドについては、表 18–4 にまとめられています。これらの各コマンドの使い方の詳細は、それぞれ該当するマニュアルページを参照してください。

表 18–4 NIS コマンドについてのまとめ

コマンド	説明
ypserv	NIS クライアントが要求する NIS マップの情報を提供します。ypserv は、完全な一群のマップが存在する NIS サーバー上で動作するデーモン。NIS サービスが機能するには、少なくとも 1 つの ypserv デーモンがネットワークに存在する必要がある
ypbind	クライアントに NIS サーバーバインド情報を提供する。ypbind は、要求元クライアントのドメイン内のマップにサービスを提供する ypserv プロセスを見つけてバインドを行う。ypbind はすべてのサーバーおよびクライアント上で実行される必要がある
ypinit	自動的に入力ファイルから NIS サーバーのマップを作成する。ypinitはまた、クライアント上に /var/yp/binding/*domain*/ypservers 初期ファイルを作成する際にも使用される。NIS マスターサーバーおよび NIS スレーブサーバーを初めて設定する場合は、ypinit を使用する

表 18–4 NIS コマンドについてのまとめ 続く

コマンド	説明
make	Makefile を読み込むことで NIS マップを更新する (make を /var/yp ディレクトリで実行した場合)。make を使うと、入力ファイルに基づいてすべてのマップを更新したり、個々のマップを更新したりできる。NIS の make の機能については、ypmake(1M) のマニュアルページで説明されている
makedbm	makedbm は入力ファイルを取得し、これを dbm.dir および dbm.pag ファイルに変換する (これらのファイルは、NIS がマップとして使用できる有効な dbm ファイル)。また、makedbm –u と入力すると、マップを分解できる。したがって、NIS 管理者は、マップを構成するキーと値のペアを参照できる
ypxfr	NIS 自体を転送媒体として使い、NIS マップをリモートサーバーから /var/yp/*domain* ローカルディレクトリに取り込む。NIS 管理者は ypxfr を対話形式で実行したり、crontab ファイルから定期的に実行したりできる。また、ypxfr が ypserv によって呼び出されると、転送が開始される
ypxfrd	ypxfr リクエスト (一般にスレーブサーバーで発生する) に対してマップ転送サービスを提供する。ypxfr は、マスターサーバー上でだけ動作する
yppush	NIS マップの新バージョンを NIS マスターサーバーからそのスレーブにコピーする。yppush の実行は、NIS マスターサーバー上で行う
ypset	指定された NIS サーバーにバインドするように ypbind プロセスに要求する。ypset は、セキュリティの関係上、通常のオペレーションで気軽に使用できるようには設計されていない。したがって、ypset はできる限り使用しない。ypbind プロセスの ypset および ypsetme オプションについては、ypset(1M) および ypbind(1M) のマニュアルページを参照
yppoll	指定されたサーバー上で NIS マップのどのバージョンが動作しているかを通知する。yppoll はまた、NIS マップのマスターサーバーを一覧表示する
ypcat	NIS マップの内容を表示する

表 18–4　NIS コマンドについてのまとめ　続く

コマンド	説明
ypmatch	NIS マップ内の指定された 1 つ以上のキーの値をプリントする。NIS 管理者は、NIS サーバーマップのバージョンを指定することはできない
ypwhich	現在どの NIS サーバーをクライアントが使用して NIS サービスを取得しているかを表示する。また、–m *mapname* オプションを指定して起動した場合は、どの NIS サーバーが各マップのマスターサーバーかが表示される。–m だけを指定した場合は、使用可能なすべてのマップ名、およびこれらのマップのマスターサーバーが表示される

NIS のバインド

NIS クライアントは、バインドプロセスにより NIS サーバーから情報を取得します。バインドプロセスは、サーバーリストおよび同報通信という 2 つのモードのどちらかで動作できます。

- サーバーリストモード

  サーバーリストモードでは ypbind プロセスは、/var/yp/binding/*domain*/ypservers リストでドメイン内のすべての NIS サーバー名を調べます。ypbindプロセスは、このファイルに存在するサーバーにだけバインドします。このファイルは、ypinit –c を動作させることで作成されます。

- 同報通信モード

  ypbind プロセスはまた、RPC 同報通信を使ってバインドを開始できます。同報通信は、これ以上送信されない唯一のローカルサブネットイベントです。したがって、同じサブネット上にクライアントとして少なくとも 1 つのサーバー (マスターまたはスレーブ) が存在しなければなりません。サーバーは、異なる複数のサブネット上に存在できます (マップはサブネット境界を超えて伝送されるため) 。サブネット環境での 1 つの一般的方法は、NIS サーバーとしてサブネットルーターを使用することです。この方法を使用すると、ドメインサーバーはどちらかのサブネットインタフェース上でクライアントにサービスを提供できます。

サーバーリストモード

サーバーリストモードでは、バインドプロセスは次のように動作します。

1. NIS マップで提供された情報を必要とする、NIS クライアントマシン上で動作しているプログラムが、ypbind にサーバー名を要求します。
2. ypbind が、/var/yp/binding/*domainname*/ypservers ファイルを調べてドメインの NIS サーバーリストを見つけます。
3. ypbind が、NIS サーバーリストの先頭サーバーへのバインドを開始します。先頭サーバーが応答しない場合は、ypbind はサーバーが見つかるまでまたは NIS サーバーリストの最後に達するまで 2 番目以降のサーバーへのバインドを順に試みます。
4. ypbind が、どのサーバーにトークすべきかをクライアントプロセスに通知します。次に、クライアントプロセスが直接、サーバーにリクエストを送信します。
5. NIS サーバー上の ypserv デーモンが、該当するマップを調べてリクエストを処理します。
6. ypserv デーモンが、要求された情報をクライアントに送り返します。

同報通信モード

同報通信モードでは、バインドプロセスは次のように動作します。

1. 同報通信オプション (broadcast) が設定されている状態で ypbind が起動されなければなりません。
2. ypbind が、RPC 同報通信を送出して NIS サーバーを探索します。

注 - このようなクライアントをサポートするには、NIS サービスを要求している各サブネット上に 1 つの NIS サーバーが存在する必要があります。

---

3. ypbind が、同報通信に応答する先頭サーバーへのバインドを開始します。
4. ypbind が、どのサーバーにトークすべきかをクライアントプロセスに通知します。次に、クライアントプロセスが直接、サーバーにリクエストを送信します。
5. NIS サーバー上の ypserv デーモンが、該当するマップを調べてリクエストを処理します。
6. ypserv デーモンが、要求された情報をクライアントに送り返します。

通常、いったんクライアントがサーバーにバインドされると、何らかの原因でバインドが解除されるまではクライアントはサーバーにバインドされたままになりま

す。たとえば、サーバーがサービスを提供できなくなると、このサーバーがサービスを提供していたクライアントは、新しいサーバーにバインドされます。

どの NIS サーバーが現在、特定クライアントにサービスを提供しているかを知りたい場合は、以下に示すコマンドを入力してください。

```
% ypwhich machinename
```

machinename は、クライアント名です。マシン名が指定されていない場合は、ypwhich はデフォルトとしてローカルマシン (コマンドが実行されるマシン) を使用します。

NIS 旧バージョンとの相違点

Solaris 8 リリース の NIS の特徴は、次のとおりです。

NSKit が存在しない

NIS サービスは、2.6 以前の Solaris リリースには組み込まれていませんでした。NIS サービスは今までは、個別販売される NSKit からインストールしなければなりませんでした。現在、NIS サービスは Solaris 8 リリースに組み込まれています。したがって、Solaris 7 以降のリリース には NSKit は存在しません。

Solaris 7 以降のリリースには NIS サービスが組み込まれたので、SUNWnsktu および SUNWnsktr パッケージはもはや存在しません。NIS のインストールは、NIS サーバークラスタにより行われています (NIS には SUNWypu および SUNWypr パッケージが含まれています) 。

NIS サービスは現在、/etc/init.d/yp スクリプトでは起動されません。/etc/init.d/yp スクリプトは現在、存在しません。Solaris 8 リリース では、まずマスターサーバーの NIS マップを ypinit スクリプトで作成し、次に ypstart で NIS を起動してください。NISサービスの停止は、ypstop コマンドで行われます。

ypupdated デーモン

ypupdated デーモンは、NSKit の 2.6 以前のバージョンには組み込まれていませんでしたが、Solaris 7 以降のリリース には組み込まれいます。

/var/yp/securenets

/var/yp/securenets ファイルは、Solaris 8 リリース でも Solaris 7 以前の NSKit リリースの場合と同様に、NIS サービスへのアクセスを制限するために使用されます。このファイルが NIS サーバーに存在する場合は、この NIS サーバーは照会に答えたり、ファイルに収められている IP アドレスのマシンおよびネットワークへのマップを与えたりするだけです。ファイルのフォーマットについては、securenets(4) のマニュアルページを参照してください。

securenets ファイルの例を以下に示します。

```
255.255.255.0 13.13.13.255
host    13.13.14.1
host    13.13.14.2
```

上記において 255.255.255.0 はネットマスクで、13.13.13.255 はネットワークアドレスです。1 行目のセットアップに関しては、ypserv はサブネットの 13.13.13.255 の範囲のこれらのアドレスにのみ応答します。/var/yp/securenets ファイルのエントリを変更したときは、ypserv と ypxfrd のデーモンを終了させて再起動をする必要があります。

マルチホームマシンのサポート

ypserv プロセスは、Solaris.8 リリース でも Solaris 7 以前の NSKit リリースの場合と同様に、複数のネットワークアドレスを持つマシンをサポートします。マシンマップが作成されると、Makefile は、複数のアドレスを持つマシンのマップに YP_MULTI_HOSTNAME エントリを作成します。このエントリには、そのマシンのすべてのアドレスがリストされます。マシンアドレスが必要な場合は、このリストに存在するアドレスのなかで、希望するアドレスに最も近いアドレスを使用しようとします。詳細は、ypserv(1) のマニュアルページを参照してください。

希望するアドレスに最も近いアドレスの判断は算術的判断なので、アドレスの妥当性検査は行われません。たとえば、マルチホームマシンが 6 つの IP アドレスを持っ

ているが、このマルチホームマシン上の 5 つのインタフェースだけが正常に動作していると仮定します。このマルチホームマシンに直接接続されていないネットワーク上のマシンは、ypserv からダウンインタフェースの IP アドレスを受け取ることができます。したがって、この仮説上のクライアントはマルチホームマシンにアクセスできません。

注 - マルチホームマシンのすべてのアドレスは、通常、アクティブでなければなりません。特定のアドレスまたはマシンでサービスが提供できなくなる恐れがある場合は、そのアドレスまたはマシンは NIS マップから削除してください。

SunOS 4.x 互換モード

Solaris 8 リリースの NIS は、パスワード構成ファイルを SunOS™ 4.x (Solarisリリース 1) フォーマットおよび Solaris リリース 2 のパスワードファイルフォーマットおよびシャドウファイルフォーマットの両方でサポートしています。

動作モードは、`$PWDIR/shadow` ファイルが存在するか否かによって決定されます (`$PWDIR` は、`/var/yp/Makefile` ファイルに設定されている `Makefile` マクロセットです) 。`shadow` ファイルが存在する場合は、NIS は Solaris リリース 2 モードで動作します。`shadow` ファイルが存在しない場合は、NIS は SunOS 4.x モードで動作します。

SunOS 4.x モードでは、すべてのパスワード情報は `passwd` ファイルに保存されています。Solaris リリース 2 モードでは、パスワード情報は `shadow` ファイルに保存され、ユーザー課金情報は `passwd` ファイルに保存されます。

`make` マクロ `PWDIR` が `/etc` ディレクトリに設定された場合は、Solaris リリース 2 の `passwd` 処理要件の関係上、NIS は Solaris リリース 2 モードでしか動作できません。しかし、`PWDIR` が `/etc` 以外のディレクトリに設定されている場合は、ユーザーは `passwd` 構成ファイルを SunOS 4.x フォーマットでも Solaris リリース 2 フォーマットでも保存できます。`rpc.yppasswdd` デーモンはこれら両方のパスワードフォーマットを認識しますが、Solaris リリース 2 フォーマットを使用することをお勧めします。

ネームサービススイッチの使用

ネームサービススイッチは、ネームサービス管理を単純化することを目的としています。クライアントマシンおよびアプリケーションは、このスイッチを使用して

ネームサービスを選択します。スイッチメカニズムは、/etc/nsswitch.conf ファイルを使って実現されます。このファイルは、各情報タイプを参照するために使用されるリソースを指定します。

この節では、NIS オペレーションに関するネームサービススイッチを正しく作成するために必要な要素についてだけ説明します。nsswitch.conf ファイルの詳細は、第 2 章を参照してください。

nsswitch.conf ファイルは、Solaris 8 リリースソフトウェアによって自動的に各マシンの /etc ディレクトリにロードされます。この際、以下に示す 3 つの代替 (テンプレート) バージョンも一緒にロードされます。

- /etc/nsswitch.nisplus
- /etc/nsswitch.nis
- /etc/nsswitch.files

これらの代替テンプレートファイルには、NIS+ サービス、NIS、ローカルファイルで使用されるデフォルトのスイッチ構成が入っています。(56ページの「nsswitch.conf テンプレートファイル」を参照してください。) DNS ではデフォルトのファイルは提供されませんが、これらの代替テンプレートファイルを編集することで DNS の使用が可能になります (60ページの「NIS クライアントでの DNS 転送」を参照)。

このスイッチ機能は SunOS 4.x 上には存在しません。したがって、4.x クライアントへの DNS 転送は、NIS サーバー上で行われなければなりません。この状況下で 4.x クライアントが、NIS サーバーの NIS マップにリストされていないホストに対して情報を要求した場合は、NIS サーバーはこの要求を DNS サーバーに転送します。

Solaris87 リリースソフトウェアがまずマシンにインストールされると、インストーラはマシンのデフォルトのネームサービス (NIS+、NIS、またはローカルファイル) を選択します。インストール時には、対応するテンプレートファイルが /etc/nsswitch.conf にコピーされます。NIS が使用されているクライアントマシンの場合は、インストール時には nsswitch.nis が nsswitch.conf にコピーされます。NIS データベースの設定が普通に行われていれば、NIS オペレーションにおいては、nsswitch.conf にコピーされたデフォルトの /etc/nsswitch.nis テンプレートファイルで十分です。

NIS 管理者は、クライアントマシンのネーミングシステム (/etc、NIS、または NIS+) を別のネーミングシステムに変更する場合は、対応するテンプレートファイルを nsswitch.conf にコピーしてください。NIS 管理者はまた、/etc/nsswitch.conf ファイルの該当行を編集することによって、クライア

ントが使用する特定タイプのネットワーク情報の発信元を変更できます。『*Solaris* ネーミングの設定と構成』、および本書の第 2 章を参照してください。

注意 - /etc/nsswitch.conf ファイルが files (nis ではない) に設定されており、サーバーが /etc/hosts ファイルに存在しない場合は、ypcat コマンドは次のようなエラーメッセージを出します。
「RPC failure: ‘‘RPC failure on yp operation’’」

第 19 章

NIS の管理

この章では、NIS の管理方法について説明します。

NIS の概要は、第 18 章を参照してください。

NIS の初期設定と構成方法については、『*Solaris* ネーミングの設定と構成』を参照してください。

パスワードファイルと名前空間のセキュリティ

セキュリティの関係上、次の点に注意してください

- マスターサーバーの NIS マップへのアクセスは制限します。
- 未許可アクセスを防止するためには、NIS パスワードマップの作成に使用されたファイルに root エントリを含めないでください。したがって、root エントリをこのパスワードファイルから削除して、このパスワードファイルをマスターサーバーの /etc ディレクトリ以外のディレクトリにおく必要があります。このディレクトリへの未許可アクセスは、防止しなければなりません。

たとえば、マスターサーバーのパスワード入力ファイルは、別のファイルへのリンクではなく Makefile に指定されている限り、/var/yp/ などのディレクトリに存在するか、または選択されたディレクトリに存在します。/usr/lib/netsvc/yp/ypstart スクリプトは、Makefile に指定された構成に従って自動的に適切なディレクトリオプションを設定します。

注 - この NIS インプリメンテーションでは、NIS パスワードマップを作成するための入力として、Solaris 1.x バージョンの passwd ファイルのフォーマットに加え、Solaris リリース 2 の passwd ファイルと shadow ファイルのフォーマットも使用できます。

NIS ユーザーの管理

この節では、ユーザーパスワードの設定、NIS ドメインへの新しいユーザーの追加、ネットグループへのユーザーの割り当てについて説明します。

NIS ドメインに新しいユーザーを追加する

NIS ドメインに新しいユーザーを追加するには、以下の手順に従ってください。

1. **NIS** マスターサーバーのルートとしてログインします。
2. useradd コマンドで新しいユーザーのログイン **ID** を作成します。

   Solarisリリース 2 のシステムの場合は、次のように入力します。

```
# useradd userID
```

userID は新しいユーザーのログイン ID です。このコマンドは、NIS マスターサーバー上の /etc/passwd ファイルと /etc/shadow ファイルにエントリを作成します。

3. 新しいユーザーの初期パスワードを作成します。

   新しいユーザーがログインするために使用できる初期パスワードを作成するには、passwd コマンドを以下の形式で実行します。

```
# passwd userID
```

   userID は新しいユーザーのログイン ID です。このユーザーに割り当てるパスワードを入力するよう指示が出ます。

   このステップが必要になるのは、useradd コマンドで作成されたパスワードエントリがロックされ、新しいユーザーがログインできないからです。初期パスワードを指定することで、このパスワードエントリのロックが解除されます。

4. 必要であれば、新しいエントリをマスターサーバーの passwd マップ入力ファイルにコピーします。

   マスターサーバー上のマップソースファイルが /etc 以外のディレクトリに存在する場合は、新しい行を /etc/passwd ファイルと /etc/shadow ファイルからマスターサーバー上の passwd マップ入力ファイルにコピー＆ペーストする必要があります。詳細は、384ページの「パスワードファイルと名前空間のセキュリティ」を参照してください。

   たとえば、新しいユーザー baruch を追加すると、/etc/passwd ファイルから passwd 入力ファイルにコピーされる行は次のようになります。

```
baruch:x:123:10:User baruch:/home/baruch:/bin/csh:
```

   /etc/shadow からコピーされる baruch 行は次のようになります。

```
baruch:W12345GkHic:6445::::::
```

注 - NIS マップの入力として Solaris リリース 1 の passwd ファイルフォーマットを使用している場合は、passwd ファイルへの新しいユーザーの追加は、テキストエディタを使用して手作業で行わなければなりません。

---

5. パスワード入力ファイルが存在するディレクトリを、Makefile で正しく指定していることを確認します。

6. 必要であれば、/etc/passwd ファイルと /etc/shadow 入力ファイルから新しいユーザーのエントリを削除します。

   セキュリティの関係上、NIS マスターサーバーの /etc/passwd ファイルと /etc/shadow 入力ファイルでユーザーエントリを保持することは望ましくありません。新しいユーザーのエントリを他のディレクトリに存在する NIS マップソースファイルにコピーした後、マスターサーバー上の userdel コマンドで新しいユーザーを削除します。

   たとえば、マスターサーバーの /etc ファイルから新しいユーザー baruch を削除するには、次のように入力します。

```
# userdel baruch
```

   userdel の詳細は、userdel(1M) のマニュアルページを参照してください。

7. **NIS** の **passwd** マップを更新します。

   マスターサーバー上の passwd 入力ファイルを更新した後、ソースファイルが存在するディレクトリで make を実行して passwd マップを更新します。

```
# userdel baruch
# cd /var/yp
# /usr/ccs/bin/make passwd
```

8. 新しいユーザーのログイン **ID** に割り当てられた初期パスワードを新しいユーザーに通知します。

   ログイン後、新しいユーザーはいつでも passwd を実行して別のパスワードに変更できます。

ユーザーパスワード

ユーザーは passwd を実行してパスワードを変更します。

```
% passwd username
```

(ユーザーの立場から見たパスワードの詳細は、208ページの「パスワードの使用」を参照してください。)

ユーザーがパスワードを変更する前に、NIS 管理者はマスターサーバー上の rpc.yppasswdd デーモンを起動してパスワードファイルを更新しなければなりません。rpc.yppasswd デーモンを起動するコマンドは、/usr/lib/netsvc/yp/ypstart ファイルにすでに存在しています。

rpc.yppasswdd デーモンは、自動的にマスターサーバー上の ypstart で起動されます。rpc.yppasswd に –m オプションが指定された場合は、ファイルが更新された直後に /var/yp の make が実行されます。passwd ファイルが更新されるたびにこの make が実行されることを回避したい場合は、ypstart スクリプトの rpc.yppasswd コマンドから –m オプションを削除して、crontab ファイルで passwd マップの転送を制御してください。

注 - rpc.yppasswd –m コマンドの後に引数を指定しないでください。別の動作を行わせるために ypstart スクリプトファイルを編集することは可能ですが、–m オプションを任意に削除すること以外の変更をこのファイルに加えることは望ましくありません。すべてのコマンドおよびデーモンは、一群の適切なコマンド行パラメータが存在するこのファイルで起動されます。このファイルを編集する場合は、rpc.yppasswdd コマンドの編集では特に注意してください。passwd.adjunct ファイルに明示的コールを追加する場合は、パスを $PWDIR/security/passwd.adjunct と正確に指定しなければなりません。$PWDIR/security/passwd.adjunct というパスを指定しない場合は、不適切な処理が行われます。

ネットグループ

NIS ネットグループは、NIS 管理者が管理目的のために定義する、ユーザーまたはマシンのグループ (集合) です。たとえば、次のようなネットグループを作成できます。

- 特定マシンにアクセスできる一群のユーザーを定義するネットグループ
- 特定のファイルシステムにアクセスできる一群の NFS クライアントマシンを定義するネットグループ
- 特定の NIS ドメインのすべてのマシンに対して管理者権限を持つ一群のユーザーを定義するネットグループ

各ネットグループには、1 つのネットグループ名が与えられます。ネットグループは、アクセス権を直接設定しません。代わりに、ユーザー名またはマシン名が一般に使用される場所では、ネットグループ名が他の NIS マップで使用されます。たとえば、netadmins というネットワーク管理者ネットグループを作成したと仮定します。netadmins ネットグループのすべてのメンバーに特定マシンへのアクセス権を与えるには、そのマシンの /etc/passwd ファイルに netadmin エントリを追加するだけで、ネットグループ名を /etc/netgroup ファイルに追加して、NIS グループマップに追加することもできます。ネットグループの使用の詳細は、netgroup(4) のマニュアルページを参照してください。

NIS が使用されているネットワーク上では、NIS マスターサーバー上の netgroup 入力ファイルを使用して、netgroup、netgroup.byuser、netgroup.byhost という 3 つのファイルが生成されます。netgroup マップには、netgroup 入力ファイルの基本情報が入っています。他の 2 つの NIS マップには、マシンまたはユーザーが指定されるとネットグループ情報の検索が迅速に行われるフォーマットで情報が入っています。

netgroup 入力ファイルのエントリのフォーマットは、*name ID* です。*name* はネットグループ名であり、*ID* は、ネットグループに属しているマシンまたはユーザー、あるいはその両方を示します。ネットグループの ID (メンバー) は、コンマで区切っていくつでも指定できます。たとえば、3 つのメンバーが存在するネットグループを作成する場合、netgroup 入力ファイルエントリのフォーマットは、*name ID*、*ID*、*ID*となります。netgroup 入力ファイルエントリのメンバー ID のフォーマットは、次のようになります。

```
(-|machine, -|user, domain)
```

machine はマシン名、*user* はユーザー ID、*domain* はマシンまたはユーザーの NIS ドメインで、それぞれコンマで区切られます。ドメインエレメントはオプションですが、他の NIS ドメインのマシンまたはユーザーを示す場合には必ず指定します。各エントリではマシンエレメントとユーザーエレメントは必須ですが、ダッシュ (-) は空であることを示すために使用されます。エントリでは、マシンエレメントとユーザーエレメントの関係を示す必要はありません。

netgroup 入力ファイルの 2 つのサンプルエントリを以下に示します。これらの各サンプルエントリでは、admins という名前のネットグループが作成されます。これらの各 admins は、リモートドメイン sales に存在するユーザー hauri と juanita、およびマシン altair と sirius で構成されます。

```
admins (altair, hauri), (sirius,juanita,sales)
admins (altair,-), (sirius,-), (-,hauri), (-,juanita,sales)
```

さまざまなプログラムでは、ログイン、リモートマウント、リモートログイン、リモートシェル作成時に NIS ネットグループマップを使用してアクセス権チェックを行います。さまざまなプログラムとは、mountd、login、rlogin、rsh などです。login コマンドは、passwd データベース内でネットグループ名を見つけた場合は、ネットグループマップでユーザー分類を調べます。mountd デーモンは、/etc/dfs/dfstab ファイル内でネットグループ名を見つけた場合は、ネットグループマップでマシン分類を調べます。rlogin と rsh (インタフェースを使用するプログラムならどれでも) は、/etc/hosts.equiv または .rhosts ファイル内でネットグループ名を見つけた場合は、ネットグループマップでマシン分類とユーザー分類の両方を調べます。

ネットワークに新しい NIS ユーザーまたはマシンを追加する場合は、netgroup 入力ファイルの該当ネットグループに追加してください。次に、make でネットグループマップを作成し、これを yppush コマンドですべての NIS サーバーに転送してください。ネットグループおよびネットグループ入力ファイル構文の詳細は、netgroup(4) のマニュアルページを参照してください。

NIS マップに関する作業

この節では、NIS マップの管理方法について説明します。

マップ情報の取得

マップ情報は、ypcat、ypwhich、ypmatch コマンドを使っていつでも取得できます。以下の例では、*mapname* はマップの正式名とニックネーム（存在する場合）の両方を意味します。

マップのすべての値を表示するには、次のように入力してください。

```
% ypcat mapname
```

マップのキーと値 (存在する場合) の両方を表示するには、次のように入力してください。

```
% ypcat -k mapname
```

マップのすべてのニックネームを表示するには、以下のコマンドのどれかを入力してください。

```
% ypcat -x
% ypwhich -x
% ypmatch -x
```

使用可能なすべてのマップとマスターサーバーを表示するには、次のように入力してください。

```
% ypwhich -m
```

特定マップのマスターサーバーを表示するには、次のように入力してください。

```
% ypwhich -m mapname
```

キーをマップのエントリと比較するには、次のように入力してください。

```
% ypmatch key    mapname
```

見つけたい項目がマップのキーでない場合は、次のように入力してください。

```
% ypcat mapname | grep item
```

item は、見つけたい情報です。他のドメインに関する情報を取得するには、これらのコマンドの –d domainname オプションを指定してください。

デフォルト以外のドメインの情報を要求するマシンが、そのドメインに対するバインドを持っていない場合は、このマシンは ypbind で /var/yp/binding/*domainname*/ypservers ファイルを参照して、そのドメインのサーバーリストを検索します。このファイルが存在しない場合は、ypbind は RPC 同報通信を送出してサーバーを検索します。この場合、検索先であるドメインのサーバーは、要求元マシンと同じサブネットに存在している必要があります。

マップのマスターサーバーの変更

選択されたマップのマスターサーバーを変更するには、まず新しい NIS マスターサーバー上にマップを作成しなければなりません。古いマスターサーバー名は既存のマップにキーと値のペアとして発生するので (このペアは makedbm で自動的に挿

入される)、ypxfr でマップを新しいマスターサーバーにコピーしたり、コピーを新しいマスターサーバーに転送するだけでは不十分です。キーと新しいマスターサーバー名との対応づけをし直す必要があります。マップに ASCII ソースファイルが存在する場合は、このファイルを新しいマスターサーバーにコピーしてください。

sites.byname というサンプル NIS マップを作成し直す手順を以下に示します。

1. 新しいマスターサーバーにスーパユーザーとしてログインし、次のように入力します。

```
newmaster# cd /var/yp
```

2. 作成するマップを指定する前に、Makefile にこの新しいマップのエントリが必要です。ない場合は、最初に Makefile を編集します。

3. マップを更新または再作成するには、次のように入力します。

```
newmaster# make sites.byname
```

4. 古いマスターサーバーが **NIS** サーバーとして残っている場合は、古いマスターサーバーにリモートログイン **(**rlogin**)** してから、Makefile を編集します。sites.byname を作成した Makefile 内のセクションをコメントアウトして、このセクションで sites.bynam が再び作成されないようにします。

5. sites.byname が ndbm ファイルとしてだけ存在している場合は、以下に示す makedbm で任意の **NIS** サーバーからコピーを取り出し、実行して sites.byname を新しいマスターサーバー上で作成し直します。

```
newmaster# cd /var/yp
newmaster# ypcat -k sites.byname | makedbm - domain /sites.byname
```

新しいマスターサーバー上でマップが作成されたら、そのコピーをこのマスターサーバーのスレーブサーバーに送信します。ただしこの場合、yppush を使用しないでください。yppush を使用すると、スレーブサーバーは新しいマスターサーバーからではなく古いマスターサーバーから新しいコピーを取得します。このような動作を回避するには、一般にマップのコピーを新しいマスターサーバー

から古いマスターサーバーに送り返すという方法が用いられます。この操作を行うには、古いマスターサーバーのスーパユーザーとなり、次のように入力します。

```
oldmaster# /usr/lib/netsvc/yp/ypxfr -h newmaster sites.byname
```

これで、yppush を使用できます。スレーブサーバーは、古いマスターサーバーを現行のマスターサーバーとして認識しているので、スレーブサーバーは、マップの現行のバージョンを古いマスターサーバーから取得しようとします。それから、スレーブサーバーは、新しいマスターサーバーが現行のマスターサーバーとして指定されている新しいマップを取得します。

この方法が失敗した場合は、各 NIS サーバーのルートとしてログインし、上記の ypxfr コマンドを実行してください。この方法は、面倒ですが必ず成功します。

構成ファイルの更新

NIS は、設定ファイルを正確に構文解析します。このため NIS 管理は容易になりますが、設定ファイルおよび構成ファイルにおける変更により、NIS の動作は影響を受けます。

ファイルには次のものがあります。

- /var/yp/Makefile

  このファイルは、サポートされているマップを追加または削除するために使用します。

- /etc/resolv.conf

  このファイルを追加または削除することで、DNS 転送が可能または不可になります。

- $PWDIR/security/passwd.adjunct

  このファイルを追加または削除することで、C2 セキュリティが可能または不可になります。$PWDIR は、/var/yp/Makefile で定義されます。

上記ファイルを更新するには、以下の手順に従ってください。

1. 次のように入力して **NIS** サーバーを停止します。

```
# /etc/init.d/yp stop
```

2. 必要に応じてファイルを変更します。

3. 次のように入力して **NIS** サーバーを再起動します。

```
# /etc/init.d/yp start
```

NIS のマップまたはマップソースファイルを更新する場合は、NIS を停止および起動する必要はありません。

以下の点に注意してください。

- NIS マスターサーバーからマップまたはソースファイルを削除しても、スレーブサーバー上の対応するマップまたはソースファイルは自動的には削除されません。スレーブサーバー上の対応するマップまたはソースファイルの削除は、NIS 管理者が手作業で行う必要があります。
- 新しいマップは、自動的には既存のスレーブサーバーに転送されません。新しいマップを既存のスレーブサーバーに転送するには、NIS 管理者がそのスレーブサーバーで `ypxfr` を実行してください。

Makefile の更新と使用

`/var/yp` で提供されたデフォルトの `Makefile` を更新することにより、NIS 管理者のニーズを満たすことができます。今後に備えて、必ずこのオリジナルの `Makefile` のコピーを保存しておいてください。`Makefile` を使用すると、マップの追加または削除、および一部のディレクトリ名の変更ができます。

新しい NIS マップを追加するには、マップの `ndbm` ファイルのコピーをドメインに存在する各 NIS サーバーの `/var/yp/`*domainname* ディレクトリに転送する必要があります。この転送は通常、`Makefile` が自動的に行います。どの NIS サーバーがマップのマスターサーバーであるかを決定したら、マップを容易に作成し直せるようにマスターサーバーの `Makefile` を更新してください。異なる複数のサーバーを異なる複数のマップのマスターサーバーとして設定することも可能ですが、このようにするとたいていの場合、管理上の混乱を招きます。したがって、1 つのサーバーだけをすべてのマップのマスターサーバーとして設定することをお勧めします。

一般に、人間が判読可能なテキストファイルは、`makedbm` に対する入力として適したものにするために `awk`、`sed`、`grep` でフィルタリングされます。デフォルトの `Makefile` を参照してください。`make` コマンドの概要については、`make(15)` のマニュアルページを参照してください。

make が認識する従属性の作成方法を決定する際には、Makefile にすでに備わっているメカニズムを使用してください。make では従属ルール内の行の始まりにタブが存在するか否かが重要であり、タブが存在しないというだけでエントリが無効になることがあるので注意してください。

Makefile エントリの追加

Makefile にエントリを追加するには、以下の作業を行ってください。

- データベース名を all ルールに追加します。
- time ルールを作成します。
- データベースのルールを追加します。

たとえば、Makefile をオートマウンタ入力ファイルで動作させるには、auto_direct.time および auto_home.time マップを NIS データベースに追加してください。

NIS データベースにこれらのマップを追加するには、以下の手順に従ってください。

1. all という語で始まる行に、追加したいデータベース名 **(1** つまたは複数**)** を追加します。

```
all: passwd group hosts ethers networks rpc services protocols \
netgroup bootparams aliases netid netmasks \
auto_direct auto_home auto_direct.time auto_home.time
```

   エントリの順序は任意ですが、継続行の始まりの空白はスペースではなくタブにしてください。

2. Makefile の終わりに以下の行を追加します。

```
auto_direct: auto_direct.time
auto_home: auto_home.time
```

3. ファイル中央に auto_direct.time エントリを追加します。

```
auto_direct.time: $(DIR)/auto_direct
@(while read L; do echo $$L; done < $(DIR)/auto_direct
$(CHKPIPE)) | \ (sed -e "/^#/d" -e "s/#.*$$//" -e "/^ *$$/d"
$(CHKPIPE)) | \ $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto_direct;
@touch auto_direct.time;
@echo "updated auto_direct";
@if [ ! $(NOPUSH) ]; then $(YPPUSH) auto_direct; fi
@if [ ! $(NOPUSH) ]; then echo "pushed auto_direct"; fi
```

- CHKPIPE は、次のコマンドに結果を渡す (パイピングする) 前に、パイプ (|) の左側の動作が正しく行われたことを確認します。パイプの左側の動作が正しく行われなかった場合は、NIS make terminated というメッセージが表示されてプロセスは終了します。
- NOPUSH は、Makefile が yppush を呼び出して新しいマップをスレーブサーバーに転送することを防止します。NOPUSH が設定されていない場合は、転送は自動的に行われます。

継続行の始まりにある while ループは、バックスラッシュで拡張された行を入力ファイルから削除するためのものです。sed スクリプトはコメント行および空行を削除し、makedbm に出力を渡します。

他のすべてのオートマウンタマップ (auto_home や他のデフォルトでないマップなど) でも、同じ手順が必要となります。

4. make を実行します。

```
# make name
```

name は、作成するマップ名です。たとえば、auto_direct などです。

Makefile エントリの削除

Makefile に特定データベースのマップを作成しない場合は、Makefile を以下のように編集してください。

1. all ルールからデータベース名を削除します。
2. 削除するデータベースのデータベースルールを削除またはコメントアウトします。

たとえば、hosts データベースを削除するには、hosts.time エントリを削除します。

3. **time** ルールを削除します。

   たとえば、hosts データベースを削除するには、hosts: hosts.time エントリを削除します。

4. マスターサーバーとスレーブサーバーからマップを削除します。

Makefile のマクロおよび変数の変更

Makefile の先頭で定義されている変数の設定値の変更は、等号 (=) の右側の値を変更するだけで行うことができます。たとえば、マップを作成するための入力として、/etc に存在するファイルではなく別のディレクトリに存在するファイル (たとえば、/var/etc/*domainname* など) を使用する場合は、DIR 値を DIR=/etc から DIR=/var/etc/*domainname* に変更してください。また、PWDIR 値を PWDIR=/etc から PWDIR=/var/etc/*domainname* に変更することもできます。

変数は次のとおりです。

- *DIR=*

  passwd と shadow を除くすべての NIS 入力ファイルが存在するディレクトリ。デフォルト値は /etc です。マスターサーバーの /etc ディレクトリのファイルを NIS 入力ファイルとして使用することは望ましくないので、この値は変更しなければなりません。

- *PWDIR*

  NIS 入力ファイル passwd と shadow が存在するディレクトリ。マスターサーバーの /etc ディレクトリのファイルを NIS 入力ファイルとして使用することは望ましくないので、この値は変更しなければなりません。

- *DOM*

  NISドメイン名。DOM のデフォルト値は、domainname コマンドで設定されます。大部分の NIS コマンドでは現在のマシンのドメイン（現在のマシンの /etc/defaultdomain ファイルに設定されている）が使用されることに注意してください。

既存のマップの更新

NIS のインストール終了後、頻繁に更新しなければならないマップとまったく更新する必要がないマップがあることに気づくかもしれません。たとえば、passwd.byname マップは、大企業のネットワークでは頻繁に更新されることがあります。一方、auto_master マップはまったくではないにしてもほとんど更新されません。

マップを更新する必要がある場合は、マップがデフォルトのマップか否かによって 2 つの更新手順のどちらかを使用できます。

- デフォルトのマップは、ネットワークデータベースから ypinit で作成されたデフォルトのセットに存在するマップです。
- デフォルトでないマップは、以下のどれかです。
  - ベンダーから購入したアプリケーションと共に提供されたマップ
  - ユーザーサイト用に特別に作成されたマップ
  - テキスト以外のファイルから作成されたマップ

この節では、さまざまな更新ツールの使用方法について説明します。実際上、これらの更新ツールは、システム起動後にデフォルトでないマップを追加する場合、または一群の NIS サーバーを変更する場合にだけ使用できます。

デフォルトのマップの更新

デフォルトのセットと共に提供されたマップを更新するには、以下の手順に従ってください。

1. マスターサーバーのルートになります。

   必ずマスターサーバーだけの NIS マップを更新します。

2. 更新するマップのソースファイルを編集します (このファイルが /etc に存在しているか、選択された他のディレクトリに存在しているかは問題ではありません)。

3. 次のように入力します。

```
# cd /var/yp# make mapname
```

make コマンドは、対応するファイルに対して NIS 管理者が行った変更に従ってマップを更新します。make コマンドはまた、これらの変更を他のサーバーに反映します。

デフォルトでないマップの更新

デフォルトでないマップを更新するには、以下の手順に従ってください。

1. 対応するテキストファイルを作成または編集します。
2. 新しいマップまたは更新されたマップを作成 (または再作成) します。マップ作成には 2 つの方法があります。
   - Makefile を使用する方法

     デフォルトでないマップを作成するには、この方法を用います。Makefile にマップのエントリが存在する場合は、make *name* を実行するだけです (*name* は作成するマップ名)。Makefile にマップのエントリが存在しない場合は、393ページの「Makefile の更新と使用」を参照してエントリを作成をしてください。
   - /usr/sbin/makedbm プログラムを使用する方法

     このコマンドの詳細は、makedbm(1M) のマニュアルページで説明されています。

デフォルトでないマップを makedbm で更新する

入力ファイルが存在しない場合は、makedbm でマップを更新する方法は 2 つあります。

- makedbm –u の出力先を一時ファイルに変更し、一時ファイルを更新し、更新済みの一時ファイルを使用します。
- makedbm –u の出力を makedbm に渡されるパイプライン内で動作させます。分解されたマップを awk、sed、または cat で更新できる場合は、この方法を用います。

新しいマップの作成

新しいマップを作成するには、入力として既存のテキストファイルを使用する方法、または入力として標準入力を使用する方法のどちらかを使用できます。

テキストファイルからマップを作成する

テキストファイル /var/yp/mymap.asc がマスターサーバー上のエディタまたはシェルスクリプトで作成されていると仮定します。この場合、このファイルから NIS マップを作成し、作成された NIS マップを *homedomain* サブディレクトリに入れるには、マスターサーバー上で次のように入力してください。

```
# cd /var/yp
# makedbm mymap .asc   homedomain/mymap
```

mymap マップは現在、マスターサーバーの *homedomain* ディレクトリに存在しています。この新しいマップをスレーブサーバーに転送するには、ypxfr を実行してください。

ファイルをベースとしたマップにエントリを追加する

mymap にエントリを追加することは簡単です。最初に、対応するテキストファイルを更新せずに実際の dbm ファイルを更新すると、更新は反映されないので、テキストファイル /var/yp/*mymap*.asc を更新します。次に、上記のような makedbm を実行してください。

標準入力からマップを作成する

オリジナルのテキストファイルが存在しない場合は、キーボードから makedbm に次のように入力して NIS マップを作成してください (最後に Control-D と入力)。

```
ypmaster# cd /var/yp
ypmaster# makedbm - homedomain/mymap    key1 value1 key2 value2   key3 value3
ypmaster#
```

標準入力から作成されたマップを更新する

後でマップを更新する必要がある場合は、makedbm でマップを取り出し、一時ファイルを作成できます。マップを分解し、一時ファイルを作成するには、次のように入力してください。

```
% cd /var/yp
% makedbm -u homedomain/ mymap > mymap.temp
```

作成される一時ファイル *mymap*.temp には、1 行につき 1 つのエントリが存在します。このファイルは、任意のテキストエディタで必要に応じて編集できます。

マップを更新するには、次のように入力して、更新後の一時ファイル名を makedbm につけます。

```
% makedbm mymap.temp  homedomain/mymap
% rm mymap.temp
```

次に、ルートになり次のように入力してマップをスレーブサーバーに反映させます。

```
# yppush mymap
```

ここでは makedbm でマップを作成する方法について説明してきましたが、実際に行わなければならないほとんどすべての作業は、ypinit と Makefile で行うことができます。ただし、システム起動後にデフォルトでないマップをデータベースに追加したり一群の NIS サーバーを変更しない場合に限ります。

/var/yp の Makefile を使用しても他の手順を使用しても、正しく作成された dbm ファイルの新しいペアをマスターサーバー上の maps ディレクトリに入れなければなりません。

NISマップを反映させる

マップが更新されると、Makefile が yppush で新しいマップをスレーブサーバーに転送させます (ただし、NOPUSH が Makefile に設定されていない場合)。転送させるための動作手順は次のとおりです。スレーブサーバー上の ypserv デーモンにマップ転送リクエストを送信します。ypserv デーモンが ypxfr プロセスを起動します。ypxfrd プロセスがマスターサーバー上の ypxfr デーモンに連絡します。いくつかの基本検査 (マップが実際に更新されていることの確認など) が行われます。マップが転送されると、スレーブサーバー上の ypxfrd が、転送が成功したことを yppush プロセスに通知します。

注 - 上記手順は、新しく作成されたマップがスレーブサーバー上に存在しない場合は動作しません。新しいマップを、スレーブサーバー上の ypxfr でスレーブサーバーに転送する必要があります。

マップ転送は失敗することがありますが、失敗した場合は ypxfr を使って手作業で新しいマップ情報を転送してください。ypxfr は、2 つの方法で使用できます。1 つはルートの crontab ファイルを定期的に使用する方法であり、もう 1 つはコマンド行から対話形式で使用する方法です。これらの方法については、以下で説明します。

cron でマップ転送を行う

マップの更新頻度は、マップによってそれぞれ異なります。たとえば、デフォルトのマップである protocols.byname やデフォルトでないマップの auto_master など一部のマップは何カ月も更新されないことがありますが、一方、passwd.byname など一部のマップは 1 日に数回更新されることがあります。crontab コマンドでマップ転送をスケジュールすると、個々のマップに対して特定の更新時間を設定できます。

マップに適切な頻度で ypxfr を定期的に実行するには、各スレーブサーバー上のルートの crontab ファイルに、該当する ypxfr エントリを入れる必要があります。ypxfr は、マスターサーバー上のコピーがローカルのコピーより新しい場合に限り、マスターサーバーと連絡をとり、マップを転送します。

注 - デフォルトの –m オプションが指定されている rpc.yppasswdd をマスターサーバー上で実行すると、誰かが yp パスワードを変更するたびに、passwd デーモンが make を実行して passwd マップを作成し直します。

---

cron と ypxfr でシェルスクリプトを使用する

NIS 管理者は、各マップに対する crontab エントリを個々に作成するという方法ではなく、ルートの crontab コマンドでシェルスクリプトを実行してすべてのマップを定期的に更新するという方法を使用することもできます。マップ更新シェルスクリプトのサンプルは、/usr/lib/netsvc/yp ディレクトリに入っています。スクリプト名は、ypxfr_1perday、ypxfr_1perhour、ypxfr_2perday です。これらのシェルスクリプトを、更新または置換することによって、容易にサイト要件に適合させることができます。例 19–1 は、デフォルトの ypxfr_1perday シェルスクリプトを示しています。

例 **19–1**　ypxfr_1perday シェルスクリプト

```
#! /bin/sh
#
# ypxfr_1perday.sh - YP マップの検査・更新を毎日行う
PATH=/bin:/usr/bin:/usr/lib/netsvc/yp:$PATH
export PATH
# set -xv
ypxfr group.byname
ypxfr group.bygid
ypxfr protocols.byname
ypxfr protocols.bynumber
ypxfr networks.byname
ypxfr networks.byaddr
ypxfr services.byname
ypxfr ypservers
```

このシェルスクリプトは、ルートの `crontab` が毎日実行されると、マップを 1 日に 1 回更新します。NIS 管理者は、1 週間に 1 回、1 ヶ月に 1回、または 1 時間に 1 回などといった頻度でマップを更新するスクリプトを使用することもできますが、マップを頻繁に更新すると性能が低下するので注意してください。

NIS ドメインに対して構成された各スレーブサーバー上で同じシェルスクリプトを root で実行してください。各サーバー上の実行時間を変更して、マスターサーバーが動作不能にならないようにしてください。

特定のスレーブサーバーからマップを転送する場合は、シェルスクリプトの `ypxfr` の `–h` *machine* オプションを使用してください。シェルスクリプトに記述するコマンドの構文は、次のとおりです。

```
/usr/lib/netsvc/yp/ypxfr -h machine [ -c ] mapname
```

machine は、転送するマップが存在するサーバー名です。*mapname* は、要求されたマップ名です。マシンを指定することなく `–h` オプションを指定すると、`ypxfr` はマスターサーバーからマップを取得しようとします。`ypxfr` 実行時に `ypserv` がローカルに実行されていない場合は、`ypxfr` がローカル `ypserver` に現在のマップリクエストの取消しを送信しないよう、`–c` フラグを使用してください。

`–s` *domain* オプションを使用すると、別のドメインからローカルドメインにマップを転送できます。これらのマップは、各ドメインにおいて同じでなければなりません。たとえば、2 つのドメインが同じ `services.byname` マップおよび `services.byaddr` マップを共有することがあります。また、より細かい制御を行うための `rcp` または `rdist` を使用してファイルを複数のドメインに転送することもできます。

ypxfr を直接起動する

2 番目の方法である ypxfr の起動とは、ypxfr をコマンドとして実行することです。一般に、ypxfr をコマンドとして実行するのは例外的状況においてだけです。たとえば、一時的に NIS サーバーを設定して試験環境を作成する場合や、他のサーバーと調和して動作不能になっていた NIS サーバーを迅速に取得しようとする場合などです。

ypxfr のアクティビティを記録する

ypxfr が試みた転送およびその転送結果は、ログファイルに記録されます。/var/yp/ypxfr.log というファイルが存在する場合は、転送結果はこのファイルに記録されます。このログファイルのサイズを制限する試みは行われません。このログファイルのサイズが無限に大きくなることを防止するには、ときどき次のように入力してこのログファイルを空にしてください。

```
# cd /var/yp
# cp ypxfr.log ypxfr.log.old
# cat /dev/null > /var/yp/ypxfr.log
```

これらのコマンドは、crontab で週間に 1 回実行させることができます。記録をオフにするには、ログファイルを削除してください。

新しいスレーブサーバーを追加する

NIS 実行後、ypinit に与えられた初期リストに含まれていなかった新しいスレーブサーバーを追加する必要があるかもしれません。

新しいスレーブサーバーを追加するには、以下の手順に従ってください。

1. ルートとしてマスターサーバーにログインします。
2. 次のように入力して **NIS** ドメインディレクトリに移ります。

```
# cd /var/yp/domainname
```

3. 次のように入力して ypservers ファイルを変換します。

```
# makedbm -u ypservers >/tmp/temp_file
```

makedbm コマンドは、ypservers を ndbm フォーマットから /tmp/temp_file (一時 ASCII ファイル) に変換します。

4. テキストエディタで /tmp/temp_file ファイルを編集します。つまり、新しいスレーブサーバー名をサーバーリストに追加します。この後、/tmp/temp_file ファイルを保存し、閉じます。

5. /temp_file で入力ファイルとして makedbm コマンドを実行し、出力ファイルとして ypservers を実行します。

```
# makedbm /tmp/temp_file ypservers
```

makedbm は、ypservers を変換して ndbm フォーマットに戻します。

6. ypservers の **ASCII** ファイルは存在しないので、次のように入力して ypservers マップが適切なものであることを確認します。

```
slave3# makedbm -u ypservers
```

makedbm コマンドは、画面に ypservers の各エントリを表示します。

注 - ypservers にマシン名が存在しない場合は、ypservers はマップファイルの更新を受信しません。これは、yppush がこのマップでスレーブサーバーリストを調べるからです。

---

7. マスターサーバーから一群の **NIS** マップをコピーして新しいスレーブサーバーの **NIS**ドメインディレクトリを設定します。

この操作を行うには、スーパユーザーとして新しい NIS スレーブサーバーにログインし、ypinit および ypbind コマンドを実行します。

```
slave3# cd /var/yp
slave3# ypinit -c list of servers
```

(続く)

```
slave3# /usr/lib/netsvc/yp/ypbind
```

8. このマシンをスレーブサーバーとして初期化するには、次のように入力します。

```
# /usr/sbin/ypinit -s ypmaster
```

ypmaster は、既存の NIS マスターサーバーのマシン名です。

9. ypstop を実行して、**NIS** クライアントとして実行されているマシンを停止します。

```
# /usr/lib/netsvc/up/ypstop
```

10. ypstart を実行して **NIS** スレーブサーバーのサービスを開始します。

```
# /usr/lib/netsvc/up/ypstart
```

NIS スレーブサーバーの設定の詳細は、『*Solaris* ネーミングの設定と構成』を参照してください。

C2 セキュリティが装備されている NIS の使用

$PWDIR/security/passwd.adjunct ファイルが存在する場合は、C2 セキュリティが自動的に起動されます。$PWDIR は、/var/yp/Makefile で定義されます。C2 セキュリティモードでは、passwd.adjunct ファイルで passwd.adjunct NIS マップが作成されます。C2 セキュリティモードでは、passwd.adjunct ファイルと shadow ファイルの両方を使用してセキュリティを管理できます。passwd.adjunct ファイルは、次のように入力された場合にだけ処理されます。

```
# make passwd.adjunct
```

make passwd コマンドは、NIS 管理者が C2 セキュリティモードで make を手作業で実行した場合にだけ passwd マップ (passwd.adjunct マップではない) を処理します。

マシンの NIS ドメインの変更

マシンの NIS ドメイン名を変更するには、以下の手順に従ってください。

1. マシンの /etc/defaultdomain ファイルを編集して、現在の内容をマシン用の新しいドメイン名に置き換えます。

   たとえば、現在のドメイン名である sales.doc.com を research.doc.com に変更したりします。

2. domainname 'cat /etc/defaultdomain' を実行します。

3. マシンを NIS クライアント、スレーブサーバー、またはマスターサーバーとして設定します。

   詳細は、『*Solaris* ネーミングの設定と構成』参照してください。

NIS を DNS と一緒に使用する

一般に NIS クライアントは、マシン名とアドレスの検索に NIS だけが使用されるように、nsswitch.conf ファイルで構成されます。このような検索が失敗した場合は、NIS サーバーはこれらの結果を DNS に転送します。

マシン名とアドレスの検索が最初に NIS で行われ、次に DNS で行われるように構成するには、以下の手順に従ってください。

1. 2 つのマップ (hosts.byname と hosts.byaddr) に YP_INTERDOMAIN キーが必要です。 このキーを設定するには、Makefile を編集します。つまり、Makefile ファイルの先頭部分の行を次のように変更します。

```
#B=-b
B=
```

を

```
B=-b
#B=
```

に変更

この変更が行われると、マップ作成時に makedbm が –b フラグで起動されるよう要求されて、また YP_INTERDOMAIN キーが ndbm ファイルに挿入されます。

2. make を実行して上記マップを作成し直します。

```
# /usr/ccs/bin/make hosts
```

3. 有効な名前のサーバーを指定している /etc/resolv.conf ファイルが **NIS** サーバーに存在することを確認します。

4. **DNS** 転送を行うには、各サーバーを ypstop コマンドで停止します。

```
# /usr/lib/netsvc/yp/ypstop
```

5. 各サーバーを ypstart コマンドで再起動します。

```
# /usr/lib/netsvc/yp/ypstart
```

この NIS インプリメンテーションでは、サーバーに /etc/resolve.conf ファイルが存在する場合は、ypstart が –d オプションで自動的に ypserv デーモンを起動して DNS にリクエストを転送します。

注 - Solaris リリース 2 が実行されていない NIS サーバーを使用している場合は、参照される DNS のホストマップに YP_INTERDOMAIN キーが存在することを確認してください。

混在 NIS ドメインにおける問題

これまでの説明の大部分では、NISドメインのマスターサーバーとスレーブサーバーの両方で Solaris リリース 2 が実行されていることが前提となっています。したがって、それ以外の場合には、問題が発生することがあります。混在 NIS ドメインにおける問題を回避する方法については、表 19–1 にまとめてあります。"4.0.3+" という表記は、「SunOS のリリース 4.0.3 以降」であることを意味します。makedbm –b コマンドは、Makefile の "–B" 変数に対する参照です。

表 19–1 混在 NIS ドメインにおける NIS/DNS

スレーブサーバー	マスターサーバー		
	4.0.3+		**Solaris NIS**
4.0.3+	マスターサーバー： makedbm -b スレーブサーバー: ypxfr	マスターサーバー： makedbm -b スレーブサーバー: ypxfr -b	マスターサーバー： ypserv -d スレーブサーバー: ypxfr -b
Solaris NIS	マスターサーバー： makedbm -b スレーブサーバー: ypxfr	マスターサーバー： makedbm -b スレーブサーバー: ypxfr	マスターサーバー： ypserv -d スレーブサーバー： ypxfr が存在する resolve.conf または ypxfr -b

NIS サービスをオフにする

マスターサーバー上の ypserv が使用不可になっている場合は、NIS マップを更新できません。ネットワーク上の NIS をオフにする場合は、リブート後に次のように入力して ypbind ファイルを ypbind.orig に名前を変更するだけで NIS を使用不可にできます。

```
% mv /usr/lib/netsvc/yp/ypbind /usr/lib/netsvc/yp/ypbind.orig
```

リブート後に特定の NIS スレーブサーバーまたはマスターサーバー上の NIS を使用不可にする場合は、その特定の NIS スレーブサーバーまたはマスターサーバー上で次のように入力してください。

```
% mv /usr/lib/netsvc/yp/ypserv /usr/lib/netsvc/yp/ypserv.orig
```

NIS を直ちに停止するには、次のように入力してください。

```
% /usr/lib/netsvc/yp/ypstop
```

NIS サービスは、リブートが行われると自動的に再起動されます。ただし、ypbind および ypserv ファイルで上記のような名前の変更が行われない場合です。

NIS の問題解決とエラーメッセージ

- NIS の問題解決については、661ページの「NIS の問題と対策」および 631ページの「NIS+ と NIS の互換性の問題」を参照してください。
- 一般的な名前空間エラーメッセージのアルファベット順のリスト、およびこれらのエラーメッセージの意味については、付録 B を参照してください。

パート V　FNS の管理

パート V では、フェデレーテッド・ネーミング・サービス (FNS) とその管理方法について説明します。

- 第 20 章
- 第 21 章
- 第 22 章
- 第 23 章
- 第 24 章
- 第 25 章
- 第 26 章
- 第 27 章

第 20 章

FNS の手引き

この章では、FNS の概要、設定と構成の手順に関する簡単な説明、およびプログラミングの例を示します。熟練の管理者であれば、この章を読むだけで必要なことがすべてわかります。

詳細は、パート V「FNS の管理」の他の章を参照してください。FNS の初期設定と構成の詳細は、『*Solaris* ネーミングの設定と構成』を参照してください。

フェデレーテッド・ネーミング・サービス (FNS)

フェデレーテッド・ネーミング・サービス (FNS) は、ネーミングとディレクトリ操作のための 1 つの単純なインタフェースのもとに、複数のネームサービスをフェデレートする方法を提供します。FNS によってリンクできるネームサービスには、NIS+、NIS、ファイルベース、DNS、および X.500/LDAP があります。

XFN (X/Open Federated Naming)

FNS がサポートするプログラミングインタフェースとそのポリシーは、XFN (X/Open Federated Naming) に述べられています。

FNS を使用する理由

FNS は、次の点で有用です。

- 単一の一貫したネーミングおよびディレクトリインタフェースがクライアントに対して用意されている。これを使用してネーミングおよびディレクトリサービスにアクセスできる。したがって、新しいネーミングおよびディレクトリサービスを追加しても、アプリケーションや既存のサービスに変更を加える必要はない
- 名前を一貫した方法で作成できる。FNS は、異なる複数のネーミングシステムから一貫した名前を作成する方法を定義する。これにより、アプリケーションは、これらの異なる複数のネーミングシステム内のオブジェクトを一定の方法でアドレス指定できる
- 共有コンテキストと共有名の使用によって、一貫したネーミングを行うことができる。異なる複数のアプリケーションでこれらの共有名と共有コンテキストを使用すると、作業を重複して行う必要がなくなる

複合名と複合コンテキスト

FNS の基本概念として、複合名と複合コンテキストがあります。

複合名

複合名とは、複数のネーミングシステムで使用される名前のことをいいます。

複合名は、複数の構成要素の順序付きリストからなります。各構成要素は、単一のネーミングシステムの名前空間からとられた名前です。各構成要素の構文は、個々のネーミングシステムにより異なります。FNS は、複数のネーミングシステムからとられた名前を使用して複合名を作成するときの構文を定義します。複合名は、スラッシュ (/) を構成要素区切り文字として使用して、左から右へ作成されます。

たとえば、複合名 `.../doc.com/site/bldg-5.alameda`
は、`...`、`doc.com`、`site`、`bldg-5.alameda` という 4 つの構成要素から構成されます。

コンテキスト

コンテキストは、次の操作を行います。

- 名前をオブジェクトに関連付ける (割り当てる)
- 名前をオブジェクトとして解釈処理する
- 割り当てを削除する
- 名前を表示する
- 名前を変更する
- 名前付きオブジェクトに属性を関連付ける
- 名前付きオブジェクトに関連付けられた属性を取り出して更新する
- 属性を使用してオブジェクトを検索する

コンテキストには、一連の名前とリファレンスの割り当てが含まれます。各リファレンスには、通信の終端またはアドレスのリストが含まれます。フェデレーテッド・ネーミング・システムは、別のネーミングシステムのコンテキストで割り当て中の、あるネーミングシステムのコンテキストによって形成されます。複合名の解

釈処理は、その名前全体が解釈処理されるまで、あるネーミングシステム内のコンテキストから、次のネーミングシステム内のコンテキストへと進みます。

属性

名前付きオブジェクトには、属性を適用できます。属性はオプションです。名前付きオブジェクトには、属性を付けないことも、1 つまたは複数の属性を付けることもできます。

各属性は、一意の属性識別子、属性構文、ゼロまたはいくつかの属性値の集合を持ちます。

XFN は、既存の名前付きオブジェクトに関連付けられた属性値を確認、変更するための基本属性インタフェースを定義します。これらのオブジェクトは、コンテキストまたは他の任意のタイプのオブジェクトにできます。コンテキストには、コンテキストによる合成名の構文解析方法を記述する構文属性が関連付けられます。

拡張属性インタフェースには、特定の属性を検索する操作と、オブジェクトとその関連属性を作成する操作があります。

エンタープライズネームサービス

エンタープライズレベルのネームサービスは、あるエンタープライズ内のオブジェクトに名前を付けるために使用されます。FNS は現在、次の 3 つのレベルのネームサービスをサポートしています。

- NIS+ (416ページの「NIS+」の項を参照)
- NIS (417ページの「NIS」の項を参照)
- ファイル (418ページの「ファイルベース」の項を参照)

NIS+

NIS+ は、Solaris 8 リリース環境で推奨されるエンタープライズ規模の情報サービスです。FNS の組織単位は、NIS+ のドメインとサブドメインに対応します。各ドメインとサブドメインごとに 1 つの `orgunit` コンテキストがあります。

NIS+ のもとで、FNS のコンテキストと属性データは、NIS+ テーブルに格納されます。これらのテーブルは、ctx_dir という名前の NIS+ ディレクトリオブジェクトに格納されます。各 NIS+ ドメインとサブドメインごとに 1 つの ctx_dir ディレクトリオブジェクトがあり、ドメインの groups_dir および org_dir の各ディレクトリオブジェクトと同じレベルにあります。したがって、ディレクトリオブジェクト ctx_dir.sales.doc.com. には、sales.doc.com. ドメインの FNS コンテキストと属性データを格納する FNS テーブルが含まれます。

NIS+ のもとでは、FNS と NIS+ のコマンドを使用して、FNS テーブル内の情報を処理します。これらのテーブルを直接編集したり、UNIX コマンドを使用して処理したりしないでください。

NIS

NIS は、Solaris 環境でのエンタープライズ規模の情報サービスです。各エンタープライズは 1 つの NIS ドメインにあたります。1 つの NIS ドメインに対応する 1 つの FNS 組織単位があります。

NIS のもとで、FNS のコンテキストと属性データは NIS マップに格納されます。これらのマップは、NIS サーバー上の /var/yp/*domainname* ディレクトリに格納されます。NIS では、スーパーユーザーは、FNS コマンドを使用して、FNS マップ内の情報を処理できます。

SKI が実行されている場合に NIS クライアントが FNS でコンテキストを更新する

一定の条件が満たされれば、NIS クライアント (マシン、プロセス、またはユーザー) はすべて、fncreate_fs、fncreate_printer などの FNS コマンドを使用して、クライアント独自のコンテキストを更新できます。これにより、NIS クライアントは、FNS コマンドを使用して、Printer Administrator、CDE カレンダマネージャ、admintool などのアプリケーションを更新できます。

スーパーユーザー以外のユーザーが、FNS コマンドを使用して独自のコンテキストを更新するには、次の条件が必要です。

- NIS マスターサーバー上で SKI (Secure Key_management Infrastructure) が使用可能
- fnsypd デーモンが、NIS マスターサーバー上で実行されている。このデーモンは、スーパーユーザー特権を持つユーザーが開始する

- クライアントユーザーまたはマシンは、独自のコンテキストの更新だけできる
- クライアントは、要求された更新を実行するための権限を持っている

注 - SKI は 64 ビットモードをサポートしていません。したがって NIS クライアントは 64 ビットモードでのコンテクストの更新は行うことができません。

ファイルベース

「ファイル」とは、通常、マシンの `/etc` ディレクトリにあるネーミングファイルを指します。これらのマシンベースファイルには、UNIX のユーザーとパスワード情報、ホスト情報、メール別名などが入っています。これらのファイルは、自動マウントマップなどの Solaris 特定のデータもサポートしています。

ファイルベースのネーミングシステムでは、FNS コンテキストと属性データはファイルに格納されます。これらの FNS ファイルは、マシンの `/var/fn` ディレクトリに格納されます。(`/var/fn` ディレクトリを各マシンに置く必要はありません。このディレクトリは、NFS ファイルサーバーからエクスポートされます。)

ファイルネーミングシステムでは、FNS コマンドを使用して FNS ファイルの情報を処理します。

グローバルネームサービス

FNS は、DNS と X.500 による NIS+ と NIS のフェデレートもサポートします。つまり、エンタープライズレベルの名前空間をグローバル名前空間に接続し、エンタープライズオブジェクトをグローバルな有効範囲でアクセス可能にできるということです。

FNS は現在、次のグローバルネームサービスをサポートしています。

- DNS
- X.500 (DAP または LDAP を経由)

FNS ネーミングポリシー

FNS は、ネーミングポリシーを定義して、ユーザーとアプリケーションが共有名前空間に依存し、それを使用できるようにします。

エンタープライズ内には、組織単位、サイト、ホスト、ユーザー、ファイルとサービスごとの名前空間があり orgunit、site、host、user、fs (ファイルシステムを示す)、および service という名前で呼ばれます。これらの名前空間は、各名前の前に下線 (_) を付けることもできます。たとえば、host と _host は同じものと見なされます。

表 20–1 は、エンタープライズレベルの名前空間に関する FNS ポリシーを要約しています。

表 20–1 FNS ポリシーの要約

コンテキストタイプ	従属コンテキスト	親コンテキスト
orgunit _orgunit	site user host fs service	enterprise root
site _site	user host fs service	enterprise root orgunit
user _user	service fs	enterprise root orgunit
host _host	service fs	enterprise root orgunit
service _service	プリンタなどのアプリケーション	enterprise root orgunit site user host
fs _fs (ファイルシステム)	(なし)	enterprise root orgunit site user host

組織名

FNS orgunit の割り当ては、基本となるネームサービスによって決まります。

- NIS+ では、組織単位は NIS+ のドメインまたはサブドメインに対応する。たとえば、NIS+ のルートドメインが doc.com. で、doc.com. のサブドメインが sales だとした場合、FNS 名の org/sales.doc.com. と org/sales は、どちらも NIS+ ドメイン sales.doc.com. に対応する組織単位を指す。(sales.doc.com. の最後のドットは、完全指定 NIS+ 名に必要であることに注意)
- NIS では、組織単位は NIS ドメインであり、必ず FNS 名 org// または org/*domainname* によって識別される。ここで、*domainname* は、doc.com. などの完全指定ドメイン名を示す。NIS の組織単位名には階層はない
- ファイルベースのネーミングシステムで、組織単位は、必ず FNS 名 org// によって識別されるシステムである

組織単位名に対応して命名されるオブジェクトのタイプは、user、host、service、fs、site です。次に例を示します。

- org/sales/site/conference1.bldg-6 は、組織単位 sales に関連するサイトの 6 号ビルにある会議室 conference1 を指定する。この例では、org/sales が sales.doc.com. に対応する場合、このオブジェクトを org/sales.doc.com./site/conference1.bldg-6 と指定することもできる (sales.doc.com. の最後のドットに注意)
- org/finance/user/mjones は、組織単位 finance のユーザー mjones を指定する
- org/finance/host/inmail は、組織単位 finance に属するマシン inmail を指定する
- org/accounts.finance/fs/pub/reports/FY92-124 は、組織単位 accounts.finance に属するファイル pub/reports/FY92-124 を指定する
- org/accounts.finance/service/calendar は、組織単位 accounts.finance のカレンダサービスを指定する。これは、組織単位のミーティングのスケジュールを管理する

サイト名

サイト名は、必要に応じて作成されます。サイト名に対応して指定されるオブジェクトのタイプは、user、host、service、fs です。次に例を示します。

- site/alameda/user/mjones は、サイト alameda にあるユーザー mjones を指定する
- site/alameda/host/sirius は、サイト alameda にあるマシン sirius を指定する
- site/alameda/service/printer/Sparc-2 は、サイト alameda にあるプリンタ Sparc-2 を指定する
- site/alameda/fs/usr/dist は、サイト alameda で使用可能なファイルディレクトリ usr/dist を指定する

ユーザー名

ユーザー名は、NIS+ の対応する passwd テーブル、NIS の passwd マップ、またはファイルの /etc/passwd ファイルにある名前に対応します。ユーザーのファイルコンテキストは、そのユーザーの passwd エントリから獲得されます。

ユーザー名に対応して指定されるオブジェクトのタイプは、service と fs です。次に例を示します。

- user/chou/service/fax は、ユーザー chou のファックスサービスを指定する
- user/esperanza/fs/projects/conf96.doc は、ユーザー esperanza のファイルシステムの projects サブディレクトリにあるファイル、conf96.doc を指定する

ホスト名

ホスト名は、NIS+ の対応する hosts、NIS の hosts マップ、またはファイルの /etc/hosts ファイルにある名前に対応します。ホストのファイルコンテキストは、ホストによってエクスポートされるファイルシステムに対応します。

ホスト名に対応して指定されるオブジェクトのタイプは、service と fs です。次に例を示します。

- host/smtp-1/service/mailbox は、マシン smtp-1 に関連するメールボックスサービスを指定する
- host/deneb/fs/etc/.cshrc は、ホスト deneb 上の /etc ディレクトリにあるファイル .cshrc を指定する

サービス名

サービス名は、サービスアプリケーションに対応し、それによって決定されます。サービスコンテキストは、組織、ユーザー、ホスト、またはサイトコンテキストに対応して指定する必要があります。次に例を示します。

- `org//service/printer` は、組織のプリンタサービスを指定する
- `host/deneb/service/printer` は、マシン `deneb` に関連するプリンタサービスを指定する
- `host/deneb/service/printer/Sparc-2` は、マシン `deneb` に関連するプリンタを指定する
- `user/charlie/service/calendar` は、ユーザー `charlie` のカレンダサービスを指定する
- `site/conf_pine.bldg-7.alameda/service/calendar` は、`Alameda` サイトの 7 号ビルにある `conf_pine` 会議室のカレンダサービスを指定する

ファイル名

ファイルシステム名は、ファイル名に対応します。次に例を示します。

- `host/altair/fs/etc/.login` は、マシン `altair` 上の `.login` ファイルを指定する
- `user/prasad/fs/projects/96draft.doc` は、ユーザー `prasad` の `projects` ディレクトリにあるファイル `96draft.doc` を指定する

開始前に必要な処置

各自のネームサービスで FNS を開始するには、`fncreate` コマンドを実行します。

`fncreate` コマンドは、FNS コンテキストが作成されるネームサービス (NIS+、NIS、ファイルベースなど) を認識します。特定のネームサービスを指定するには、下記の 423ページの「デフォルト以外のネームサービスの指定」で説明する、`fnselect` コマンドを実行する必要があります。

デフォルト以外のネームサービスの指定

デフォルトでは次のようになります。

- `fncreate` を NIS+ クライアントまたはサーバーであるマシン上で実行すると、FNS 名前空間が NIS+ に設定されます。FNS NIS+ マスターサーバーとして別のマシンを指定する必要がある場合は、『*Solaris* ネーミングの設定と構成』を参照してください。
- マシンが NIS クライアントの場合、名前空間は NIS で設定されます。
- マシンが上記のどちらでもない場合、名前空間はマシンで `/var/fn` ディレクトリに設定されます。基本となるネームサービスがファイルベースの場合、各マシンで `fncreate` を実行することによって `/var/fn` を作成する点は共通です。ただし、一方のマシンで `/var/fn` を作成し、NFS によってそれをエクスポートして、別のクライアントによってマウントできます。

`fnselect` コマンドを使用すると、デフォルト以外のターゲットネームサービスを明確に指定することもできます。たとえば、次のコマンドは、ターゲットネームサービスに NIS を選択します。

```
# fnselect nis
```

FNS 名前空間の作成

デフォルトポリシー、または `fnselect` によって明示的にネームサービスを指定したら、次のコマンドを使用して、FNS 名前空間を作成できます。

```
# fncreate -t org org//
```

このコマンドは、対応するネームサービス内のユーザーとホストに必要なすべてのコンテキストを作成します。

NIS+ についての考慮事項

各自のエンタープライズレベルの基本ネームサービスが NIS+ の場合は、次の点を考慮に入れてください。

NIS+ のドメインとサブドメイン

上記のコマンド構文は、ルート NIS+ ドメインの FNS 名前空間を作成します。ルート以外のドメインを指定するには、次のように、2 つのスラッシュの間にドメイン名を追加します。

```
# fncreate -t org org/sales.doc.com./
```

完全指定の sales.doc.com. の最後のドットに注意してください。

空間とパフォーマンスの考慮事項

fncreate コマンドは、ctx_dir ディレクトリに NIS+ のテーブルとディレクトリを作成します。ctx_dir ディレクトリオブジェクトは、ドメインの NIS+ の groups_dir と org_dir ディレクトリオブジェクトと同じレベルにあります。

- 大きなドメインでは、NIS+ サーバー上で必要な追加スペースが多く、大きなインストール環境では、FNS と標準 NIS+ の各テーブルに個別のサーバーを使用することによって、パフォーマンスを改善できる場合があります。FNS と NIS+ に個別のサーバーを使用する方法については、『*Solaris* ネーミングの設定と構成』を参照してください。
- 大きいドメイン、または重要なドメインでは、FNS サービスを複製する必要があります。FNS サービスの複製方法については、『*Solaris* ネーミングの設定と構成』を参照してください。

NIS+ のセキュリティ要件

fncreate などの FNS コマンドを実行するユーザーは、必要な NIS+ 資格を持つことが前提とされます。

環境変数 NIS_GROUP は、fncreate によって作成された NIS+ オブジェクトのグループ所有者を指定します。NIS+ オブジェクトの管理を容易にするために、NIS_GROUP には、そのドメインの FNS 管理を担当する NIS+ グループ名を設定してから、fncreate などの FNS コマンドを実行する必要があります。

デフォルトのアクセス制御権を含む NIS+ 関連属性は、コンテキストの作成後、NIS+ の管理ツールやインタフェースを使用して変更できます。FNS 複合名に対応する NIS+ オブジェクト名は、後で説明する fnlookup および fnlist によって獲得できます。

NIS についての考慮事項

`fncreate` コマンドは、FNS マップの NIS マスターサーバーとして機能する NIS システム上のスーパーユーザーが実行してください。

FNS によって使用される NIS マップは、`/var/yp/`*domainname* に保存されています。

FNS NIS マスターサーバー上のスーパーユーザーだけが、FNS コマンドを使用して FNS 情報を変更できます。

ファイルについての考慮事項

`fncreate` に `–t org` オプションを付けて FNS 名前空間を作成する場合は、`/var` が存在するファイルシステムを所有するマシン上のスーパーユーザーが、コマンドを実行してください。FNS が使用するファイルは、`/var/fn` ディレクトリに格納されています。

ユーザーのコンテキストを作成すれば、ユーザーは、各自の UNIX 資格に基いて、各自のコンテキストを変更できます。

ファイルシステム `/var/fn` は、エクスポートすることにより、別のシステムでマウントして、FNS 名前空間にアクセスできます。

FNS 名前空間の表示

名前空間を設定したら、次のコマンドを使用して表示できます。

- `fnlist` - コンテキストの内容を表示 (425ページの「コンテキストの内容の表示」を参照)
- `fnlookup` - 複合名の割り当てを表示 (426ページの「複合名の割り当ての表示」を参照)
- `fnattr` - 複合名の属性を表示 (427ページの「複合名の属性の表示」を参照)

コンテキストの内容の表示

次の `fnlist` コマンドは、*name* のコンテキスト名とリファレンスの割り当てを表示します。

```
fnlist [-lvA] [name]
```

表 20–2 fnlist コマンドのオプション

オプション	説明
name	複合名。*name* のコンテキストでの名前の割り当てを表示する
–v	詳細。割り当ての詳細を表示する
–l	指定のコンテキストで割り当てられた名前の割り当ても表示する
–A	fnlist で、権限のあるサーバーからの情報を強制的に獲得する。NIS と NIS+ で、このサーバーはドメインのマスターサーバーになる。–A オプションは、基本ネームサービスがファイルベースの場合、無効である

次に例を示します。

初期コンテキストの名前をリスト表示する場合

```
% fnlist
```

現在の組織単位内のユーザーすべての詳細をリスト表示する場合

```
% fnlist -v user
```

ユーザー pug の service コンテキストの内容をリスト表示する場合

```
% fnlist user/pug/service
```

権限のあるサーバーからの名前と割り当てをリスト表示する場合

```
% fnlist -l -A
```

複合名の割り当ての表示

fnlookup コマンドは、指定の複合名の割り当てを表示します。

```
fnlookup [-vAL] [name]
```

表 **20–3** fnlookup コマンドのオプション

オプション	説明
name	コンテキスト名。*name* の割り当てと XFN リンクを表示する
–v	詳細。割り当てをより詳しく表示する
–L	名前に割り当てられている XFN リンクを表示する
–A	fnlist で、権限のあるサーバーからの情報を強制的に獲得する。NIS と NIS+ で、これは、ドメインのマスターサーバーになる。–A オプションは、基本ネームサービスがファイルベースの場合、無効である

たとえば、user/ana/service/printer の割り当てを表示するには、次のコマンドを実行します。

```
# fnlookup user/ana/service/printer
```

複合名の属性の表示

fnattr コマンドは、指定の複合名の属性を表示、更新します。

たとえば、ユーザー ada に関連する属性を検索するには、次のコマンドを実行します。

```
# fnattr user/ada
```

プリンタ laser-9 に関連する属性を検索するには、次のコマンドを実行します。

```
# fnattr thisorgunit/service/printer/laser-9
```

詳細は、436ページの「属性の処理」を参照してください。

FNS 情報の検索

fnsearch コマンドは、属性が所定の検索基準を満たす複合名以下で割り当てられたオブジェクト名を表示し、必要に応じてその属性とリファレンスを表示します。

たとえば、realname という属性を持つユーザーとその属性をリスト表示するには、次のコマンドを実行します。

```
% fnsearch user realname
```

値が Ravi Chattha の属性 realname を持つユーザーをリスト表示するには、次のコマンドを実行します。

```
% fnsearch user ``realname == 'Ravi Chattha'''
```

fnsearch コマンドは、共通のブール型演算子を使用します。上記の例では、二重引用符と一重引用符、2 つの等号の使用に注意してください。

名前空間の更新

名前空間を設定したら、次のコマンドを使用して、要素の追加、削除、および変更できます。

- fnbind - 複合名に新しいリファレンスを割り当てる (429ページの「複合名へのリファレンスの割り当て」を参照)
- fnunbind - 割り当てを削除する (431ページの「割り当ての削除」を参照)
- fncreate - 新しい組織、ユーザー、ホスト、サイト、サービスコンテキストを作成する (431ページの「新しいコンテキストの作成」を参照)
- fncreate_fs - 新しいファイルシステムコンテキストを作成する (433ページの「ファイルコンテキストの作成」を参照)
- fncreate_printer - 新しいプリンタコンテキストを作成する (434ページの「プリンタコンテキストの作成」を参照)
- fndestroy - コンテキストを削除する (436ページの「コンテキストの削除」を参照)
- fnattr - 属性を表示、作成、変更、削除する (436ページの「属性の処理」を参照)
- fncopy - あるネームサービスから別のネームサービスへ FNS コンテキストと属性をコピーする (438ページの「FNS コンテキストのコピーと変換」を参照)

FNS 管理特権

FNS システム管理は、基本となるネームサービスによって異なります。

- 「NIS+」では、FNS システム管理タスクは、その権限を持つユーザーだけが実行できます。システム管理特権を付与する通常の方法では、NIS+ グループを作成し、そのドメインに必要な特権をそのグループに割り当てます。これにより、そのグループのメンバーはすべて、システム管理機能を実行できるようになります。
- 「NIS」では、FNS 管理タスクは、NIS マスターサーバー上の `root` で実行しなければなりません。
- 「ファイルベース」のネーミングシステムでは、FNS 管理タスクは、`/var/fn` ディレクトリへの `root` アクセス権を持つユーザーが実行しなければなりません。

ユーザーが各自のサブコンテキストを変更できるかどうかは、基本となるネームサービスによって異なります。

- 「NIS+」では、ユーザーのコンテキストと関連サブコンテキストは、ユーザーが所有します。NIS+ ポリシーでログインした場合、適切な資格と特権を持つユーザーは、`fncreate`、`fnbind`、`fnunbind` などのコマンドを使用して、各自のコンテキストを変更できます。
- 「NIS」では、ユーザーはどの FNS データも変更できません。NIS マスターサーバーへの `root` アクセス権を持つユーザーだけが、FNS データを変更できます。
- 「ファイルベース」のネーミングシステムでは、ユーザーは独自のコンテキストを所有します。標準の UNIX アクセス制御が FNS ファイルに適用されます。

複合名へのリファレンスの割り当て

`fnbind` コマンドは、既存のリファレンス (名前) を新しい複合名に割り当てるために使用されます。

```
fnbind -r [-s][-v][-L] name [-O|-U]  newname reftype addrtype  [-c|-x] address
```

表 20–4 `fnbind` コマンドのオプション

オプション	説明
name	既存の複合名
newname	新しい割り当ての複合名
addrtype	使用するアドレスのタイプ。`onc_cal_str` のようにアプリケーション特定

表 **20–4** fnbind コマンドのオプション 続く

オプション	説明
address	使用するアドレスの内容。たとえば、tsvi@altair
reftype	使用するリファレンスのタイプ。one_calendar のようにアプリケーション特定
–s	すでに割り当てられている場合でも、*newname* に割り当てられる。これは、以前の *newname* の割り当てを置き換える。–s を指定しないと、*newname* がまだ割り当てられていない場合、fnbind は失敗する
–v	*newname* に割り当てられるリファレンスを表示する
–L	*oldname* を使用して XFN リンクを作成し、それを *newname* に割り当てる
–r	コマンド行引数によって作成されたリファレンスに *newname* を割り当てる
–c	入力された形式で *address* の内容を格納する。XDR コードは使用しない
–x	*address* を XDR コードに変換しないで、16 進文字列に変換する
–O	識別子の形式は FN_ID_ISO_OID_STRING。これは、ASN.1 のドットで区切った整数リスト文字列である
–U	識別子の形式は FN_ID_DCE_UUID。これは、文字列形式の DCE UUID である

次に例を示します。

ユーザー jamal にカレンダ割り当てを追加する場合

```
# fnbind -r user/jamal/service/calendar onc_calendar onc_cal_str jamal@cygnus
```

org//service/Sparc-4 の既存の割り当てを、org//service/printer の割り当てによって置換する場合

```
# fnbind -s org//service/printer org//service/Sparc-4
```

site/bldg-5/service/printer のリファレンスを user/ando/service/printer にコピーする場合

```
# fnbind site/bldg-5/service/printer user/ando/service/printer
```

シンボリックリンクを使用して、site/bldg-5/service/printer のリファレンスを user/ando/service/printer に割り当てる場合

```
# fnbind -L site/bldg-5/service/printer user/ando/service/printer
```

staff@altair が onc_cal タイプのリファレンスであり、かつタイプ onc_cal_str のアドレスである場合、thisens/service/calendar の名前をアドレス staff@altair に割り当てる場合

```
# fnbind -r thisens/service/calendar onc_calendar onc_cal_str staff@altair
```

コマンド行 *address* によって作成されたリファレンスとして *newname* を割り当てる場合

```
# fnbind -r [-sv] newname [-O|-U] reftype {[-O|-U] addrtype [-c|-x] address}
```

割り当ての削除

fnunbind コマンドは、割り当てを削除するために使用されます。

たとえば、user/jsmith/service/calendar の割り当てを削除するには、次のコマンドを実行します。

```
# fnunbind user/jsmith/service/calendar
```

新しいコンテキストの作成

fncreate コマンドは、コンテキストを作成するために使用されます。

```
fncreate -t context [-f file] [-o] [-r reference] [-s] [-v] [-D] name
```

表 **20–5** fncreate コマンドのオプション

オプション	説明
–t *context*	*context* タイプのコンテキストを作成する。*context* タイプは、org、hostname、host、username、user、service、fs、site、nsid、generic
–f *file*	入力ファイルを使用して、コンテキストを作成するユーザーとホストを一覧表示する
–r *reference*	リファレンスのタイプ。–r *reference* オプションは、–t generic とともにしか使用できない
name	複合名
–o	*name* によって識別されるコンテキストだけを作成する
–s	既存の割り当てすべてを上書き (置換) する。–s を使用しないと、名前がすでに割り当てられている場合、fncreate は失敗する
–D	各コンテキストの作成時に、そのコンテキストと、対応するテーブル、ディレクトリ、ファイルに関する情報を表示する
–v	詳細。各テキストの表示時にその情報を表示する

次に例を示します。

ルート組織のコンテキストとサブコンテキストを作成する場合

```
# fncreate -t org org//
```

ホスト deneb のコンテキストとサブコンテキストを作成する場合

```
# fncreate -t host host/deneb
```

コンテキスト、サービス、ファイルサブコンテキストを作成して、ユーザー sisulu のカレンダ割り当てを追加する場合

```
# fncreate -t user user/sisulu
# fnbind -r user/sisulu onc_calendar onc_cal_str sisulu@deneb
```

sales 組織のサイトコンテキストを作成する場合

```
# fncreate -t site org/sales/site/
```

サイトコンテキストは、ドットで区切られた右から左へという順番の名前によって、階層名前空間をサポートします。これにより、地理上のカバレージ関係によってサイトを区分化できます。たとえば、サイトコンテキスト alameda とサイトサブコンテキスト bldg-6.alameda を作成するには、次のコマンドを実行します。

```
# fncreate -t site org/sales/site/alameda
# fncreate -t site org/sales/site/bldg-6.alameda
```

ファイルコンテキストの作成

- fncreate_fs コマンドは、コマンド行から入力された割り当て記述によって、組織とサイトのファイルコンテキストを作成します。

  ```
  fncreate_fs [-r] [-v] name [options] [mount]
  ```

- fncreate_fs コマンドは、入力ファイルから提供された割り当ての記述によって、組織とサイトのファイルコンテキストを作成します。

  ```
  fncreate_fs [-r] [-v] -f file  name
  ```

表 20–6 fncreate_fs コマンドのオプション

オプション	説明
name	ファイルコンテキスト名
options	マウントオプション
mount	マウント位置
–f *file*	入力ファイル
–v	詳細。作成中のコンテキストに関する情報を表示
–r	コンテキスト *name* の割り当てを、入力で指定された割り当てによって置換する

表 **20–6**　fncreate_fs コマンドのオプション　続く

次に例を示します。

FNS サーバー server4 の /export/data パスに割り当てられた sales 組織に関して、ファイルシステムコンテキスト data を作成する場合

```
# fncreate_fs org/sales/fs/data server4:/export/data
```

2 つの異なるサーバーにマウントされた buyers および buyers/orders という名前の sales 組織に関して、ファイルシステムコンテキストの階層を作成する場合

```
# fncreate_fs org/sales/fs/buyers server2:/export/buyers
# fncreate_fs org/sales/fs/buyers/orders server3:/export/orders
```

input_a という名前の入力ファイルによって指定されたサーバーとパスに割り当てられた sales 組織に関して、leads という名前のファイルシステムコンテキストを作成する場合

```
# fncreate_fs -f input_a org/sales/fs/leads
```

(入力ファイル形式については、fncreate_fs(1M) のマニュアルページを参照してください。)

プリンタコンテキストの作成

fncreate_printer コマンドは、組織、ユーザー、ホスト、サイトの各コンテキストのプリンタコンテキストを作成します。プリンタコンテキストは、対応する複合名のサービスコンテキストのもとで作成されます。

```
fncreate_printer [-vs] name printer [prntaddr]
```

```
fncreate_printer [-vs] [-f [file]] name
```

表 20–7 fncreate_printer コマンドのオプション

オプション	説明
name	プリンタの組織、ホスト、ユーザー、またはサイトの名前
printer	プリンタ名
prntaddr	プリンタアドレス。<addresstype>=<address> という形式をとる
–f *file*	指定の ファイル を、作成するプリンタリストへの入力として使用。入力ファイルは /etc/printers.conf ファイルの形式をとる。プリンタ名 も –f ファイル も指定されていない場合、fncreate_printer は、fncreate_printer がデフォルト入力ファイルとして実行されるマシン上の /etc/printer.conf ファイルを使用する
–s	既存のアドレスを、同じアドレスタイプによって置換する
–v	詳細。割り当ての詳細を表示する

次に例を示します。

fncreate_printer が実行されるマシンの /etc/printers.conf ファイルに示されたプリンタに基いて sales 組識のプリンタを作成する場合

```
# fncreate_printer -s org/sales/
```

マシン altair が、プリンタ Sparc-5 のサーバーであると想定します。ユーザー nguyen に対して、実際に Sparc-5 プリンタである invoices という名前のプリンタを作成する場合、次のように入力します。

```
# fncreate_printer user/nguyen invoices bsdaddr=altair,Sparc-5
```

プリンタを階層構造で構成することもできます。たとえば、fncreate_printer コマンドは、プリンタ color、color/inkjet、color/Sparc のプリンタコンテキストを作成できます。コンテキストは次のようになります。

```
org/doc.com/service/printer/color
org/doc.com/service/printer/color/inkjet
org/doc.com/service/printer/color/Sparc
```

上記のコンテキストを作成するには、次のコマンドを実行します。

```
# fncreate_printer org/doc.com color bsdaddr=colorful,color
# fncreate_printer org/doc.com color/inkjet bsdaddr=colorjet,inkjet
# fncreate_printer org/doc.com color/Sparc bsdaddr=colorprt,Sparc
```

コンテキストの削除

fndestroy コマンドは、空のコンテキストを削除するために使用されます。

たとえば、ユーザー patel の service コンテキストを削除するには、次のコマンドを実行します。

```
# fndestroy user/patel/service
```

属性の処理

fnattr コマンドは、名前に関連する属性の追加、削除、または変更に使用できます。変更は一度に 1 つずつ行うことも、同じコマンド内で何回かバッチ処理することもできます。

- fnattr [–l] *name*

  name の属性をリスト表示

- fnattr *name* –a–s –U –O *attrib values*

  属性の追加

- fnattr *name* –m –O –U *attrib oldvalue newvalue*

  属性の変更

- fnattr *name* –d –O | –U [*values attrib*]

  属性の削除

表 20–8 fnattr コマンドのオプション

オプション	説明
name	複合名
attrib	属性の識別子

表 **20–8** fnattr コマンドのオプション 続く

オプション	説明
values	1 つまたは複数の属性値
oldvalue	新しい値によって置換される属性値
newvalue	古い値を置換する属性値
`-a`	属性の追加
`-d`	属性の削除
`-l`	属性のリスト表示
`-m`	属性の変更
`-s`	指定された属性の新しい値によって、すべての古い属性値を置換する
`-O`	識別子の形式は `FN_ID_ISO_OID_STRING`。これは、ASN.1 のドットで区切られた整数リスト文字列である
`-U`	識別子の形式は `FN_ID_DCE_UUID`。これは、文字列形式での DCE UUID である

次に例を示します。

ユーザー名 `rosa` に関連する属性すべてを表示する場合

```
# fnattr user/rosa
```

ユーザー `uri` に関連する `size` 属性を表示する場合

```
# fnattr user/uri/ size
```

`devlin` という名前のユーザーについて、値が `small` の属性 `shoesize` を追加するには、`hatsize` 属性を削除して、`dresssize` 属性の値を `12` から `8` に変更します。

```
# fnattr user/devlin -a shoesize small -d hatsize -m dresssize 12 8
```

グローバル名前空間のフェデレート

NIS+ または NIS は、DNS や X.500 などのグローバルネームサービスにフェデレートできます。

NIS+ または NIS の名前空間を DNS または X.500 のもとでフェデレートするには、まず NIS+ 階層または NIS ドメインのルートリファレンスを獲得する必要があります。

グローバルネームサービスでは、ルートリファレンスは、「次のネーミングシステムリファレンス」として知られています。これは、このリファレンスが、DNS ドメインまたは X.500 エントリの下にある次のネーミングシステムを指すためです。NIS+ または NIS をグローバルネームサービスによってフェデレートするには、そのグローバルサービスに、ルートリファレンス情報を追加します。

ルートリファレンス情報をグローバルサービスに追加すると、各自の NIS+ 階層または NIS ドメインの外部のクライアントは、その NIS+ 階層または NIS ドメインのコンテキストにアクセスして、操作を実行できます。外部 NIS+ クライアントは、未認証 NIS+ クライアントとしてその階層にアクセスします。

次に例を示します。

NIS+ が、DNS ドメイン `doc.com.` の下でフェデレートされる場合は、次のコマンドを使用して NIS+ エンタープライズのルートをリスト表示できます。

```
# fnlist .../doc.com/
```

NIS+ が、X.500 エントリ `/c=us/o=doc` の下でフェデレートされる場合は、次のコマンドを使用して NIS+ エンタープライズのルートをリスト表示できます。

```
# fnlist .../c=us/o=doc/
```

どちらの例でも、最後にスラッシュが必要であることに注意してください。

FNS コンテキストのコピーと変換

`fncopy` コマンドは、FNS のコンテキストと属性を新しい FNS コンテキストにコピー、または変換するために使用できます。

–i および –o オプションを使用すると、あるエンタープライズレベルのネームサービスに基く FNS コンテキストを、異なるネームサービスに基くコンテキストにコピーできます。たとえば、NIS の上部で実行される FNS インストール環境があって、NIS サービスを NIS+ にアップグレードする場合、fncopy を使用すると、NIS+ を使って新しいコンテキストを作成できます。

次の点に注意してください。

- 古いコンテキストをコピー中の新しい FNS コンテキストが、そのターゲットネームサービスにすでに存在する場合は、新しいコンテキストと割り当てだけがコピーされる。これらのコンテキストは上書きも変更もされない
- fncopy は、リンクに従わないが、名前に割り当てられた FNS リンクを、新しいコンテキストの名前空間にコピーする

表 20–9 fncopy コマンドのオプション

オプション	説明
–i *oldservice*	エンタープライズレベルの古い (入力) 基本ネームサービス。たとえば、–i nis は、古いサービスが NIS であることを指定する。指定できる値は、files、nis、nisplus
–o *newservice*	エンタープライズレベルの新しい (出力) ネームサービス。たとえば、-o nisplus は、新しいサービスが NIS+ であることを指定する。指定できる値は、files、nis、nisplus
–f *filename*	コピーされる FNS コンテキストのリストが入っているテキストファイル。–i および –o オプションを指定しない場合、コンテキストは、グローバル名を使用する識別子でなければならない
oldcontext	コピーされるコンテキスト名
newcontext	作成先またはコピー先のコンテキスト名

たとえば、doc.com プリンタコンテキスト (およびサブコンテキスト) と割り当てを orgunit/east/doc.com にコピーするには、次のコマンドを実行します。

```
# fncopy .../doc.com/service/printer .../doc.com/orgunit/east/service/printer
```

ファイル user_list に指定された NIS FNS ユーザーのコンテキストを、orgunit west/doc.com の NIS+ FNS ユーザーのコンテキストにコピーするには、次のコマンドを実行します。

```
# fncopy -i nis -o nisplus -f /etc/user_list thisorgunit/user org/doc.com/user
```

名前空間ブラウザのプログラムの例

この節のプログラミング例は、次の操作を実行するための XFN API の使用法を示しています。

- 440ページの「コンテキストに割り当てられた名前のリスト表示」
- 441ページの「バインディングの作成」
- 443ページの「オブジェクト属性のリスト表示と処理」
- 444ページの「オブジェクト属性の追加、削除、変更」
- 446ページの「コンテキスト内のオブジェクトの検索」

コンテキストに割り当てられた名前のリスト表示

次の例は、コンテキストをリスト表示するための XFN 操作を示しています。

```
#include <stdio.h>
#include <xfn/xfn.h>
#include <string.h>
#include <stdlib.h>
/*
 このルーチンは与えられたコンテキスト (ctx_name) の下で
 割り当てられた名前のリストを返す。
 ctx_name の例としては "user"、"thisorgunit/service"、
 host/alto/service、user/jsmit/service/calendar などがある
*/
typedef struct fns_listing {
 char *name;
 struct fns_listing *next;
} fns_listing;
fns_listing *
fns_list_names(const char *ctx_name)
{
 FN_status_t *status;
 FN_ctx_t *initial_context;
```

(続く)

```
 FN_composite_name_t *context_name;
 FN_namelist_t *name_list;
 FN_string_t *name;
 unsigned int stat;
 fns_listing *head = 0, *current, *prev;
 int no_names = 0;
 status = fn_status_create();
 /* 初期コンテキストの獲得 */
 initial_context = fn_ctx_handle_from_initial(0, status);
 if (!fn_status_is_success(status)) {
  fprintf(stderr, "Unable to obtain intial context\n");
  return (0);
 }
 context_name = fn_composite_name_from_str((unsigned char *)
                    ctx_name);
 /* FNS によるリスト名の呼び出し */
 name_list = fn_ctx_list_names(initial_context, context_name,
              status);
 if (!fn_status_is_success(status)) {
  fprintf(stderr, "Unable to list names\n");
  return (0);
 }
 /* 名前を個々に呼び出す */
 while (name = fn_namelist_next(name_list, status)) {
  no_names++;
 current = (fns_listing *) malloc(sizeof(fns_listing));
 current->name = (char *)
  malloc(strlen((char *) fn_string_str(name, &stat)) + 1);
 strcpy(current->name, (char *) fn_string_str(name, &stat));
 current->next = 0;
 if (head) {
  prev->next = current;
  prev = current;
 } else {
  head = current;
  prev = current;
 }
 fn_string_destroy(name);
}
fn_namelist_destroy(name_list);
fn_status_destroy(status);
fn_ctx_destroy(initial_context);
return (head);
```

バインディングの作成

次の例は、バインディングの作成方法を示しています。

例 **20–1**　バインディングの作成

```
#include <stdio.h>
#include <xfn/xfn.h>
#include <string.h>
/*
 このルーチンは "name" によって提供される名前を用いて
 バインディングを作成する。提供される名前のリファレンスのタイプは
 "reference_type" で、アドレスのタイプは "address_type"である。
 関数の使用例として
  fns_create_bindings(
  "user/jsmith/service/calendar"、
  "onc_calendar"、
  "onc_cal_str"、
  "jsmith&calserver");
 がある
*/
int fns_create_bindings(
 char *name,
 char *reference_type,
 char *address_type,
 char *data)
{
 int return_status;
 FN_composite_name_t *binding_name;
 FN_identifier_t ref_id, addr_id;
 FN_status_t *status;
 FN_ref_t *reference;
 FN_ref_addr_t *address;
 FN_ctx_t *initial_context;
 /* 初期コンテキストの獲得 */
 status = fn_status_create();
 initial_context = fn_ctx_handle_from_initial(0, status);
 /* エラーメッセージの状態のチェック */
 if ((return_status = fn_status_code(status)) != FN_SUCCESS) {
  fprintf(stderr, "Unable to obtain the initial context\n");
  return (return_status);
 }
 /* プリンタ名に付ける複合名の獲得 */
 binding_name = fn_composite_name_from_str((unsigned char *) name);
 /* アドレスのコンストラクト */
 addr_id.format = FN_ID_STRING;
 addr_id.length = strlen(address_type);
 addr_id.contents = (void *) address_type;
 address = fn_ref_addr_create(&addr_id,
  strlen(data), (const void *) data);
 /* リファレンスのコンストラクト */
 ref_id.format = FN_ID_STRING;
 ref_id.length = strlen(reference_type);
 ref_id.contents = (void *) reference_type;
 reference = fn_ref_create(&ref_id);
 /* リファレンスにアドレスを追加する */
 fn_ref_append_addr(reference, address);

 /* 割り当ての作成 */
 fn_ctx_bind(initial_context, binding_name, reference, 0, status);
```

(続く)

```
 /* エラー状態をチェックして返す */
 return_status = fn_status_code(status);
 fn_composite_name_destroy(binding_name);
 fn_ref_addr_destroy(address);
 fn_ref_destroy(reference);
 fn_ctx_destroy(initial_context);
 return (return_status);
}
```

オブジェクト属性のリスト表示と処理

次の例は、オブジェクト属性をリスト表示して処理する方法を示しています。

オブジェクト属性のリスト表示

次の例は、オブジェクト属性をリスト表示する方法を示しています。

```
#include <stdio.h>
#include <xfn/xfn.h>
/*
 このルーチンは名前付きオブジェクトに関連付けられたすべての属性を
 標準出力に出力する。関数の使用例として fns_attr_list("user/jsmith"); や
 fns_attr_list("thisorgunit/service/printer/color"); がある
*/
void fns_attr_list(const char *name)
{
 FN_composite_name_t *name_comp;
 const FN_identifier_t *identifier;
 FN_attribute_t *attribute;
 const FN_attrvalue_t *values;
 char *id, *val;
 FN_multigetlist_t *attrset;
 void *ip;
 FN_status_t *status;
 FN_ctx_t *initial_context;
 name_comp = fn_composite_name_from_str((unsigned char *) name);
 status = fn_status_create();
 /* 初期コンテキストの獲得 */
 initial_context = fn_ctx_handle_from_initial(0, status);
 if (!fn_status_is_success(status)) {
  fprintf(stderr, "Unable to obtain intial context\n");
  return;
 }
 /* 全属性の獲得 */
 attrset = fn_attr_multi_get(initial_context, name_comp, 0, 0,
        status);
 if (!fn_status_is_success(status)) {
```

(続く)

```
  fprintf(stderr, "Unable to obtain attributes\n");
  return;  }
 /* 全属性の表示 */
 while (attribute = fn_multigetlist_next(attrset, status)) {
  identifier = fn_attribute_identifier(attribute);
  switch(identifier->format) {
  case FN_ID_STRING:
  id = (char *) malloc(identifier->length + 1);
  memcpy(id, identifier->contents, identifier->length);
  id[identifier->length] = '\0';
  printf("Attribute Identifier: %s", id);
  free(id);
  break;
 default:
  printf("Attribute of non-string format\n\n");
  continue;
 }
for (values = fn_attribute_first(attribute, &ip);
 values != NULL;
 values = fn_attribute_next(attribute, &ip)) {
 val = (char *) malloc(values->length + 1);
 memcpy(val, values->contents, values->length);
 val[values->length] = '\0';
 printf("Value: %s", val);
  free(val);
 }
 fn_attribute_destroy(attribute);
 printf("\n");
 }
 fn_multigetlist_destroy(attrset);
 fn_ctx_destroy(initial_context);
 fn_status_destroy(status);
 fn_composite_name_destroy(name_comp);
}
```

オブジェクト属性の追加、削除、変更

次の例は、オブジェクト属性の追加、削除、変更を示しています。

```
#include <stdio.h>
#include <xfn/xfn.h>
/*
 このルーチンは名前付きオブジェクトに関連付けられた属性を変更する。
 変更としては、
 FN_ATTR_OP_ADD
 FN_ATTR_OP_ADD_EXCLUSIVE
 FN_ATTR_OP_REMOVE
 FN_ATTR_OP_ADD_VALUES
 FN_ATTR_OP_REMOVE_VALUES
 が有効である。この関数は属性値が文字列であることを前提とする。
```

(続く)

```
 関数の使用例として、以下に "James Smith" という値を持つ識別子 "realname"
 の属性を、ユーザーオブジェクト "user/jsmith" に追加する。
  fns_attr_modify(
  "user/jsmith",
  "realname",
  "James Smith",
  FN_ATTR_OP_ADD);
 次の関数は識別子 "location" の属性をプリンタオブジェクト
 "thisorgunit/service/printer/color" から削除する。
  fns_attr_modify(
  "thisorgunit/service/printer/color",
  "location",
  NULL,
  FN_ATTR_OP_REMOVE);
*/
static const char *attr_id_syntax = "fn_attr_syntax_ascii";
void fns_attr_modify(const char *name,
 const char *attr_id,
 const char *attr_value,
 unsigned int operation)
{
 FN_composite_name_t *name_comp;
 FN_identifier_t identifier, syntax;
 FN_attrvalue_t *values;
 FN_attribute_t *attribute;
 FN_status_t *status;
 FN_ctx_t *initial_context;
 name_comp = fn_composite_name_from_str((unsigned char *) name);
 status = fn_status_create();
 /* 初期コンテキストの獲得 */
 initial_context = fn_ctx_handle_from_initial(0, status);
 if (!fn_status_is_success(status)) {
  fprintf(stderr, "Unable to obtain intial context\n");
  return;
 }
 /* 追加する属性の作成 */
 /* 最初に識別子 */
 identifier.format = FN_ID_STRING;
 identifier.length = strlen(attr_id);
 identifier.contents = (void *) strdup(attr_id);
 /* 次に構文 */
 syntax.format = FN_ID_STRING;
 syntax.length = strlen(attr_id_syntax);
 syntax.contents = (void *) strdup(attr_id_syntax);
 /* 次に属性値 */
 if (attr_value) {
  values = (FN_attrvalue_t *) malloc(sizeof(FN_attrvalue_t));
  values->length = strlen(attr_value);
  values->contents = (void *) strdup(attr_value);
 } else
  values = NULL;
 /* 次に属性の作成 */
 attribute = fn_attribute_create(&identifier, &syntax);
```

(続く)

```
 /*次に属性値の追加 */
 if (values)
 fn_attribute_add(attribute, values, 0);
 /* XFN オペレーションの実行 */
 fn_attr_modify(initial_context, name_comp, operation, attribute, 0,
status);
 if (!fn_status_is_success(status))
  fprintf(stderr, "Unable to perform attribute operation\n");
 fn_ctx_destroy(initial_context);
 fn_status_destroy(status);
 fn_composite_name_destroy(name_comp);
 fn_attibute_destroy(attribute);
 free(identifier.contents);  free(syntax.contents);
 if (values) {
  free(values->contents);
  free(values);
 ]
]
```

コンテキスト内のオブジェクトの検索

次の例は、特定の属性識別子と値によって、コンテキスト内のオブジェクトを検索する方法を示しています。

```
#include <stdio.h>
#include <xfn/xfn.h>
#include <string.h>
#include <stdlib.h>
/*
 このルーチンは、指定された属性識別子と値を持つコンテキスト内の
 オブジェクトを検索します。
*/
typedef struct fns_search_results {
 char *name;
 struct fns_search_results *next;
} fns_search_results;
static const char *attr_id_syntax = "fn_attr_syntax_ascii";
fns_search_results *
fns_attr_search(const char *name,
 const char *attr_id,
 const char *attr_value)
{
 FN_status_t *status;
 FN_ctx_t *initial_context;
 FN_composite_name_t *context_name;
 FN_searchlist_t *search_list;
 FN_string_t *search_name;
 FN_attribute_t *attribute;
 FN_attrset_t *attrset;
```

(続く)

```
FN_identifier_t identifier, syntax;
FN_attrvalue_t *values;
unsigned stat;
fns_search_results *head = 0, *current, *prev;
int no_names = 0;
context_name = fn_composite_name_from_str((unsigned char *) name);
status = fn_status_create();
initial_context = fn_ctx_handle_from_initial(0, status);
if (!fn_status_is_success(status)) {
 fprintf(stderr, "Unable to obtain intial context\n");
 return (0);
}
/* 検索する属性を持つ attrset のコンストラクト */
/* 最初に識別子 */
identifier.format = FN_ID_STRING;
identifier.length = strlen(attr_id);
identifier.contents = (void *) strdup(attr_id);
/* 次に構文 */
syntax.format = FN_ID_STRING;
syntax.length = strlen(attr_id_syntax);
syntax.contents = (void *) strdup(attr_id_syntax);
/* 次に属性値 */
values = (FN_attrvalue_t *) malloc(sizeof(FN_attrvalue_t));
values->length = strlen(attr_value);
values->contents = (void *) strdup(attr_value);
/* 次に属性の作成 */
attribute = fn_attribute_create(&identifier, &syntax);
/* 次に属性値の作成 */
fn_attribute_add(attribute, values, 0);
/* 次に attrset の作成と属性の追加 */
attrset = fn_attrset_create();
fn_attrset_add(attrset, attribute, 0);
search_list = prelim_fn_attr_search(initial_context,
 context_name, attrset, 0, 0, status);
if (!fn_status_is_success(status)) {
 fprintf(stderr, "Unable to list names\n");
 return (0);
}
while (search_name = prelim_fn_searchlist_next(search_list,
 0, 0, status)) {
 no_names++;
 current = (fns_search_results *)
  malloc(sizeof(fns_search_results));
 current->name = (char *)
  malloc(strlen((char *) fn_string_str(search_name, &stat)) + 1);
 strcpy(current->name, (char *) fn_string_str(search_name, &stat));
 current->next = 0;
 if (head) {
  prev->next = current;
  prev = current;
 } else {
  head = current;
  prev = current;
```

(続く)

```
 }
 fn_string_destroy(search_name);
}
fn_searchlist_destroy(search_list);
fn_status_destroy(status);
fn_ctx_destroy(initial_context);
fn_attrset_destroy(attrset);
fn_attribute_destroy(attribute);
free(identifier.contents);
free(syntax.contents);
free(values->contents);
free(values);
 return (head);
}
```

第 21 章

FNS の概要

この章では、X/OPEN XFN フェデレーテッド・ネーミング標準に基づいて Sun Microsysmtems, Inc. が設計した FNS (フェデレーテッド・ネーミング・サービス) について説明します。

XFN と FNS

コンピューティング環境に組み込まれているネームサービスは、アプリケーションやサービスによって異なります。異なる複数のネームサービスを処理する場合、アプリケーション開発者は多くの問題に直面します。ほとんどのアプリケーションは、1つのネームサービスを使用するように設計されており、分散コンピューティング環境内のオブジェクトへのアクセスは非常に制限されています。アプリケーションによって、使用するネームサービスが異なるため、名前の作成方法も異なります。ユーザーが非常に似ていると見なすオブジェクトについても、異なる名前が使用される場合がよくあります。たとえば、友人の Johanna に対して彼女の名前 `johanna@admin.doc.com` を使用してメールを送信できますが、彼女のカレンダにアクセスするには、別の名前 `jsmith@altair` を使用しなければなりません。

Sun Microsysmtems, Inc. による XFN 標準の実装である FNS を使用すると、オブジェクトを一貫した方法でネーミングし、しかもアプリケーションと開発者が必要とする機能を提供することもできます。

- 「XFN」は、X/Open フェデレーテッド・ネーミングのことを指します。XFN は、Sun Microsystems, Inc.、IBM、Hewlett-Packard、DEC、Siemens、OSF などの組織によって実際にサポートされている標準です。Solari 7 リリースでの XFN の実装である FNS は、「X/Open Preliminary Specification for Federated Naming」(1994 年 7 月) に準拠しています。FNS を使用するアプリケーションは、プラットフォーム間で移植可能です。これは、FNS によって、エクスポートされるインタフェースが、他のベンダーと X/Open によって認められた公共のオープンインタフェースである XFN だからです。X/Open Company Ltd. は、主要なコンピュータベンダーによって推奨され支持されるコンピューティング標準の定義を委託された国際標準機構です。
- 「FNS」は、基本的なネーミング操作用の 1 つの単純な一貫したインタフェースによって、複数のネーミングサービスをフェデレートする方法です。このサービスは、複合名、つまり、複数のネーミングシステムにまたがる名前を、ネーミングインタフェースによって解決します。フェデレーションの各メンバーは、ネーミング規則、管理インタフェース、名前解決以外の特定操作の集合を独自に選択できます。

  Solaris 環境での FNS 実装は、DNS や X.500 などのグローバルネームサービス (460ページの「Solaris エンタープライズレベルのネームサービス」を参照) のサポートだけでなく、組織、ユーザー、ホスト、サイト、サービスをネーミングするための特定のポリシーと規則によって、一連のエンタープライズレベルのネー

ムサービス (463ページの「グローバル ネームサービス」を参照) から構成されます。

FNS は次の理由で便利です。

- 異なる複数のネームサービスにアクセスするために、クライアントに対して、一貫した 1 つのネーミングインタフェースが提供される。したがって、新しいネームサービスを追加しても、アプリケーションや既存のメンバー名サービスを変更する必要がない
- 名前を一貫した方法で作成でき、作成される複合名に任意の数の構成要素を使用できる
- 共有コンテキストと共有名を使用するときに、一貫したネーミングが可能になる

注 - このマニュアルでは、XFN と FNS を区別することが重要です。FNS ポリシーには XFN ポリシーの拡張がいくつか含まれており、これらのポリシーは、注によって明確に定義されます。XFN プログラミングインタフェースに属するオブジェクトは、他のプログラミングインタフェースとの混同を避けるために、XFN オブジェクトとして指定されます。

XFN モデル

この項では、XFN のネーミングモデルについて、いくつかの観点から説明します。

XFN アーキテクチャモデル

フェデレーテッド・ネーミング・システムによって提供される基本サービスは、複合名とリファレンスのマッピングと、名前付きオブジェクトに関連する属性へのアクセス権の提供です。この項では、XFN ネーミングモデルの各要素を定義します。

原子名

分割不可能な名前の最小構成要素を「原子名」といいます。たとえば、マシン名 `nismaster` やユーザー名 `chou` などです。原子名には 1 つまたは複数の属性がある場合や、属性がまったくない場合があります (453ページの「属性」を参照)。

リファレンス

リファレンスとは、オブジェクトに到達する方法についての情報のことをいいます。リファレンスには、アドレスのリストが入っています。1 つのアドレスは、1 つの通信の終端 (オブジェクト) を識別します。たとえば、`nismaster.doc.com` などのマシンアドレスや、`chou@doc.com` などのユーザーの電子メールアドレスです。

リファレンスには、1 つの概念上のオブジェクトまたはサービスを示す、複数の通信終端を識別する複数のアドレスが含まれる場合があります。たとえば、オブジェクトが分配されるか、または複数の通信メカニズムによってオブジェクトにアクセスできるため、アドレスのリストが必要になる場合などです。

注 - XFN は、安定性、有効性、到達の確実性などのアドレス属性を保証できません。クライアントは名前を検索できても、返されたリファレンスを使用できない場合があります。これは、そのクライアントが必要な通信メカニズムをサポートしていないか、あるいはそのアドレスに到達するために必要なネットワークの接続がないために起こります。また、アドレスが最初から無効であるか、古い場合があります。これらは、そのアドレスに指定された名前のバインダ、クライアント、サービスプロバイダ間の規則の問題です。

コンテキスト

コンテキストとは、図 21–1 に示すようにリファレンスに割り当てられた原子名の集合のことです。どのコンテキストにも、関連のネーミング規則があります。コンテキストは、検索 (解決) 操作を行います。この操作では、リファレンスが返され、名前のバインド、名前のバインド解除、バインド名のリスト表示を行うこともあります。コンテキストは、検索操作とバインド操作の中心となるものです。

図 21–1　XFN のコンテキスト

属性

属性は、名前付きオブジェクトに適用できます。属性はオプションです。名前付きオブジェクトには、属性を付けないことも、あるいは 1 つまたは複数の属性を付けることもできます。

各属性は、固有の属性識別子と属性構文を持ちます。個別の属性値の集合のある場合とない場合があります。属性は、図 21–1 に示すように点線で示されます。

XFN は、既存の名前付きオブジェクトに関連する属性の値を調べて変更するための基本属性インタフェースを定義します。これらのオブジェクトは、コンテキストまたは他のタイプのオブジェクトとなります。コンテキストには、コンテキストによる合成名 (compound name) の構文解析方法を記述する構造属性が関連付けられます。

拡張属性インタフェースには、特定の属性を検索する操作と、オブジェクトおよびその関連属性を作成する操作が含まれます。

合成名 (compound name)

合成名とは、1 つ以上の原子名が連なったものをいいます。あるコンテキスト内の原子名は、サブコンテキストと呼ばれる同じタイプの別のコンテキストへのリファレンスに割り当てられます。サブコンテキスト内のオブジェクトは、合成名を使用してネーミングされます。合成名は、連続する各コンテキスト内の連続する各原子名を検索することによって解決されます。

これは、UNIX ユーザーのファイルネーミングモデルに似ています。この場合、ディレクトリはコンテキストにあたり、パス名は合成名として機能します。また、コンテキストは、ディレクトリと同様に「ツリー」構造で構成されます。合成名は階層名前空間を形成します。

次に例を示します。

- 「UNIX」の `usr/local/bin`

  UNIX の原子名は、左から右へ順序付けられ、スラッシュ (/) によって区切られる。`usr` という名前は、`local` がバインドされているコンテキストにバインドされる。`local` という名前は、`bin` がバインドされているコンテキストにバインドされる

- 「DNS」の `sales.doc.com`

  DNS の原子名は、右から左へ順序付けられ、ドット (.) によって区切られる。ドメイン名 `com` は、`doc` がバインドされているコンテキストにバインドされる。`doc` は、`sales` がバインドされているコンテキストにバインドされる

- 「X.500」の c=us/o=doc/ou=sales

  X.500 の原子名は、属性タイプと属性値を構成する。原子名は、X.500 では、「相対識別名」と呼ばれる。この文字列表記で、X.500 原子名は、左から右に順序付けられ、スラッシュ (/) によって区切られる。属性タイプは、等号 (=) によって、属性値と区切られる。一般的に使用される属性タイプには省略名が定義される (たとえば、"c" は国名を表す)。国名 US は、docがバインドされているコンテキストにバインドされる。組織名 doc は、組織ユニット名 sales がバインドされているコンテキストにバインドされる

複合名 (composite name)

複合名とは、複数のネーミングシステムにまたがる 1 つの名前のことをいいます。各構成要素は、単一のネーミングシステムの名前空間からとられた名前です。複合名の名前解決は、複数のネーミングシステムにまたがる名前の解決プロセスです。

構成要素はスラッシュ (/) によって区切られ、XFN 複合名構文に従って、左から右へ順序付けられます。たとえば、複合名

```
sales.doc.com/usr/local/bin
```

には、DNS 名 (sales.doc.com) と UNIX パス名 (usr/local/bin) という 2 つの構成要素があります。

FNS 名前空間

原子名とリファレンスアドレスは、1 つまたは複数の「名前空間」に対応して解決することもできます。デフォルトにより、FNS には、org (組織を示す)、site、host、user、service、fs (ファイルを示す) の 6 つの名前空間があります。

FNS のポリシーは、名前空間に関連する名前が相互に関連付け合う方法を決めるために使用されます。たとえば、あるユーザーが、ユーザー名前空間で sergei とネーミングされていて、/user/sergei と識別されるとします。カレンダアプリケーションは、サービス名前空間でネーミングされ、/service/calendar と識別されます。このシステムでは、sergei のカレンダサービスを /user/sergei/service/calendar と識別できます。名前空間とその使用方法の詳細は、470ページの「FNS および XFN のポリシーの概要」を参照してください。

あるアプリケーションでユーザー名の入力が必要な場合、そのアプリケーションでは、入力する名前の前に名前空間識別子 user/ を付けることができます。アプ

リケーションで、ユーザーのデフォルトのファックス機などのユーザーのサービスの 1 つをネーミングする必要がある場合は、入力された名前に service 名前空間とサービスの名前 (/service/fax) を追加できます。したがって、ファックスツールは、入力値として、ユーザー名 jacques をとり、ユーザー jacques のデフォルトファックスの完全指定名 user/jacques/service/fax を作成します。同様に、ユーザー名を入力するだけで、あるユーザーのカレンダにアクセスできます。アプリケーションは入力値 raj をとり、その値を使用して複合名を作成します。この場合は、user/raj/service/calendar です。

XFN リンク

XFN リンクは、コンテキスト内の原子名に割り当てられたリファレンスの特殊な形式です。リンクには、アドレスではなく複合名が含まれます。ネーミングシステムの多くは、そのネーミングシステム自体で使用可能なリンクという基本概念をサポートしています。XFN は、このような基本リンクと XFN リンクの間に何らかの関係があるかどうかを指定しません。

初期コンテキスト

XFN 名はすべていくつかのコンテキストに対応して解釈され、XFN ネーミング操作は、コンテキストオブジェクトに対して実行されます。「初期コンテキスト」オブジェクトは、複合名解決の開始地点となるものです。XFN インタフェースは、クライアントが初期コンテキストを獲得するための機能を提供します。

第 22 章で説明しているように、このコンテキストとそのバインドの意味を見つけるためにクライアントが必要とする名前の集合を指定します。これにより、別の XFN コンテキストを見つけることが可能となります。

ユーザーへの表示

ユーザーは、アプリケーションによりフェデレーテッド・ネーミングを利用できます。通常、ユーザーは、オブジェクトの完全複合名を作成する必要も、知る必要もありません。これは、アプリケーションが複合名の作成を行うためです。これにより、ユーザーは、XFN を認識するアプリケーションと、単純な一貫した方法で対話することができます。

ファイルシステムの表示

ユーザーとアプリケーションは、ファイルシステムによっても、フェデレーテッドネーミングを利用できます。初期コンテキストは、ルートディレクトリの /xfn に入っています。たとえば、ingrid の to_do ファイルの XFN 名が xfn/user/ingrid/fs/to_do だとします。

このファイルを読むには、次のように入力します。

```
% cat /xfn/user/ingrid/fs/to_do
```

アプリケーションは、他のファイルの場合と同様に、/xfn のもとにあるファイルにアクセスします。アプリケーションを変更する必要も、XFN API を使用する必要もありません。

アプリケーションの表示

次の一連の図は、クライアントアプリケーションが XFN と対話して異なるネーミングシステムにアクセスする方法を示します。図 21–2 は、XFN API とライブラリを使用するアプリケーションを示します。

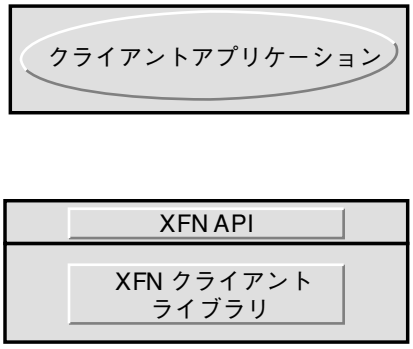


図 21–2　クライアントアプリケーションと XFN との対話

図 21–3 は、API の詳細を示しています。フェデレートされるネームサービスは、XFN クライアントライブラリと「コンテキスト共有オブジェクトモジュール」によってアクセスされます。このモジュールは、XFN 呼び出しをネームサービス特定呼び出しに変換します。

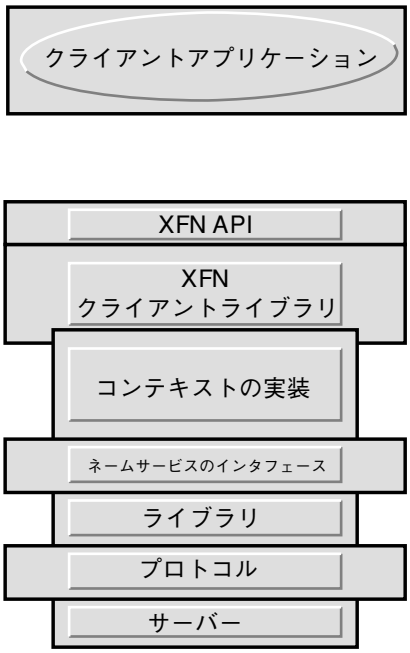


図 21–3　XFN API の詳細

API 使用モデル

XFN インタフェースのクライアントの多くは、検索だけを対象とします。これらのクライアントは、インタフェースを次の目的で使用します。

- 初期コンテキストの獲得
- 初期コンテキストに対応する 1 つまたは複数の名前の検索

クライアントは、検索操作で必要なリファレンスを獲得すると、そのリファレンスからオブジェクトのクライアント側での表示を作成します。

フェデレーテッド・ネーミング・サービス

Solaris 環境では、ネームサービスは、ファイルシステム、ネットワーク情報サービス、メールシステム、カレンダサービスなどの他のサービスに統合されます。たとえば、ファイルシステムには、ファイルとディレクトリ用のネーミングシステム

が含まれます。NIS+ サービスでは、ネーミングシステムと特殊な情報サービスを結合します。

FNS がない場合、Solaris 環境のユーザーは、整合性のない複数の名前を使用して、オブジェクトを参照しなければなりません。たとえば、`jsmith@admin` という名前を使用して Joan にメールを送り、`jsmith@altair` という名前を使用してカレンダにアクセスし、`/home/jsmith/.cshrc` という名前を使用して、彼女のホームディレクトリにあるファイルに到達しなければなりません。このため、ユーザーが名前を形式化するのが困難になります。また、アプリケーションがユーザーに代わって名前を自動生成することも難しくなります。FNS ポリシーでは、これらのオブジェクトをネーミングする一貫した方法を定義します。

FNS とアプリケーションの開発

アプリケーションもネームサービスを必要とします。Solaris 環境のアプリケーションでは、多くのネームサービスインタフェースを処理する必要があります。1 つのアプリケーションが、Solaris 環境の外部にある各種の互換性のないネーミングシステムと関わる場合もあります。ローカルネットワークと広域ネットワークでは、異機種の複数のハードウェアとオペレーティングシステムが接続されているので、各種のインタフェースが混在する可能性が高くなります。これらのネーミングインタフェースは大幅に異なるだけでなく、基本的なネーミング操作も通常明確ではありません。

FNS は、これらの問題を次の 2 つの方法で解決します。

- 基本ネーミング機能に単一の標準インタフェースを提供して、開発者がアプリケーションで使用できるようにする
- 既存のアプリケーションを変更しなくても、ネットワークのネームサービスの変更や追加ができるようにする

FNS と複合名

アプリケーションによっては、複合名を使用して、Solaris 環境のオブジェクトにアクセスするものがあります。コマンドの `mail` と `rcp` は、このようなアプリケーションの一例です。

`rcp` は、`sirius:/usr/jsmith/memo` などの複合名を使用します。これは、ホスト名 `sirius` とファイル名 (パス) `/usr/jsmith/memo` の 2 つの構成要素からなり

ます。mail プログラムは、jsmith@altair などの複合名を使用します。これは、ユーザ名 jsmith とホスト名 altair の 2 つの構成要素からなります。

各アプリケーションは、独自の名前作成規則を定義し、複合名を構文解析して、複合名を解決します。複合規則は、通常、アプリケーションごとに異なります。

FNS がない場合、ユーザーは、どのアプリケーションで複合ネーミングが可能であり、どれが可能でないかを覚えておく必要があります。たとえば、複合名 sirius:/tmp/saleslist は、rcp コマンドでは受け入れられますが、cp コマンドでは受け入れられません。

FNS がない場合、ユーザーは、アプリケーションごとに使用される異なる作成規則も覚えておく必要があります。独自の複合名をサポートするアプリケーションは、小さい特定のネーミングシステムしか使用できません。また、新しいタイプのネーミングシステムが追加されるたびに変更する必要があります。

複合ネーミングの一貫したポリシーを、コンピューティングプラットフォームに組み込むと、どのアプリケーションでも一貫した方法で複合名をサポートできます。アプリケーションは、1 つの名前を 1 つのインタフェースに渡します。

FNS ポリシーの原則

FNS ポリシーには、次の原則があります。

- 「特定のオブジェクトに対応して他のオブジェクトをネーミングする場合、そのオブジェクトはネーミングコンテキストとなる」

  たとえば、ユーザーに対応してさまざまな事柄をネーミングする場合、ユーザーオブジェクトがネーミングコンテキストになります。

- 「共通構成要素を使用した名前の作成が可能」

  これにより、ユーザーが覚える必要がある名前の数が減り、アプリケーションとユーザーによる、共通構成要素とその論理構成に関する知識に基く名前の作成が容易になります。

- 「名前は視覚的で自明のものでなければならない」

  たとえば、FNS 名 /user/wong/service/calendar は、ユーザー Wong によって使用されるカレンダサービスを明示します。これに対して、カレンダ名 wong@deneb は、Wong のカレンダサービスが提供されるホスト (deneb) を表します。ただし、他のユーザーにとって、ホスト名は余分であり、検索しにくく覚えずらいので、このユーザーのカレンダとホストの間に明確な関連性はありません。

- 「1 つのコンテキストで間に合う場合には、2 つのコンテキストを使用しない。」

  上記の例では、メールアドレス、カレンダ、ファイルのディレクトリを、ユーザー Wong に対応してネーミングした方が効果的です。コンテキストとその名前を共有すると、ネーミングの整合性が高まり、管理が簡単になります。

Solaris 環境での FNS

Solaris 環境での FNS 実装は、現在次のものの上に実装されたネームサービスで構成されています。

- NIS+、NIS、ローカルファイル、などの組織レベルのネームサービス (460ページの「Solaris エンタープライズレベルのネームサービス」を参照)
- ファイルネーミング、プリンタネーミング、他のアプリケーションのサポート (第 25 章を参照)
- DNS と X.500/LDAP を使用するグローバルレベルのネーミングシステム (第 26 章を参照)

FNS は、FNS を使用するアプリケーションとシステムが増えるほど、Solaris のユーザーには利用しやすいものとなります。

Solaris エンタープライズレベルのネームサービス

「エンタープライズレベル」のネームサービスは、エンタープライズレベルのネットワーク内のマシン (ホスト)、ユーザー、ファイルを識別 (ネーミング) します。FNS でも、組織ユニット、地理的なサイト、アプリケーションサービスのネーミングが可能です。「エンタープライズレベル」のネットワークは、ケーブル、赤外線、ラジオブロードキャストを通して通信する単一のローカルエリアネットワーク (LAN) にできます。または、ケーブルや直接通話接続によってリンクされた複数の LAN にできます。エンタープライズレベルのネットワーク内では、DNS や X.500/LDAP などのグローバルネームサービスを参照しなくても、すべてのマシンが他のマシンと通信できます。

FNS は現在、次に 3 つのエンタープライズレベルのネームサービスをサポートしています。

- NIS+

  下記の、461ページの「FNS と NIS+ のネーミング」、497ページの「FNS ポリシーと NIS+ との関連」、518ページの「FNS と NIS+ の詳細情報」を参照

- NIS

  詳細は、462ページの「FNS と NIS のネーミング」、500ページの「FNS ポリシーと NIS の関連」、520ページの「FNS と NIS の詳細情報」を参照

- ファイル

  詳細は、463ページの「FNS とファイルベースのネーミング」、501ページの「FNS ポリシーとファイルベースのネーミングの関連」、523ページの「FNS とファイルベースのネーミングの詳細情報」を参照

FNS とエンタープライズレベルネームサービスの管理情報については、第 23 章を参照してください。

FNS と NIS+ のネーミング

NIS+ とその用語については、このマニュアルのパート I「ネーミングの紹介」と用語集を参照してください。典型的な NIS+ 環境の構造を理解しておくと有効です。

NIS+ は、Solaris 環境で推奨される組織規模の情報サービスです。NIS+ では、NIS とローカルファイルの両方を使用できます。NIS+ を使用すると、ドメインとサブドメインからなる階層組織レベルに組織を分割できます。

FNS 組織ユニットは、NIS+ のドメインとサブドメインに対応します。各ドメインとサブドメインに 1 つの `orgunit` コンテキストがあります。

FNS は NIS+、NIS、ローカルファイルをフェデレートして、Solaris 環境でのネーミングポリシーをサポートします。FNS はこのために、`organization`、`site`、`user`、`host` の各オブジェクトに対してネーミング操作を実行するための XFN インタフェースを提供します。このインタフェースは、ファイル、ディレクトリ、テーブルにアクセスするために適切なプログラミングインタフェースを使用して、これらの操作を実装します。

NIS+ のもとでは、FNS コンテキストと属性データは、NIS+ タイプテーブルに格納されます。これらのテーブルは、`ctx_dir` という名前の NIS+ タイプのディレクトリオブジェクトに格納されます。各 NIS+ ドメインとサブドメインに `ctx_dir` ディレクトリオブジェクトがあり、ドメインの `groups_dir` と `org_dir` ディレクトリオブジェクトと同じレベルにあります。したがって、ディレクトリオブジェクト

ctx_dir.sales.doc.com. には、sales.doc.com ドメインの FNS コンテキストと属性データを格納する FNS テーブルが含まれます。

NIS+ では、FNS と NIS+ のコマンドを使用して、FNS テーブル内の情報を処理します。これらのテーブルを直接編集したり、UNIX コマンドで操作したりしないでください。

FNS と NIS のネーミング

NIS は Solaris 環境でのエンタープライズ規模の情報サービスです。ローカルファイルは、NIS とともに使用できます。NIS のもとでは、エンタープライズは単一の NIS ドメインとして構成されます。

各エンタープライズは、単一の NIS ドメインです。1 つの NIS ドメインに対応して、1 つの FNS エンタープライズユニットがあります。

FNS は、NIS とローカルファイルをフェデレートして、Solaris 環境でのネーミングサービスをサポートします。FNS はこのため
に、organization、site、user、host の各マップに対してネーミング操作を実行するための XFN インタフェースを提供します。このインタフェースは、ファイル、ディレクトリにアクセスするために適切なプログラミングインタフェースを使用して、これらの操作を可能にします。

NIS では、FNS コンテキストと属性データは NIS マップに格納されます。これらのマップは、NIS サーバー上の /var/yp/*domainname* ディレクトリに格納されます。NIS では、スーパーユーザーが FNS コマンドを使用して、FNS マップ内の情報を処理できます。

NIS クライアントが SKI の実行中 FNS によってコンテキストを更新する

一定の条件が合えば、NIS クライアント (マシン、プロセス、またはユーザー) はすべて、fncreate_fs や fncreate_printer などの FNS コマンドを使用して、クライアント自身のコンテキストを更新できます。これにより、NIS クライアントは、FNS コマンドを使用して、Printer Administrator、CDE カレンダマネージャ、admintool などを更新できます。

スーパーユーザー以外のユーザーが、FNS コマンドによって各自のコンテキストを更新するには、次の条件が必要です。

- SKI (Secure Key_management Infrastructure) が NIS マスターサーバー上で使用可能
- `fnsypd` デーモンが NIS マスターサーバー上で実行されている。このデーモンは、スーパーユーザー特権を持つユーザーが開始する
- クライアントユーザーまたはマシンだけが各自のコンテキストを更新できる
- クライアントは、要求された更新の実行権限を持っている

FNS とファイルベースのネーミング

「ファイル」とは、通常、マシンの `/etc` ディレクトリにあるネーミングファイルのことを指します。これらのマシンベースのファイルには、UNIX ユーザーとパスワードの情報、ホスト情報、メールエイリアスなどが入っています。これらは、オートマウントマップなどの Solaris 特定のデータもサポートします。

FNS は、ローカルファイルをフェデレートして、Solaris 環境でのネームサービスをサポートします。FNS はこのために、`organization`、`site`、`user`、`host` の各ファイルに対してネーミング操作を実行するための XFN インタフェースを提供します。これらの操作は、ファイル、ディレクトリにアクセスするための適切なプログラミングインタフェースを使用して実装されます。

ファイルベースのネーミングシステムでは、FNS コンテキストと属性は、ファイルに格納されます。これらのファイルは、NFS ファイルサーバーからエクスポートされた `/var/fn` ディレクトリに格納されます。

ファイルベースのネーミングシステムでは、FNS コマンドを使用して、FNS ファイル内の情報を処理します。

グローバルネームサービス

グローバルネームサービスは、電話、衛星などの通信システムによって世界中でリンクされた組織レベルのネットワークを識別 (ネーミング) します。この世界規模でリンクされたネットワークの集合は、「インターネット」と呼ばれています。ネーミングネットワークだけでなく、グローバルネームサービスも、ネットワーク内の個々のマシンとユーザーを識別します。

FNS は現在、次の 2 つのグローバルネームサービスをサポートしています。

- DNS

詳細は、464ページの「FNS と DNS」、558ページの「DNS でフェデレートする」を参照

- X.500/LDAP

  詳細は、464ページの「FNS と X.500」、560ページの「X.500/LDAP でフェデレートする」を参照

注 - エンタープライズレベルのネームサービスが NIS+ または NIS の場合、グローバルネームサービスだけをフェデレートできます。ファイルベースのネームサービスをエンタープライズで使用している場合は、DNS も X.500/LDAP もフェデレートできません。

FNS とエンタープライズレベルネームサービスの管理情報については、第 26 章を参照してください。

FNS と DNS

インターネットドメイン名システム (DNS) は、世界のインターネットにホストとドメインの名前解決を提供するネームサーバーの階層集合です。FNS は DNS を使用してエンタープライズオブジェクトをグローバルにネーミングします。

ドメイン名とは、DNS がエンタープライズレベルネットワーク (LAN または WAN) を識別するために使用する名前のことをいいます。NIS+ を使用するネットワークでは、親ドメイン内でのサブドメインの作成が可能であり、DNS はこのようなサブドメインを識別できます。

名前は、インターネット上でアクセス可能なすべてのエンタープライズについて作成できます。したがって、名前は、これらのエンタープライズからエクスポートされたオブジェクトにも作成できます。FNS と DNS の詳細は、558ページの「DNS でフェデレートする」を参照してください。

FNS と X.500

X.500 はグローバルディレクトリサービスです。この各構成要素は、世界規模の範囲内のオブジェクトに関する情報を管理するためにともに機能します。このようなオブジェクトには、国、組織、ユーザー、マシンが含まれます。FNS は、X.500 をフェデレートして、エンタープライズネームサービスへのグローバルアクセスを可能にします。次の 2 つの API のうち 1 つを使用して、X.500 グローバルディレクトリサービスにアクセスできます。

- XDS/XOM API
- LDAP (軽量ディレクトリアクセスプロトコル) API

X.500 のフェデレートについては、560ページの「X.500/LDAP でフェデレートする」を参照してください。

FNS とアプリケーション

FNS は、次のものをサポートします。

- Solaris NFS ファイルサービス (下記の、465ページの「FNS ファイルネーミング」を参照)
- プリンタネーミング (下記の、465ページの「FNS プリンタネーミング」を参照)
- その他のアプリケーション (下記の、466ページの「FNS アプリケーションサポート」を参照)

FNS ファイルネーミング

FNS ベースのファイルネーミングは、FNS ネーミングを Solaris ファイルサービスに統合します。FNS ベースのファイルネーミングを使用すると、他のファイルに関連しないアプリケーションと共有される FNS ポリシーによって、ユーザー、ホスト、サイト、組織に対応してファイルをネーミングできます。

FNS ベースのファイルネーミングは、クライアントに対して、グローバルかつエンタープライズ規模のファイル名前空間の共通表示を提供します。ファイルシステムにアクセスする Solaris アプリケーションは、FNS によってサポートされるファイル名前空間に、変更せずにアクセスできます。

FNS プリンタネーミング

FNS ベースのプリンタネーミングは、バンドルされていない SunSoft Print Client (SSPC) の基本ネーミング機能となります。FNS ベースのプリンタネーミングを使用すると、他の印刷関連ではないアプリケーションと共有される FNS ポリシーを使用して、ユーザー、ホスト、サイト、組織に対応してプリンタをネーミングできます。

FNS ベースのプリンタネーミングは、クライアントに対して、グローバルかつエンタープライズ規模のプリンタ名前空間の共通表示を提供します。また、これを使用すると、プリンタ名前空間を中央で管理できます。

FNS アプリケーションサポート

FNS を認識するアプリケーションでは、名前空を FNS ポリシーに従って構成できます。また、FNS 名前空で名前を割り当てるアプリケーションは、これらのポリシーに従う必要があります。

アプリケーションは、FNS を次の 3 つの方法で使用します。

- 「アプリケーションは、FNS インタフェースとポリシーを使用すればクライアントとなる。」

  ファイルシステム、印刷サービス、デスクトップツール (カレンダマネージャ、ファイルマネージャ) などのアプリケーションレベルのユーティリティは、FNS インタフェースを直接使用するクライアントの一例である

- 「アプリケーションは、既存のインタフェースにより FNS を使用できる。」

  FNS の大部分は、既存のアプリケーションプログラミングインタフェースにより使用される。たとえば、後で UNIX open() 関数に与えられるファイル名を獲得する UNIX アプリケーションを例にとる。この場合、FNS サポートを使用してファイル名を解決すると、アプリケーションは、処理対象の文字列が従来のローカルパス名ではなく複合名であることを認識しなくてもすむ。したがって、アプリケーションの多くは、変更なしに合成名をサポートできる

- 「システムが FNS インタフェースをエクスポートできる。」

  DNS や X.500 などのネーミングシステム、ファイルシステムや印刷サービスなどのサービスに組み込まれたネーミングシステムは、FNS インタフェースをエクスポートするネーミングシステムの例である

FNS の管理

FNS システムの管理は、基本となるネームサービスによって異なります。

- 「NIS+」

  NIS+ では、FNS システム管理タスクは、その権限を持つユーザーだけが実行できる。システム管理特権は、通常、NIS+ グループを作成して、そのグループに

ドメインで必要な特権を割り当てることによって付与される。これにより、グループのメンバーはすべて、システム管理機能を実行できる

- 「NIS」

  NIS では、NIS マスターサーバー上の `root` が FNS 管理タスクを実行しなければならない

- 「ファイルベース」

  ファイルベースのネーミングシステムでは、`/var/fn` ディレクトリに `root` アクセス権を持つユーザーが、FNS 管理タスクを実行しなければならない

各自のユーザーサブコンテキストをユーザーが変更できるかどうかは、基本となるネームサービスによって異なります。

- 「NIS+」

  NIS+ では、ユーザーのコンテキスト (およびサブコンテキスト) は、ユーザーが所有する。NIS+ のポリシーでログインした場合、適切な資格と特権を持つユーザーは、`fncreate`、`fnbind`、`fnunbind` などのコマンドを使用して、各自のコンテキストを変更できる

- 「NIS」

  NIS では、ユーザーはどの FNS データも変更できない。NIS マスターサーバーの `root` アクセス権を持つユーザーだけが FNS データを変更できる

- 「ファイルベース」

  ファイルベースのネーミングシステムでは、ユーザーが独自のコンテキストを所有する。標準 UNIX アクセスでは、FNS ファイルへの適用が制御される

問題解決とエラーメッセージ

一般的な FNS の問題を追跡して解決する方法については、676ページの「FNS の問題と対策」を参照してください。FNS のエラーメッセージは、付録 B に記載されています。

第 22 章

FNS ポリシー

この章では、FNS ポリシーについて説明します。

FNS および XFN のポリシーの概要

XFN は、フェデレートされた名前空間にあるオブジェクトをネーミングするためのポリシーを定義します。これらのポリシーは、次のような目的を持っています。

- 統一性のある複合名を簡単に作成する
- アプリケーションおよびサービスでネーミングの一貫性を持たせる
- 簡単ながら十分な機能をもつポリシーを提供し、アプリケーションで特定環境に特別のポリシーを作成したり、実行したりする必要をなくす
- アプリケーションの移植性を拡張する
- 異機種システム混在コンピュータ環境でのプラットフォーム同士の相互運用性を高める

FNS ポリシーで指定されるもの

FNS ポリシーには、すべての XFN ポリシーと Solaris 環境用の拡張機能が含まれます。

現在、コンピュータ環境は世界的な規模となり、広範囲なサービスを提供しています。ユーザーは、どのレベルのコンピュータ環境のサービスにもアクセスできます。FNS ポリシーは、グローバル、エンタープライズ、アプリケーションの 3 つのレベルのサービスに共通の枠組みを提供します。

FNS は、ネームサービスをどのように調整して使用するかについて次のようなポリシーをアプリケーション側に提供します。

- エンタープライズ名前空間をフェデレートして、グローバル名前空間でアクセス可能にする方法を指定する
- 各プロセスの初期コンテキストにある名前およびバインドを指定する

- 組織、ホスト、ユーザー、サイト、ファイル、サービスなどのエンタープライズのオブジェクトに対するネームサービス
- 組織、ホスト、サイト、ファイル、サービスのエンタープライズのオブジェクト間の関係を定義する
- これらのエンタープライズのオブジェクトのリファレンスに使用される名前の構文を指定する

FNS ポリシーで指定されないもの

FNS ポリシーでは、次のものは指定されません。

- ネームサービスで使用される実際の名前
- アプリケーション内のネーミング。アプリケーションレベルのネーミングは、個別のアプリケーションまたは関連アプリケーションのグループで行われる
- オブジェクトがネーミングされた後に使用する属性

エンタープライズ名前空間のポリシー

FNS ポリシーでは、エンタープライズ内の名前空間のタイプおよび配列と、アプリケーションが名前空間を使用する方法が指定されます。たとえば、名前空間が他のどの名前空間と関連するかが指定されます。ここで説明する FNS ポリシーには、XFN ポリシーへの拡張機能があります。これらは、注釈によって明確に定義されます。

デフォルトの FNS エンタープライズ名前空間

FNS エンタープライズのポリシーでは、名前空間内でのエンタープライズのオブジェクトの配列が処理されます。各エンタープライズのオブジェクトは、独自の名前空間を持っています。

デフォルトでは、FNS には 7 つのエンタープライズオブジェクトと対応する名前空間があります。

- 「組織」(`orgunit`)

  部、センター、課などのエンティティ。サイト、ホスト、ユーザー、サービスには、組織に関連する名前を付けることができる。組織に対する XFN 用語は、

「組織単位」となる。初期コンテキストで使用されたときは、識別子 org は、orgunit の別名として使用できる

- 「サイト」(site)

  建物、建物内のマシン、建物内の会議室などの物理的な位置。サイトは、それらと関連するファイルおよびサービスを持つ

- 「ホスト」(host)

  マシン。ホストは、それらと関連するファイルおよびサービスを持つ

- 「ユーザー」(user)

  人間のユーザー。ユーザーは、それらと関連するファイルおよびサービスを持つ

- 「ファイル」(fs)

  ファイルシステム内のファイル

- 「サービス」(service)

  プリンタ、FAX、メール、電子カレンダなどのサービス

- 「プリンタ」(service/printer)

  プリンタの名前空間は、サービスの名前空間に従属する

図 22–1 は、エンタープライズの名前空間が配列される方法を示しています。

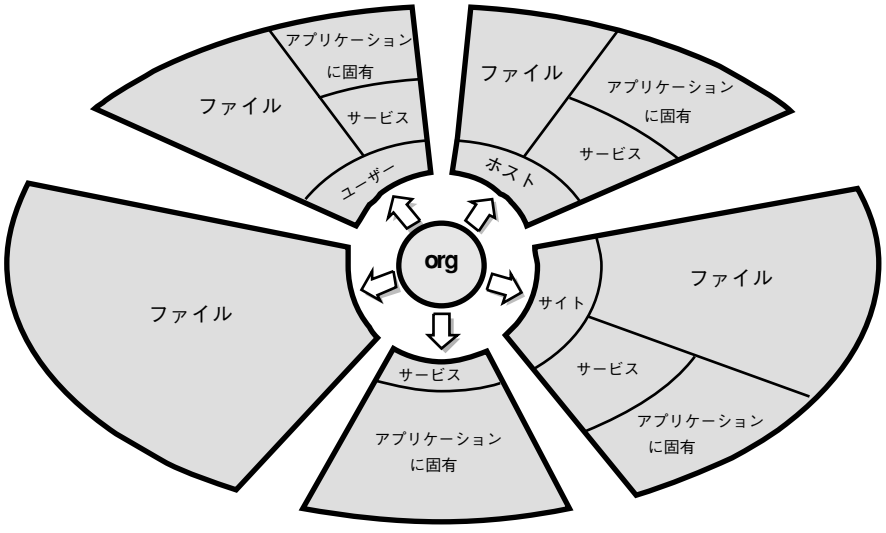


図 22–1　FNS ポリシーが配列するもの

ユーザーやホストなどの名前空間には、フェデレートされた名前空間で複数回表示されるものもあります。

これらの名前空間に適用されるポリシーは、表 22–2 に要約しています。

エンタープライズ名前空間の識別子

エンタープライズ名前空間は、フェデレートされた名前空間の原子名によって参照されます。

XFN では、先行する下線 (_) 文字を使用して、エンタープライズ名前空間の識別子が示されます。たとえば、_site となります。FNS では、下線 (_) なしの識別子の使用もサポートされます。下線なしの名前は、XFN ポリシーの範疇には入りません。site および printer のコンテキストも、XFN ポリシーの範疇には入りません。これらの原子名は、表 22–1 に挙げられています。

表 22–1 エンタープライズでのエンタープライズ名前空間の識別子

名前空間	**XFN** 識別子	**FNS** 識別子	解釈処理
組織	_orgunit	orgunit または org	組織単位をネーミングするためのコンテキスト
サイト	_site	site	サイトをネーミングするためのコンテキスト
ホスト	_host	host	ホストをネーミングするためのコンテキスト
ユーザー	_user	user	ユーザーをネーミングするのためのコンテキスト
ファイルシステム	_fs	fs	ファイルをネーミングするためのコンテキスト
サービス	_service	service	サービスをネーミングするためのコンテキスト
プリンタ		printer	プリンタをネーミングするためのコンテキスト (サービスの名前空間に従属)

注 - XFN の用語では、先行する下線のある名前は、「正規」名前空間識別子です。下線のない名前は、Solaris 環境用に「カスタマイズ」された名前空間識別子です。これらのカスタマイズされた名前空間識別子に `printer` を追加しても、Solaris 以外の環境では認識されません。正規の名前空間識別子は、他の環境でも常に認識され、互換性があります。

構成要素の区切り文字

XFN 構成要素の区切り文字 (/) で、名前空間識別子を区切ります。たとえば、名前空間識別子 `orgunit` を組織単位名 `west.sales` とともに作成すると、複合名は `orgunit/west.sales` となります。

デフォルトの FNS 名前空間

FNS では、次の 7 つの名前空間が提供されます。

- 「組織」(474ページの「組織単位の名前空間」を参照)
- 「サイト」(476ページの「サイトの名前空間」を参照)
- 「ホスト」(476ページの「ホストの名前空間」を参照)
- 「ユーザー」(477ページの「ユーザーの名前空間」を参照)
- 「ファイル」(477ページの「ファイルの名前空間」を参照)
- 「サービス」(477ページの「サービスの名前空間」を参照)
- 「プリンタ」(477ページの「サービスの名前空間」を参照)

組織単位の名前空間

組織単位の名前空間では、エンタープライズのサブ単位をネーミングするために階層になった名前空間が提供されます。各組識単位名は組織単位を表す「組織単位コンテキスト」と結合されています。組織単位名は、接頭辞 `org/`、`orgunit/`、または `_orgunit/` によってネーミングされます。短縮形別名の `org/` は、内部コンテキストだけで使用され、複合名の途中では使用されません。490ページの「エンタープライズ内のネーミングに対する初期コンテキストのバインド」および、479ページの「複合名の例」を参照してください。

NIS+ 環境

NIS+ 環境では、組織単位は NIS+ のドメインおよびサブドメインに対応します。

NIS+ では、組織単位は必ずドメインおよびサブドメインにマップされます。それぞれの NIS+ ドメインおよびサブドメインに組織単位を与える必要があります。ドメインまたはサブドメインに論理、組織ユニットを与えることはできません。つまり、NIS+ ドメインまたはサブドメインを小規模な組織単位に分割することはできません。そのため、NIS+ ドメイン `doc.com.` と 2 つのサブドメイン `sales.doc.com.` および `manf.doc.com.` がある場合は、これら 3 つのドメインに対応する 3 つの FNS 組織単位を持つ必要があります。

組織単位は、ドットで区切られた右から左への複合名を使用してネーミングされます。ここでは、各原子要素がより大きな単位内の組織単位をネーミングします。たとえば、`org/sales.doc.com.` という名前は、`doc.com.` という名前のより大きな単位内にある `sales` をネーミングします。この例では、`sales` は、`doc.com.` の NIS+ サブドメインです。

組織単位名は、完全指定の NIS+ ドメインまたは相対名の NIS+ 名のどちらかになります。完全指定名には終端にドットが付きますが、相対名には付きません。そのため、終端のドットが組織名にある場合は、その名前は完全指定の NIS+ ドメイン名として処理されます。終端にドットがない場合は、その組織名は最上部の組織階層に対して相対的に解釈処理されます。たとえば、`orgunit/west.sales.doc.com.` は、`west` 組織単位をネーミングする完全指定名で、`_orgunit/west.sales` は、同じサブドメインをネーミングする相対的な名前です。

NIS 環境

NIS 環境では、NIS ドメインに対応する組織単位は各エンタープライズ 1 つずつあります。この `orgunit` には、`orgunit/`*domainname* という名前が付けられ、ここでの *domainname* は NIS ドメイン名です。たとえば、NIS ドメイン名が `doc.com` である場合は、組織単位は `org/doc.com` です。

NIS 環境では、空の文字列を組織単位の省略形として使用できます。したがって、`org//` は、`org/`*domainname* に相当します。

ファイルベースの環境

エンタープライズレベルのネームサービスがファイルベースであるときには、1 つだけの FNS 組織単位が存在し、サブ単位はありません。ファイルベースのネーミングで許容される唯一の組織単位は `org//` です。

サイトの名前空間

サイトの名前空間では、物理的位置でそのまま識別される地域名前空間が提供されます。たとえば、これらのオブジェクトには、キャンパスにある建物、ある階のマシンやプリンタ、建物内の会議室とその使用スケジュール、隣接するオフィスにいるユーザーなどがあります。サイト名は、接頭辞 `site/` または `_site/` で識別されます。

Solaris 環境では、サイトは複合名を使用してネーミングされます。複合名では、各原子名がより大きなサイト内のサイトを表わします。サイト名の構文は、ドット区切りの右から左であり、もっとも一般的な位置記述からもっとも特定の位置記述に配列された構成要素が含まれます。たとえば、`_site/pine.bldg5` は、建物 5 にある Pine 会議室を表わし、`site/bldg7.alameda` は、あるエンタープライズの Alameda にある建物 7 を表わします。

ホストの名前空間

ホストの名前空間では、コンピュータをネーミングするための名前空間が提供されます。ホスト名は、接頭辞 `host/` または `_host/` で識別されます。たとえば、`host/deneb` は、`deneb` という名前のマシンを識別します。

ホストは、「`hostname`」コンテキストでネーミングされます。ホストコンテキストには、フラットな名前空間があり、ホストコンテキストへのホスト名のバインドが含まれます。「ホストコンテキスト」では、そのホストで見つかったファイルやプリンタなどの、マシンに相対的なオブジェクトをネーミングできます。

Solaris 環境では、ホスト名は Solaris のホスト名に対応します。単一のマシンに対する別名で同じコンテキストが共有されます。たとえば、`mail_server` という名前がマシン `deneb` および `altair` の別名である場合、`deneb` と `altair` の両方で、`mail_server` に作成されたコンテキストが共有されます。

ネットワーク資源は、必要に応じてホストに関連付けてネーミングする必要があります。たいていの場合、組織、ユーザー、サイトなどの構成要素に関連して資源をネーミングした方がわかりやすくなります。ホスト名に依存すると、ユーザーは、あいまいで変更される可能性のある情報を覚えなくてはならなくなります。たとえば、ハードウェアの変更、ファイルスペースの使用、ネットワークの再構成などのために、ユーザーのファイルがあるサーバーから他のサーバーに移動されてしまう場合もあります。さらに、ユーザーは、同じファイルサーバーを共有することが多くなり、ファイルがホストに関連付けてネーミングされた場合に混乱を招きます。しかし、ファイルがユーザーに関連付けてネーミングされた場合は、このような変更はファイルのネーミング方法に影響しません。

ホスト名を使用した方が適切な場合もあります。たとえば、資源が特定のマシンだけで使用可能であり、他の構成要素に関連付けてその資源をネーミングする論理的方法がない場合には、ホストに関連付けてネーミングする意味があります。または、ファイルシステムでは、ファイルが多くのユーザーに共有されている場合、格納されているマシンに関連付けてファイルをネーミングする意味があります。

ユーザーの名前空間

ユーザーの名前空間では、コンピュータ環境内のユーザーをネーミングするための名前空間が提供されます。ユーザー名は、接頭辞 `user/` または `_user/` で識別されます。

ユーザーは、「ユーザーコンテキスト」でネーミングされます。ユーザーコンテキストには、単一レベルの名前空間があり、「ユーザーコンテキスト」へのユーザー名のバインドが含まれます。ユーザーコンテキストでは、ユーザーに関連するファイル、サービス、資源などのオブジェクトをユーザーに関連付けてネーミングすることができます。

Solaris 環境では、ユーザー名は Solaris ログイン ID に対応します。たとえば、`_user/inga` は、ログイン ID が `inga` のユーザーを識別します。

ファイルの名前空間

ファイルの名前空間 (またはファイルシステム) では、ファイルをネーミングするための名前空間が提供されます。ファイル名は、接頭辞 `fs/` または `_fs/` で識別されます。たとえば、`fs/etc/motd` という名前では、`/etc` ディレクトリに格納されているファイル `motd` が識別されます。

ファイルの名前空間については、465ページの「FNS ファイルネーミング」で詳しく説明し、ファイルのコンテキストについては、547ページの「ファイルコンテキストの管理」で説明します。

サービスの名前空間

サービスの名前空間では、エンタープライズ内のオブジェクトに使用される、または関連するサービスのための名前空間が提供されます。このようなサービスには、電子カレンダ、FAX、メール、印刷などがあります。サービス名は、接頭辞 `service/` または `_service/` で識別されます。

Solaris 環境では、サービスの名前空間は階層になっています。サービス名は、スラッシュ (/) で区切られた左から右の複合名です。サービスの名前空間を使用するアプリケーションは、この階層属性を利用し、そのアプリケーションのサブツリーを獲保します。たとえば、プリンタサービスはサービスの名前空間のサブツリー printer を獲保します。

FNS は、サービス名またはリファレンスタイプが選択される方法を指定します。これらは、サービスの名前空間を共有するサービス提供者によって決定されます。たとえば、カレンダサービスは、サービスのコンテキストにある名前 _service/calendar を使用してカレンダサービスをネーミングし、名前 calendar にバインドされたものを決定します。

サービス名およびリファレンスの登録

米国 Sun Microsystems, Inc. は、service の名前空間の最初のレベルにある名前の登録を行います。新規名を登録するには、電子メールのリクエストを fns-register@sun.com まで送信するか、または書面で次の住所までご郵送ください。

```
FNS Registration
Sun Microsystems, Inc.
901 San Antonio Road.
Palo Alto, CA 94303
U.S.A.
```

名前の使用目的についての簡単な説明と、その名前にバインドされるリファレンスのフォーマットの説明をお書き添えください。

プリンタの名前空間

プリンタの名前空間では、プリンタをネーミングするための名前空間が提供されます。プリンタの名前空間は、サービスの名前空間に関連 (従属) します。つまり、プリンタのサービスは、サービスの名前空間のサービスの一部です。プリンタ名は、接頭辞 service/printer または _service/printer で識別されます。たとえば、service/printer/laser1 では、laser1 という名前のプリンタが識別されます。

後続スラッシュの意味

後続の / は、次のネーミングシステムにあるオブジェクトをネーミングします。あるネーミングシステムから他のネーミングシステムに移動するときに必ず必要となります。たとえば、名前 org/east.sales/service/printer では、org と east.sales の間のスラッシュは、上で説明した構成要素の区切り文字であり、組織名 (sales/) の最後の要素に続くスラッシュは、service 名前空間を組織単位名前空間から区切ります。したがって、org/west.sales/service/printer では、west.sales という組織単位の printer サービスがネーミングされます。

FNS 予約名

FNS は、表 22–1 に挙げたすべての原子名をそれ自身の使用のために名前空間識別子として獲保します。たとえば、_orgunit、org、_site、site などです。この制限事項は、481ページの「エンタープライズ名前空間の構造」にある名前空間の配列で定義されているように、名前空間識別子が現れるコンテキストに適用されます。それ以外の場合は、FNS は、他のコンテキストではこれらの原子名の使用を制限します。

たとえば、原子名 service は、user/fatima/service/calendar のように、ユーザー名に相対的な名前空間識別子として使用され、ユーザー fatima のサービスの名前空間のルートであることを意味します。FNS では、名前 user/ がバインドされるコンテキストは、名前空間識別子にではなく、ユーザー名に指定されるため、user/service のようにユーザー名として名前 service を使用することが禁止されるわけではありません。この場合、service は明確にユーザー名として解釈されます。一方で、/user/mikhail を使用すると、ユーザー mikhail とファイル (またはサブディレクトリ) /user/mikhail の区別がつきにくくなるため、user という名前のディレクトリは作成しないでください。

複合名の例

この項では、FNS ポリシーに従う名前の例を示します。これらのポリシーについては、表 22–2 を参照してください。

組織名、サイト名、ユーザー名、ホスト名、ファイル名、サービス名 (calendar や printer など) の名前は、例として使用されています。これらの名前は、FNS ポリシーでは指定されません。

組織に関連する名前を作成する

組織の名前空間 (_orgunit、orgunit、または org) に関連付けることのできる名前空間は、user、host、service、fs、と site です。

以下に例を示します。

- orgunit/doc.com/site/videoconf.bldg-5 では、組織 doc.com に関連するサイトの建物 5 にある会議室 videoconf がネーミングされる (orgunit に別名 org を使用して、org/doc.com/site/videoconf.bldg-5 を作成することもできる)
- orgunit/doc.com/user/mjones では、組織 doc.com のユーザー mjones がネーミングされる
- orgunit/doc.com/host/smptserver1 では、組織 doc.com に所属するマシン smptserver1 がネーミングされる
- orgunit/doc.com/fs/staff/agenda9604/ では、組織 doc.com に所属するファイル staff/agenda9604 がネーミングされる
- orgunit/doc.com/service/calendar では、組織 doc.com の calendar サービスがネーミングされます。ここでは、組織の会議スケジュールを管理する

ユーザーに関連する名前を作成する

user の名前空間に関連付けることができる名前空間は、service および fs です。

- user/helga/service/calendar では、helga という名前のユーザーの calendar サービスがネーミングされる
- user/helga/fs/tasklist では、ユーザー helga のホームディレクトリにあるファイル tasklist がネーミングされる

ホストに関連する名前を作成する

hosts の名前空間に関連付けることができる名前空間は、service および fs です。

- host/mailhop/service/mailbox では、マシン mailhop と関連する mailbox サービスがネーミングされる
- host/mailhop/fs/pub/saf/archives.96 では、マシン mailhop によってエクスポートされたファイルシステムのルートディレクトリにあるディレクトリ pub/saf/archives.96 がネーミングされる

サイトに関連する名前を作成する

sites の名前空間に関連付けることができる名前空間は、service および fs です。

- site/bldg-7.alameda/service/printer/speedy では、bldg-7.alameda サイトにあるプリンタ speedy がネーミングされる
- site/alameda/fs/usr/dist では、alameda サイトで使用可能なファイルディレクトリ usr/dist がネーミングされる

サービスおよびファイルに関連する名前を作成する

ファイル (fs) またはサービス (service) の名前空間に関連付けることができる名前空間はありません。たとえば、/services/calendar/orgunit/doc.com などの名前は作成できません。つまり、ファイルまたはサービスの名前空間に関連付けた複合名は作成できません。もちろん、/user/esperanza/service/calendar のように、他の名前空間に関連付けてファイル名またはサービス名を作成することは可能です。

エンタープライズ名前空間の構造

FNS ポリシーでは、エンタープライズ名前空間の構造が定義されます。この構造の目的は、統一性のある名前を簡単に作成することです。このエンタープライズ名前空間構造には、2 つの主要なルールがあります。

- 狭い有効範囲のオブジェクトは、広い有効範囲のオブジェクトに関連してネーミングされる
- 名前空間識別子は、ある名前空間から次の名前空間への移行を表すために使用される

表 22–2 は、エンタープライズ名前空間を配列するための FNS ポリシーのまとめです。図 22–2 に、これらの FNS ポリシーに従う名前空間の配置の例を示します。

表 22–2 フェデレートされたエンタープライズ名前空間用のポリシー

名前空間 識別子	従属する名前空間	親コンテキスト	名前空間編成	構文
orgunit _orgunit org	サイト、ユーザー、ホスト、ファイルシステム、サービス	エンタープライズのルート	階層	ドット区切り、右から左
site _site	サービス、ファイルシステム	エンタープライズのルート、組織単位	階層	ドット区切り、右から左
user _user	サービス、ファイルシステム	エンタープライズのルート、組織単位	フラット	Solaris ログイン名
host _host	サービス、ファイルシステム	エンタープライズのルート、組織単位	フラット	Solaris ホスト名
service _service	アプリケーション固有	エンタープライズのルート、組織単位、サイト、ユーザー、ホスト	階層	/ 区切り、左から右
fs _fs	なし	エンタープライズのルート、組織単位、サイト、ユーザー、ホスト	階層	/ 区切り、左から右
printer	なし	サービス	階層	/ 区切り、左から右

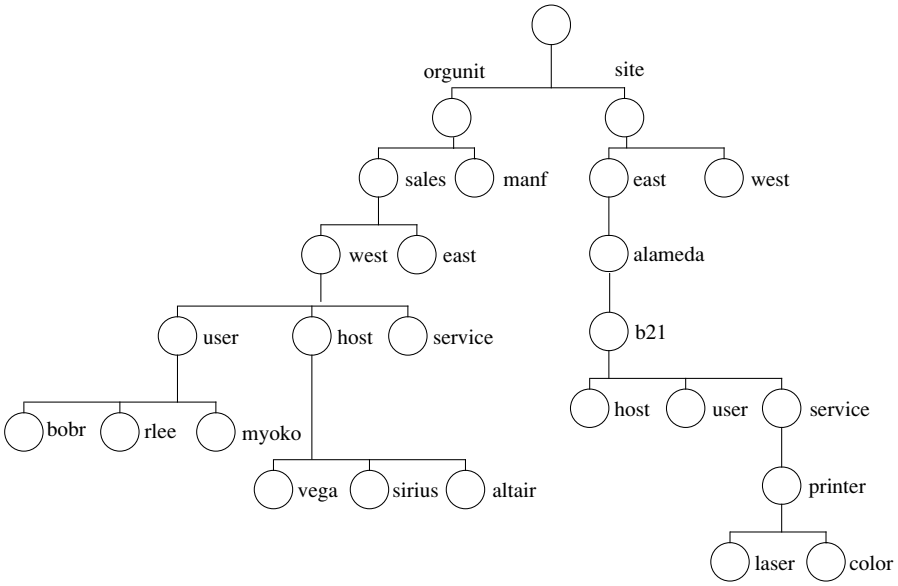


図 22–2　エンタープライズ名前空間の例

エンタープライズの名前空間は、組織単位の周辺に構成されます。適切な名前空間識別子とオブジェクト名を使用して組織単位名を作成して、サイト、ホスト、ユーザー、ファイル、サービスの名前は、組織単位に関連付けてネーミングできます。

図 22–2 では、エンタープライズの sales 組織の西部事業部のユーザー `myoko` は、名前 `orgunit/west.sales/user/myoko` を使用してネーミングされます。

名前空間識別子 `user` を使用して、`orgunit` 名前空間から `user` 名前空間への移行を表していることに注意してください。同様に (適切な名前空間識別子を使用して)、ファイル名およびサービス名も、サイト、ユーザー、またはホストの名前に関連付けてネーミングできます。サイト名は、組織単位名に関連付けてネーミングできます。

簡単に均質な名前を作成することは、この構造を使用して達成されます。たとえば、エンタープライズ (`orgunit/west`) 内の組織単位名が分かれば、`user` の名前空間識別子とユーザーのログイン名で名前を作成して `orgunit/west/user/josepha` などの名前を作成し、エンタープライズに関連付けてユーザーをネーミングできます。

このユーザーのファイルシステムにあるファイルをネーミングするときには、`orgunit/west/user/josepha/fs/notes` などの名前を使用できます。

エンタープライズのルート

エンタープライズのルートコンテキストは、エンタープライズ名前空間のルートレベルにあるオブジェクトをネーミングするためのコンテキストです。エンタープライズのルートは、グローバルの名前空間でバインドされます。

エンタープライズのルートをネーミングする方法には、次の 2 つがあります。

- `.../`*rootdomain*
- `org//`

3 つのドットを使用してエンタープライズのルートを識別する

`.../`*rootdomain*`/` を使用して、エンタープライズのルートを識別できます。

- 最初の 3 つのドット (`...`) は、グローバルのコンテキストを示す原子名 (508ページの「グローバル名前空間のポリシー」のグローバルのコンテキストについての説明を参照)
- *rootdomain*`/` は、エンタープライズのルートドメイン。たとえば、`doc.com/` など

  したがって、`.../doc.com/` では、ルートドメインが `doc.com` である会社のエンタープライズのルートが識別されます。この例では、エンタープライズのルートに関連するサイトをネーミングするためのコンテキストは、`.../doc.com/site/alameda` または `.../doc.com/site/alameda.bldg5` などの `.../doc.com/site/` となります。

注 - DNS のグローバルの建物を設定した場合は、`.../`*rootdomain* フォーマットだけを使用できます。

---

org// を使用してエンタープライズのルートを識別する

`org//` を使用して、エンタープライズのルートを識別できます。本質的に `org//` は、`.../`*domainname*`/` に対する別名または機能的に相当するものです。`org//` を使用するときは、二重のスラッシュによって、ルートのエンタープライズコンテキストとそれに関連する名前空間が識別されます。

たとえば、org//site/alameda では、エンタープライズのルートに関連する Alameda サイトがネーミングされます。

反対に、org/ または orgunit/ (一重のスラッシュ) では、エンタープライズのルートに必ずしも相対的にネーミングされていない組織のコンテキストを示します。たとえば、org/sales/site/alameda のようになります。

エンタープライズのルートの従属するコンテキスト

次のオブジェクトは、エンタープライズのルートに関連付けてネーミングできます。

- エンタープライズの組織単位
- エンタープライズの最上位の組織単位にあるサイト (XFN ポリシーへの拡張)
- エンタープライズの最上位の組織単位にいるユーザー
- エンタープライズの最上位の組織単位にあるホスト
- エンタープライズの最上位の組織単位に対するサービス
- エンタープライズの最上位の組織単位に対するファイルサービス

  これらのオブジェクトは、ターゲットオブジェクトの名前でターゲットオブジェクトの名前空間の名前空間識別子を作成して、ネーミングします。

エンタープライズのルートと組織的なサブ単位

組織のサブ単位は、エンタープライズのルートに関連付けてネーミングできます。

組織ルート名の場合は、orgunit または _orgunit のどちらかの名前空間識別子を使用して、従属する組織単位コンテキストに名前を作成できます。

たとえば、.../doc.com がエンタープライズ名である場合、組織単位にネーミングするためのコンテキストのルートは、.../doc.com/orgunit/ であり、組織単位名は .../doc.com/orgunit/sales や .../doc.com/orgunit/west.sales のようになります。または、 org//orgunit/sales でも同じ結果を得ることができます。

次のオブジェクトは、組織単位名に関連付けてネーミングできます。

- その組織単位のサイト (XFN ポリシーからの拡張)
- その組織単位にあるホスト
- その組織単位にいるユーザー

- その組織単位に対するサービス
- その組織単位に対するファイルサービス

  たとえば、名前 ...doc.com/orgunit/sales/service/calendar では、sales 組織単位のカレンダのサービスが識別されます。組織単位に関連付けてオブジェクトをネーミングする際の詳細は、474ページの「組織単位の名前空間」および 480ページの「組織に関連する名前を作成する」を参照してください。

エンタープライズのルートとサイト

サイトは、XFN ポリシーから拡張されています。

サイトは、次のものに関連付けてネーミングできます。

- エンタープライズのルート
- 組織単位

  エンタープライズのルートに関連付けてネーミングされたサイトは、最上位の組織単位に関連付けてネーミングされたサイトと同一になります。組織名の場合は、名前空間識別子の site または _site を使用して、サイトのコンテキストに名前を作成できます。たとえば、エンタープライズのルートが ../doc.com である場合、エンタープライズのルートに関連付けてサイトにネーミングするためのコンテキストは、../doc.com/site となります。サイトは、../doc.com/site/alameda のような名前になります。

  次のオブジェクトは、サイト名に関連付けてネーミングできます。

- サイトのスケジュールまたはカレンダ、プリンタ、および FAX などのサイトでのサービス
- サイトで利用可能なファイルサービス

  これらのオブジェクトは、ターゲットオブジェクトの名前空間とターゲットオブジェクト名の名前空間識別子でサイト名を作成して、ネーミングできます。たとえば、名前 site/Clark.bldg-5/service/calendar は、会議室 Clark.bldg-5 のカレンダのサービスでネーミングし、サイト名 site/Clark.bldg-5 とサービス名 service/calendar から作成されます。サイトに関連付けてオブジェクトにネーミングする際の詳細は、481ページの「サイトに関連する名前を作成する」を参照してください。

エンタープライズのルートとユーザー

ユーザーは、次のものに関連付けてネーミングできます。

- 組織単位
- エンタープライズのルート

  エンタープライズのルートに関連付けてネーミングされたユーザーは、最上位の組織単位に関連付けてネーミングされたユーザーと同一になります。組織名の場合は、名前空間識別子の user または _user のどちらかを使用して、コンテキストに名前を作成できます。したがって、orgunit/east.sales で組織がネーミングされる場合は、orgunit/east.sales/user/hirokani で east.sales 組織単位にいるユーザー hirokani がネーミングされます。

  次のオブジェクトは、ユーザー名に関連付けてネーミングできます。

- ユーザーに関連するサービス
- ユーザーのファイル

  これらのオブジェクトは、ターゲットオブジェクトの名前空間とターゲットオブジェクト名の名前空間識別子でユーザー名を作成して、ネーミングされます。たとえば、名前 user/sophia/service/calendar では、ユーザー sophia に対するカレンダがネーミングされます。ユーザーに関連付けてオブジェクトにネーミングする際の詳細は、477ページの「ユーザーの名前空間」および 、487ページの「エンタープライズのルートとユーザー」を参照してください。

エンタープライズのルートとホスト

ホストは、次のものに関連付けてネーミングできます。

- 組織単位
- エンタープライズのルート

  エンタープライズのルートに関連付けてネーミングされたホストは、最上位の組織単位に関連付けてネーミングされたホストと同一になります。組織名の場合は、名前空間識別子の host または _host のどちらかを追加して、hostname のコンテキストに名前を作成できます。したがって、orgunit/west.sales で組織がネーミングされる場合は、名前 org/west.sales/host/altair で west.sales 組織単位にあるマシン altair がネーミングされます。

  次のオブジェクトは、ホスト名に関連付けてネーミングできます。

- ホストに関連するサービス

- ホストによってエクスポートされたファイル

  これらのオブジェクトは、ターゲットオブジェクトの名前空間とターゲットオブジェクト名の名前空間識別子でホスト名を作成して、ネーミングされます。たとえば、名前 host/sirius/fs/release では、マシン sirius によってエクスポートされたファイルディレクトリ release がネーミングされます。(ホストに関連付けてオブジェクトにネーミングする際の詳細は、476ページの「ホストの名前空間」および480ページの「ホストに関連する名前を作成する」を参照してください。)

エンタープライズのルートとサービス

サービスは、次のものに関連付けてネーミングできます。

- 組織単位
- エンタープライズのルート
- ユーザー
- ホスト
- サイト

  エンタープライズのルートに関連付けてネーミングされたサービスは、最上位の組織単位に関連付けてネーミングされたサービスと同一になります。

  サービスのコンテキストは、名前空間識別子 service または _service を使用して、関連する組織、サイト、ユーザー、またはホストに関連付けてネーミングされます。たとえば、orgunit/corp.finance で組織単位がネーミングされる場合は、orgunit/corp.finance/service/calendar で組織単位 corp.finance の calendar サービスがネーミングされます。ユーザーの名前空間および、ユーザーに関連付けてオブジェクトをネーミングする際の詳細は、477ページの「サービスの名前空間」および、481ページの「サービスおよびファイルに関連する名前を作成する」を参照してください。

  FNS では、サービスの名前空間でのバインドのタイプが制限されるわけではありません。アプリケーションは、サービスのコンテキスト以外のタイプのコンテキストを作成し、サービスの名前空間でバインドできます。

  FNS は、サービスのコンテキストに「汎用」コンテキストを作成することをサポートします。汎用コンテキストは、アプリケーション決定のリファレンスタイプを持つこと以外は、サービスのコンテキストに類似しています。汎用コンテキストの他のすべての属性は、サービスのコンテキストと同一です。

たとえば、World Intrinsic Designs Corp (WIDC) という名前の会社は、サービスの名前空間で名前 extcomm を獲保し、製品の外部通信ラインに関連するバインドを追加するための汎用コンテキストを参照します。extcomm にバインドされるコンテキストは、リファレンスタイプ WIDC_comm を持つ汎用コンテキストです。このコンテキストとサービスのコンテキストの違いは、このコンテキストが異なるリファレンスタイプを持っていることだけです。

サービス名は、478ページの「サービス名およびリファレンスの登録」で説明しているように、米国 Sun Microsystems, Inc. に登録する必要があります。

エンタープライズのルートとファイル

ファイルの名前空間は、次のものに関連付けてネーミングできます。

- エンタープライズのルート
- 組織単位
- ユーザー
- ホスト
- サイト

エンタープライズのルートに関連付けてネーミングされたファイルは、最上位の組織単位に関連付けてネーミングされたファイルと同一になります。ファイルのコンテキストは、名前空間識別子の fs または _fs を使用して、関連する組織、サイト、ユーザー、またはホストに関連付けてネーミングされます。たとえば、orgunit/accountspayable.finance で組織単位がネーミングされる場合は、名前 user/jsmith/fs/report96.doc でファイル report96.doc がネーミングされます。ユーザーのファイルサービスは、NIS+ の passwd テーブルで指定されているように、デフォルトではホームディレクトリになります。ユーザーの名前空間の詳細は、477ページの「ファイルの名前空間」を参照してください。

この他には、ファイルシステムに従属するコンテキストのタイプはありません。

エンタープライズのルートとプリンタ

プリンタのコンテキストは、XFN ポリシーの拡張です。

プリンタの名前空間は、サービスのコンテキストでネーミングできます。プリンタのコンテキストは、名前空間識別子の printer を使用して、次のものに関連したサービスのコンテキストでネーミングされます。

- 組織単位
- ユーザー
- ホスト
- サイト

  たとえば、org/east.sales で組織単位がネーミングされる場合は、org/eastsales/service/printer では組織単位 east.sales のプリンタのサービスがネーミングされます。したがって lp1 という名前の固有のプリンタは、次のように識別されます。

  org/east.sales/service/printer/lp1

  この他には、プリンタに従属するコンテキストのタイプはありません。

エンタープライズ内のネーミングに対する初期コンテキストのバインド

初期コンテキストは、クライアントのユーザー、ホスト、およびアプリケーションがエンタープライズ名前空間の任意のオブジェクトに (結果的に) ネーミングできる開始点です。

図 22–3 に示すネーミングシステムは、初期コンテキストのバインドが影付きでイタリックになっている点を除いて図 22–2 に挙げたネーミングシステムと同一です。これらの初期コンテキストは、解決する名前を決定するユーザー、ホスト、またはアプリケーションの観点から示されています。

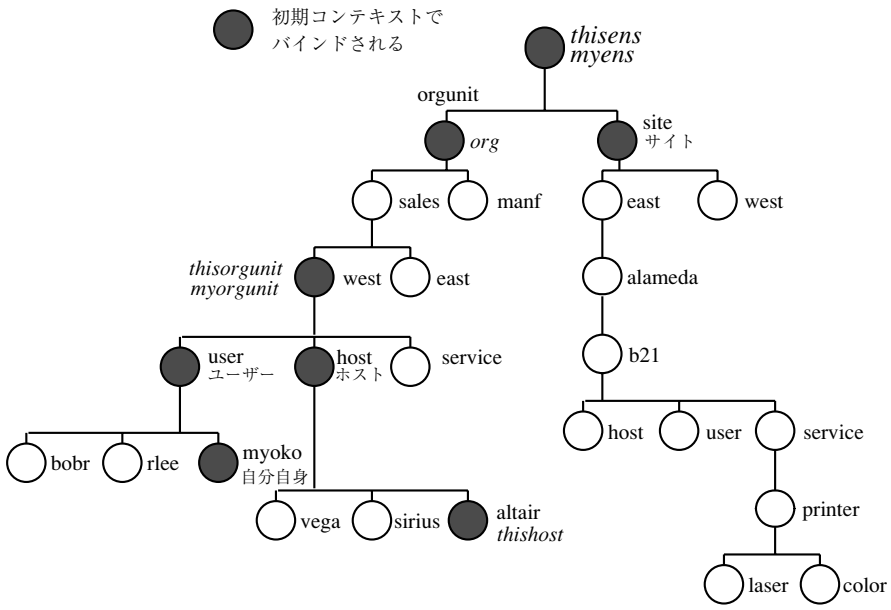


図 22–3　初期コンテキストのエンタープライズバインドの例

XFN では、「初期コンテキスト関数」の `fn_ctx_handle_from_initial()` が提供され、クライアントは初期コンテキストのオブジェクトを名前解決の開始点として得ることが可能になります。初期コンテキストには、名前空間識別子に対するフラットな名前空間があります。これらの初期コンテキスト識別子のバインドについては表 22–3 に要約し、その後の節でさらに詳細に説明しています。これらの名前は、必ずしもすべての初期コンテキストに表示されるわけではありません。たとえば、スーパーユーザーがプログラムを起動したときには、ホストとクライアントに関連するバインドのみが初期コンテキストに表示され、ユーザーに関連するバインドは表示されません。

表 22–3　エンタープライズ内のネーミングに対する初期コンテキストのバインド

名前空間 識別子	バインド
`myself`, `_myself`, `thisuser`	名前解決しようとするユーザーのコンテキスト
`myens`, `_myens`	名前解決しようとするユーザーのエンタープライズのルート

表 22–3　エンタープライズ内のネーミングに対する初期コンテキストのバインド　続く

名前空間識別子	バインド
myorgunit, _myorgunit	ユーザーの主要な組織単位のコンテキスト。たとえば、NIS+ 環境では、主組織単位はユーザーの NIS+ ホームドメインとなる
thishost, _thishost	名前解決しようとするホストのコンテキスト
thisens, _thisens	名前解決しようとするホストのエンタープライズのルート
thisorgunit, _thisorgunit	ホストの主要な組織単位のコンテキスト。たとえば、NIS+ 環境では、主組織単位はホストの NIS+ ホームドメインとなる
user, _user	ホストと同じ組織単位にいるユーザーがネーミングされるコンテキスト
host, _host	ホストと同じ組織単位にいるホストがネーミングされるコンテキスト
org, orgunit, _orgunit	ホストのエンタープライズにある組織単位の名前空間のルートコンテキスト。たとえば、NIS+ 環境では、これは NIS+ ルートドメインに対応する
site, _site	サイトの名前空間が構成された場合、最上位の組織単位にあるサイトの名前空間のルートコンテキスト

XFN 用語では、接頭辞として使用される下線は、「正規」名前空間識別子と呼ばれます。この下線がなく、org および thisuser が追加された名前は、Solaris 用にカスタマイズされた名前です。Solaris カスタマイズの名前空間識別子は、他の Solaris 以外の環境で認識されるとは限りません。正規名前空間識別子は、他の環境でも必ず認識されるため移植性があります。

注 - 現在実装されている FNS では、初期コンテキストにある名前およびバインドの追加や修正はサポートされません。

初期コンテキストのバインドは、基本的に次の 3 つに分けられます。

- ユーザーに関連するバインド (493ページの「ユーザーに関連するバインド」を参照)

- ホストに関連するバインド (494ページの「ホストに関連するバインド」を参照)
- 短縮形のバインド (495ページの「短縮形のバインド」を参照)

ユーザーに関連するバインド

FNS では、XFN 初期コンテキスト関数が呼び出されたときに、プロセスに関連付けられたユーザーが存在するものと仮定されます。この関連付けは、プロセスの実効ユーザー ID (*euid*) に基づいています。プロセスに対するユーザーの関連付けがプロセスの途中で変更されることはありますが、元のコンテキストハンドルは変更されません。

FNS では、名前解決を要求するユーザーに関連する初期コンテキストにある次のバインドが定義されます。

myself

初期コンテキストにある名前空間識別子 `myself` (あるいは同義語の `_myself` または `thisuser` のどちらか) は、要求を行うユーザーのコンテキストで解決されます。ユーザー `jsmith` が所有するプロセスが名前解決を要求したときの例を示します。

- 名前 `myself` は、`jsmith` のユーザーのコンテキストとして初期コンテキストで解決される
- `myself/fs/.cshrc` は、`jsmith` のファイル `cshrc` にネーミングする

myorgunit

FNS では、各ユーザーがエンタープライズの組織単位に関連するものと仮定されます。1 人のユーザーを複数の組織単位に関連させることはできますが、組織の名前空間での位置または組織内でのユーザーの役割によって、主な組織単位が必ず 1 つ必要です。

- 「NIS+」

  NIS+ 名前空間では、この組織単位はユーザーのホームドメイン (これは、サブドメインになることもある) に対応する

- 「NIS」

  NIS 名前空間では、ユーザーのドメインに対応するエンタープライズレベルの組織単位は 1 つしかない

- 「ファイル」

  ファイルベースの名前空間では、組織単位は myorgunit にマップする org// しかない

  名前空間識別子 myorgunit (または _myorgunit) は、要求を行うユーザーが所属する主組織単位のコンテキストとして初期コンテキストで解決されます。たとえば、要求を行うユーザーが jsmith で、jsmith のホームドメインが east.sales である場合は、myorgunit は east.sales の組織単位コンテキストとして初期コンテキストで解決され、名前 myorgunit/service/calendar は east.sales のカレンダのサービスで解決されます。

myens

FNS では、各ユーザーがエンタープライズに関連するものと仮定されます。これは、myorgunit を保持する名前空間に対応します。

名前空間識別子 myens (および _myens) は、要求を行うユーザーが所属するエンタープライズのルートとして初期コンテキストで解決されます。たとえば、jsmith が要求を行い、jsmith の NIS+ ホームドメインが east.sales である場合、それは doc.com. のルートドメイン名を含む NIS+ 階層にあります。名前 myens/orgunit/ は、doc.com の最上部の組織単位で解決されます。

注 - myorgunit または myself/service などのユーザーに関連する複合名を使用するときは、これらのバインドはプロセスの実効ユーザー ID に依存するため、ユーザー ID 設定プログラムに注意してください。ユーザー ID を root に設定して、呼び出し元に代わってシステムリソースにアクセスするプログラムでは、通常は fn_ctx_handle_from_initial() を呼び出す前に seteuid(getuid()) を呼び出してください。

ホストに関連するバインド

XFN 初期コンテキスト関数が呼び出されたときには、特定のホストでプロセスが実行中です。FNS では、プロセス実行中のホストに関連する初期コンテキストにある次のバインドが定義されます。

thishost

名前空間識別子 thishost (または _thishost) は、プロセス実行中のホストのホストコンテキストにバインドされます。たとえば、プロセスがマシン cygnus で実行中である場合、thishost は cygnus のホストのコンテキストにバインドされ、名前 thishost/service/calendar はマシン cygnus のカレンダのサービスを参照します。

thisorgunit

FNS では、ホストは組織単位に関連付けられていると仮定します。1 つのホストを複数の組織単位に関連付けることができますが、主となる組織単位が必要です。NIS+ 名前空間では、組織単位はホストのホームドメインに対応します。

名前空間識別子 thisorgunit (または _thisorgunit) は、プロセス実行中のホストの主要な組織単位に解決されます。たとえば、ホストがマシン cygnus であり、cygnus の NIS+ ホームドメインが west.sales である場合は、thisorgunit は west.sales として組織単位コンテキストで解決され、名前 thisorgunit/service/fax は組織単位 west.sales の FAX サービスを参照します。

thisens

FNS では、ホストがエンタープライズに関連するものと仮定されます。これは、thisorgunit を保持する名前空間に対応します。

名前空間識別子 thisens (または _thisens) は、プロセス実行中のホストのエンタープライズのルートで解決されます。たとえば、NIS+ で、ホストのホームドメインが sales.doc.com である場合、名前 thisens/site/ は doc.com. のサイトの名前空間のルートで解決されます。

短縮形のバインド

FNS では、初期コンテキストにある次の短縮形バインドを定義し、短い名前を使用して、共通に参照される特定の名前空間を参照できます。

user

名前空間識別子 user (または _user) は、プロセス実行中のホストの組織単位にある username コンテキストとして初期コンテキストにバインドされます。これにより、同じ組織単位にいる他のユーザーをこのコンテキストからネーミングできます。

初期コンテキストから、名前 user および thisorgunit/user は同じコンテキストで解決されます。たとえば、プロセス実行中のホストがマシン altairであり、altair が east.sales 組織単位にある場合は、名前 user/medici で east.sales にいるユーザー medici がネーミングされます。

host

名前空間識別子 host (または _host) は、プロセス実行中のホストの組織単位にある hostname コンテキストとして初期コンテキストにバインドされます。これにより、同じ組織単位にいる他のホストをこのコンテキストからネーミングできます。

初期コンテキストから、名前 host および thisorgunit/host は同じコンテキストで解決されます。たとえば、プロセス実行中のホストがマシン sirius であり、sirius が east.sales 組織単位にある場合は、名前 host/sirius で east.sales にあるマシン sirius がネーミングされます。

org

名前空間識別子 org (または orgunit、_orgunit) は、プロセス実行中のホストが所属するエンタープライズの組織のルートコンテキストとして初期コンテキストでバインドされます。

初期コンテキストから、名前 org および thisens/orgunit は同じコンテキストで解決されます。たとえば、プロセス実行中のホストがマシン aldebaran であり、aldebaran がエンタープライズ doc.com にある場合は、名前 org/east.sales で doc.com. にある組織単位 east.sales がネーミングされます。

site

名前空間識別子 site (または _site) は、プロセス実行中のホストが所属するエンタープライズの最上位の組織単位のサイトネーミングシステムのルートに対する初期コンテキストにバインドされます。

初期コンテキストから、名前 site および thisens/site は同じコンテキストで解決されます。たとえば、プロセス実行中のホストがマシン aldebaran であ

り、aldebaran がエンタープライズ doc.com にある場合は、名前 site/pine.bldg-5 で doc.com. の建物 5 にある会議室 pine がネーミングされます。

FNS およびエンタープライズのレベルのネーミング

FNS では、基本ネーミング操作の単一の簡単なインタフェースで、複数のネームサービスをフェデレートする方法が提供されます。FNS は、次の 3 つのエンタープライズレベルのネームサービスで機能するように設計されています。

- 「NIS+」(497ページの「FNS ポリシーと NIS+ との関連」および461ページの「FNS と NIS+ のネーミング」を参照)
- 「NIS」(500ページの「FNS ポリシーと NIS の関連」および462ページの「FNS と NIS のネーミング」を参照)
- 「ファイル」(501ページの「FNS ポリシーとファイルベースのネーミングの関連」および463ページの「FNS とファイルベースのネーミング」を参照)

  また FNS は、502ページの「FNS ポリシーの対象となるクライアントアプリケーション」で説明しているようなプリンタおよびカレンダサービスなどのアプリケーションで機能するようにも設計されています。

FNS ポリシーと NIS+ との関連

FNS および NIS+ に関連する内容説明の概要については、461ページの「FNS と NIS+ のネーミング」を参照してください。NIS+ およびその用語を調べる場合は、このマニュアルのパート I「ネーミングの紹介」および用語集を参照してください。一般的な NIS+ 環境の構造を学ぶときに便利です。

FNS では、NIS+ サーバー上のドメインレベルの org_dir NIS+ ディレクトリにある FNS テーブルに、エンタープライズのオブジェクトに対するバインドが格納されます。FNS テーブルは NIS+ テーブルに類似しています。これらの FNS テーブルには、次のエンタープライズ名前空間に対するバインドが格納されます。

- 498ページの「NIS+ ドメインと FNS 組織単位」で説明する「組織」の名前空間
- 499ページの「NIS+ ホストと FNS ホスト」で説明する「ホスト」の名前空間
- 499ページの「NIS+ ユーザーと FNS ユーザー」で説明する「ユーザー」の名前空間

- 組織、ホスト、ユーザーに関連付けて地域のサイトにネーミング可能な「サイト」の名前空間
- 組織、ホスト、ユーザーに関連付けてプリンタやカレンダなどのサービスにネーミング可能な「サービス」の名前空間

NIS+ ドメインと FNS 組織単位

FNS では、優先 Solaris エンタープライズのネームサービスである NIS+ 内の組織、ユーザー、ホストのエンタープライズオブジェクトがネーミングされます。NIS+ ドメインは、ユーザーおよびマシンの論理コレクションとそれらについての情報で構成され、エンタープライズ内の階層組織構造のフォームを反映するように配列されます。

FNS は、NIS+ ドメインを FNS 組織にマッピングして、NIS+ で実行されます。組織単位名は、NIS+ ドメイン名に対応しており、完全指定された形式の NIS+ ドメイン名または NIS+ ルートに関連した NIS+ ドメイン名のどちらかを使用して識別されます。最上位の FNS 組織名前空間は、NIS+ ルートドメインにマップされ、初期コンテキストから名前 `org/` を使用してアクセスされます。

NIS+ では、ユーザーおよびホストに「ホームドメイン」の概念があります。ホストまたはユーザーのホームドメインは、それらの関連付けられた情報を保持する NIS+ ドメインです。ユーザーまたはホストのホームドメインは、直接 NIS+「主体名」を使用して決定できます。NIS+ 主体名は、原子ユーザー (ログイン) 名または原子ホスト名と、NIS+ ホームドメイン名からなります。たとえば、ホームドメイン `doc.com.` を持つユーザー `sekou` には NIS+ 主体名 `sekou.doc.com` が付けられ、マシン名 `vega` には NIS+ 主体名 `vega.doc.com` が付けられます。

ユーザーの NIS+ ホームドメインは、ユーザーの FNS 組織単位に対応します。同様に、ホストのホームドメインは、その FNS 組織単位に対応します。

組織名の後のドット

組織名の後のドットは、その名前が完全指定の NIS+ ドメイン名であることを示します。このドットがない場合は、その組織名は NIS+ ルートドメインに関連付けて解決される NIS+ ドメイン名です。

たとえば、NIS+ ルートドメインが `doc.com.` であり、サブドメイン `ales.doc.com.` を含む場合は、次の名前で同じ組織が参照されます。

表 22–4　NIS+ での相対および完全指定の組織名の例

相対名	完全指定名
org/	org/doc.com.
org/sales	org/sales.doc.com.

manf という名前だけを含む NIS+ ドメインは存在しないため、名前 org/manf. (ドット付き) はありません。

NIS+ ホストと FNS ホスト

NIS+ 名前空間のホストは、ホストのホームドメインの hosts.org_dir テーブルにあります。FNS 組織にあるホストは、対応する NIS+ ドメインの hosts.org_dir テーブル内のホストに対応します。FNS では、hosts テーブル内の各ホストにコンテキストが提供されます。

NIS+ ユーザーと FNS ユーザー

NIS+ 名前空間にいるユーザーは、ユーザーのホームドメインの passwd.org_dir テーブルのリストに入っています。FNS 組織にいるユーザーは、対応する NIS+ ドメインの passwd.org_dir テーブル内のユーザーに対応します。FNS では、passwd テーブル中の各ホストにコンテキストが提供されます。

NIS+ セキュリティと FNS

FNS の fncreate コマンドを使用して、コマンドが実行されたホストのドメインに関連付けられた NIS+ 階層に FNS テーブルとディレクトリを作成します。fncreate を実行するには、そのドメインの NIS+ オブジェクトの読み取り、作成、変更、削除を許可する資格を得て、認証された NIS+ 主体になる必要があります。fncreate で作成した FNS テーブルの「所有者」になります。この許可を得る 1 つの方法は、ドメインの管理者特権を持つ NIS+ グループのメンバーになることです。

fncreate を実行する前に、NIS_GROUP 環境変数をドメインに対する NIS+ 管理グループ名に設定する必要があります。個別のユーザーが、ユーザーに関連する FNS データを変更可能かどうかを指定できます。

NIS+ セキュリティの説明については、第 6 章を参照してください。

FNS ポリシーと NIS の関連

FNS および NIS に関連する概要と内容説明については、462ページの「FNS と NIS のネーミング」を参照してください。

FNS では、NIS をネームサービスとして使用する基本的なネーミングおよび属性の操作用に XFN インタフェースが提供されます。

FNS では、NIS マスターサーバー (および存在する場合は NIS スレーブサーバー) `/var/yp/`*domainname* ディレクトリにある FNS マップのエンタープライズオブジェクトへのバインドが格納されます。FNS マップは、構造と機能の面で FNS マップと類似しています。これらの NIS マップでは、次のエンタープライズ名前空間に対するバインドが格納されます。

- エンタープライズ全体に関連するオブジェクトをネーミングするための名前空間を提供する「組織」。NIS がネームサービスの下にあるときは、NIS ドメインに対応する 1 つの組織単位コンテキストがある。この組織単位コンテキストは、NIS ドメイン名または、マシン NIS ドメイン名をデフォルトとする空白名によって FNS で識別される。
- NIS ドメインの `hosts.byname` マップに対応する「ホスト」の名前空間。FNS では、`hosts.byname` マップにある各ホストにコンテキストが提供される
- `passwd.byname` マップに対応する「ユーザー」の名前空間。FNS では、ドメインの `passwd.byname` にある各ユーザーにコンテキストが提供される
- 組織、ホスト、ユーザーに関連する地域サイトにネーミングするための「サイト」の名前空間
- 組織、ホスト、ユーザーに関連付けてプリンタやカレンダなどのサービスにネーミングするための「サービス」の名前空間

  FNS では、他のオブジェクトをこれら 5 つの名前空間に関連付けてネーミングするためのコンテキストが提供されます。

  FNS の `fncreate` コマンドを使用して、NIS マスターサーバーの `/var/yp/`*domainname* ディレクトリに FNS マップを作成します。これは、NIS ネームサービスのマスターサーバーと同じマシンか、または FNS マスターサーバーとして機能する異なるマシンで行えます。スレーブサーバーが存在する場合は、NIS は、通常の処理の一部として FNS マップをそれらにプッシュしま

す。fncreate を実行するには、FNS マップのホストとなるサーバーの特権ユーザーになる必要があります。各ユーザーは、FNS データを変更できません。

FNS ポリシーとファイルベースのネーミングの関連

FNS とファイルに関連する概要および内容説明については、463ページの「FNS とファイルベースのネーミング」を参照してください。

FNS では、ローカルのファイルをネームサービスとして使用する基本的なネーミングおよび属性の操作用に XFN インタフェースが提供されます。

FNS では、通常各マシンに NFS マウントされた /var/fn ディレクトリにあるファイル内のエンタープライズオブジェクトに対するバインドが格納されます。これらの FNS ファイルでは、次のエンタープライズ名前空間に対するバインドが格納されます。

- エンタープライズ全体に関連付けてオブジェクトをネーミングする名前空間を提供する「組織」。ローカルのファイルがネームサービスの下にある場合は、システム全体を表す 1 つの組織単位コンテキストがある。この組織単位コンテキストは、FNS では常に org// として識別される
- /etc/hosts ファイルに対応する「ホスト」の名前空間。FNS では、/etc/hosts ファイルにある各ホストにコンテキストが提供される
- /etc/passwd ファイルに対応する「ユーザー」の名前空間。FNS では、/etc/passwd ファイルにある各ユーザーにコンテキストが提供される
- 組織、ホスト、ユーザーに関連付けて地域サイトをネーミングするための「サイト」の名前空間
- 組織、ホスト、ユーザーに関連付けてプリンタやカレンダなどのサービスにネーミングするための「サービス」の名前空間

  FNS では、他のオブジェクトをこれら 5 つの名前空間に関連付けてネーミングするためのコンテキストが提供されます。

  FNS の fncreate コマンドによって、コマンドが実行されるマシンの /var/fn ディレクトリに FNS ファイルを作成します。fncreate を実行するには、そのマシンでのスーパーユーザー特権を持つ必要があります。UNIX ユーザー ID に基づいて、各ユーザーは、FNS コマンドを使用して自身のコンテキスト、バインド、属性を変更することが許可されます。

FNS ポリシーの対象となるクライアントアプリケーション

FNS ポリシーの 1 つの目的は、ファイルシステム、およびカレンダマネージャ、印刷ツール、ファイルマネージャ、メールツールなどの DeskSet ツール、またこれらのツールをサポートする RPC、電子メール、印刷サブシステムなどのサービスなどの、共通して使用されるツールに一貫性を持たせることです。

注 - これらの例のいくつかは、現在 Solaris 環境には導入されていませんが、FNS の使用方法を示すためにここに挙げます。

- 「カレンダ」

  「username@hostname」の形の名前を使用して誰かのカレンダにアクセスする代わりに、たいていの場合は単にサイト、組織、またはユーザー名を入力する。カレンダをネーミングするときには、複合名を使用することもできる。たとえば、FNS でフェデレートするときには、次の形式の名前がカレンダマネージャで使用可能となる

  - bernadette
  - user/bernadette
  - site/pine.bldg-5 (Pine 会議室用のカレンダ)
  - org/sales (sales 組織用のカレンダ)

- 「印刷」

  特定のプリンタを名前でネーミングする代わりに、ユーザー、サイト、または組織に関連付けてプリンタをネーミングできる。次のようなものがある

  - ilych (ilych のデフォルトのプリンタ)
  - org/sales (組織のデフォルトのプリンタ)
  - site/pine.bldg-5 (Pine 会議室のプリンタ)

- 「ファイルアクセス」

  ファイルシステムやファイルをネーミングするときに複合名を使用できる。オートマウンタは、FNS を使用して複合名の解決を可能にする。たとえば、/xfn/user/baruch/fs/.cshrc などのファイル名を使用して、ユーザー baruch の .cshrc ファイルを参照できる

- 「RPC」

サービスをホスト名、プログラム、およびバージョン番号でアドレス指定する代わりに、複合名を使用してサービスをネーミングできる。たとえば、user/hatori/service/rpc のようにユーザーまたは組織に関連付けて RPC サービスをネーミングできる

- 「メール」

  同様に、複合名はメールの宛先をネーミングするために使用される。次のような名前を使用できる

  - angus
  - user/angus
  - org/mlist (組織のメールリスト)
  - site/pine.bldg-5 (会議室の調整係のメールボックス)

- 「他のデスクトップアプリケーション」

  スプレッドシート、文書作成ツール、FAX ツールなどの他のデスクトップアプリケーションに複合名を引き渡すことができる。これらのアプリケーションには、それ自身の名前空間をサービスの名前空間に接続し、FNS フェデレーションの一部となるものもある

アプリケーションの例 - カレンダサービス

ここでは、カレンダサービスというアプリケーションを変更して、FNS ポリシーを使用する方法について説明します。この例では、ユーザーから FNS 複合名が提示され、承認される手順を示します。

DeskSet のカレンダサービスは、一般的なクライアントサーバーアプリケーションです。カレンダサーバーは、複数のマシンで動作し、ユーザーのカレンダを保持します。カレンダマネージャ (cm) は、デスクトップで動作し、適切なサーバーにコンタクトして必要なカレンダを入手します。

カレンダサービスは、次のような簡単なレジストリまたは検索のモデルを使用して FNS を利用します。

- 「バインド」

  起動時に、サーバーは、user/jsmith/service/calendar などの管理するそれぞれのカレンダの複合名に対するそれ自身の ONC+ RPC アドレス (ホスト、プログラム、バージョン) を含むリファレンスをバインドし、サーバーの管理するカレンダを登録する

■ 「検索」

cm を使用するときには、ユーザーは、単にユーザー名 (hirokani など) を入力するか、または以前に入力した名前のリストからユーザーを選択して、他のユーザーのカレンダを指定する。ユーザー名が hirokani の場合は、cm で複合名 user/hirokani/service/calendar が作成され、これを使用してカレンダを管理するサーバーと通信する必要のある RPC アドレスが検索される

前の例では、名前 calendar を使用して、カレンダのバインドを示しています。RPC プログラムを RPC 管理者に登録するのと同じように、カレンダサービスの開発者は、名前 calendar を FNS 管理者に登録します。478ページの「サービス名およびリファレンスの登録」を参照してください。

注 - ここで使用した名前 calendar は 1 つの例です。FNS ポリシーによって、特定のサービス名が指定されるわけではありません。

カレンダサービスでは、カレンダをサイト、組織、およびホストに関連付け、一定の方法でそれらをネーミングして、FNS ポリシーをさらに利用できます。たとえば、カレンダを会議室 (サイト) と関連付け、ユーザーのカレンダのセットでその部屋の会議に利用可能な時間を見つけるのと同じように、サービスを使用して会議室のカレンダを多重表示できます。同様に、カレンダは、グループ会議の組織や、保守スケジュールを管理するホストに関連付けることができます。

カレンダマネージャ (cm) は、いくつかの簡単な手順に従って、ユーザーが指定する必要のあることを明確にします。

1. cm では、ユーザーからの複合名を承認して、カレンダが要求されたオブジェクト名を構成するためのツールが使用されます。

   オブジェクトは、ユーザー、サイト、ホスト、または組織の名前になります。たとえば、ユーザーが名前 kuanda を入力すると、カレンダマネージャで複合名 user/kuanda が生成されます。このツールは、DeskSet アプリケーションのグループ間で共有できます。
2. cm は、XFN インタフェースを使用して、接尾辞 /service/calendar を含む名前を作成し、カレンダ名を入手します。
3. このカレンダ名は、プロセスの初期コンテキストに関連付けて解決されます。

   続いて、名前 user/kuanda/service/calendar が解決されます。同様に、ユーザーがサイト名 pine.bldg-5 を入力した場合は、cm で名前 site/pine.bldg-5/service/calendar が生成され解決されます。

印刷やメールなどの他のサービスでも、類似の方法で FNS ポリシーを利用できます。

FNS ファイルシステム名前空間

FNS では、ファイル名はユーザー、ホスト、組織、サイトとの関係から設定できます。この場合実際には、オブジェクトの名前の後にエンタープライズの名前空間識別子 `fs` を指定し、その後にファイル名を指定します。たとえば、Sales 部門の、`budget` というファイルなら、`org/sales/fs/budget/draft96`という名前になります。

初期コンテキストは、ルートディレクトリの `/xfn` の下にあります。したがって、以下のように入力して参照できます。

```
% more /xfn/org/sales/fs/budget/draft96
```

アプリケーションからこのディレクトリへアクセスする方法は、他のディレクトリの場合とまったく同様です。アプリケーションを修正したり、XFN API を使用したりする必要はありません。

NFS ファイルサーバー

NFS は Sun Microsystems, Inc. の製品で、分散ファイルシステムの一種です。オブジェクトのファイルは、通常 1 つ以上のリモート NFS ファイルサーバーにあります。最も単純なのは、エクスポートされた NFS ファイルシステムに名前空間識別子 `fs` が対応するという図 22–4 のような場合です。

図 22–4　NFS ファイルシステム - 単純な場合

しかしオブジェクトのファイルシステムの中には、複数の (重なり合う場合もある) リモートマウントで構成され、FNS によって管理される仮想ディレクトリ構造にまとめられているものもあります。

図 22–5 は、3 つの異なるファイルサーバーをまとめて、組織のファイルシステムを 1 つ作成している例です。`project` ディレクトリと `lib` サブディレクトリは同じファイルサーバー上に常駐していますが、`src` サブディレクトリの常駐先は別です。しかしユーザー、およびアプリケーションの側からは 1 つのシームレスな名前空間に見えるので、複数のサーバーを使用していることを意識する必要はありません。

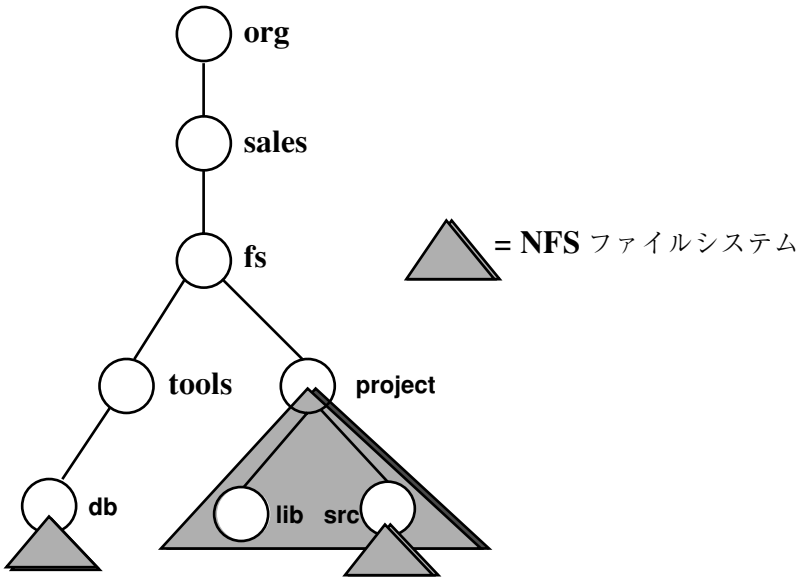


図 22–5　NFS ファイルシステム - 複数サーバーで構成されている場合

オートマウンタ

効率を上げるため、FNS ディレクトリのマウントには必要に応じてオートマウンタが使用されます。`/etc/auto_master` 構成ファイルには、デフォルトでは以下のような行が含まれます。

```
/xfn -xfn
```

これは、「FNS 名前空間が、XFN の指定どおり `/xfn` の下にマウントされる」という意味です。

オートマウンタは、FNS によって指定されたディレクトリのマウントに使用されるため、FNS ディレクトリのサブディレクトリは、マウントされるまで表示できません。たとえば、sales 組織のファイルシステムが複数のファイルシステムで構成されているとします。以下の `ls` コマンドは最近アクセスして現在もマウントされている 2 つのファイルシステムだけを表示します。

```
% ls /xfn/org/sales/fs
customers products
```

完全なリストを表示するには、`fnlist` コマンドを使用します。

```
% fnlist org/sales/fs
Listing 'org/sales/fs':
products
goals
customers
incentives
```

FNS プリンタの名前空間

`printer` コンテキストは、XFN ポリシーには含まれていません。FNS に提供されているのは、プリンタバインディングを保存するためです。

FNS では、FNS 名前空間にプリンタバインディングを保存するための機能が提供されています。この機能によって、プリントサーバーは自分自身のサービスをユーザーに知らせることができ、ユーザーはクライアント側の管理がなくてもプリンタのブラウズおよび選択ができます。

プリンタバインディングは、組織、ユーザー、ホスト、サイトごとに存在する `printer` コンテキストに格納されます。したがって、組織、ユーザー、ホスト、サイトは、それぞれ専用の `printer` コンテキストを所有できます。

`printer` コンテキストは、個々の複合名の `service` コンテキストの下に作成されます。例を以下に示します。

```
org/doc.com./service/printer
```

ホストのプリンタ名が `deneb` である場合、`printer` コンテキストは以下のように作成されます。

```
host/deneb/service/printer/laser
```

グローバル名前空間のポリシー

グローバルネームサービスには、固有の適用範囲があります。ここでは、グローバルネームサービスを使用するオブジェクトをネーミングするためのポリシーについて説明します。

ネーミングに関しては、エンタープライズは、グローバル名前空間にあるエンタープライズのルートをバインドして、フェデレートされたグローバル名前空間にリンクします。これにより、エンタープライズ外部のアプリケーションおよびユーザーがそのエンタープライズ内部のオブジェクトをネーミングできます。たとえば、エンタープライズ内部のユーザーは、他のエンタープライズにいる同僚にファイルのグローバル名を渡すことができます。

DNS および X.500 のコンテキストは、エンタープライズをネーミングするためのグローバルレベルのネームサービスを提供します。FNS では、DNS と X.500 コンテキストの両方がサポートされます。FNS がない場合、DNS および X.500 では、限定された部分のエンタープライズ名前空間だけに外部からアクセスできます。FNS では、カレンダマネージャなどのサービスを含むエンタープライズ名前空間全体に外部からアクセスできます。

グローバルネーミングに対する初期コンテキストのバインディング

原子名「...」(3 つのドット) は、各 FNS クライアントの初期コンテキストに表示されます。原子名「...」は、グローバル名が解決されるコンテキストにバインドされます。

表 22–5　グローバルネーミングに対する初期コンテキストバインド

原子名	バインド
. . .	DNS または X.500 の名前を解決するためのグローバルコンテキスト
/ . . .	3 つのドットの同義語

グローバル名は、完全指定のインターネットドメイン名か、または X.500 の識別名になります。

- インターネットドメイン名は、インターネット RFC 1035 で指定された構文に表示される。たとえば、`. . . /doc.com` (509ページの「DNS のフェデレーティング」を参照) など
- X.500 名は、X/Open DCE ディレクトリで決定された構文に表示される。たとえば、`. . . /c=us/o=doc` など (510ページの「X.500/LDAP のフェデレーティング」を参照)

  名前「`...`」および「`/...`」は、初期コンテキストで解決されるときには等価になります。たとえば、名前 `/. . . /c=us/o=doc` および `. . . /c=us/o=doc` は、初期コンテキストで同じオブジェクトとして解決されます。

DNS のフェデレーティング

完全指定の DNS 名は、必ずグローバルコンテキストで使用できます。DNS 名がグローバル名前空間で見つかると、リゾルバライブラリを使用して解決されます。リゾルバライブラリは、DNS 名前解決メカニズムです。一般的 DNS 名は、インターネットのホストアドレスまたは DNS ドメインレコードで解決されます。グローバルコンテキストで DNS 名が検出されると、名前は DNS リゾルバに引き渡されて解決されます。結果は、XFN リファレンス構造に変換され、要求者に返されます。

DNS ドメインの内容は、表示できます。しかし、表示操作は、インターネットでの連結性とセキュリティなどの実用上の理由によって限定されることもあります。たとえば、DNS ドメインのグローバルルート表示は、一般的にルート DNS サーバーではサポートされません。しかし、ルートの下にあるたいていのエンティティは、表示操作をサポートします。

DNS ホストおよびドメインは、DNS リソース名と関連付けられたネームサービス (NS) のリソースレコードの有無によって確認されます。

- 「DNS ドメイン名」

  NS レコードがリソース名に存在する場合は、その名前はドメイン名であると考えられ、返されたリファレンスのタイプは `inet_domain` となる

- 「DNS ホスト名」

  NS レコードがリソース名に存在しない場合は、その名前はホスト名であると考えられ、返されたリファレンスのタイプは `inet_host` となる

DNS は、端末以外のネーミングシステムのように機能し、他のネーミングシステムをフェデレートするために使用されます。

たとえば、エンタープライズネーミングシステムは、FNS 名 `.../doc.com/` がそのエンタープライズの FNS 名前空間のルートを参照するような DNS にある `doc.com` にバインドできます。

エンタープライズのネーミングシステムは、適切なテキスト (TXT) レコードをそのドメインの DNS マップに追加して、DNS ドメインにバインドされます。そのドメインに対する FNS 名には後に続くスラッシュ (/) が含まれ、TXT リソースレコードは、エンタープライズのネーミングシステムへのリファレンスを構成するために使用されます。

DNS に関する一般的な情報については、`in.named(1M)` のマニュアルページか、または『*Solaris* ネーミングの設定と構成』の DNS についての章を参照してください。

X.500/LDAP のフェデレーティング

X.500 は、グローバルのディレクトリサービスです。ここでは、情報が格納され、情報を表示して検索する機能と同様に、名前で情報を検索する機能が提供されます。

X.500 の情報は、ディレクトリ情報ベース (DIB) に格納されます。DIB にあるエントリは、ツリー構造で配列されます。各エントリは名前付きオブジェクトで、定義された属性のセットを含んでいます。各属性には、定義済みの属性タイプと 1 つまたは複数の値があります。

エントリは、ルートから名前付きエントリまでのパスでツリー内の各エントリから選択された属性の連結である「識別名」によって明確に識別されます。たとえば、図 22–6 に示す DIB を使用すると、`c=us/o=doc` は、合衆国にある `doc` 組織の識別名となります。X.500 ディレクトリのユーザーは、DIB にあるエントリと属性を問い合わせたり、変更したりすることができます。

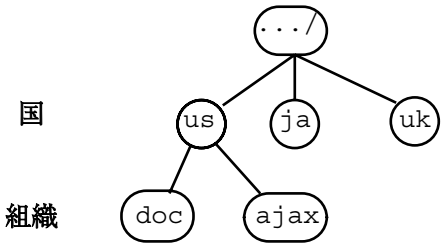


図 22–6　X.500 ディレクトリ情報ベースの例

FNS は、名前空間をグローバル X.500 名前空間の下にシームレスに接続して表示するために必要なサポートを提供し、X.500 をフェデレートします。

たとえば、FNS は、X.500 の下の `doc` 組織にエンタープライズネーミングシステムをリンクします。初期コンテキストから始まって、`doc` 組織の `sales` 組織単位を識別する FNS 名は次のようになります

```
.../c=us/o=doc/orgunit/sales
```

エンタープライズ内の名前は、単純にグローバルの X.500 名上に連結されます。FNS 名が初期コンテキストで名前「`...`」を使用して、グローバル名が後に続くことを示すことに注意してください。

FNS 名の名前解決は、次のように行われます。X.500 名がグローバル名前空間で見つかると、X.500 名前解決メカニズムを使用して解決されます。次の 3 つの結果が考えられます。

- フルネームが X.500 エントリに解決される。これは、エントリが X.500 に格納されていることを示す。要求された FNS 操作がそのエントリで実行される
- フルネームの接頭辞は、X.500 エントリに解決される。これは、名前の残りの部分が従属するネーミングシステムに所属することを示す

  従属するネーミングシステムへの次のネーミングシステムのポインタ (NNSP) が調べられ、XFN リファレンスが返されます。この後、名前解決は従属するネーミングシステムで続けられます。
- エラーが報告される

  X.500 エントリは、FNS 操作を使用して調べたり、変更したりできます (アクセス制御に従う)。しかし、現在は FNS を使用して X.500 名前空間のルートの下にある従属するエントリを表示できません。

第 23 章

FNS とエンタープライズのネームサービス

この章では、FNS およびエンタープライズレベルのネームサービスの管理について説明します。

FNS とエンタープライズレベルのネームサービス

エンタープライズレベルのネームサービスは、組織内のオブジェクトをネーミングするために使用されます。現在 FNS は、次の 3 つのエンタープライズレベルのネームサービスをサポートしています。

- NIS+ (461ページの「FNS と NIS+ のネーミング」および、497ページの「FNS ポリシーと NIS+ との関連」を参照)

- NIS (462ページの「FNS と NIS のネーミング」および、500ページの「FNS ポリシーと NIS の関連」を参照)
- ローカルファイル (463ページの「FNS とファイルベースのネーミング」および、501ページの「FNS ポリシーとファイルベースのネーミングの関連」を参照)

エンタープライズレベルのネームサービスを選択する

『*Solaris* ネーミングの設定と構成』で説明している `fncreate` コマンドで FNS 名前空間を初期設定して構成するときには、正しいデフォルトのネームサービスが自動的に各マシンに選択されます。

マシンの主要なエンタープライズレベルのネームサービスを後で変更する場合は、そのマシンで `fnselect` コマンドを実行する必要があります。詳細は、516ページの「ネームサービスを選択する」を参照してください。

FNS とネームサービスとの整合性

タスクの 1 つにシステム管理者の機能が割り当てられていて、FNS とその下層のネームサービス間の整合性を維持しています。これは、ネームサービスのファイル、マップ、またはテーブルと FNS コンテキストが正しく対応しているかどうかをチェックするということです。

『*Solaris* ネーミングの設定と構成』で説明している `fncreate` コマンドで FNS 名前空間を初期設定して構成するとき、`fncreate` では、FNS コンテキストが適切に作成され、下層のネームサービスのデータと整合されたことが確認されます。FNS コンテキストが設定された後、ユーザー、ホスト、プリンタなどがシステムに追加または削除されるときに、この対応関係は維持される必要があります。以下の項では、FNS とネームサービスとの整合性を維持する方法について説明します。

FNS と Solstice AdminSuite

Solstice AdminSuite 製品で、基本的なネームサービスでのユーザーとホストの情報を追加、変更、削除できます。AdminSuite ツールでは対応する FNS 名前空間が自動的に更新されるため、この方法をお勧めします。

ネーミングの不一致をチェックする

Solstice AdminSuite 製品を使用せずに FNS または主要なネームサービスを更新するときに、不一致が発生した場合は、FNS ツールの `fncheck` を使用して解決します。`fncheck` コマンドは、FNS の `hostname` と `user` のコンテキストと、次のものとの不一致をチェックします。

- 「NIS+」- NIS+ の `hosts.org_dir` と `passwd.org_dir` のシステムテーブル
- 「NIS」- NIS の `hosts.byname` と `passwd.byname` のマップ
- 「ファイル」- `etc/hosts` と `etc/passwd` のファイル

`fncheck` コマンドは、FNS 名前空間にあってネームサービスのデータにないホストとユーザー名およびネームサービスのデータにあって FNS 名前空間にないホストとユーザー名を表示します。

コマンド構文は以下のとおりです。

```
fncheck [-r] [-s] [-u] [-t hostname|username] [domain_name]
```

表 23–1 `fncheck` コマンドオプション

オプション	説明
domain	コマンドを実行しているドメイン以外の NIS+ ドメインにコマンドを適用する
`-t`	チェックするコンテキストのタイプを指定する。許容されるタイプは、`hostname` または `username`
`-s`	FNS 名前空間にない名前空間のデータセットからホスト名とユーザー名を表示する

表 23–1　fncheck コマンドオプション　続く

オプション	説明
–r	対応する名前空間のデータセットにないエントリを持たない FNS 名前空間からホスト名またはユーザー名を表示する
–u	関連した名前空間のデータセットにある情報に基づいて FNS 名前空間を更新する

–t オプションは、チェックするコンテキスト (ホストまたはユーザー) を指定するために使用します。–t オプションを省略した場合は、hostname と username のコンテキストの両方がチェックされます。

–r オプションを –u オプションとともに使用すると、FNS コンテキストにしか表示されない項目が FNS コンテキストから削除されます。–s オプションを –u オプションとともに使用すると、名前空間のデータセットにしか表示されない項目が FNS コンテキストに追加されます。–r と –s のどちらも指定しない場合は、項目が FNS コンテキストに追加および削除され、対応する名前空間のデータとの整合性が保たれます。

ネームサービスを選択する

FNS がマシンに対する初期コンテキストにバインドを構成するときには、特定のネームサービスに基づいて行います。

FNS では fnselect コマンドで使用するネームサービスを選択できます。fnselect で指定したネームサービスの設定は、マシン全体、そのマシンで動作するすべてのアプリケーション、およびそのマシンにログインしたすべてのユーザーに影響します。

スーパーユーザーだけが fnselect を実行できます。コマンド構文は以下のとおりです。

```
fnselect [-D] [namesvc]
```

表 23–2 fnselect コマンドオプション

オプション	説明
namesvc	選択するネームサービス。必ず次のどれかになる。default、nisplus、nis または files
–D	FNS 初期コンテキストを生成するために使用されるネームサービスを表示する

たとえば、マシンのネームサービスとして NIS+ を選択するには以下のコマンドを使用します。

```
# fnselect nisplus
```

たとえば、マシンのネームサービスとして default を選択し、FNS 初期コンテキストを生成するために使用されるサービスの名前を印刷するには次のように入力します。

```
# fnselect -D default
```

デフォルトのネームサービス

fnselect でネームサービスを指定しない場合、FNS はデフォルトのネームサービスを使用します。デフォルトのネームサービスは、マシンの使用するネームサービスに基づいて FNS で決定されます。マシンが NIS+ クライアントである場合は、FNS は NIS+ をネームサービスとして使用します。マシンが NIS クライアントである場合は、FNS は NIS を使用します。マシンが NIS+ および NIS クライアントのどちらでもない場合は、FNS は /etc ファイルをマシンのネームサービスとして使用します。

NIS+ と NIS が共存する場合

ごくまれに、NIS+ と NIS ベースのコンテキストの両方にアクセスする必要が生じることがあります。たとえば、それ自体が NIS+ クライアントである NIS サーバーを稼動している場合です。この場合、fnselect コマンドを使用して、使用するエンタープライズレベルのネームサービスを選択します。

FNS と NIS+ の詳細情報

この節では、NIS+ オブジェクトと FNS オブジェクトの関係の詳細について説明します。この情報は、FNS オブジェクトのアクセス制御を変更するときに有効です。

NIS またはファイルベースのネーミングから NIS+ への移行

次を参照してください。

- 522ページの「NIS から NIS+ への変更」
- 525ページの「ファイルベースのネーミングから NIS または NIS+ への変更」

FNS コンテキストを NIS+ オブジェクトにマップする

FNS コンテキストは、NIS+ オブジェクトに格納されます。組織に関連するすべてのコンテキストは、関連する NIS+ ドメインの `ctx_dir` ディレクトリに格納されます。`ctx_dir` ディレクトリは、同じドメインの `org_dir` と同じレベルにあります。すなわち、FNS と同時に実行される時には、それぞれの NIS+ ドメインまたはサブドメインには対応する `org_dir`、`groups_dir`、および `ctx_dir` というディレクトリオブジェクトが存在します。

`fnlookup` または `fnlist` のコマンドで `–v` オプションを使用して、リファレンスについての詳細な説明を表示します。内部名フィールドに、対応する NIS+ オブジェクトの名前が表示されます。

NIS+ コマンドを使用して FNS 構造を表示する

NIS+ コマンドの `nisls` を使用して、FNS で使用される NIS+ オブジェクトを表示できます。たとえば、次のコマンドでは、NIS+ ドメインのディレクトリおよび `ctx_dir` サブディレクトリの内容が表示されます。

```
# nisls doc.com.
doc.com.:
manf
sales
```

(続く)

```
groups_dir
org_dir
ctx_dir
```

```
# nisls ctx_dir.doc.com.
ctx_dir.DOC.COM.:
fns
fns_user
fns_host
fns_host_alto
fns_host_mladd
fns_host_elvira
fns_user_jjones
fns_user_jsmith
fns_user_aw
```

niscat コマンドを使用して、fns_hosts テーブルの内容を表示します。

```
# niscat fns_host.ctx_dir
altair *BINARY* *BINARY*
cygnus *BINARY* *BINARY*
centauri *BINARY* *BINARY*
```

アクセス制御をチェックする

niscat コマンドを –o オプションつきで使用して、コンテキストのアクセス制御を確認します。特定のバインディングのアクセス制御を確認するには、親コンテキストのバインドテーブルにあるバインドエントリの名前 (つまり、fnlookup –v および fnlist –v の出力の内部名フィールドに表示される名前) を使用します。

```
# niscat -o fns_host.ctx_dir
Object Name      : fns_host
Owner            : alto.doc.com.
Group             : admin.doc.com.
Domain            : ctx_dir.doc.com.
Access Rights    : r-c-rmcdrmcdr-c-
Time to Live     : 53:0:56
Object Type      : TABLE
Table Type       : H
Number of Columns : 3
Character Separator
Search Path      :
```

(続く)

```
Columns         :
[0] Name    : atomicname
 Attributes   : (検索可能、テキストデータ、大文字・小文字の区別なし)
 Access Rights  : r-c-rmcdrmcdr-c-
[1] Name    : reference
 Attributes   : (2 進データ)
 Access Rights  : r-c-rmcdrmcdr-c-
[2] Name    : flags
 Attributes   : (2 進データ)
 Access Rights  : r-c-rmcdrmcdr-c-
```

```
# niscat -o "[atomicname=altair],fns_host.ctx_dir"
Object Name : fns_host
Owner    : altair.doc.com.
Group    : admin.doc.com.
Domain   : ctx_dir.doc.com.
Access Rights : r-c-rmcdrmcdr-c-
Time to Live : 12:0:0
Object Type : ENTRY
Entry data of type H
[1] - [5 bytes] 'alto'
[2] - [104 bytes] '0x00 ...'
[3] - [1 bytes] 0x01
```

niscat コマンドに関する追加情報については、298ページの「niscat コマンド」を参照してください。

特定のコンテキストのアクセス権または所有権を変更するには、次のコマンドを使用します：

- nischown
- nischmod
- nischgrp

操作が影響するオブジェクトに応じて、バインドエントリまたはバインドテーブルのどちらかを引数として与えます。

FNS と NIS の詳細情報

ここでは、NIS と FNS の関係についての特定情報を説明します。

NIS と FNS のマップと Makefile

FNS は、NIS マスターおよびスレーブのサーバー上の /var/yp/*domainname* ディレクトリに格納されている 6 つのマップを使用します。

- fns_host.ctx

  ホスト属性とサブコンテキストのデータを格納する。これが最初に作成されると、hosts.byname マップから情報が得られる

- fns_host.ctx

  ユーザー属性とサブコンテキストのデータを格納する。これが最初に作成されると、passwd.byname マップから情報が得られる

- fns_org.ctx

  エンタープライズ属性とサブコンテキストのデータを格納する

- fns_host.attr

  属性による検索のためのホスト属性を格納する

- fns_user.attr

  属性による検索のためのユーザー属性を格納する

- fns_org.attr

  属性による検索のためのエンタープライズ属性を格納する

ホスト、ユーザー、およびエンタープライズのサービスとファイルのコンテキスト情報は、それぞれ fns_host.ctx、fns_user.ctx、および fns_org.ctx のマップに格納されます。プリンタのコンテキスト情報は、他のサービスのコンテキスト情報と同じマップに格納されます。しかし、古い printers.conf.byname マップはここでもサポートされます。

サイトは、エンタープライズのサブコンテキストであり、サイトのコンテキスト情報は fns_org.ctx マップに格納されます。

注 - これらの FNS マップは、直接編集しないでください。fncreate、fndestroy、fnbind、fnunbind、fnrename、fnattr、fnlookup、および fnlist などの適切な FNS コマンドを実行して、これらのマップで変更または作業を行います。これらのコマンドは、必ず NIS マスターサーバーで実行します。スレーブサーバーまたはクライアントマシンでこれらを実行できません。

FNS マップファイルは、/var/yp/*domainname* ディレクトリにあります。/var/yp にある NIS Makefile は変更され、/etc/fn/*domainname* にある FNS Makefile が認識されます。

大型 FNS コンテキスト

NIS では、NIS マップに含まれるエントリ数に 64K という制限があります。サービスおよびプリンタのコンテキストだけが各オブジェクト (ホストまたはユーザー) に作成された場合、ユーザーまたはホストの数が 7K を超えると、エントリ数がその制限に達します。通常行われるように、ホストまたはユーザーに追加のコンテキストが作成された場合、ずっと少ないホストまたはユーザーでエントリ数が 64,000 の上限に達します。

FNS は、古いマップが最大サイズに達したら新規マップを自動的に作成して、この問題を解決します。各新規マップは、マップ名に数字の接尾辞を追加して識別されます。たとえば、2 つめの fns_user.ctx マップが作成されると、そのマップには名前 fns_user_0.ctx という名前が与えられます。3 つめのマップが必要になると、そのマップには名前 fns_user_1.ctx という名前が与えられます。追加のマップが作成されるにつれて、数字は毎回増加していきます。

プリンタの下位互換

Solaris リリース 2.5 では、FNS は、printers.conf.byname という名前のマップを使用して、組織コンテキストで NIS のプリンタのネーミングをサポートします。現在の Solaris では、組織コンテキストのプリンタサポートは、fns_org.ctx マップに保持されています。つまり、ここでの fncreate_printer コマンドでは fns_org.ctx マップが変更され、printers.conf.byname マップは変更されません。

NIS から NIS+ への変更

fncopy コマンドは、エンタープライズレベルのネームサービスを NIS から NIS+ に変更するときに、FNS 関連の側面を処理します。このコマンドは、NIS ベースの FNS コンテキストを NIS+ ベースのコンテキストにコピーして変換します。

コマンド構文は以下のとおりです。

```
fncopy [-i oldsvc – o newsvc] [–f  filename] oldctx newctx
```

表 23–3 fncopyコマンドオプション

オプション	説明
–i *oldsvc*	ソースのネームサービス。nis または files だけが指定される
–o *newsvc*	目標のネームサービス。nisplus または nis だけが指定される
–f *filename*	コピーされる FNS コンテキストを表示するファイル名
oldctx	コピーされる古い FNS コンテキスト
newctx	目標の新規 FNS コンテキスト

たとえば、ファイル /etc/sales_users に表示されるコンテキストを、NIS ベースのネームサービスの doc.com ドメインから NIS+ ネームサービスの sales.doc.com ドメインにコピーするには、次のように入力します。

```
fncopy -i nis -o nisplus -f /etc/sales_users org/sales.doc.com/user
```

FNS とファイルベースのネーミングの詳細情報

この節では、ファイルベースのネーミングと FNS の関係についての特定の情報を説明します。

FNS ファイル

FNS では、各マシンの /var/fn ディレクトリに格納された新規ファイルが使用されます。通常 /var/fn ディレクトリは各マシンに格納されていますが、NFS 経由で中央の /var/fn ディレクトリをマウントしたり、エクスポートしたりできます。

新規 FNS ファイルを以下に示します。

- fns_host.ctx

ホスト属性とサブコンテキストのデータを格納する。これが最初に作成されると、/etc/hosts ファイルから情報が得られる

- fns_user.ctx

  ユーザー属性とサブコンテキストのデータを格納する。これが最初に作成されると、/etc/passwd ファイルから情報が得られる

- fns_org.ctx

  エンタープライズ属性とサブコンテキストのデータを格納する

- fns_host.attr

  属性による検索のためのホスト属性を格納する

- fns_user.attr

  属性による検索のためのユーザー属性を格納する

- fns_org.attr

  属性による検索のためのエンタープライズ属性を格納する

- ユーザーのサブコンテキストと属性の情報は、各ユーザーが所有する別個の /var/fn ファイルに格納される。これにより、ユーザーは FNS コマンドを使用して、自分のデータを変更できる。これらのユーザー固有のファイルは、fns_user_*username*.ctx とネーミングされる。ここでの *username* は、個々のユーザーのログイン ID になる

ホスト、ユーザー、およびエンタープライズのサービスとファイルのコンテキスト情報は、それぞれ fns_host.ctx、fns_user.ctx、および fns_org.ctx のファイルに格納されます。プリンタのコンテキスト情報は、他のサービスのコンテキスト情報と同じファイルに格納されます。

サイトは、エンタープライズのサブコンテキストであり、サイトのコンテキスト情報は fns_org.ctx ファイルに格納されます。

注 - これらの FNS マップは、直接編集しないでください。fncreate 、fndestroy、fnbind、fnunbind、fnrename、fnattr、fnlookup、および fnlist などの適切な FNS コマンドを実行して、これらのファイルで変更または作業を行います。スーパーユーザーとしてこれらのコマンドを実行すると、ホスト、サイト、および組織単位などの、コマンドが適用されるコンテキストが影響されます。ユーザーとしてこれらのコマンドを実行すると、自分のユーザーのサブコンテキストだけが影響されます。

ファイルベースのネーミングから NIS または NIS+ への変更

fncopy コマンドは、エンタープライズレベルのネームサービスをファイルから NIS または NIS+ に変更するときに、FNS 関連の側面を処理します。このコマンドは、ファイルベースの FNS コンテキストを NIS または NIS+ ベースのコンテキストにコピーして変換します。

コマンド構文は以下の通りです。

```
fncopy [-i oldsvc -o newsvc] [-f filename] oldctx newctx
```

たとえば、ファイル /etc/host_list に表示される内容を NIS+ ネームサービスの doc.com ドメインにコピーするには、次のように入力します。

```
fncopy -i files -o nisplus -f /etc/host_list //doc.com/host
```

プリンタの下位互換

Solaris リリース 2.5 では、FNS は、printers.conf.byname という名前のファイルを使用して、組織コンテキストでファイルへのプリンタのネーミングをサポートします。現在の Solaris では、組織コンテキストのプリンタサポートは、fns_org.ctx マップに保持されています。つまり、ここでの fncreate_printer コマンドでは fns_org.ctx マップが変更され、printers.conf.byname マップは変更されません。

第 **24** 章

エンタープライズレベルのコンテキスト

この章では、すでに存在しているエンタープライズレベルのコンテキストを、個別に作成、管理する方法について説明します。

- 545ページの「コンテキストにバインドされた名前を変更する」
- 545ページの「コンテキストを削除する」

FNS コンテキストを個別に作成する

FNS コンテキストは、fncreate コマンドを使用して作成します。この節では、『*Solaris* ネーミングの設定と構成』に説明されているような組織全体の FNS コンテキストではなく、個別に FNS コンテキストを作成する方法について説明します。fncreate コマンドは、指定されたタイプのコンテキストを作成し、そのコンテキストを任意の複合名とバインドします。また、コンテキストのサブコンテキストを作成することもできます。

fncreate コマンドの構文は次のとおりです。

```
fncreate -t context_type [-f input_file] [-o] [-r reference_type] [-s] [-v] [-D] composite_name
```

表 24–1 fncreate コマンドオプション

オプション	説明
–t *context*	作成するコンテキストのタイプを指定する。*context* には、org、hostname、username、host、user、service、site、nsid、generic、fs のうちいずれかを使用する
–f	*input_file* に指定された個々のホストまたはユーザーのコンテキストを作成する。このオプションは、–t username オプションまたは –t hostname オプションと併用される場合にだけ有効で、対応する NIS+ passwd テーブル (または hosts テーブル) 中のユーザー (またはホスト) のサブセットのコンテキストを個々に作成する場合に使用できる
–o	指定されたコンテキストだけを作成する。–o オプションを指定しないと、サブコンテキストは FNS ポリシーに基づいて作成される
–r	作成される汎用コンテキストの *reference_type* を指定する。これは –t generic オプションと併用される場合にだけ有効
–s	すでに使用されている複合名で、新しいコンテキストを作成する。このオプションを使用しなければ、すでにバインドされている名前で新しいコンテキストを作成できない

表 24–1　fncreate コマンドオプション　続く

オプション	説明
–D	コンテキストが作成されるたびに、コンテキストに関連付けられた NIS+ オブジェクトの情報を表示する。このオプションはデバッグに役立つ
–v	コンテキスト作成時に、作成に関する情報を表示する

注 - 組織コンテキスト作成時に –o オプションを指定した場合、host、user、service のコンテキストは作成されてもサブコンテキストは生成されません。

名前空間識別子にバインドされるコンテキストを作成する場合、下線のない名前 (user など) はコンテキストの作成に使用され、下線の付いた名前 (_user など) は新しく作成されたコンテキストのリファレンスにバインドされます。この点は、コマンド行で指定された名前が下線のついてものであるか否かに関わらず常に同じです。

たとえば、以下のコマンドでは、org/sales/user のコンテキストが作成され、org/sales/_user が org/sales/user のコンテキストのリファレンスにバインドされます。

```
fncreate -t username org/sales/_user
```

組織コンテキストを作成する

組織コンテキストを作成するには、コンテキストタイプに org を指定します。複合名には、使用しているネームサービスによって以下のいずれかにする必要があります。

- 「NIS+」の場合

  すでに存在している NIS+ ドメイン (またはサブドメイン) の名前。NIS+ ドメインとは、org_dir サブディレクトリを含む NIS+ ディレクトリオブジェクトのことです。ドメインの org_dir サブディレクトリには、そのドメインの生成された host テーブルと passwd テーブルが必要です。

- 「NIS」の場合

  NIS ドメインの名前。host マップと passwd マップが必要です。

- /etc ファイルの場合

  /etc ファイルを使用するのは、org// 組織コンテキストだけです。

NIS+ の場合の組織コンテキストの例

NIS+ のルートドメインが doc.com で、sales.doc.com というサブドメインがあるとします。sales というサブドメインに対応する sales という組織コンテキストを作成するには、以下のコマンドを入力します。

```
fncreate -t org org/sales/
```

新しいコンテキストの作成時には、ドメインである sales.doc.com の下に ctx_dir というディレクトリが作成されます。すでに ctx_dir が存在する場合は作成されません。

上記の例では –t オプションだけを使用し、–o オプションは使用しません。これによって複合名 org/sales/ の組織コンテキストが作成されると同時に、サブコンテキストとして hostname、username、service が作成されます。また host コンテキストと user コンテキスト、ホストとユーザーの service サブコンテキストも作成されます。実際には以下のコマンドが実行されます。

```
fncreate -t hostname org/sales/host/
fncreate -t username org/sales/user/
fncreate -t service org/sales/service/
```

代わりに fncreate -o -t org を使用すると、org コンテキストだけが作成されます。hostname、username、service のコンテキストは作成されますが、host コンテキストと user コンテキストの生成はされません。

org コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。この点は、hostname、username、service といったサブコンテキストについても同様です。ただし、host コンテキスト、user コンテキストとそのサブコンテキストの所有者となるのは、コンテキストの作成対象となったホストおよびユーザーです。管理者が引き続き host コンテキストや user コンテキストを処理するには、fncreate を実行する時に NIS_GROUP 環境変数を適宜設定する必要があります。たとえば、C シェルの場合、NIS_GROUP を以下のように fns_admins.doc.com に設定します。

```
rootmaster# setenv NIS_GROUP fns_admins.doc.com
```

すべてのホストのコンテキスト

fncreate で hostname をコンテキストのタイプとして指定すると hostname コンテキストが作成されます。hostname コンテキストでは、host コンテキストの作成、バインドができます。この場合、–o オプションを指定しなければ、host コン

テキスト (およびそのサブコンテキスト) も NIS+ の host.org_dir テーブル中のホストごとに作成されます。–o オプションを指定した場合は、コンテキストだけが作成されます。

たとえば、以下のコマンドを実行すると、

```
fncreate -t hostname org/sales/host/
```

hostname コンテキストが作成されます。

```
fncreate -t host org/sales/host/hname
```

hname とは、各マシンの hosts.org_dir テーブル中にある名前です。また org/sales/_host/ の、org/sales/host/ のリファレンスへのバインドも行われます。

hostname コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。また host コンテキスト (およびそのサブコンテキスト) の所有者となるのは、コンテキストの作成対象となったホストです。つまり、個々のホストがそれぞれ自分のホストのコンテキスト (およびそのサブコンテキスト) を持つことになります。

–f オプションを使用すると、NIS+ の hosts.org_dir テーブル中のホストの、サブセットのコンテキストを作成できます。このオプションでは、input_file に指定されたホストのコンテキストが作成されます。

1 台のホストのコンテキスト

fncreate で host をコンテキストのタイプとして指定すると、ホストのコンテキストとサブコンテキストが作成されます。–o オプションを指定しなければ、ホストの service コンテキストと fs のバインディングが作成されます。–o オプションを指定した場合は、host コンテキストだけが作成されます。

たとえば、以下のコマンドでは、

```
# fncreate -t host org/sales/host/antares/
```

antares というホストのコンテキストが作成されます。実際には以下のコマンドが実行されます。

```
fncreate -t service org/sales/host/antares/service/
fncreate -t fs org/sales/host/antares/fs/
```

host コンテキスト (およびそのサブコンテキスト) の所有者となるのはホストです。上記の例では antares というホスト (NIS+ 主体名 antares.sales.doc.com) が以下のコンテキストの所有者となります。

- org/sales/host/antares/
- org/sales/host/capsule/service/
- org/sales/host/capsule/fs

ただしこの場合、ホストの属する hostname コンテキスト (上記の例では、org/sales/host/) が必要です。また NIS+ テーブル hosts.org_dir に、ホスト名を指定する必要があります。

ホストの別名

NIS+ テーブル hosts.org_dir には、別名を持つホストが存在することもあります。この種のホストは、テーブル中では「1 つの正式名にいくつかの別名が対応づけられている」という形で表されます。

FNSにおいては、たとえ複数の別名を持っていても、1 つのホストが持つ host コンテキストは 1 つだけです。hostname コンテキスト中のホストの別名がバインドされるのは、正式名と同じコンテキストのリファレンスです。

すべてのユーザーのコンテキスト

fncreate で username をコンテキストのタイプとして指定すると、username コンテキストが作成されます。username コンテキストでは、個々の user コンテキストの作成とバインドが行われます。–o オプションを指定しないと、NIS+ テーブル passwd.org_dir に存在するユーザー名ごとに、user コンテキスト (およびそのサブコンテキスト) が作成されます。–o オプションを指定した場合は、username コンテキストだけが作成されます。

たとえば、以下のコマンドを実行すると、

```
# fncreate -t username org/sales/user/
```

username コンテキストが作成されます。

```
fncreate -t user org/sales/user/uname
```

実際には、passwd.org_dir テーブルに存在するユーザー *uname* ごとに、以下のコマンドが実行されます。また、org/sales/_user/ の org/sales/user/ へのバインドも行われます。

username コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。個々の user コンテキスト (およびそのサブコンテキスト) の所有者となるのは、コンテキストの作成対象となったユーザーです。ユーザーがそれぞれ独自の user コンテキスト (およびそのサブコンテキスト) を所有することになります。

–f オプションを使用すると、NIS+ テーブル passwd.org_dir に存在するユーザーのサブセットのコンテキストを作成できます。このオプションでは、input_file に指定されたホストのコンテキストが作成されます。

1 人のユーザーのコンテキスト

fncreate で user をコンテキストタイプとして指定すると、ユーザーの user コンテキスト (およびそのサブコンテキスト) が作成されます。–o オプションを指定しなければ、user コンテキストの下に service サブコンテキストと fs のバインディングが作成されます。–o オプションを指定した場合は、user コンテキストだけが作成されます。

たとえば、以下のコマンドでは、

```
# fncreate -t user org/sales/user/jjones/
```

jjones という名前のユーザーの user コンテキストが作成されます。実際には、以下のコマンドが実行されます。

```
fncreate -t service org/sales/user/jjones/service/
fncreate -t fs org/sales/user/jjones/fs/
```

user コンテキスト (およびそのサブコンテキスト) の所有者となるのは、コンテキストの作成対象となったユーザーです。上記の場合、作成されたコンテキストの所有者となるのは、jjones というユーザー (NIS+ 主体名 jjones.sales.doc.com) です。

ただしこの場合、ユーザーの属する username コンテキスト (上記の org/sales/user) が必要です。また、指定されたユーザー名が NIS+ テーブル passwd.org_dir 中に存在している必要があります。

サービスのコンテキスト

fncreate でコンテキストタイプとして service を指定すると、service コンテキストが作成されます。service コンテキストでは、サービス名のバインドが行えます。service コンテキスト中でバインドできるリファレンスのタイプに制約はありません。ただし、service コンテキストを使用するアプリケーションによっては制約が設けられます。たとえば、デスクトップアプリケーションのグループなら、service コンテキスト中でバインドされるリファレンスのタイプは、カレンダ、カードファイル、ファックスサービス、プリンタなどになるでしょう。

たとえば、以下のコマンドでは、

```
# fncreate -t service org/sales/service/
```

sales という組織の service コンテキストが作成されます。末尾の名前 (service) は名前空間識別子なので、fncreate では org/sales/_service/ の org/sales/service/ のリファレンスへのバインドも行われます。このコマンドの実行後は、org/sales/service/calendar、org/sales/service/fax といった名前をこのサービスコンテキストでバインドできます。

service コンテキストでは、階層型の名前空間 (左から順に名前が並べられ、名前と名前の間はスラッシュで区切られる) がサポートされています。これによってアプリケーションは、名前空間をサービスごとに区切ることができます。デスクトップアプリケーションの場合、plotter コンテキストを作成した後に以下のコマンドを使用すれば、service コンテキストの下に plotter のグループを作成することができます。

```
# fncreate -t service org/sales/service/plotter
```

さらにこの下には、org/sales/service/plotter/speedy、org/sales/service/plotter/color といった名前をバインドできます。

注 - ただし、末尾の名前が名前空間識別子ではないので、service と _service のようなバインドが行われることはありません。

作成される service コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です 。

プリンタコンテキスト

プリンタコンテキストは、複合名の service コンテキストの下に作成されます。

汎用コンテキスト

fncreate でコンテキストタイプに generic を使用すると、バインディング名のコンテキスト (アプリケーションによって使用される) が作成されます。

generic コンテキストは、リファレンスのタイプを変更できるという点以外は service コンテキストと同じです。generic コンテキストのリファレンスのタイプを指定するには、–r オプションを使用します。このオプションが省略された場合は、親コンテキストが generic タイプであればそれがそのまま使用され、別のタイプであればデフォルトとして指定されたものが使用されます。

service コンテキストと同様、generic コンテキストにバインドされるリファレンスのタイプには制約がありません。ただし generic コンテキストを使用するアプリケーションによって制約が設けられる場合があります。

たとえば、以下のコマンドでは、

```
# fncreate -t generic -r WIDC_comm org/sales/service/extcomm
```

sales という組織の service コンテキストの下に、リファレンスタイプ WIDC_comm の generic コンテキストが作成されます。この generic コンテキストでは、org/sales/service/extcomm/modem などの名前がバインドできます。

generic コンテキストでは、階層型の名前空間 (左から順に名前が並べられ、名前と名前の間はスラッシュで区切られる) がサポートされています。これによってアプリケーションは、名前空間をサービスごとに区切ることができます。上の例の場合、以下のコマンドを使用すれば、modem 用の generic サブコンテキストが作成できます。

```
# fncreate -t generic org/sales/service/extcomm/modem
```

さらにこの下には、
org/sales/service/extcomm/modem/secure、

org/sales/service/extcomm/modem/public といった名前をバインドできます。

作成された generic コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。

Site コンテキスト

site コンテキストでは、サイト名のバインドが行えます。

site コンテキストを作成するには、以下のようなコマンドを使用します。

```
# fncreate -t site org/sales/site/
```

ここでは末尾の名前 (site) が名前空間識別子なので、org/sales/site/ のリファレンスに org/sales/_site/ がバインドされます。

site コンテキストでは、階層型の名前空間 (左から順に名前が並べられ、名前と名前の間はドットで区切られる) がサポートされています。これによって、地理的な位置関係にもとづいてサイトを区分できます。

たとえば、site コンテキスト alameda と、site サブコンテキスト alameda.bldg-5 を作成するには、以下のコマンドを使用します。

```
# fncreate -t site org/sales/alameda
# fncreate -t site org/sales/site/alameda.bldg-5
```

注 - ただし末尾の名前が名前空間識別子ではないので、site と _site の場合のようなバインドが行われることはありません。

作成された site コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。

ファイルのコンテキスト

fncreate でコンテキストタイプに fs を指定すると、ユーザーまたはホストのファイルシステムコンテキスト (ファイルコンテキストとも呼ばれる) が作成されま

す。たとえば以下のコマンドでは、ユーザー petrova のファイルコンテキストが作成されます。

```
# fncreate -t fs org/sales/user/petrova/fs/
```

ここでは末尾の名前 (fs) が名前空間識別子なので、org/sales/user/petrova/fs/ のリファレンスに org/sales/user/petrova/_fs/ がバインドされます。

ユーザーの fs コンテキストは、NIS+ テーブル passwd.org_dir に保存されているユーザーのホームディレクトリと同じものです。一方ホストの fs コンテキストは、ホストによってエクスポートされる NFS ファイルシステムをいくつか集めたものです。

組織 およびサイトの file コンテキストを作成する場合、あるいはユーザーおよびホストのデフォルトでない file コンテキストを作成する場合は、fncreate_fs コマンドを使用します。詳細については、547ページの「ファイルコンテキストの管理」を参照してください。

作成された fs コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。

名前空間識別子のコンテキスト

nsid (namspace identifier = 名前空間識別子) タイプのコンテキストでは、名前空間識別子がバインドできます。

たとえば、以下のコマンドでは、サイト alameda.bldg-5 コンテキストが作成され、service/ などサブコンテキストが作成できるようになります。

```
# fncreate -t nsid org/sales/site/alameda.bldg-5/
```

この場合以下のコマンドを実行すると、alameda.bldg-5 の service コンテキストが作成できます。

```
# fncreate -t service org/sales/site/alameda.bldg-5/service/
```

作成された nsid コンテキストの所有者となるのは、fncreate コマンドを実行した管理者です。

エンタープライズレベルのコンテキストを管理する

FNS コンテキストを管理する (コンテキストの内容を確認する) ためのツールには様々な種類があります。以下の表は、そのツール (コマンド) の構文を示したものです。

バインドに関する情報を表示する

fnlookup は、複合名のバインドに関する情報を表示するのに使用します。

```
fnlookup [-v][-L] composite_name
```

表 24–2 fnlookup コマンドのオプション

オプション	説明
–v	詳細なバインド関連情報を表示する
–L	「XFN リンクのバインド先となるのはどのリファレンスか」を表示する

たとえば、以下の例では、ユーザー darwin のバインドに関する情報が表示されます。

```
# fnlookup -v user/darwin/
Reference type: onc_fn_user
Address type: onc_fn_nisplus
length: 52
context type: user
representation: normal
version: 0
internal name: fns_user_darwin.ctx_dir.sales.doc.com.
```

ここで user/Charles.Darwin が user/darwin にリンクされているとすると、下の例の最初のコマンドでは、user/Charles.Darwin のバインド先 (XFNリンク)

が表示されます。2 番目のコマンドでは、XFNリンク user/jjones が調べられ、user/darwin のバインド先 (user コンテキスト) が表示されます。

```
# fnlookup user/Charles.Darwin
Reference type: fn_link_ref
Address type: fn_link_addr
Link name: user/darwin
# fnlookup -L user/Charles.Darwin
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
```

コンテキストの内容を表示する

fnlist は、コンテキストの内容を表示するのに使用されます。

```
fnlist [-lv] [name]
```

表 24–3 fnlist コマンドのオプション

オプション	説明
–v	バインドに関する詳細な情報を表示する
–l	コンテキスト中でバインドされている名前に関する情報を表示する

以下の例では、user コンテキストにバインドされている名前が表示されます。

```
# fnlist user/
Listing 'user/':
jjones
julio
chaim
James.Jones
```

名前を指定しないと、初期コンテキストの内容が表示されます。

```
# fnlist
Listing '':
_myorgunit
...
_myself
thishost
myself
_orgunit
_x500
_host
 _thisens
 myens
 thisens
 org
 orgunit
 _dns
 thisuser
 _thishost
 myorgunit
 _user
 thisorgunit
 host
 _thisorgunit
 _myens
 user
```

-l オプションを指定すると、指定されたコンテキスト中でバインドされている名前に関する情報が表示されます。

```
# fnlist -l user/
Listing bindings 'user/':
name: julio
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
name: chaim
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
name: James.Jones
Reference type: fn_link_ref
Address type: fn_link_addr
Link name: user/jjones
name: jjones
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
```

-l、-v オプションを同時に指定すると、バインドされた名前に関するさらに詳細な情報が表示されます。

```
# fnlist -lv user/
Listing bindings 'user/':
name: julio
Reference type: onc_fn_user
Address type: onc_fn_nisplus
length: 52
context type: user
representation: normal
version: 0
internal name: fns_user_julio.ctx_dir.sales.doc.com.
name: chaim
Reference type: onc_fn_user
Address type: onc_fn_nisplus
length: 52
context type: user
representation: normal
version: 0
internal name: fns_user_chaim.ctx_dir.sales.doc.com.
name: James.Jones
Reference type: fn_link_ref
Address type: fn_link_addr
length: 11
data: 0x75 0x73 0x65 0x72 0x2f 0x6a 0x6a 0x6f 0x6e 0x65
user/jjones
name: jjones
Reference type: onc_fn_user
Address type: onc_fn_nisplus
length: 52
context type: user
representation: normal
version: 0
internal name: fns_user_jjones.ctx_dir.sales.doc.com.
```

複合名をリファレンスにバインドする

fnbind は、複合名をリファレンスにバインドするためのコマンドです。

このコマンドには、2 種類の構文があります。

- すでに存在する名前のリファレンスを新しい名前にバインドするもの (以下参照)
- コマンド行の引数にもとづいて作成されたリファレンスを、別の名前にバインドするもの (543ページの「コマンド行にリファレンスを作成する」を参照)

既存の名前を新しい名前にバインドする

既存の名前を新しい名前にバインドするための fnbind の構文は以下のようになります。

```
fnbind [-s][-v][-L] oldname  newname
```

表 24–4　fnbind コマンドのオプション (バインド名)

オプション	説明
oldname	すでに存在している複合名
newname	すでに存在している複合名にバインドする新しい名前
–s	複合名がすでにバインドされている場合、バインドを上書きする
–v	バインドに使用されたリファレンスに関する情報を表示する
–L	*name* を使用して XFN リンクを作成し、それを *new_name* にバインドする

たとえば、以下のコマンドでは、user/julio/service/printer という名前が myorgunit/service/printer のリファレンスにバインドされます。

```
# fnbind myorgunit/service/printer user/julio/service/printer
```

newname がすでにバインドされている場合は、fnbind –s を使用しないと処理が正しく行われません。つまり上記の例で user/julio/service/printer がすでにバインドされていれば、以下のように –s オプションを使用して myorgunit/service/printer とのバインドで上書きする必要があるということになります。

```
# fnbind -s myorgunit/service/printer user/julio/service/printer
```

–v オプションは、バインドに使用されたリファレンスに関する情報を表示するのに使用されます。

```
# fnbind -v myorgunit/service/printer user/julio/service/printer
Reference type: onc_printers
Address type: onc_fn_printer_nisplus
```

以下の例では、user/jjones によって XFN リンクが作成され、user/James.Jones という名前にバインドされます。

```
# fnbind -L user/jjones user/James.Jones
```

同様に以下の例では、user/julio/service/printer から myorgunit/service/printer へのリンクが作成されます。

```
# fnbind -sL myorgunit/service/printer user/julio/service/printer
```

コマンド行にリファレンスを作成する

コマンド行にリファレンスを作成するための fnbind の構文は次のようになります。

```
fnbind -r [-s] [-v] newname [-O | -U] reftype {[-O | -U] | addresstype [-c|-x] addresscontents}+
```

表 24–5 fnbind コマンドオプション (リファレンス作成)

オプション	説明
newname	リファレンスを作成する新しい名前
reftype	作成するリファレンスのタイプ。–O、–U オプションを使用しない場合は、*reftype* の識別子として FN_ID_STRING を使用する
addresstype	作成するアドレスのタイプ。–O、–U オプションを使用しない場合は、*addresstype* の識別子として FN_ID_STRING を使用する
addresscontents	作成するリファレンスのアドレス。–c あるいは –x オプションを使用しない場合、アドレスは XDR によって符号化された文字列として保存される
–s	複合名がすでにバインドされている場合、バインドを上書きする
–v	バインドに使用されたリファレンスに関する情報を表示する
–c	アドレス内容を、XDR による符号化を行わずに保存する
–x	アドレス内容を 16 進数で入力された文字列であると解釈し、そのまま保存する
–r	指定タイプのリファレンスを作成し、コマンド行で指定された名前にバインドする

表 24–5　fnbind コマンドオプション (リファレンス作成)　続く

オプション	説明
–O	タイプを指定する文字列を、ASN.1 の形式 (整数をいくつか並べてドットで区切ったもの) として解釈して保存する
–U	タイプを指定する文字列を、DCE UUID として解釈して保存する

たとえば、以下の例では、thisorgunit/service/calendar という名前が、「タイプ onc_calendar、アドレスタイプ onc_cal_str、アドレス staff@cygnus」というリファレンスにバインドされます。

```
# fnbind -r thisorgunit/service/calendar onc_calendar onc_cal_str staff@cygnus
```

コマンド行で指定されたアドレスは、デフォルトでは XDR で符号化された後、リファレンスに保存されます。–c オプションが指定された場合は、XDR による符号化は行われず、そのままの形で保存されます。–x オプションが指定された場合は、16 進文字列として解釈されて保存されます (XDR による符号化は行われない)。

リファレンス (およびそのアドレス) のタイプの指定には、デフォルトでは FN_ID_STRING 識別子の形式が使用されます。–O オプションでは FN_ID_ISO_OID_STRING (ASN.1、10進数を並べてドットで区切る) の形式が、–U オプションでは FN_ID_DCE_UUID (DCE UUID、文字列を使用する) の形式が使用されます。

注 - ASN.1 の詳細は、『*ISO 8824: 1990, Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*』を参照してください。DCE UUID の詳細は、『*X/Open Preliminary Specification, October 1993, X/Open DCE: Remote Procedure Call (ISBN: 1-872630-95-2)*』を参照してください。

以下の例では、thisorgunit/service/nx という名前にバインドされるリファレンス (およびそのアドレス) の、タイプが OIDs の形式で、アドレスが 16 進の文字列で指定されています。

```
# fnbind -r thisorgunit/service/nx -O 1.2.99.6.2.1 -O 1.2.99.6.2.3 -x ef12eab67290
```

複合名を削除する

fnunbind は、複合名を名前空間から削除するのに使用されます。ただし、名前に対応するオブジェクトは削除されず、オブジェクトと名前のバインドを解除するだけであるという点に注意してください。

以下の例では、user/jjones/service/printer/color という複合名が削除されます。

```
# fnunbind user/jjones/service/printer/color
```

コンテキストにバインドされた名前を変更する

fnrename は、コンテキストにバインドされている名前を変更するのに使用されます。

以下の例では user/jjones/service/ というコンテキストにバインドされた、clndr という名前が calendar に変更されます。

```
# fnunbind user/jjones/service/printer/color
```

コンテキストを削除する

fndestroy は、指定された複合名を名前空間から削除し、その名前を持つコンテキストを削除するのに使用されます。

たとえば、user/jones/ という名前を名前空間とのバインドから解除し、user/jjones/ という名前のコンテキストを削除するには、以下のように入力します。

```
# fndestroy user/jjones/
```

削除の対象となるコンテキスト (指定された複合名を持つもの) にサブコンテキストが存在する場合、コマンドは正常に動作しません。

第 25 章

ファイルコンテキストの管理

この章では、アプリケーション固有のコンテキストをどのように管理するかについて説明します。

ファイルコンテキストの管理

ファイルコンテキストには、以下のようなものがあります。

- `fncreate_fs` コマンドを使用して作成したもの (548ページの「fncreate_fs を使ってファイルコンテキストを作成する」を参照)
- `fnlist` コマンド (539ページの「コンテキストの内容を表示する」を参照)、あるいは `fnlookup` コマンド (538ページの「バインドに関する情報を表示する」を参照) を使用して、検索されたもの
- `fnunbind` コマンド (545ページの「複合名を削除する」を参照)、あるいは `fndestroy` コマンド (545ページの「コンテキストを削除する」を参照) を使用して、切り取られたものあるいは削除されたもの

fncreate_fs を使ってファイルコンテキストを作成する

fncreate_fs コマンドにより、組織やサイト用にファイルコンテキストが作成できます。fncreate コマンドによって作成されたホストやユーザーのためのデフォルトのファイルコンテキストを無効にするために使用されることもあります。

fncreate_fs コマンドの使用には、以下のような 2 つの方法があります。

- 「入力ファイル」

  入力ファイルによって、ファイルコンテキストのバインディングが提供されます(549ページの「入力ファイルを使って、ファイルコンテキストを作成する」を参照)。

- 「コマンド行」

  コマンド行によって、ファイルコンテキストのバインディングが作成されます(551ページの「コマンド行の入力によりファイルコンテキストを作成する」を参照)。

fncreate_fs の 2 つの方法には、以下のような構文があります。

```
fncreate_fs [-v] [-r] -f file   composite_name
fncreate_fs [-v] [-r]  composite_name [options] [ location...]
```

表 25–1 fncreate_fs コマンドのオプション

オプション	説明
composite_name	ファイルコンテキストの複合名
–f *file*	*file* という名前の入力ファイルを使用する
options	マウントのオプション
location	マウントする位置

表 25–1 fncreate_fs コマンドのオプション 続く

オプション	説明
-v	詳細出力に設定。作成および修正されるコンテキストに関する情報を表示
-r	*composite_name* で名前付きコンテキスト、およびそのサブコンテキストのすべてのバインディングを、入力で指定したものだけで置き換える。これは、コンテキスト (およびそのサブコンテキスト) を削除し、このオプションなしで、fncreate_fs を実行するのと同じ。-r オプションは、注意して使用する必要がある

fncreate_fs コマンドは、FNS コンテキスト、および onc_fn_fs リファレンスタイプのバインディングを操作します。これは、onc_fn_fs_mount タイプのアドレスを使って、それぞれのリモートのマウント先を表わします。このタイプのアドレスに関連するデータは、XDR で符号化された 1 つの文字列で示した、対応するマウントのオプションやマウント先になっています。

入力ファイルを使って、ファイルコンテキストを作成する

入力ファイルには、composite_name のコンテキストでバインドされる名前や値が入っています。この形式は、間接自動マウントマップの形式に基づいてはいますが、全く同じではありません。入力ファイルには、以下のような形式のエントリが 1 つあるいは複数含まれています。

```
name [options] [location...]
```

- *name* には、リファレンス名が入ります。*name* フィールドは、単純な原子名のこともあれば、あるいはスラッシュで区切った階層構造を示す名前の場合もあります。さらに (.) といったようにドットを使って示すこともあり、この場合リファレンスは、*composite_name* に直接バインディングされます。
- *options* には、マウントのオプションがあれば、それが入ります。*options* フィールドは、ハイフン (-) で始まります。この後に、マウントオプションのコンマで区切ったリスト (スペースなし) が続き、これはディレクトリをマウントするのに使用されます。またこれらのオプションは、*composite_name/name* サブコンテキスト

にも適用されます。なお、*composite_name/name* は、それ自身のマウントオプションは指定しません。

- *location* には、マウントの位置が入ります。*location* フィールドでは、*composite_name/name* のファイルを提供する 1 つあるいは複数のホストを指定します。簡単な NFS マウントでは、*location* は、以下のような形式をとります。

```
host:path
```

- *host* には、ファイルシステムをマウントするサーバー名が入ります。*path* には、マウントするディレクトリのパス名が入ります。

各エントリについて、マウントの位置や対応するマウントのオプションのリファレンスは、*composite_name*/*name* に割り当てられます。

options と *location* の両方が省略されると、リファレンスは *composite_name/name* にバインドされません。既存のリファレンスもすべてバインドされません。

たとえば、kuanda のファイルシステムを図 22–4 に示すように、ホスト `altair` からディレクトリ `/export/home/kuanda` に NFS マウントするとしましょう。コマンドは以下のように実行されます。

```
% fncreate_fs -f infile user/kuanda/fs
```

ここでは、以下を含む `infile` を使用します。

```
. altair:/export/home/kuanda
```

図 22–5 に示されるような、複数のサーバーに分散された複雑なファイルシステムを設定する場合、以下のコマンドを実行します。

```
% fncreate_fs -f infile org/sales/fs
```

ここでは、以下を含む `infile` を使用します。

```
tools/db   altair:/export/db
project    altair:/export/proj
project/lib             altair:/export/lib
project/src   deneb:/export/src
```

プロジェクトやそのサブコンテキストの `src` や `lib` の NFS マウントを、読み取り専用に変更する場合、`infile` を以下のように変更します。

```
tools/db          svr1:/export/db
project          -ro     svr1:/export/projproject/lib altair:/export/lib
project/src          svr2:/export/src
```

–ro は、3 行目と 4 行目では必要ありません。なぜならば、src および lib は、project のサブコンテキストであり、これらは、上から –ro マウントオプションを継承するからです。

以下の入力ファイルは、org/sales/fs/project/src を除いて、すべてのマウントを読み取り専用にします。

```
.       -ro
tools/db    svr1:/export/db
project     svr1:/export/proj
project/lib altair:/export/lib
project/src      -rw      svr2:/export/src
```

コマンド行の入力によりファイルコンテキストを作成する

fncreate_fs コマンドにより、以下のようにバインディングの説明をコマンド行で入れることもできます。

```
fncreate_fs composite_name [ mmount_options] [mount_location ...]
```

これは、コマンドの入力ファイル書式を使用し、キーボードから個々のバインディングを入力するのと同じです。先の kuanda のファイルシステムを設定した例は、コマンド行からは、以下のように設定できます。

```
% fncreate_fs user/kuanda/fs altair:/export/home/kuanda
```

同様に、図 22–5 に示した階層構造は、以下のような一連のコマンドを実行することによって、設定できます。

```
% fncreate_fs org/sales/fs/tools/db altair:/export/db
% fncreate_fs org/sales/fs/project altair:/export/proj
% fncreate_fs org/sales/fs/project/lib altair:/export/lib
% fncreate_fs org/sales/fs/project/src deneb:/export/src
```

これら 3 つすべてのマウントを読み取り専用にするには、以下のコマンドを実行します。

```
% fncreate_fs org/sales/fs -ro
```

詳細入力フォーマット

以下の 2 つの項で述べる内容は、入力ファイル、およびコマンド行入力の両方の形式に適用されます。

多重マウントの位置

複数の *location* フィールドが、機能的には同じである複数の位置からエクスポートされた NFS ファイルシステムに対して指定できます。

```
% fncreate_fs org/sales/fs altair:/sales cygnus:/sales
```

オートマウンタが、選択肢の中から最適のサーバーを選択します。リストの中のいくつかの位置が同じパス名を共有している場合、これらは以下のようにコンマで区切ったホスト名のリストを使用して、結合されます。

```
% fncreate_fs org/sales/fs altair,cygnus:/sales
```

ホストは、丸括弧で囲った整数の形でホストに追加された重み係数により加重されることがあります。この場合、数字が小さければ小さいほど、望ましいサーバーであることを示します。デフォルトの重み係数は、ゼロ (もっとも望ましい) です。マイナスの数字は使用できません。

以下の例では、cygnus が望ましいサーバーになっていることを示す 1 つの方法を表わしています。

```
% fncreate_fs org/sales/fs altair(2),cygnus(1):/sales
```

変数の置き換え

前に $ の付く変数名を、fncreate_fs の *options* や *location* のフィールドで使用できます。たとえば、マウントの位置は、以下のように指定できます。

```
altair:/export/$CPU
```

対応するファイルシステムにマウントする際、オートマウンタにより、これらの変数がクライアントに固有の値に置き換えられます。上の例の場合、$CPU は、uname –p の出力結果、たとえば sparc で置き換えられます。

下位互換入力フォーマット

さらにオートマウントマップとの互換として、以下の入力ファイル書式も fncreate_fs で使用できます。

```
name   [mount_options ] [mount_location ...] \
  / offset1 [mount_options1]   mount_location1 ... \
  / offset2   [mount_options2] mount_location2 ... \
  . . .
```

ここでは、各 *offset* フィールドは、スラッシュで区切った階層構造になっています。バックスラッシュ (\) は、1 つの長い行の続きであることを示します。これは、以下を行うのと同じことです。

```
name [mount_options ] [mount_location ...] \
   name/offset1 [mount_options1 ] mount_location1 ... \
   name/offset2 [mount_options2 ] mount_location2 ... . . .
```

最初の行は、*mount_options* と *mount_location* の両方が省略されている場合、省略します。この書式は、互換のためだけのものです。これ以外の機能はないので、あまり使用しないでください。

第 26 章

FNS およびグローバルネーミングシステム

この章では、2 つのグローバルネーミングシステム (DNS と X.500/LDAP) とそれらを FNS でフェデレートする方法について説明します。

FNS およびグローバルネーミングシステム

FNS とグローバルネームサービスの関係についての概要および背景に関しては、463ページの「グローバルネームサービス」を参照してください。

FNS は、エンタープライズネーミングシステムのグローバルネーミングシステム、DNS、および X.500/LDAP のフェデレーションをサポートします。この章では、NIS+ を DNS および X.500 でフェデレートするための手順について説明します。通常、この手順には以下の内容が含まれます。

- NIS+ 階層構造用に NIS+ ルートリファレンスを決定する
- この情報をグローバルネーミングシステムで必要なフォーマットで追加する

注 - エンタープライズレベルのネームサービスが NIS+ もしくは NIS の場合、グローバルネーミングサービスしかフェデレートできません。企業用にファイルベースのネームサービスを使用している場合、DNS も X.500/LDAP もフェデレートできません。

ルートリファレンスを取得する

DNS、あるいは X.500/LDAP でエンタープライズネームサービスをフェデレートする場合、企業や企業外部のグローバルインターネットからの、あるいはそれらへのアクセスができるようにするために、情報をこれらそれぞれのネーミングシステムに追加する必要があります。この情報とは「ルートリファレンス」のことで、これは、ある特定の企業の名前空間の最上部にどのようにして到達するかを示すネットワークアドレス情報からできています。

ルートリファレンスは、XDR で符号化した 1 つの文字列を含む 1 つのアドレスからできています。アドレスのタイプと内容は、使用しているエンタープライズレベルのネームサービスによって、つまり NIS+ かあるいは NIS かによって異なります。

NIS+ ルートリファレンス

エンタープライズレベルのネームサービスが NIS+ の場合、ルートリファレンスのアドレスタイプは、`onc_fn_nisplus_root` になります。ルートリファレンスネットワークアドレスには、必要 (必須の) 要素が 2 つと、オプションの要素が 1 つあります。要素は、以下のように空白文字で区切られます。

```
root-domain  server  [server_IP_address]
```

表 **26–1** NIS+ ルートリファレンス

アドレス要素	説明
root_domain	NIS+ ルートドメインの完全指定名 (末尾にドットが必要)
server	*nis+_root_domain* にサービスを行なっている NIS+ サーバー (マスターあるいは複製) のうちの 1 つのホスト名
server_IP_address	*nis+server* の IP アドレス。これは、*nis+server* の名前がわかっている場合は、オプション。このことは、`/etc/nsswitch.conf` ファイルのリストにあるネームサービスのどれか 1 つにより、これが取得できなければならないことを意味している

たとえば、NIS+ ルートドメインが `doc.com.` (末尾のドットに注意) で、ホスト `nismaster.doc.com` を使ってこれに到達できるとします。この場合ルートリファレンスは以下のようになります。

```
doc.com. nismaster.doc.com
```

上の例で、サーバーの IP アドレスが指定されていないのは、これ以外の方法で取得できるからです。何らかの理由で、その IP アドレスがこれ以外の方法で取得できない場合、ルートリファレンスは以下のようになります。

```
doc.com. nismaster.doc.com 123.123.123.33
```

NIS ルートリファレンス

エンタープライズレベルのネームサービスが NIS の場合、ルートリファレンスのアドレスタイプは、`onc_fn_nis_root` になります。ルートリファレンスネットワークアドレスには必要 (必須の) 要素が 2 つと、オプションの要素が 1 つあります。要素は、以下のように空白文字で区切られます。

```
root_domain  server  [server_IP_address]
```

表 **26–2** NIS ルートリファレンス

アドレスの要素	説明
root_domain	NISドメインの完全指定名 (末尾にスラッシュが必要)
server	*root_domain* にサービスを行なっている NIS サーバー (マスターか、あるいは複製) のうちの 1 つのホスト名
server_IP_address	*nis-server* の IP アドレス。これは、*nis-server* のアドレスがわかっている場合は、オプション。このことは、`/etc/nsswitch.conf` ファイルのリストにあるネームサービスのどれか 1 つにより、これが取得できなければならないことを意味している

たとえば、NIS ドメインが `doc.com` で、ホスト `ypmaster.doc.com` を使ってこれに到達できるとします。ルートリファレンスは以下のようになります。

```
doc.com/ ypmaster.doc.com
```

上の例で、サーバーの IP アドレスが指定されていないのは、これ以外の方法で取得できるからです。何らかの理由でその IP アドレスが、これ以外の方法で取得できない場合、ルートリファレンスは以下のようになります。

```
doc.com/ ypmaster.doc.com 123.123.123.37
```

DNS でフェデレートする

この項では、NIS+ あるいは NIS が使われている下位のエンタープライズネーミングシステムのための、TXT (テキスト) レコードを追加するのに必要な手順を説明します。DNS で下位のネーミングシステムをフェデレートする場合、DNS にリファレンス情報を追加し、下位のネーミングシステムのルートリファレンスへの到達の仕方を記述する必要があります。

1. **ルートリファレンスを取得します。**

   詳細は、556ページの「ルートリファレンスを取得する」を参照してください。

2. **ルートリファレンス TXT レコードを DNS ループバックファイルに追加します。**

デフォルトで、このマニュアルでは、このファイル用に /etc/named.local の名前を使用します (これ以外で、このファイルによく使われる名前は、*domain*.127.0.0 あるいは db.127.0.0 です)。

ルートリファレンス TXT レコードには以下の形式があります。

NIS+ の場合

```
TXT ''XFNNISPLUS rootdomain server [server_IP_address]''
```

たとえば、次のようになります。

```
TXT ''XFNNISPLUS doc.com. nismaster.doc.com''
```

NIS の場合

```
TXT ''XFNNIS rootdomain server [server_IP_address]''
```

たとえば、次のようになります。

```
TXT ''XFNNIS doc.com/ ypmaster.doc.com''
```

TXT レコードは、NS (ネームサーバー) レコードのエントリを含む DNS ドメインに関連していなければなりません。以下は、DNS ドメインの例を、その中でバインドされている NIS+ への参照情報とともに示したものです。

```
ORIGIN doc.com
@ IN SOA foo bar.eng.doc.com
 (
 100 ;; Serial
 3600 ;; Refresh
 3600 ;; Retry
 3600 ;; Expire
 3600 ;; Minimum
 )
 NS nshost
 TXT "XFNNISPLUS doc.com. nismaster 123.123.123.33"
nshost IN A 133.33.33.34
```

DNS ファイルの詳細は、591ページの「DNS の構成ファイルとデータファイル」を参照してください。

3. **TXT** レコードを **DNS** テーブルに追加した後、**DNS** サーバーを再スタートするか、あるいはこれにテーブルを再度読むようシグナルを送ります。

```
# kill -HUP pid
```

pid のところには、in.named のプロセス ID 番号が入ります。

DNS TXT を XFN リファレンス用に使用する方法の詳細は、763ページの「XFN リファレンス用 DNS 文書レコードの書式」を参照してください。

X.500/LDAP でフェデレートする

X.500/LDAP で下位のネーミングシステム (NIS+ または NIS) をフェデレートする場合、以下の規則があります。

- ルートリファレンスの情報は、下位のネーミングシステムへどのように到達するかを示すもので、X.500 に追加する必要があります。
- X.500 クライアント API を指定する必要があります。

X.500 ルートリファレンスを指定する

1. **NIS+** 階層構造用の **NIS+** ルートリファレンスを取得します。

   詳細は、556ページの「ルートリファレンスを取得する」を参照してください。

2. **XFN** リファレンス属性をサポートする **X.500** エントリを作成します。

   たとえば、以下のコマンドは、オブジェクトクラスである top、organization、および XFN-supplement (1.2.840.1135436.25) を使って、c=us/o=doc と呼ばれる新しい X.500 エントリを作成します。XFN-supplement オブジェクトクラスにより、c=us/o=doc エントリに、下位のネーミングシステム用のリファレンス情報を保存できます。

```
# fnattr -a .../c=us/o=doc object-class \
top organization XFN-supplement
```

X.500 エントリがすでに存在していて、XFN-supplement オブジェクトクラスで定義されたものでない場合、これを削除し、オブジェクトクラスを追加して作

成し直す必要があります。そうでないと、下位のネーミングシステムに関するリファレンス情報を保持しておくことができなくなります。

3. 下位のシステムに関するリファレンス情報をエントリに追加します。

   X.500 エントリを作成した後、適切なルートリファレンスを指定したエントリにバインドすることにより、下位のシステムに関する情報を追加できます。

   たとえば、下位のネーミングシステムが NIS+ で、使用する NIS+ サーバーが `nismaster` の場合、以下のように入力します。

```
# fnbind -r .../c=us/o=doc/ onc_fn_enterprise onc_fn_nisplus_root \
"doc.com. nismaster
```

   下位のネーミングシステムが NIS で、使用する NIS サーバーが `ypmaster` の場合、以下のように入力します。

```
# fnbind -r .../c=us/o=doc/ onc_fn_enterprise onc_fn_nis_root \
"doc.com/ ypmaster"
```

   これらの例では、ルートのドメイン名 `doc.com.` を使って、NIS+ あるいは NIS 階層構造のリファレンスを X.500 エントリ `c=us/o=doc` の次のネーミングシステムポインタ (NNSP) にバインドしており、このようにして `doc.com.` 名前空間を x.500 名前空間にリンクしています。

   使用しているアドレスのフォーマットは、556ページの「ルートリファレンスを取得する」で説明したルートリファレンスの形式です。`fnbind` に対する名前の引数 `.../c=us/o=doc/` の末尾に付いているスラッシュは、リファレンスがエントリ自体ではなく、エントリの NNSP にバインドされることを示すために使っていることに注意してください。

   X.500 エントリおよび XFN リファレンスの詳細は、766ページの「XFN リファレンス用 X.500 属性の構文」を参照してください。

X.500 クライアント API を指定する

X.500 クライアント API は、FNS を使って X.500 にアクセスする場合に必要です。以下の 2 つの異なるクライアントのうちどちらか 1 つを使うことができます。

- 「XDS/XOM API」

  XDS/XOM API がインストールされていることが必要です。これは共用オブジェクトである `/opt/SUNWxds/lib/libxomxds.so`からエクスポートされます。X.500 製品については、『*Getting Started with the SunLink X.500 Client Toolkit*』を読んでください。

- 「LDAP (Lightweight Directory Access Protocol - 軽量ディレクトリアクセスプロトコル)」。Solaris リリース 2.6 の一部として、LDAP API が自動的にインストールされています。

使用する API は、各マシンの `/etc/fn/x500.conf` のファイルで指定されています。このファイルには、X.500 および LDAP の設定に関する情報が入っています。このファイルは、直接編集できます。デフォルトの `x500.conf` ファイルには、以下の 2 つのエントリが入っています。

```
x500-access: xds ldap
ldap-servers: localhost ldap
```

localhost および *ldap* のところには、1 つあるいは複数の LDAP サーバーの IP アドレス、またはホスト名が入ります。

最初のエントリでは、X.500 が API にアクセスする順序を指定します。上の例の場合、X.500 はまず XDS/XOM を使用しようとします。XDS/XOM が使用できない場合、LDAP を使用します。エントリが `x500-access: ldap xds` になっている場合、x.500 は LDAP を使い、LDAP が使用できない場合にだけ XDS に戻ります。

2 番目のエントリでは、LDAP を実行しているサーバーの IP アドレス、またはホスト名を表示します。各サーバーが、LDAP 接続が成功するまで、次々に試されます。上の例の場合、*localhost* が最初に試されます。LDAP がそのサーバーで使用できない場合、次のサーバーが試されます。

第 27 章

FNS 属性の管理

この章では、FNS 属性と、その管理の方法について説明します。

属性の概要

属性は、名前付きオブジェクトに使用することができます。属性はオプション指定ですから、属性のないオブジェクトもあれば、1 つあるいは複数の属性を持つオブジェクトもあります。

属性はユニークな属性識別子、属性構文、および 0 個以上の明確な属性値のセットからなります。

XFN で、既存の名前付きオブジェクトに対応する属性値の検索や変更のための、基本的な属性インタフェースが定義されます。これらのオブジェクトはコンテキストの場合もあれば、別のタイプのオブジェクトの場合もあります。コンテキストに関連しているのは、構文属性で、これはコンテキストがどのように複合名を解析するかを示します。

拡張属性インタフェースには、特定の属性を検索したり、オブジェクトやそれに対応する属性を作成するオペレーションが含まれます。

属性を検索する

fnsearch コマンドで、属性を検索します。

fnsearch コマンドの構文を以下に示します。

```
fnsearch [-ALlv] [-n max] [-s scope] name [-a ident ]... [-O|-U] filter_expr [ filter_arg]
```

表 27–1 fnsearch コマンドのオプション

オプション	説明
–n *max*	オブジェクトの最大数だけを表示する
–s *scope*	検索の範囲を設定する
–a *ident*	*ident* に一致する属性だけを表示する
name	複合名
filter_expr	ブール、論理、グルーピング、リレーショナル、比較演算子 (表 27–2 参照)
filter_arg	フィルタ表現の引数 (表 27–2 参照)
–A	信頼できるソースだけを参照する
–L	XFN リンクに従う
–l	一致するオブジェクトのオブジェクトリファレンスを表示する
–v	詳細表示。一致するオブジェクトの詳細なオブジェクトリファレンスを表示する
–O	識別子にOSI OIDを使用する
–U	識別子にDCE UUIDを使用する

表 27–1　fnsearch コマンドのオプション　続く

属性に対応するオブジェクトを検索する

fnsearch コマンドを使って、選択した属性に対応するオブジェクトを検索できます。

たとえば、orgunit/sales/site/ の中の for_sale という属性に対応している属性をすべて見つけ出す場合、以下のコマンドを入力します。

```
% fnsearch orgunit/sales/site/ for_sale
```

属性検索をカスタマイズする

検索パターンにおいて、フィルタ表現のうち以下のすべてを使用することもできます。

表 27–2　fnsearch フィルタ表現の演算子

フィルタ表現演算子	シンボル、あるいは形式
論理演算子	or、and、not
グルーピングのための丸括弧	( )
関係演算子：ある属性を入力した値と比較する	== 少なくとも 1 つの属性値が、入力した値に等しい場合、True (真) != 入力した値に等しい属性値がまったくない場合、True (真) < 少なくとも 1 つの属性値が、入力した値より小さい場合、True (真) <= 少なくとも 1 つの属性値が、入力した値よりも小さいか等しい場合、True (真) >少なくとも 1 つの属性値が、入力した値よりも大きい場合、True (真) >= 少なくとも 1 つの属性値が、入力した値よりも大きいか、等しい場合、True (真) ~= 少なくとも 1 つの属性値が、いくつかのコンテキスト特有のおおまかな一致条件に照らして、入力した値に一致している場合、True (真)。この条件は厳密な一致を含む必要がある
例	**% fnsearch** *name* **"not (make == 'olds' and year == 1983)"**

表 27–2 fnsearch フィルタ表現の演算子 続く

フィルタ表現演算子	シンボル、あるいは形式
置換トークン： シェルスクリプトを書くときに役に立ち、–O および –U オプションを指定すると、OSI OID および DCE UUID を使用できる	**%a** は、属性に置き換えられる **%s** は、文字列に置き換えられる **%i** は、識別子に置き換えられる **%v** は、属性値に置きかえられる (現在は、fn_attr_syntax_ascii しかサポートされていない)
例：	以下の 3 つの例はどれも同じ **% fnsearch** *name* **"color == 'red'"** **% fnsearch** *name* **"%a == 'red'" color** **% fnsearch** *name* **"%a == %s" color red**
ワイルドカード文字列	*, *string, string*, str*ing, %s*
拡張演算子	'name'(ワイルドカードを使った文字列), 'reftype'(識別子), 'addrtype' (識別子)
例：	名前が "Bill" で始まり、IQ 属性が 80 以上のオブジェクトを検索する **% fnsearch name "'name'('bill'*) and IQ > 80"**

検索パターン作成の詳細は、fnsearch(1) マニュアルページを参照してください。

属性を更新する

fnattr コマンドにより、FNS 名前付きオブジェクトに関連する属性を更新したり検索できます。fnattr コマンドを使って、以下の 4 つの属性に関する操作が行えます。

- 属性を追加する

```
fnattr -a [-s] name [-O|-U]  identifier values
```

- 属性を削除する

```
fnattr -d name [[-O|-U]  identifier [values]]
```

■ 属性を変更する

```
fnattr -m name [-O|-U]  identifier oldvalue newvalue
```

■ 属性を表示する

```
fnattr -l name [[-O|-U]  identifier]
```

表 27–3 `fnattr` コマンドのオプション

オプション	説明
name	複合名
identifier	属性名
values	1 つあるいは複数の属性値
oldvalue	変更する属性値
newvalue	新しい属性値
`-a`	新しい属性値を追加 (作成) する
`-d`	属性を削除する
`-m`	属性を変更 (修正) する
`-l`	属性値を一覧表示する
`-s`	「代用」モードで追加する。*identifier* 属性に対する既存の値をすべて削除し、新しい属性値を作成する
`-l`	属性と値を一覧表示する
`-O`	識別子として OSI OID を使用する
`-U`	識別子として DCE UUID を使用する

表 27–3　fnattr コマンドのオプション　続く

各オプションでの識別子の形式は、–O あるいは –U のオプションを使用していない限り、FN_ID_STRING になります。

属性を追加する

属性を追加したり、属性に値を追加したりするには、–a オプションを使用します。この場合、複合名、属性識別子、追加する値も指定します。

```
fnattr -a [-s] name [-O | -U]  identifier value1 [ value2+]
```

以下の例では、属性識別子 model と値 hplaser が thisorgunit/service/printer に追加されます。

```
# fnattr -a thisorgunit/service/printer model hplaser
```

–s は、「代用モードで追加する」という意味のオプションです。指定の識別子を持つ属性がすでに存在する場合、–s はその値をすべて削除し、それらを追加した値に置き換えます。このオプションを指定しないと、指定した属性の値には既存の値と追加した新しい値の両方が含まれることになります。

```
# fnattr -as thisorgunit/service/printer model hplaser
```

上の例では最初に model に対応する値をすべて削除し、hplaser を値として追加します。

属性を削除する

FNS 名前付きオブジェクトの属性を削除するには、–d オプションを使用します。

```
fnattr -d name [[-O | -U]  identifier value1 [ value2+]]
```

削除対象の指定には以下の規則があります。

- 名前のみ。複合名だけを指定し、属性識別子を指定しないと、名前付きオブジェクトの属性はすべて削除される
- 名前および識別子のみ。複合名と属性識別子だけを指定し、属性値を指定しない場合、「*identifier*」によって識別された属性全体が削除される

- 名前、識別子、および値。複合名、属性識別子、および 1 つあるいは複数の属性値を指定した場合、これらの値だけが属性から削除される。属性の最後に残っている値が削除されると、属性自体が削除されたのと同じ結果になる

以下の例では、thisorgunit/service/printer のすべての属性が削除されます。

```
# fnattr -d thisorgunit/service/printer
```

属性の内容を表示する

属性の内容を表示するには、–l オプションを使用します。構文は以下のとおりです。

```
fnattr -l name [[-O | -U]  identifier]
```

以下の例では、thisorgunit/service/printer の model 属性の値が表示されます。

```
# fnattr -l thisorgunit/service/printer model laser postscript
```

識別子を指定しない場合、オブジェクトのすべての属性の内容が表示されます。

属性を変更する

属性値を変更するには、–m オプションを使用します。構文は以下のとおりです。

```
fnattr -m name [-O | -U]  identifier old_value  new_value
```

以下の例では、postscript という値が laser に変更されます。thisorgunit/service/printer の他の属性 (およびその値) には影響がありません。

```
# fnattr -m thisorgunit/service/printer model postscript laser
```

その他のオプション

–O オプションを指定した場合、属性識別子の形式には、ASN.1 の整数を並べてドットで区切る形式 (FN_ID_ISO_OID_STRING、10 進数を並べてドットで区切る形式) を使用します。

–U オプションを指定した場合、属性識別子の形式には、DCE UUID 文字列 (FN_ID_DCE_UUID) を使用します。

パート VI　DNS の管理

パート VI では DNS とその管理の方法について説明します。

- 第 28 章
- 第 29 章

第 28 章

DNS の紹介

この章ではドメイン名システム (DNS) の構造と概要を説明します。

DNS サービスの初期設定と構成に関する情報は、『*Solaris* ネーミングの設定と構成』を参照してください。

注 - DNS の利用で最も一般的で重要なことは、ネットワークをグローバルなインターネットに接続することです。インターネットに接続するには、ローカルなネットワークの IP アドレスが親ドメインの管理者のところに登録されている必要があります。管理者はそれぞれの地理的条件や親ドメインのタイプによって異なります。

DNS の基礎

ドメイン名システム (DNS) は、標準 TCP/IP プロトコル群のひとつのアプリケーション層プロトコルです。これによって DNS ネームサービスを実現します。DNS ネームサービスはインターネットで使用される ネームサービスです。

ここでは基本的な DNS のコンセプトを紹介します。説明に際しては、ネットワークの管理 (特に TCP/IP) に精通していて、NIS+ や NIS といった他のネームサービスについての知識があることを前提とします。

DNS の初期設定と構成に関しては、『*Solaris* ネーミングの設定と構成』を参照してください。

注 - DNS、NIS+、NIS、FNS には同じような機能があり、時として異なる構成要素を定義するのに同じ用語を使用している場合がありますが、この章では、ドメインやネームサーバーといった用語は DNS の機能に合わせて定義しています。なぜなら、DNS の機能は NIS+ や NIS のドメインやサーバーとは大きく異なるからです。

名前のアドレス解決

DNS はインターネット上の複雑で全世界的なコンピュータの階層をサポートしますが、それ自身の基本機能は非常に簡単なものです。すなわち、TCP/IP に準拠したネットワーク用に「名前のアドレス解決」を提供するということです。名前のアドレス解決は「マッピング」ともいい、あるコンピュータのホスト名をインデックスとして用い、その IP アドレスをデータベースから見つけだすプロセスのことです。

名前のアドレスマッピングは、ローカルなコンピュータ上で実行されているプログラムが遠隔コンピュータに接続する必要がある時に行われます。ほとんどの場合、このプログラムでは遠隔コンピュータのホスト名はわかっても、その場所を特定できないことがあります。特に遠隔マシンが自分のサイトから何マイルも離れた他の

会社にある場合には場所を特定できません。プログラムは遠隔マシンのアドレスを得るために、ローカルマシン上で動作している DNS ソフトウェアに要求をに送ります。この時のローカルマシンを「DNS クライアント」と呼びます。

ローカルマシンは、「DNS ネームサーバー」にリクエストを送ります。DNS ネームサーバーは 分散 DNS データベースを保持しています。NIS+ のホストテーブル、ipnodes やローカルの /etc/hosts ファイル、/etc/inet/ipnodes ファイルも同様の情報、すなわち、ホスト名、ipnodes 名、IPv4、IPv6 のアドレス、その他コンピュータの特定のグループに関する情報を保持していますが、DNS データベースはそれらとはあまり似ていません。ローカルマシンは、遠隔マシンの IP アドレスを見つけたり、解決したりするためのリクエストの一部として自分自身のホスト名を送信しますが、ネームサーバーは、それも利用します。そして、もしリクエストしたホスト名が DNS データベースにあれば、その IP アドレスがネームサーバーによってローカルマシンに返されます。

図 28–1 に、DNS クライアントのローカルネットワーク上でクライアントとネームサーバーの間での名前のアドレスマッピングを示します。

図 28–1　名前のアドレス解決

もし、ホスト名がネームサーバーの DNS データベースになければ、そのマシンは権限の外側にある、すなわち、DNS の用語でいう「ローカル管理ドメイン」の外側にあることを意味します。このために、各ネームサーバーは、ローカル管理ドメインに対して権限があるとされています。

幸い、ローカルネームサーバーには、ルートドメインネームサーバーのホスト名と IP アドレスのリストがあります。そしてローカルネームサーバーは、ローカルマシンからのリクエストを「ルートドメインネームサーバー」に転送します。ルートネームサーバーは、580ページの「DNS 階層とインターネット」で説明したように、大きな組織のドメインに対して権限があります。その階層は、上下のツリー構造で編成された UNIX のファイルシステムに似ています。

各ルートネームサーバーには、会社、大学、その他大規模な組織における最上位のドメインネームサーバーのホスト名と IP アドレスがあります。ルートネームサーバーは、ホスト名と IP アドレスがわかっているすべての最上位のネームサーバーにローカルマシンからのリクエストを送信します。リクエストされたホストの IP アド

レスを最上位のネームサーバーのどれかが持っている場合、その情報がローカルマシンに返されます。最上位のサーバーがリクエストされたホストに関する情報を持っていない場合、第 2 レベルのネームサーバーにリクエストが渡されます。このようにローカルマシンからのリクエストは組織の巨大なツリー構造の下へ向かって順に渡されます。最終的にローカルマシンからのリクエストに対する情報をデータベースに持っているネームサーバーが IP アドレスを返します。

図 28–2 はローカルドメインの外側での名前のアドレス解決を示します。

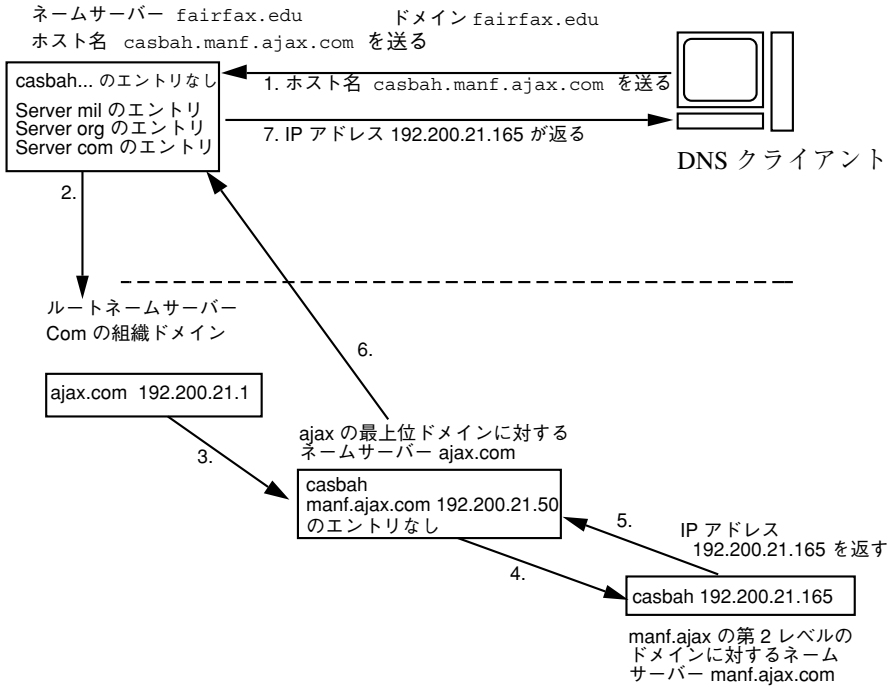


図 28–2　遠隔ホストに対する名前のアドレス解決

DNS 管理ドメイン

DNS から見ると「管理ドメイン」は 1 つの単位として管理されるマシン群を構成しています。このドメインに関する情報は、ドメインに対して権限を持つ 2 つ以上のネームサーバーで管理されます。DNS のドメインはマシンを論理的にグループ分けしたものですが、小規模のビジネスにおいて全マシンが Ethernet に接続された場合のように、マシンの物理的グループと対応している場合もあります。しかし、ローカル DNS ドメインには大学の大規模なネットワークのすべてのマシンが含ま

れることもあります。大学のネットワークにはコンピュータ学科や管理部門に属する多くのコンピュータがあります。

たとえば、Ajax という会社がサンフランシスコとシアトルの 2 つのサイト持っているとします。そして、シアトルのドメインが `Retail.Sales.Ajax.com.` で、`Wholesale.Sales.Ajax.com.` ドメインはサンフランシスコにあるとします。また、`Sales.Ajax.com.` ドメインが 2 つの市に分かれているとします。

各管理ドメインは、固有のサブドメイン名を持たなければなりません。さらに、ネットワークをインターネットに接続するのであれば、そのネットワークの属する管理ドメインはすでに登録されたものでなければなりません。ドメイン名とドメインの登録に関する詳細は 582ページの「インターネットへの参加」の節を参照してください。

in.named と DNS ネームサーバー

すでに説明したように、管理ドメイン内のネームサーバーは、DNS データベースを保持しています。また、ネームサーバーでは `in.named` デーモンが動作しています。このデーモンで DNS サービスが実装されます。そのサービスの中でも重要なのが名前のアドレスマッピングです。`in.named` はパブリックドメインの TCP/IP プログラムで Solaris 8 リリース環境に含まれています。

注 - `in.named` デーモンは、カリフォルニア大学バークレー校で開発されたので、Berkeley Internet Name Domain service (BIND) と呼ばれることもあります。

以下のような、3 つのタイプの DNS ネームサーバーがあります。

- 主サーバー
- 副サーバー
- キャッシュオンリーサーバー

各ドメインには、主サーバーが 1 つとバックアップのための副サーバーが少なくとも 1 つ必要です。主サーバーと副サーバーの詳細は、584ページの「ゾーン」を参照してください。

DNS クライアントとリゾルバ

あるマシンを DNS クライアントにするには、「リゾルバ」を実行する必要があります。リゾルバはデーモンでもなければ単一のプログラムでもなく、アプリケー

ションによって使用される動的ライブラリルーチンの集合です。マシン名を知る必要があるときに、このライブラリが使用されます。リゾルバの機能はユーザーの照会を解決することです。それを行うために、リゾルバはネームサーバーに照会します。するとネームサーバーは要求された情報か、または他のサーバーに照会する旨を返します。一度リゾルバを設定すれば、そのマシンはネームサーバーに DNS サービスを要求できるようになります。

あるマシンの `/etc/nsswitch.conf` ファイルで `hosts:dns` (または `hosts` 行に `dns` を含む別のパターン) が指定されていると、リゾルバのライブラリが自動的に使用されます。もし、`nsswitch.conf` ファイルが `dns` より前に、他のネームサービスを指定した場合、最初にそのネームサービスに対してホスト情報を問い合わせ、要求されたホストの情報が見つからなかった場合にだけリゾルバのライブラリが使用されます。

たとえば、`nsswitch.conf` ファイル内の `hosts` の行で `hosts:nisplus dns` と指定されている場合、ホストの情報を得るために NIS+ ネームサービスが最初に検索されます。もし、NIS+ で情報が見つからなかったとき、DNS リゾルバが使用されます。NIS+ や NIS といったネームサービスではそれ自身のネットワーク内にあるホストに関する情報だけを保持しているため、結果的に `hosts:nisplus dns` と `switch` ファイルで指定することは、ローカルホストに関する情報に対しては NIS+ を使用し、インターネット上の遠隔システムに関する情報に対しては DNS を使用するように指定することになります。

DNS クライアントには以下のような 2 つの種類があります。

- クライアント専用

  クライアント専用 DNS は `in.named` を実行せず、代わりにリゾルバを実行します。リゾルバはドメインに対するネームサーバーのリストを保持しているので、そこに問い合わせが転送されます。

- クライアント兼サーバー

  クライアントマシンのリゾルバによって転送されてきた問い合わせを解決するために `in.named` で提供されるサービスを使用します。

Solaris 8 リリース環境には、リゾルバを構成している動的ライブラリルーチンが含まれています。『*Solaris* ネーミングの設定と構成』で、ホストを DNS クライアントとして設定する方法が説明されています。

DNS 名前空間について

全世界の DNS 管理ドメインの集合全体は「 DNS 名前空間」と呼ばれる階層構造を形成しています。ここでは、名前空間の組織がどのようにローカルドメインやインターネットに影響するのか説明します。

DNS 名前空間の階層

DNS ドメインは、UNIX のファイルシステムと同様、木の根に似た下に向きの枝分かれで構成されています。各枝分かれがドメインであり、そこから分かれる各枝が「サブドメイン」です。「ドメイン」と「サブドメイン」は相対的な関係を示します。階層の中で、あるドメインは、その上にあるドメインに対するサブドメインになり、その下にあるサブドメインの親ドメインになります。

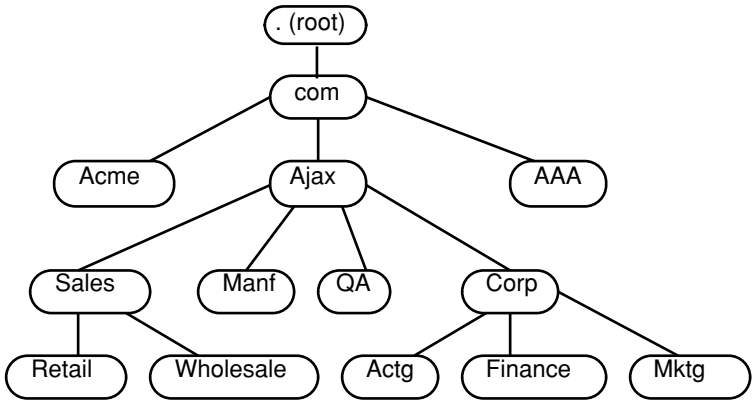


図 28–3　ドメインとサブドメイン

たとえば、図 28–3 において com は Acme、Ajax、AAA の各ドメインの親ドメインです。あるいは、これらのドメインは com ドメインのサブドメインということもできます。このように考えると、Ajax ドメインは 4 つのサブドメイン (Sales、Manf、QA、Corp) の親ドメインになっています。

あるドメインには、1 つの親 (あるいは最上位) ドメインと、(もしあれば) その下のサブドメインが含まれます。ドメインの名前付けは最下位 (階層の底) のサブドメインから始まり、最後がルートドメインとなっています。図 28–3 の Mktg.Corp.Ajax.Com. がその例です。

ローカルドメイン内の DNS 階層

大規模な会社であれば、多くのドメインをサポートしており、ローカルな名前空間が構成されていることでしょう。図 28–4 に、ある会社に存在するドメインの階層構造の例を示します。この会社の最上位のドメイン、すなわちルートドメインは `ajax.com` で、`sales.ajax.com`、`test.ajax.com`、`manf.ajax.com` の 3 つのサブドメインがあります。

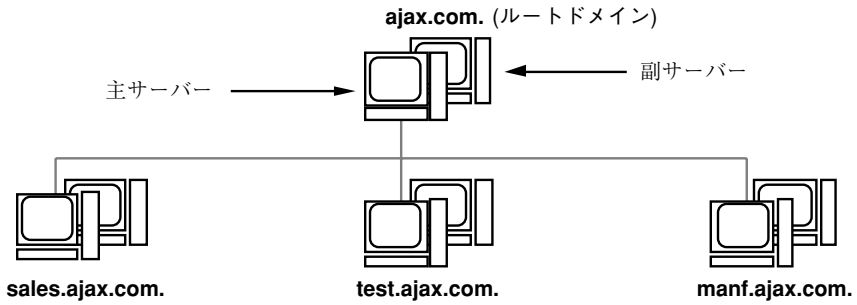


図 28–4　ある組織での DNS ドメインの階層

DNS クライアントは、そのドメインをサポートするサーバーだけにサービスをリクエストします。もし、クライアントの必要としている情報がそのドメインサーバーにない場合、リクエストは親サーバーに転送されます。親サーバーは、1 つ上の階層のドメインサーバーです。リクエストが最上位のサーバーに達した場合、最上位のサーバーはリクエストされたドメインが有効かどうか調べます。それが有効でない場合、サーバーは「not found」というメッセージをクライアントに返します。リクエストされたドメインが有効な場合、最上位のサーバーはそのドメインをサポートしているサーバーに要求を転送します。

DNS 階層とインターネット

図 28–4 に示したドメインの階層は、概念的には世界規模のインターネット上でサポートされる巨大な DNS 名前空間の枝葉のようなものです。

インターネットの DNS 名前空間は 図 28–5 で示すように階層状に組織されます。それは、ピリオド (.) で表されるルートディレクトリと最上位のドメインの階層 2 つ (組織的なものと地理的なもの) で構成されます。com ドメイン (図 28–3) はインターネットに存在する最上位の組織ドメインの 1 つです。

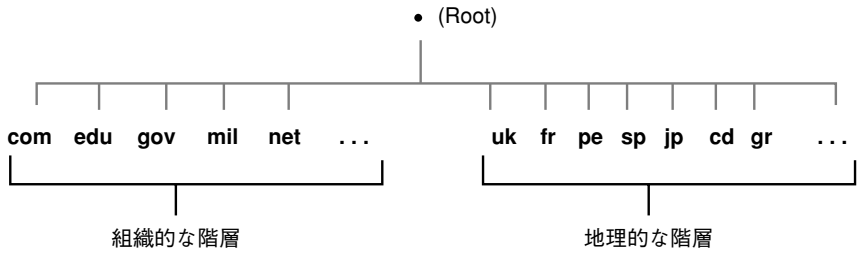


図 28–5　インターネットのドメインの階層

現在、組織的な階層の名前空間は、表 28–1 に示した最上位のドメインに分けられます。将来、これ以外にも最上位の組織ドメインが追加される可能性があります。

表 28–1　インターネットの組織ドメイン

ドメイン	目的
com	営利団体
edu	教育機関
gov	行政機関
mil	軍事組織
net	大手のネットワークサポートセンター
org	非営利団体、その他
int	国際組織

地理的な階層においては、各国に対して 2、3 文字の識別子が割り当てられ、それが、各国に対する公式名として用いられます。たとえば、イギリス国内のドメインは、uk という最上位のドメインのサブドメインになります。日本国内のドメインは、jp のサブドメインで、他の国に関しても同様です。

インターネットへの参加

インターネットのルートドメイン、すなわち最上位のドメイン (組織的および地理的) は、様々なインターネット運営体によって管理されています。規模にかかわらずネットワークを有する人は、そのネットワークのドメイン名を組織的な階層か地理的な階層に登録することによってインターネットに参加できます。

DNS ドメインはドメイン名を持つ必要があります。インターネットに接続しないで、ネームサービスとして DNS を使用したいのであれば、ドメインやサブドメインに任意の名前を付けることができます。しかし、インターネットに参加するのであれば、そのドメイン名はインターネット運営体に登録する必要があります。

インターネットに接続するために必要な条件は以下のとおりです。

- DNS ドメイン名を、適当なインターネット運営体に登録します。
- そのインターネット運営体から、ネットワークの IP アドレスを入手します。

上記のことを行うには、以下のような 2 つの方法があります。

- 適当なインターネット運営体またはその代理団体と直接連絡をとる方法。アメリカ合衆国では、現在 InterNIC 社がネットワークアドレスとドメインの登録を取り扱っています。
- インターネットサービスプロバイダ (ISP) と契約する方法。ISP は、コンサルティングから実際の接続まで広範なサービスを提供しています。

ドメイン名

ドメイン名は、DNS 名前空間全体でのドメインの位置を表します。それは UNIX のファイルシステムでパス名がファイルの位置を表しているのと同じです。ローカルドメインが登録されると、そのドメイン名は所属するインターネットの階層の名前に追加されます。たとえば、図 28–4 で示した Ajax ドメインはインターネットの階層である com の一部として登録されました。したがって、そのインターネットドメイン名は `ajax.com` となります。

図 28–6 は、`ajax.com` ドメインがインターネット上の DNS 名前空間のどこに位置しているのかを示しています。

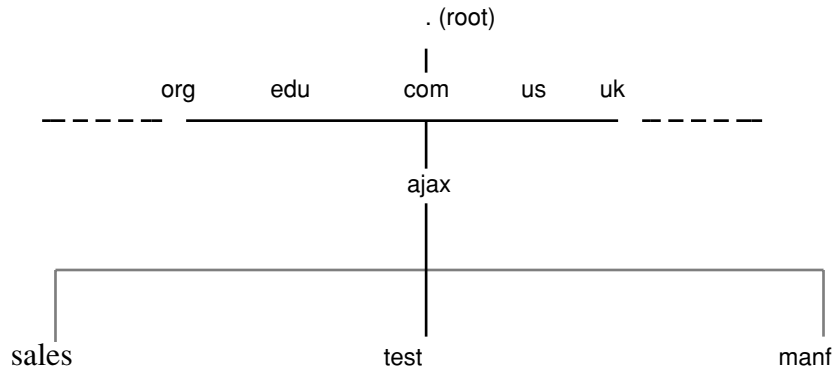


図 28–6　DNS 名前空間における Ajax ドメインの位置

ajax.com のサブドメインには以下のような名前があります。

```
sales.ajax.com
test.ajax.com
manf.ajax.com
```

DNS ではドメイン名を大文字にすることは可能ですが、必須ではありません。次にマシン名とドメイン名の例を挙げます。

```
Boss.manf.ajax.com
quota.Sales.ajax.com
```

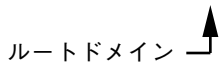
インターネットではドメインの管理は、ドメイン内のホスト名に対する権限を各ドメインに保証し、また各ドメインがそれ以下のレベルに権限を委任することによって行われます。したがって、com ドメインはそのドメイン内のホスト名に対して権限があります。また com ドメインは、Ajax.com ドメインを構成することに対して権限を与え、そのドメイン内の名前に対する権限を委任します。それを受けて Ajax.com はドメイン内のホストに名前を割り当てて、Sales.Ajax.com、Test.Ajax.com、Manf.Ajax.com の各ドメインの編成を承認します。

完全指定ドメイン名

ドメイン名にローカルドメインから DNS のルートドメイン (.) までのすべての DNS のドメインが含まれているとき、そのドメイン名は「完全指定されている」といいます。概念的には完全指定ドメイン名は、UNIX ファイルの絶対パス名と同様に、

ルートへのパスを示しています。しかし、完全指定ドメイン名を読む場合、左から右に行くにしたがって最下位から最上位となります。したがって、完全指定ドメイン名は次のような構文になっています。

<local_domain_name>.<Internet_Org_name> .



ajax ドメインとそのサブドメインの完全指定ドメイン名は次のとおりです。

```
ajax.com.
sales.ajax.com
test.ajax.com.
manf.ajax.com
```

ここで、一番右に付けられたドット (.) に注意してください。

ゾーン

あるドメインの DNS サービスは、577ページの「in.named と DNS ネームサーバー」で最初に説明したネームサーバーの集合で管理されます。ネームサーバーは、単一のドメイン、複数のドメイン、複数のドメインとその下のサブドメインの一部または全部を管理できます。あるネームサーバーによって制御される名前空間の一部は、「ゾーン」と呼ばれます。したがって、ネームサーバーはゾーンに対して権限があるといわれます。ネームサーバーの責任者は、ゾーン管理者とも呼ばれます。

ネームサーバーのデータベース内のデータは、「ゾーンファイル」と呼ばれます。ゾーンファイルの 1 つには IP アドレスとホスト名が格納されています。ftp または telnet のようなユーティリティでホスト名を用いて遠隔ホストに接続しようとすると、DNS は名前のアドレスマッピングを実行し、ゾーンファイルの中でホスト名を探して、IP アドレスに変換します。

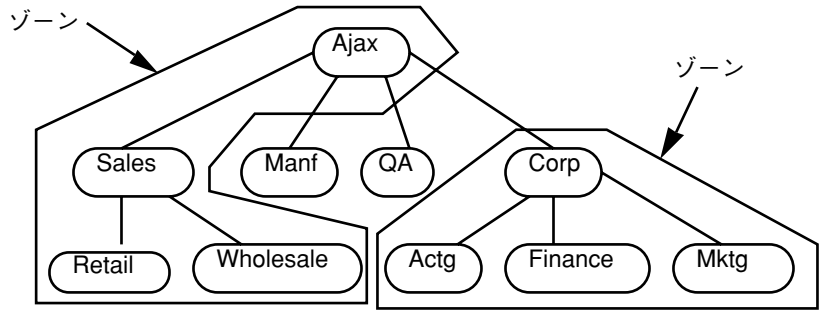


図 28–7　ドメインとゾーン

たとえば、図 28–7 で示した Ajax ドメインは最上位のドメインである Ajax と、4 つのサブドメイン、そして 5 つのサブサブドメインからなります。それは、太線で示す 4 つのゾーンに分けられます。したがって、Ajax ネームサーバーは、Ajax、Sales、Retail、Wholesale の各ドメインからなるゾーンを管理します。Manf と QA の両ドメインは別のゾーンで、それ自身のネームサーバーからサービスを受けます。Corp ネームサーバーは Corp、Actg、Finance、Mktg ドメインからなるゾーンを管理します。

逆マッピング

DNS データベースには、マシンのホスト名を見つけだすキーとして IP アドレスを用いるゾーンファイルもあります。このファイルで IP アドレスのホスト名解決が可能になります。このプロセスを、「逆解決」または、より一般的には逆マッピングと呼びます。逆マッピングは基本的にはメッセージを送ってきたマシンの識別情報を確認したり、ローカルホスト上でのリモート操作を許可したりするために使用されます。

in-addr.arpa ドメイン

in-addr.arpa ドメインとは、DNS 名前空間において概念的な存在で認証 (許可) のためにドメインでなく IP アドレスを用います。このドメインは、ゾーンの一部ですが、これによってアドレスから名前のマッピングが可能になります。

DNS ドメイン名では、もっとも左が最下位のサブドメイン、最も右がルートとして読まれるので、in-addr.arpa ドメインでも、IP アドレスは最下位のレベルからルートになるように書き換えられます。すなわち、IP アドレスは逆転します。たとえば、あるホストの IP アドレスが 192.200.21.165 であるとします。in-addr.arpa ゾーンファイルでは、そのアドレスは

165.21.200.192.in-addr.arpa. と書かれます。ここで最後のドットは in-addr.arpa ドメインのルートを示しています。

DNS サーバー

DNS サーバーには複数の機能があります。

- 「ゾーンマスターサーバー」

  マスターネームサーバーは、ゾーンに対応するすべてのデータを保持しています。マスターサーバーは一般に、権限があるネームサーバー (587ページの「マスターサーバー」を参照) と呼ばれます。

  マスターサーバーには 2 つのタイプがあります。

  - 「ゾーン主マスターサーバー」

    各ゾーンには、そのゾーンの主マスターサーバー (587ページの「主マスターサーバー」を参照) と呼ばれるサーバーが 1 つあります。

  - 「ゾーン副マスターサーバー」

    ゾーンには、1 つ以上の副マスターサーバーがあります。副マスターサーバーは、DNS データをゾーンの主マスターサーバー (587ページの「主マスターサーバー」を参照) から入手します。

- 「キャッシュオンリーサーバー」

  サーバーは、DNS データのキャッシュを保持するという意味で、すべてキャッシュサーバーです。キャッシュオンリーサーバーは in-addr.arpa. ドメイン (588ページの「キャッシュサーバーとキャッシュオンリーサーバー」を参照) だけに対してのマスターサーバーです。

- 「ルートドメインサーバー」

  ルートドメインサーバーは、ユーザーの DNS ドメインの階層の最上位に対して権限があるサーバーです。もしユーザーのネットワークがインターネットに接続されているなら、ルートドメインサーバーはインターネット上にあります。もしネットワークがインターネットに接続されていないなら、ルートドメインサーバーを設定する必要があります (588ページの「ルートドメインネームサーバー」を参照)。

個々の異なるサーバーの機能は、同一のマシン上でも実行可能です。たとえば、あるマシンを、あるゾーンの主マスターサーバーにし、同時に別のゾーンの副マスターサーバーにできます。このマニュアルでは、主サーバー、副サーバー、キャッシュオンリーサーバーというのは特定のマシンの呼び名ではなく、マシンがあるゾーンに対して果たしている役割のことを指しています。

マスターサーバー

マスターネームサーバーは、ゾーンに対応するすべてのデータを保持しています。マスターサーバーは一般に、権限があるネームサーバーと呼ばれます。任意のゾーンに対応するデータは 2 つ以上の権威があるサーバーで使用可能になっている必要があります。ここで、1 つのネームサーバーは主マスターサーバーとし、少なくとも 1 つ以上は副マスターサーバーとする必要があります。副マスターサーバーは、主マスターサーバーが使用できないときや負荷過剰のときのバックアップとして機能します。

サーバーは複数のゾーンに対するサーバーとして機能できます。すなわち、あるゾーンに対しては主マスターで、別のゾーンに対しては副マスターとなります。

主マスターサーバー

主マスターサーバーは、自分用のデータのマスターコピーを `in.named` の起動時にディスクから読み込む DNS ネームサーバーです。ゾーンの主マスターサーバーは、そのゾーンに対する変更を行なった場所にあります。主マスターサーバーは、そのゾーンに関する DNS 情報の情報源です。また、主マスターサーバーはそのゾーン内の副サーバーやゾーン外のサーバーに権限を任せることもあります。

副マスターサーバー

副マスターサーバーにはそのゾーン用のデータのコピーがあります。主サーバーはそのデータを副サーバーに送り、権限を任せます。クライアントは、DNS 情報を副サーバーに照会できます。副サーバーを使用することによって、負荷が複数のマシンに分散され、応答時間を短縮してネットワークのオーバーヘッドを減らすことができます。また、副サーバーは、主サーバーが使用できないときに代わりの機能を果たします。

副サーバーは `in.named` を起動するとき、ゾーンに関するすべての情報を主サーバーにリクエストします。副サーバーはデータベースを更新する必要があるかどう

かを調べるために主サーバーを定期的にチェックします。最新のゾーンデータベースを主サーバーから副サーバーに送信するプロセスを「ゾーン転送」と呼びます。したがって、副サーバー上のデータファイルは変更しません。ゾーンの主サーバーのデータサーバーを変更します。そして、副サーバーのファイルは主サーバーから変更されます。

キャッシュサーバーとキャッシュオンリーサーバー

すべてのネームサーバーは「キャッシュサーバー」です。ネームサーバーは受け取ったデータをそれが期限切れになるまでキャッシュに書き込みます。期限切れのプロセスは、データに結びついている生存期間 (time-to-live = TTL) フィールドで制御されます。

さらに、どんなゾーンに対しても権限がない「キャッシュオンリーサーバー」を設定することもできます。キャッシュオンリーサーバーは、`in-addr.arpa.` ドメイン以外のゾーンだけに対してのマスターサーバーです。キャッシュオンリーサーバーは、権限があるネームサーバーが処理するのと同様の照会を扱いますが、権威データ自体は保持しません。

キャッシュオンリーサーバーは、権限があるサーバーより少ないメモリーで動作しますが、もし主サーバーまたは副サーバーが使用できない場合、キャッシュオンリーサーバーだけでは機能しません。

ルートドメインネームサーバー

DNS 名前空間には、ルートドメインに対して権限がある「ルートドメインネームサーバー」が 1 つ以上必要です。

- ネットワークがインターネットに接続されている場合、ルートドメインサーバーはルートドメインであるインターネットのサイトにあります。この場合、589ページの「インターネットルートドメインサーバー」で説明しているように、ルートドメインサーバーがあるサイトのインターネット IP アドレスをキャッシュファイルに入れてください。
- ネットワークがインターネットに接続されていない場合、589ページの「非インターネットルートドメインサーバー」で説明しているように、ローカルネットワーク上のルートレベルのドメイン内に主ネームサーバーと副ネームサーバーを設定する必要があります。これによってローカルネットワーク内のすべてのド

メインは、参照すべき権限があるサーバーを持つことになります。この設定がされていないと、マシンは要求された照会を解決できません。

ルートドメインネームサーバーを識別する情報はキャッシュファイルに格納されています。このマニュアルでも、またほとんどの Solaris のサイトでも、このキャッシュファイルを named.ca (このファイルのどこでも共通の別名は root.cache、named.root、db.cache) と呼びます。各サーバーのブートファイルには、ルートドメインネームサーバーの情報を保持しているファイルを識別するレコードがあります。

インターネットルートドメインサーバー

ネットワークがインターネットに接続されている場合、DNS ネームサーバーのブートファイルでは、共通キャッシュファイル (一般に named.ca と呼ばれる) を指定して、ルートドメインネームサーバーを識別する必要があります。このファイルのテンプレートは、以下のいずれかを通して、InterNIC から入手できます。

- 匿名 FTP

  FTP のサイトは、ftp.rs.internic.net、ファイル名は、/domain/named.root

- Gopher

  Gopher のサイトは、rs.internic.net で、ファイルは、named.root。このファイルは「InterNIC Registration Services」メニューの「InterNIC Registration Archives」サブメニューにある

このマニュアルに記載の方法で DNS のファイルに名前を付けるのであれば、上記の方法で入手したファイルは、/var/named/named.ca とする必要があります。

非インターネットルートドメインサーバー

ネットワークがインターネットに接続されていない場合、ルートドメインネームサーバーとして機能するように 1 つ以上のサーバーを設定する必要があります。ネットワーク上のすべての DNS ネームサーバーのブートファイルで共通のキャッシュファイル (一般的には named.ca) を指示する必要があります。これで、ルートドメインネームサーバーが識別されます。次に、ルートドメインネームサーバーを識別するキャッシュファイルを作成します。

1 台のマシンを複数のマシンの主ドメインネームサーバーとすることができますので、ルートドメインネームサーバーを作成する最も簡単な方法は、最上位のドメインにサーバーを持ち、それを論理的な (.) ドメインのサーバーにすることです。

たとえば、ネットワークに `solo` というドメイン名を与えたとします。DNS マスターネームサーバーは `dnsmaster.solo.` (最後にドットを付ける) となります。この場合、`dnsmaster` を (.) ドメインのルートマスターサーバーとしたことになります。

ネットワークに最上位のドメインが複数ある場合、ルートドメインサーバー名は最上位のすべてのドメインの主ネームサーバーである必要があります。たとえば、ネットワークが、`solo` と `private` という名前の非階層の 2 つのドメインに分かれているとすると、同じサーバーが両方のルートマスターサーバーである必要があります。上の例では、`dnsmaster.solo.` が、`solo` と `private` の両方のドメインのルートドメインマスターとなります。

DNS のメールデリバリへの影響について

DNS には 2 つの基本的なサービスがあります。1 つは、574ページの「名前のアドレス解決」で 説明しているように 、名前からアドレス (またはその逆のアドレスから名前) のマッピングを行うサービスで、もう 1 つは、インターネット上でメールを配信する `sendmail` や POP といったメール配信エージェントを助けるサービスです。

インターネット上でメールを配信するのに、DNS は「メール交換レコード」(MX レコード) を用います。ほとんどの組織は、その組織内にあるホストに宛てられたインターネットから来るメールを直接配信することを許していません。そのかわりに、1 台のセントラルメールホスト (またはメールホストの集合) を用いて、入ってくるメールメッセージを途中で止めて宛先に振り分けます。

メール交換レコードで、ドメイン内の各マシンにサービスを提供しているメールホストが識別されるため、メール交換レコードには遠隔組織の DNS ドメイン名と、その IP アドレスまたは対応するメールホストのホスト名のいずれかが列挙されています。

DNS の構成ファイルとデータファイル

DNS のネームサーバーには、in.named デーモンに加えて、named.conf と呼ばれるブートファイル、resolv.conf という名前のリゾルバファイル、4 つのタイプのゾーンデータファイルがあります。

DNS データファイル名

内部的に矛盾がなければ、ゾーンデータファイルはどんな名前にもできます。このため、異なるサイトで作業をしようとする場合や DNS 関連のマニュアルや本を参照する場合に、混乱するかもしれません。たとえば、Sun のマニュアルやほとんどの Solaris のサイトのファイル名は、『*DNS and BIND*』(Paul Albitz & Cricket Liu 著 浅羽登志也／上水流由香 監訳、アスキー出版局、1995年) で使用されている名前とは異なりますし、その両方の命名法は、『*Name Server Operations Guide for BIND*』(カリフォルニア州立大学刊、パブリックドメイン) での命名法とも若干異なります。

さらに、このマニュアルや他の DNS の文書では、そのファイルの主要な目的を表すために汎用的な名前が用いられ、レコードの例ではそのファイルに対して例として特定の名前が用いられています。たとえば、Solaris のネームサービスに関するマニュアルでは、ファイルの機能と役割を記述するために hosts が総称名として用いられ、db.doc と db.sales.doc という名前が、例として用いられています。

参考として、表 28–2 では上で述べた 3 つのマニュアルにおける BIND のファイル名を比較します。

表 **28–2** BIND ファイル名 例

Solaris	**O'Reilly** その他	カリフォルニア州立大学バークレイ校	ファイルの内容と役割
/etc/named.conf	/etc/named.conf	/etc/named.conf	構成ファイルは、それが実行されるサーバーのタイプ、および「マスター」、「スレーブ」、または「スタブ」として機能するゾーンを指定する。また、セキュリティ、ロギング、およびゾーンに適用されるオプションの細かい細分性を定義する
/etc/resolv.conf	/etc/resolv.conf	/etc/resolv.conf	各クライアント (DNS サーバーを含む) 上に存在するファイル。DNS 情報を探すためにクライアントが照会するサーバーを示す
named.ca	db.cache db.root	root.cache	ルートサーバーの名前とそのアドレスがリストされている
総称名: hosts 例: db.doc db.sales	総称名: db.domain 例: db.movie db.fx	総称名: hosts 例: ucbhosts	サーバーがサービスを提供するローカルゾーン内のマシンに関する全データが格納されている
総称名: hosts.rev 例: doc.rev	総称名: db.ADDR 例: db.192.249.249 db.192.249.253	hosts.rev	逆変換 (アドレスから名前への変換) が可能な特殊ドメイン in-addr.arpa. 内のゾーンを指定する
named.local	総称名: db.cache 例: db.127.0.0	named.local	ローカルループバックインタフェース (local host) 用のアドレスを指定する
$INCLUDE ファイル	$INCLUDE ファイル	$INCLUDE ファイル	データファイル内の $INCLUDE() ステートメントによって識別されるファイル

表 28–2　BIND ファイル名 例　続く

named.conf ファイル

BIND 8.1 は、新しい構成ファイル /etc/named.conf を追加し、/etc/named.conf ファイルと置き換えます。/etc/named.conf ファイルは、サーバーを、主、副、またはキャッシュ専用ネームサーバーとして確立します。また、サーバーが権限を持つゾーンを指定し、どのデータファイルを読み取って初期データを取得するかを指定します。

/etc/named.conf ファイルには、以下を実現するステートメントが含まれています。

- NIS+ ホストが読み取り権／書き込み権を持つ IP アドレスのコレクションを定義するアクセス制御リスト (ACL) によるセキュリティ
- ログ仕様
- すべてのゾーンに対してではなく、ゾーンのセットに対して選択的に適用されるオプション

構成ファイルは、デーモンがサーバーの起動スクリプト /etc/init.d/inetsvc によって起動されるとき、in.named によって読み取られます。構成ファイルは、in.named を他のサーバーまたは指定されたドメインのローカルデータファイルに導きます。

named.conf ステートメント

named.conf ファイルには、ステートメントおよびコメントが含まれています。ステートメントはセミコロンで終わります。一部のステートメントは、ステートメントのブロックを含むことができます。ブロックの中の各ステートメントもセミコロンで終わります。

named.conf ファイルは、以下のステートメントをサポートします。

表 **28–3**

acl	アクセス制御に使用する名前付き IP アドレス一致リストを定義。このアドレス一致リストは、1 つ以上の IP アドレス (小数点表記) または IP 接頭辞 (小数点表記の後にスラッシュとネットマスクのビット数をつけたもの) を指定する。acl ステートメントによって名前付き IP アドレス一致リストを定義しなければ、他の場所で名前付き IP 一致リストを使用することはできまない。前方参照は不可
include	include ステートメントのある箇所にインクルードファイルを挿入する。include を使用すると管理の簡単なかたまりに構成を分割できる
key	特定のネームサーバー上での認証および承認に使用する鍵の ID を指定する。server ステートメントを参照
logging	サーバーが記録する情報およびログメッセージの宛先を指定
options	グローバルサーバー構成オプションを制御し、他のステートメントのデフォルト値を設定
server	リモートネームサーバーに関連付ける指定された構成オプションを設定する。すべてのサーバーに対してではなく、サーバーごとに選択的にオプションを適用
zone	ゾーンを定義する。すべてのゾーンに対してではなく、ゾーンごとに選択的にオプションを適用

例 **28–1** 主サーバー用マスター構成ファイルの例

```
options {
         directory "/var/named";
         datasize 2098;
         forward only;
         forwarders {
                  99.11.33.44;
         };
         recursion no;
         transfers-in 10;
         transfers-per-ns 2;
         allow-transfer {
                  127.0.1.1/24;
         };
};

logging {
         category queries { default_syslog; };
};

include "/var/named/abcZones.conf"
```

(続く)

```
// これは主ファイルの名前です
zone "cities.zn" {
         type master;
         file "db.cities.zn";
};

zone "0.0.127.in-addr.arpa." {
          type master;
         file "db.127.cities.zn";
};

zone "168.192.in-addr.arpa" {
         type master;
         file "db.cities.zn.rev";
};

zone "sales.doc.com" {
         type slave;
         file "slave/db.sales.doc";
         masters {
         192.168.1.151;
 };
};

zone "168.192.in-addr.arpa" {
          type slave;
         file "slave/db.sales.doc.rev";
         masters {
                  192.168.1.151;
         };
};
```

BIND 4.9.x から BIND 8.1.x への移行

スーパーユーザーになって、Korn シェルスクリプト /usr/sbin/named-bootconf を実行して BIND 4.9.x の named.boot ファイルを BIND 8.1.x の named.conf ファイルに変換します。named-bootconf(1M) を参照してください。

注 - Solaris 8 では、named.boot ファイルは無視されます。

---

named.ca ファイル

named.ca ファイルによって、ルートサーバー名が確立され、そのアドレスが列挙されます。ネットワークがインターネットに接続されているなら、named.ca に

は、インターネットのネームサーバーが表示されます。接続されていなければ、ローカルネットワークのルートドメインネームサーバーが表示されます。in.named デーモンは、サーバーの 1 つに接続できるまで、サーバーのリストを一巡します。そして、そのサーバーから現在のルートサーバーのリストを入手します。デーモンは、このリストを named.ca の更新のために用います。

hosts ファイル

hosts ファイルには、ローカルゾーン内のマシンに関するすべてのデータが含まれています。このファイル名は、ブートファイル内で指定します。/etc/hosts との混同を避けるために、hosts 以外の名前を付けます。たとえば、これらのファイルに db.*domain* パターンを使用して名前を付けることができます。この命名方法により、doc.com と sales.doc.com ドメインのホストファイルは db.doc と db.sales になります。

hosts.rev ファイル

hosts.rev ファイルで、逆 (アドレスから名前) マッピングを行うための特別な in-addr.arpa. ドメインのゾーンを指定します。このファイル名は、ブートファイル内で指定します。

named.local ファイル

named.local で、ローカルループバックインタフェースのアドレスまたはローカルホストをネットワークアドレス 127.0.0.1 とともに指定します。このファイル名はブートファイルで指定します。他のファイルと同様、このマニュアルで使われていない名前を付けることもできます。

$INCLUDE ファイル

インクルードファイルは、DNS データファイルの中の $INCLUDE() 文で指定された任意のファイルです。$INCLUDE ファイルは、異なるタイプのデータを便宜的に複数のファイルに分けるのに使うことができます。

たとえば、データファイルに次のような行が含まれているとします。

```
$INCLUDE /etc/named/data/mailboxes
```

この行によって、/etc/named/data/mailboxes ファイルがその時点で読み込まれます。この例では、/etc/named/data/mailboxes が、$INCLUDE ファイルです。$INCLUDE ファイルは必要に応じて、必要な数だけ使用します。必要がなければ使う必要はありません。

データファイルのリソースレコード書式

DNS のデーモン in.named によって使用されるすべてのデータファイルは標準リソースレコード書式で書かれます。各 DNS データファイルには、必ずリソースレコードが含まれる必要があります。ここでは、DNS データファイルと各ファイルに含まれるべきリソースレコードについて説明します。

標準リソースレコード書式

標準リソースレコード書式では、データファイルの各行は、「リソースレコード」(RR) と呼ばれます。リソースレコードには空白で区切られた次のようなフィールドがあります。

namettl	*class*	*record-type*	*record-specific-data*

フィールドの順は常に同じですが、最初の 2 行はオプション (カッコ付きで示す) です。また、最後は *record-type* フィールドによって変化します。

name フィールド

最初のフィールドは、そのレコードに適用するドメイン名のフィールドです。RR でこのフィールドが空白のままであれば、デフォルトとして直前の RR の *name* フィールドの値が用いられます。

ゾーンファイルのドメイン名は、ドットで終わる完全指定名でも、相対名でもかまいません。相対名の場合、現在のドメインが相対名に付加されます。

ttl フィールド

2 番目のフィールドは、オプションの有効期限フィールドです。このフィールドで、データを破棄する前にデータベース内にデータをキャッシュに保存しておく時間 (秒)、すなわちサーバーに新しい情報を次回リクエストするまでの時間を指定します。このフィールドを空白のままにすると、*ttl* には、Start-Of-Authority (SOA) リソースレコードで指定された最小時間がデフォルトとして用いられます。

ttl の設定値があまりにも小さいと、サーバーはデータ更新のためのリクエストを頻繁に繰り返します。逆に、*ttl* の設定値があまりにも大きいと、情報の変更がタイムリーに反映されなくなります。

ほとんどの場合、*ttl* の値は、初期値として 1 日 (86400) から 1 週間 (604800) の間に設定するとよいでしょう。実際の情報の変更の頻度にあわせて、*ttl* の値を適切な値に変更してください。また、ほとんど変化することがないデータと関連しているということで *ttl* の値を大きく設定していた場合、そのデータが変更されるとわかった時点で、*ttl* の値を、データの変更があるときまで小さな値 (3600 から 86400) にし、その後またもとの大きな値に戻すこともできます。

同じ名前、クラス、タイプを持つすべての RR では、*ttl* は同じ値に設定してください。

class フィールド

3 番目のフィールドは、レコードのクラスです。現在のところ、1 つのクラスだけがあります。それは、TCP/IP プロトコルのファミリーであることを示す `IN` です。

record-type フィールド

4 番目のフィールドでは、リソースレコードのタイプを記述します。RR にはたくさんのタイプがあります。最も一般的に使用されるタイプは、601ページの「リソースレコードのタイプ」で説明されています。

record-specific-data フィールド

record-specific-data フィールドの中身は、特定のリソースレコードのタイプで異なります。

ネームフィールドやデータフィールドの大文字と小文字の区別は、ネームサーバーに読み込まれたときには保存されていますが、ネームサーバーのデータベースを比

較して検索する際には大文字と小文字の区別はしません。しかし、これは将来的には変更される可能性がありますので、大文字と小文字の使用に関しては一貫性を保つように心がけてください。

特殊なリソースレコード文字

次に挙げる文字には特別な意味があります。

表 28–4 特別な意味を持つリソースレコード文字

文字	定義
.	ネームフィールドで、1 つのドットだけが指定された場合は現在のドメインを指す
@	ネームフィールドで、1 つの @だけが指定された場合は現在の起点を示す
. .	2 つのドットがネームフィールドに指定された場合は null ドメイン名を表す
X	*X* は数字以外 (0-9) の任意の文字で、\ を付けることによって文字に特別な意味を持たせないようにする。たとえば、\. を指定して、ラベル内にドット文字おくことができる
DDD	各 *D* は一桁の数字で、\ を付けることによって *DDD* で表される 10 進数に対応する 8 進数を表現する。結果的に得られる 8 進数は、テキストとみなされ、そのテキストに特別な意味があるかどうかはチェックされない
( )	データが 1 行に収まらないようなとき、データをグループ化するのにカッコを使用する。結果的に、カッコの間では行の終わりが認識されない
;	セミコロンでコメントが開始する。その行でセミコロン以降は無視される
*	アスタリスクはワイルドカードを表す

ほとんどのリソースレコードには、現在の起点があり、名前の最後にドット (.) が付いていなければ、現在の起点が名前に追加されます。この機能は、マシン名などのデータに現在のドメイン名を追加する際には便利ですが、追加したくない場合に

は、問題を引き起こすかもしれません。データファイルを作成しているドメイン内に名前がない場合、ピリオドで終わる完全指定名を使してください。

制御エントリ

データファイルで制御エントリの行だけは標準 RR 書式に従わない行です。制御エントリには、$INCLUDE() と $ORIGIN()の 2 つのタイプがあります。

$INCLUDE

インクルード行は 1 列目から $INCLUDE で始まり、後にファイル名 ($INCLUDE ファイル) が続きます。次の例に示すように、この機能は異なるタイプのデータを複数のファイルに分けるのに特に便利です。

```
$INCLUDE /etc/named/data/mailboxes
```

この行は、/etc/named/data/mailboxes ファイルを読み込むリクエストとして解釈されます。$INCLUDE コマンドでは、異なるゾーンまたはツリーにデータは読み込まれません。このコマンドを使用しても、あるゾーンのデータを別々のファイルに入れるだけです。たとえば、mailbox のデータはこの機能を使ってホストデータとは別に保存されます。

$INCLUDE の文とファイルは必要に応じて必要な数だけ使用できます。

$ORIGIN()

$ORIGIN コマンドによって、データファイル内の起点を変更できます。この行は 1 列目から始まり、ドメイン名が続きます。これによって、相対ドメイン名 (たとえば、完全指定されていないドメイン名) の現在の起点を指定の名前に変更します。これは、1 つのデータファイルに複数のドメインを入れるのに便利です。

注 - 1 つのデータファイルに複数のゾーンを入れるために $ORIGIN() を使うことはできません。

データファイルでの $ORIGIN コマンドは、必要に応じて使用します。もし、$ORIGIN() 文がない場合、DNS データファイルのデフォルトの起点は named.conf ファイルの primary または secondary の行の 2 番目のフィールドにあるドメイン名です。

リソースレコードのタイプ

最も一般的に使用されるリソースレコードのタイプを 表 28–5 に列挙します。通常、表 28–5 に並んだ順で入力しますが、必須というわけではありません。

表 28–5 一般的に利用されるリソースレコードのタイプ

タイプ	説明
SOA	権限の開始
NS	ネームサーバー
A	インターネットアドレス (名前からアドレス)
PTR	ポインタ (アドレスから名前)
CNAME	正規名 (ニックネーム)
TXT	テキスト情報
WKS	既知サービス
HINFO	ホスト情報
MX	メール交換

SOA - 権限の開始

例 28–2 に SOA リソースレコードの構文を示します。

例 28–2 SOA レコードの書式

```
name  class  SOA  origin  person-in-charge  (  serial  number
   refresh
  retry
  expire
  ttl)
```

権限の開始 (SOA) レコードでゾーンの開始を指定します。次の SOA レコードでそのゾーンは終了します。次に SOA レコードのフィールドについて述べます。

name

ゾーン名を指定します。ゾーン名の後にはドットを付ける必要があります。たとえば、doc.com. は正しいですが、一方、doc.com は誤りです。

class

アドレスのクラスです。たとえば、IN は、インターネットを示し、最も一般的に用いられるクラスです。

SOA

リソースレコードのタイプを示します。

origin

データファイルがあるホスト名のフィールドです。ホスト名の後にはドットを付ける必要があります。たとえば、dnsmaster.doc.com. は正しいですが、dnsmaster.doc.com は誤りです。

person-in-charge

ネームサーバーの責任者のメールアドレスのフィールドです。たとえば、kjd.nismaster.doc.com. です。この名前も終わりにドットを付ける必要があります。

serial

データファイルのバージョン番号のフィールドです。データを変更するたびにこの番号を増やしてください。副サーバーは serial フィールドを使って、最後にマスターサーバーからデータファイルをコピーしたときから変更があったかどうかを検出します。

refresh

更新が必要かどうかを見るために副ネームサーバーがどのくらいの頻度で主ネームサーバーをチェックすべきかを秒で指定します。たとえば、7200 は、2 時間を意味します。

retry

リフレッシュのためのチェックに失敗した後、副サーバーがどのくらいの時間再試行するかを秒で指定します。

expire

リフレッシュが頻繁に行われない場合に、データの期限が切れる前に、副ネームサーバーがそのデータを使用する上限の時間を秒で指定します。

ttl

リソースレコードの time-to-live フィールドで使用されるデフォルトの秒数を指定します。このデフォルト値はリソースレコードで *ttl* が指定されていないときに適用されます。

SOA は各ゾーンに 1 つだけ指定してください。例 28–3 に SOA リソースレコードの例を示します。

例 **28–3** SOA リソースレコードの例

```
;name class   SOA  origin     person-in-charge
doc.com. IN  SOA dnsmaster.doc.com. root.nismaster.doc.com. (
       101   ;Serial
       7200  ;Refresh
       3600  ;Retry
       432000  ;Expire
       86400)  ;Minimum    )
```

NS - ネームサーバー

例 28–4 に、ネームサーバー (NS) リソースレコードの構文を示します。

例 **28–4** NS レコードフォーマット

domainname [optional *TTL*] *class* NS *name-server-name*

ネームサーバーレコードには、対象としているドメインを受け持つサーバーの名前を指定します。*name* フィールドには、指定したネームサーバーからサービスを受けるドメインを指定します。もし、*name* フィールドが指定されない場合、そのデフォルトは最後に指定された名前になります。そのドメインの主マスターサーバーと副

マスターサーバーのそれぞれに 1 つの NS レコードが必要です。例 28–5 に NS リソースレコードの例を示します。

例 **28–5** NS リソースレコードの例

```
;domainname     [TTL]   class NS nameserver
doc.com         90000    IN NS  sirius.doc.com.
```

A アドレス

例 28–6 に、アドレス (A) リソースレコードの構文を示します。

例 **28–6** アドレスレコードの書式

```
machinename [optional TTL] class A address
```

アドレス (A) レコードでは、対象としているマシンのアドレスを指定します。*name* フィールドは、ホスト名のフィールドです。*address* は、IP アドレスです。各マシンのアドレスに 1 つの A レコードが必要です。ルーターやゲートウェイには 2 つ以上のエントリが必要で、IP アドレスを含む別々のエントリがネットワークインタフェースに必要です。

例 **28–7** アドレスレコードの例

```
;machinename [TTL] class A address
sirius  IN  A 123.45.6.1
```

HINFO - ホスト情報

例 28–8 に、ホスト情報 (HINFO) リソースレコードの構文を示します。

例 **28–8** HINFO レコードの書式

```
[optional name] [optional TTL]  class HINFO hardware OS
```

ホスト情報 (HINFO) リソースレコードでは、ホスト固有のデータを指定します。ハードウェアとこのホストで動作しているオペレーティング環境を指定します。マシン名や `hardware` フィールドのエントリに空白を含めるには、エントリを引用符で囲む必要があります。`name` フィールドでは、ホスト名を指定します。名前が指定されなければ、`in.named` での最後のホストがデフォルトになります。各ホストに1

つの HINFO レコードが必要です。例 28–9 に HINFO リソースレコードの例を示します。

例 **28–9**　HINFO リソースレコードの例

```
;[name]    [TTL] class HINFO   hardware    OS
                 IN    HINFO   Sparc-10    UNIX
```

注意 - HINFO フィールドにはネットワーク上のマシンについての情報があるので、多くのサイトで、この情報はセキュリティ上危険であると考えられ、現在はほとんど使われていません。

WKS - 既知サービス

例 28–10 に、既知サービス (WKS) リソースレコードの構文を示します。

例 **28–10**　WKS レコードの書式

```
[Optional name] [TTL] class WKS address protocol-list-of-services
```

既知サービス (WKS) レコードには、指定されたアドレスの特定のプロトコルでサポートされるよく知られたサービスが記述されます。サービスのリストとポート番号はサービスデータベースで指定されたサービスのリストから得られます。各プロトコルの各アドレスに 1 つの WKS レコードが必要です。例 28–11 に WKS リソースレコードの例を示します。

例 **28–11**　WKS リソースレコードの例

```
;[name] [TTL] class WKS address   protocol-list-of-services
altair IN WKS 123.45.6.1  TCP ( smtp discard rpc
                                                sftp uucp-
path systat daytime
        netstat qotd nntp doc.com )
```

注意 - WKS レコードはオプションです。セキュリティ上の理由から、ほとんどのサイトでは現在この情報は提供していません。

CNAME - 正規名

例 28–12 に、正規名 (CNAME) リソースレコードの構文を示します。

例 **28–12** CNAME レコードの書式

```
nickname [optional TTL] class CNAME canonical-name
```

正規名 (CNAME) リソースレコードでは、正規名に対するニックネームまたは別名を指定します。ニックネームは一意である必要があります。他のすべてのリソースレコードは、ニックネームではなく正規名と結びつけるようにする必要があります。ニックネームの作成と他のリソースレコードでの使用はしないでください。ニックネームは、マシン名が変更されたけれども古い名前でのアクセスを許可する移行期に特に有用です。また、ニックネームはメールサーバーなど特定の目的で使用するマシンを識別するためにも使用できます。例 28–13 に CNAME リソースレコードの例を示します。

例 **28–13** CNAME リソースレコードの例

```
;nickname      [TTL] class CNAME canonical-name
mailhost  IN CNAME antares.doc.com
```

PTR - ポインタレコード

例 28–14 に、ポインター (PTR) リソースレコードの構文を示します。

例 **28–14** PTR レコードの書式

```
special-name [optional TTL]   class   PTR-real-name
```

特別名でドメイン内の他の場所を指すことが、ポインタレコードによって可能になります。例では、PTR は、アドレス (特別名) の実名への変換のために、主に `in-addr.arpa.` レコードで使用されます。アドレスを解釈するときは、ドメインが完全指定であれば、指定する必要があるのはマシンの識別番号だけです。PTR の名前付けは、ゾーンに対して一意である必要があります。PTR レコードによって特別なドメインである `in-addr.arpa.`ドメインに対する逆ポインタが設定 (例 28–15) されます。

例 **28–15**　PTR リソースレコードのサンプル

```
;special name   [TTL] class PTR-real-name
1   IN PTR sirius.doc.com.
```

MX - メール交換

例 28–16 に、メール交換 (MX) リソースレコードの構文を示します。

例 **28–16**　MX リソースレコードの書式

```
name [optional TTL] class MX preference-value mailer-exchanger
```

メール交換リソースレコードは、あるドメインまたはドメイン内の特定のマシンにメールを配信するマシンを指定するために使用します。対象としている名前に対して複数の MX リソースレコードがあるかもしれません。例 28–17 では、Seismo.CSS.GOV (完全指定のドメイン名) は、Munnari.OZ.AU にメールを配信するメールゲートウェイです。ネットワーク上の他のマシンは、Munnari に直接メールを配信できません。Seismo と Munnari は、専用接続を持っている場合も、異なるトランスポートメディアを使用している場合もあります。優先値フィールドで、メールプログラムが従うべき順を指定します。単一のマシンに複数の方法でメールが配信される場合に指定します。値が 0 (ゼロ) は最優先であることを意味します。同じ名前に対して複数の MX リソースレコードがある場合、そのレコードの優先値は同じであることも、同じでないこともあります。

メールをルーティングするために、MX レコードでワイルドカードであるアスタリスク (*) を名前に使うこともできます。あるドメイン宛のメールはリレー経由で配信されることを単に示すサーバーがネットワーク上にはよくあります。例 28–17 では、foo.com ドメイン内のホスト宛のすべてのメールは RELAY.CS.NET. を経由して送られます。これを指定するには、ワイルドカードを用いて MX リソースレコードを作成し、*.foo.com のメール交換は RELAY.CS.NET. により行われると指定します。アスタリスクは foo.com の任意のホストまたはサブドメインに一致します。しかし、foo.com 自体には一致しません。

例 **28–17**　MX リソースレコードの例

```
;name   [TTL] class  MX  preference mailer-exchanger
Munnari.OZ.AU. IN  MX  0 Seismo.CSS.GOV.
foo.com.  IN  MX  10 RELAY.CS.NET.
*.foo.com. IN  MX  20 RELAY.CS.NET.
```

Solaris DNS BIND の実装

Solaris リリース 8 では、コンパイル版の Berkeley Internet Name Domain (BIND) 8.1.2 を提供します。Solaris 8 リリースでは、コンパイル版の BIND を提供しています。このソフトウェアのコンパイル時には、たくさんのサイトでの様々なニーズに対応するためにオプションの設定や選択が行われました。しかし、もしこのコンパイル版の BIND がニーズに合わない場合、公に入手可能なソースコードを用いて、再コンパイルし、自分用のオリジナルのバージョンの BIND を作成することもできます。

Solaris 7 以降のリリースで提供される BIND バージョンのコンパイルでは、以下の選択が行われました。

- 「RFC1535」

  実装によって暗黙の検索リストが削除されるため、現在は実装されていません。

- 「逆参照」

  これがないと SunOS 4.x の `nslookup` が機能しないため許可されています。

- 「Bogus ネームロギング」

  不要なメッセージが生成されるので bogus ネームサーバーのロギングは実装されていません。

- 「デフォルトのドメイン名」

  DNS ドメイン名が `/etc/resolv.conf` 内または `LOCALDOMAIN` 環境変数で設定されていない場合、`libresolv` は、デフォルトのドメイン名を NIS または NIS+ のドメイン名から導き出します。ただし、それは `/etc/nsswitch.conf` ファイルが `hosts` 行の最初の要素として `nisplus` または `nis` を含んでいる場合です。

- 「ユーティリティスクリプト」

BIND のユーティリティスクリプトは、今回の Solaris リリースには含まれていません。

- 「テストプログラム」

  BIND のテストプログラムである `dig`、`dnsquery`、`host` は、今回の Solaris リリースには含まれていません。これらのテストプログラムの目的は、`nslookup` と `nstest` の目的と同様です。

第 29 章

DNS の管理

この章では、DNS の管理方法を説明します。詳しくは、Cricket Liu、Paul Albitz 著『*DNS and Bind*』(O'Reilly, 1992) と『*Name Server Operations Guide for BIND*』, University of California, Berkeley をお読みください。

ドメイン名の終わりにつけるドット

DNS 関連のファイルで作業する際、ドメイン名の終わりにつけるドットに関しては以下に述べるルールに従ってください。

- `hosts`、`hosts.rev`、`named.ca`、`named.local` の各データファイル内のドメイン名には `sales.doc.com.`のように終わりにドットを付ける
- `named.boot` または `resolv.conf` ファイルのドメイン名には `sales.doc.com` のように終わりにドットを付けない

DNS データファイルの変更

マスター DNS サーバー内の DNS データファイルのひとつに対して、ホストの追加および削除、またはそれ以外の何らかの変更、あるいは DNS データファイルの修正を行なったときには、以下のことも行なってください。

- 副サーバーがそのデータを変更するように、SOA リソースレコードのシリアル番号を変更します (612ページの「SOA のシリアル番号の変更」を参照)。
- データファイルを再度読み込み、内部データベースを更新するためにマスターサーバーの `in.named` に情報を与えます (613ページの「in.named に DNS データを強制的に再読み込みさせる」を参照)。

SOA のシリアル番号の変更

すべての DNS データベースファイルには権限の開始 (SOA) リソースレコードがあります。DNS データベースのデータを変更したときはいつでも、SOA シリアル番号を 1 増加させる必要があります。

たとえば、データファイルの SOA のシリアル番号が現在 `101` で、ファイルのデータに変更を加えたなら、シリアル番号を `101` から `102` に変更する必要があります。SOA のシリアル番号を変更しないと、ドメインの副サーバーは、新しい情報でデータベースファイルのコピーを更新しません。その結果主サーバーと副サーバーは同期しなくなります。

一般的な SOA レコードの例である `hosts` ファイルは以下のとおりです。

```
; sample  hosts  file
@ IN       SOA  nismaster.doc.com. root.nismaster.doc.com. (
   109 ; Serial
   10800 ; Refresh
   1800 ; Retry
   3600000 ; Expire
   86400 ) ; Minimum
```

したがって、`hosts` ファイルを変更すると、シリアル番号は `109` から `110` に変更して、次にファイルを変更した場合、`110` から `111` に変更します。

in.named に DNS データを強制的に再読み込みさせる

in.named が無事起動すると、デーモンはそのプロセス ID を /etc/named.pid ファイルに書き込みます。in.named で named.boot を再び読み込み、データベースを再度読み込むには、以下のとおり入力します。

```
# kill -HUP `cat /etc/named.pid`
```

これを行うと以前のキャッシュはすべて削除され、キャッシュの処理が再スタートされます。

注意 - inetd から in.named を実行しないでください。これを行うとネームサーバーは、繰り返しリスタートし、そしてキャッシュを持つ意味がなくなります。

マシンの追加と削除

マシンを追加または削除するときはいつでも、主 DNS サーバーに格納されたデータファイルを変更してください。副サーバーのファイルは変更しないでください。SOA のシリアル番号の変化に基づいて副サーバーは主サーバーから自動的に更新されてしまいます。

マシンの追加

DNS ドメインにマシンを追加するには、新しいマシンを DNS クライアントとして設定して、新しいマシンのレコードを hosts と hosts.rev の各ファイルに追加します。

たとえば、rigel というホストを doc.com ドメインに追加する場合は次のように実行します。

1. /etc/resolv.conf ファイルを rigel 上に作成します。
2. rigel の /etc/nsswitch.conf ファイルの hosts の行に dns を追加します **(**59ページの「DNS とインターネットでのアクセス」を参照**)**。
3. 主サーバーの hosts ファイルに、rigel のためのアドレス **(**A**)** レコードを追加します。

例を以下に示します。

```
rigel  IN  A  123.45.6.112
```

4. 主サーバーの hosts ファイルに rigel のためのオプションのレコードを追加します。

   オプションのレコードには以下のものがあります。

   - エイリアス (CNAME)
   - メール交換 (MX)
   - よく知られたサービス (WKS)
   - ホスト情報 (HINFO)

5. hosts.rev ファイルに rigel のための PTR レコードを追加します。

6. 主サーバーの hosts ファイルと hosts.rev ファイルの **SOA** シリアル番号を増やします。

7. サーバーのデータを再ロードします。

   サーバーをリブートするか、または以下のとおり入力します。

```
# kill -HUP `cat /etc/named.pid`
```

   これらの手順は、『*Solaris* ネーミングの設定と構成』で詳しく説明されています。

マシンの削除

DNS ドメインからマシンを取り除く場合は次のように実行します。

1. 削除するマシンの nsswitch.conf ファイルの hosts の行から dns を削除します。

2. マシンの /etc/resolv.conf ファイルを削除します。

3. 主サーバーの hosts ファイルと hosts.rev ファイルからそのマシンのレコードを削除します。

4. **CNAME** レコードがそのマシンにあるなら、その **CNAME** のレコードも `hosts` ファイルから削除される必要があります。

5. 削除されるマシンによってサポートされていたサービスの代替を設定します。

   マシンが主サーバー、メールホスト、その他必要なプロセスまたはサービスのホストである場合、そのマシンが行なっていたサービスを他のマシンで実行できるように設定する必要があります。

DNS サーバーの追加

ネットワークに主サーバーや副サーバーを追加できます。DNS サーバーを追加するには、以下のとおりにしてください。

1. **DNS** クライアントとしてサーバーを設定します。

   詳細は、613ページの「マシンの追加」を参照してください。

2. サーバーのブートファイルを設定します。

3. サーバーの `named.ca` ファイルを設定します。

4. サーバーの `hosts` ファイルを設定します。

5. サーバーの `hosts.rev` ファイルを設定します。

6. サーバーの `named.local` ファイルを設定します。

7. サーバーを初期化します。

8. サーバーをテストします。

   以上の手順については、『*Solaris* ネーミングの設定と構成』に詳しい説明があります。

DNS サブドメインの作成

ネットワークは大きくなっていくので、ネットワークを複数の DNS サブドメインに分割すると便利です (DNS のドメインの階層と構造については、579ページの「DNS 名前空間について」を参照)。

ネットワークを親ドメインとひとつ以上のサブドメインに分割すると、負担が複数のドメインに分散して、各 DNS サーバーの負荷は減ります。この方法で、ネットワークのパフォーマンスを改善できます。たとえば、ネットワークに 900 台のマシンがあり、すべてがひとつのドメインにあるとします。この場合、1 台の主サーバー、1 台以上の副サーバーとキャッシュオンリーサーバーからなる DNS のサーバーの集合は 900 台のマシンをサポートしなければなりません。もし、このネットワークを、各ドメインが 300 台ずつの、1 つの親ドメインと 2 つのサブドメインに分割すると、主サーバーと副サーバーの組は 3 つとなり、それぞれは 300 台だけを担当することになります。

ネットワークを、物理的または組織的な構成に合うように、複数のドメインに分割することによって、DNS ドメイン名は、どこに対象とするマシンがあるか、電子メールのアドレスが組織のどこにあたるのかを指し示すことになります。たとえば、`rigel@alameda.doc.com` は、マシン `rigel` は Alameda というサイトにあることを意味し、電子メールのアドレス `barnum@sales.doc.com` は、ユーザーの `barnum` が Sales の部署の者であることを意味します。

ネットワークを複数のドメインに分けると、すべてをひとつのドメインに置く場合よりも設定作業が増えます。また、ドメインを互いにつなぐ委託データを保持しなければなりません。一方で、複数のドメインを持つと、各ドメインの保守をそれぞれの管理者またはチームに分散させることができます。

サブドメインの設計

ネットワークをひとつの親ドメインと 1 つ以上のサブドメインに分割する前に考慮しておくべきポイントをここで述べます。

- 「サブドメインの数」

  作成するサブドメインが多ければ、それだけ初期設定の作業が増え、運用時の親ドメインでの管理者の調整作業も増えます。多くのサブドメインがあれば、親ドメインのサーバーのために委任される作業もその分増加します。一方で、ドメインの数が少ないということは各ドメインが大きいということを意味し、ドメイン

が大きければ、それをサポートするためにサーバーのスピードやメモリーが余計必要だということです。

- 「ネットワークの分割方法」

  ネットワークはどのようにでも複数のドメインに分割できます。最も一般的な方法が 3 つあります。まず、組織の構成によるもので、各部署 (営業、研究開発、製造、など) に別々のサブドメインを持つ方法です。次に、地理的なもので、各サイトに別々のサブドメインを持つ方法です。最後に、ネットワークの構造によるもので、大きなネットワーク構成要素ごとに別々のドメインを持つ方法です。ドメインの構造が、統一されていて、論理的で、明確であれば、管理も利用もより簡単になります。

- 「将来に対する考慮」

  もっとも問題が多いのは、時間の経過とともに、新しいサイトや部が増えて、サブドメインが無計画に追加されて大きくなったドメインです。可能な限り、将来のドメインの拡大を見越してドメインの階層を設計してください。安定性も考慮に入れてください。最も安定しているものを基礎として、サブドメインが分けられているのがベストです。たとえば、地理的なサイトは比較的安定しているけれども、部や課が頻繁に再編成されるなら、サブドメインは組織よりも地理的な位置に基づいて決定すべきです。一方、組織は比較的安定しているが、サイトが頻繁に追加されたり、変更されたりする場合は、サブドメインは組織の階層に基づいて決定してください。

- 「広域ネットワーク (WAN) とのリンク」

  ネットワークがモデムまたは専用回線で接続された複数のサイトに広がっている場合、広域ネットワーク (WAN) のリンクにドメインが広がっていないなら、パフォーマンスと信頼性は高くなります。WAN のリンクは連続的なネットワーク接続より遅く、失敗に終わる傾向があります。サーバーが、WAN のリンク経由だけで接続できるマシンをサポートしなければならないときは、より遅いリンクを経由するトラフィックファネリングが増加することになります。この場合、もしひとつのサイトで停電や何か他の問題が起こった場合には、相手のサイトのマシンにも影響を及ぼしてしまうかもしれません。(同様のパフォーマンスと信頼性の配慮は DNS ゾーンについても必要です。経験的な知恵として、ゾーンが WAN のリンクに広がらなければベストです。)

- 「NIS+ ネームサービス」

  もし、エンタープライズレベルのネームサービスが NIS+ である場合、DNS と NIS+ のドメイン、サブドメインの構造が一致するなら管理はより容易です。

- 「サブドメイン名」

可能な限り、サブドメインに名前を付けるのに一貫したポリシーを確立して、それを守るのが理想的です。ドメイン名が一貫していれば、ユーザーは簡単に記憶し、正しく指定できます。ドメイン名はすべての DNS データファイルで重要な要素であること、サブドメインを変更すると古いドメイン名があるすべてのファイルを修正する必要があることに注意してください。したがって、安定していて変更の必要がないと思われるサブドメイン名を選択するようにしてください。サブドメイン名として `manufacturing` のように完全指定することもできますし、`manf` のように短縮して指定することもできます。しかし、あるサブドメインが短縮形で指定され、他が完全指定された場合、利用者は混乱します。もし、短縮形を用いるのであれば、名前を識別するのに十分な文字数にしてください。短かすぎる名前は利用や記憶が困難だからです。最上位のインターネットのドメイン名として使用されている名前をサブドメイン名として使わないで下さい。すなわち、`org`、`net`、`com`、`gov`、`edu` と、2 文字の国のコード `jp`、`uk`、`ca`、`it` 等は、すでに使用されているのでサブドメイン名として使うことはできません。

サブドメインの設定

ほとんどの場合、新しいサブドメインは、新しいネットワークやマシンをつなぐか、既にあるドメインを分ける場合に作成されます。どちらの場合も、プロセスはよく似ています。

新しいサブドメインを計画したなら、以下に述べる手順に従って設定してください。

1. **新しいサブドメイン内のすべてのマシンが DNS クライアントとして正しく設定されていることを確認してください。**

   既存のドメインから新しいサブドメインを分けている場合には、ほとんどのマシンは DNS クライアントの設定が既にされているはずです。新しいサブドメインを最初から構築している、あるいは既存のネットワークに新しいマシンを追加しているのであれば、正しく設定された `resolv.conf` ファイルと `nsswitch.conf` ファイルを各マシンにインストールする必要があります。これらのファイルについては、『*Solaris* ネーミングの設定と構成』を参照してください。

2. **正しく設定されたブートファイルと DNS データファイルをサブドメインの主マスターサーバーにインストールします。**

   各サーバーに以下のファイルをインストールします (詳細は、『*Solaris* ネーミングの設定と構成』を参照)。

   - `/etc/named.boot`.

- /var/named/named.ca.
- /var/named/hosts.
- /var/named/hosts.rev.
- /var/named/named.local.

サーバーのホストファイルには、サブドメイン内の各マシン対するアドレス (A) レコードと場合によっては CNAME レコードが必要で、サーバーの hosts.rev ファイルはサブドメイン内の各マシンに対するポインター (PTR) レコードを持つ必要があることに注意してください。オプションの HINFO と WKS レコードも追加できます。

3. 既存のドメインを分割しているなら、親ドメインのマスターサーバーの hosts ファイルと hosts.rev ファイルから新しいサブドメインのマシンのレコードを削除してください。

   そのためには、今は新しいサブドメイン内にあるマシンの A レコードを古いドメインサーバーのホストファイルから削除します。また、同じマシンの PTR レコードを古いドメインサーバーの hosts.rev ファイルから削除します。移動するマシンのオプションの HINFO レコードと WKS レコードも、削除する必要があります。

4. もし、既存のドメインを分割しているなら、新しいサブドメイン名を、親ドメインのマスターサーバーのホストファイル内の **CNAME** レコードに追加します。

   たとえば、aldebaran というマシンをファックスサーバーとして使っているとします。また親ドメインのサーバーのホストファイル内の CNAME レコードが次のとおりであるとします。

```
faxserver   IN   CNAME   aldebaran
```

   新しい faxserver の CNAME レコードを aldebaran に対して、新しいサブドメインのマスターサーバーにあるホストファイル内に作成するのに加えて、下記のように aldebaran のサブドメインが含まれるように、親ドメインのホストファイル内の CNAME レコードも変更する必要があります。

```
faxserver   IN   CNAME   aldebaran.manf.doc.com
```

5. 新しいサブドメインのサーバーの **NS** レコードを、親ドメインのホストファイルに追加します。

たとえば、親ドメインは doc.com で、manf.doc.com という新しいサブドメインを作成しているとします。また、この時 rigel というマシンを manf のマスターサーバーとして aldebaran を副マスターサーバーとして指定するとします。すると、以下に示すレコードを aldebaran の主マスターサーバーのホストファイルに追加することになります。

```
manf.doc.com 99999 IN NS rigel.manf.doc.com
             99999 IN NS aldebaran.manf.doc.com
```

6. 新しいサブドメインのサーバーの **A** レコードを、親ドメインのホストファイルに追加します。

   引き続き上の例で考えると、以下に示すレコードを doc.com の主マスターサーバーのホストファイルに追加することになります。

```
rigel.manf.doc.com       99999  IN  A  1.22.333.121
aldebaran.manf.doc.com   99999  IN  A  1.22.333.136
```

7. サブドメインのサーバー上の named を以下のように起動します。

```
# /usr/sbin/in.named
```

   コマンド行から in.named を実行する代わりに、リブートもできます。詳しくは、『*Solaris* ネーミングの設定と構成』を参照してください。

DNS エラーメッセージと問題解決

DNS の問題解決とエラーメッセージに関しては、付録 A と付録 B を参照してください。

パート VII　付録

この付録は参考資料を提供します。

- 付録 A
- 付録 B
- 付録 C
- 付録 D

付録 **A**

問題と解決方法

この章では、Solaris 8 リリースの名前空間を管理する際に発生する可能性のある問題のいくつかを説明します。

NIS+ の問題解決

この章は問題のタイプで分類されています。各問題には、一般的な現象、問題の説明、および対策が示されています。この章の他に、NIS+ のさらに一般的なエラーメッセージをアルファベット順に表示した付録があります。エラーメッセージが特定できていれば、最初に付録 B を参照してください。問題が簡単か、または 1 つのエラーメッセージに特定できる場合には、その対策は付録 B に掲載されています。

NIS+ のデバッグオプション

NIS_OPTIONS の環境変数を設定して、NIS+ デバッグオプションを制御できます。

オプションは、二重引用符で囲まれたオプションセットと共にスペースで区切られた NIS_OPTIONS コマンドの後に指定されます。各オプションに *name=value* のフォーマットがあります。値は、特定のオプションに従って整数、文字列、ファイル名になります。整数値のオプションに対して、値が指定されていない場合には、デフォルト値は 1 になります。

NIS_OPTIONS は次のオプションを認識します。

表 A–1 NIS_OPTIONS オプションと値

オプション	値	アクション
debug_file	*filename*	デバッグの指定ファイルへの出力を指示する。このオプションが指定されていない場合は、デバッグは stdout に出力する
debug_bind	*Number*	サーバーの選択プロセスに関する情報を表示する
debug_rpc	1 または 2	値が 1 の場合には、NIS+ に対する RPC コールを表示する。値が 2 の場合には、RPC コールと RPC の内容、引数、結果の両方を表示する
debug_calls	*Number*	NIS+ API へのコールと、アプリケーションに戻される結果を表示する
pref_srvr	*String*	nisprefadm コマンドによって生成されるのと同じフォーマットで、優先サーバーを指定する (表 15–1 を参照)。これは nis_cachemgr で指定された優先サーバーを上書きする

表 A–1 NIS_OPTIONS オプションと値 続く

オプション	値	アクション
server	*String*	特定のサーバーを設定する
pref_type	*String*	現在実装されていない

例を以下に示します。(C シェルの使用を前提)

- デバッグメッセージを表示するには、以下を入力します。

```
setenv NIS_OPTIONS ''debug_calls=2 debug_bind debug_rpc''
```

- API コールの簡単なリストを得て、それをファイル /tmp/CALLS に格納するには、以下を入力します。

```
setenv NIS_OPTIONS ''debug_calls debug_file=/tmp/CALLS''
```

- 特定のサーバーに送られた API コールの簡単なリストを得るには、以下を入力します。

```
setenv NIS_OPTIONS ''debug_calls server = sirius''
```

NIS+ の管理上の問題

この節では、日常的な NIS+ 名前空間の管理作業を行なっているときに発生する可能性のある問題について説明します。一般的な症状には次のようなものがあります。

- 「Illegal object type for operation」メッセージ
- その他の「オブジェクトの問題」のエラーメッセージ
- 初期設定のエラー
- チェックポイントのエラー
- ユーザーをグループに追加するときのエラー
- ログが大きすぎる場合、ディスク容量が不足した場合、ログの切り捨てのエラー
- groups_dir や org_dir が削除できない

無効なオブジェクトの問題

「症状」

- 「Illegal object type for operation」メッセージ
- その他の「オブジェクトの問題」のエラーメッセージ

このエラーメッセージには、いくつもの原因が考えられます。

- 検索可能な列がない状態で、テーブルを作成しようとしました。
- データベース処理が DB_BADOBJECT の状態を返しました (db(3N) のエラーコードの詳細は、nis_db(3N) のマニュアルページを参照してください)。
- 長さを 0 に指定して、データベースオブジェクトを追加または変更しようとしました。
- 所有者を指定せずに、オブジェクトを追加しようとしました。
- その処理はディレクトリオブジェクトを想定していますが、指定したオブジェクトは、ディレクトリオブジェクトではありませんでした。
- ディレクトリを LINK オブジェクトにリンクしようとしました。
- テーブルエントリにリンクしようとしました。
- グループオブジェクトではないオブジェクトを、nisgrpadm コマンドに渡しました。
- グループオブジェクトの処理が想定されていましたが、指定したオブジェクトのタイプはグループオブジェクトではありませんでした。
- テーブルオブジェクトの処理が想定されていましたが、指定したオブジェクトはテーブルオブジェクトではありませんでした。

nisinit のエラー

次の点をチェックしてください。

- NIS+ が動作していることを、ping によって確認できるか
- –H オプションで指定した NIS+ サーバーが、適切なサーバーであるか、また現在も動作しているか
- rpc.nisd がサーバー上で動作しているか
- その他クラスが、このドメインの読み取り権を持っているか
- このマシンで、ネットマスクが正しく設定されているか

チェックポイントのエラーが続く

チェックポイント処理 (たとえば、nisping –C コマンドによる操作) が、継続してエラーを起こしている場合は、十分なスワップ空間やディスク容量があるかどうか確認してください。syslog 内のエラーメッセージもチェックします。core ファイルがディスク空間を使い果たしていないかどうかチェックします。

ユーザーをグループに追加することができない

ユーザーをそのドメイン内のグループのメンバーとして追加する前に、そのユーザーは、ドメインの cred テーブルの中で LOCAL の資格を持つ NIS+ 主体クライアントになっていなければなりません。DES の資格を持っているだけでは、十分ではありません。

ログが大きくなりすぎた

nisping –C を使用して定期的にチェックポイントを実行していないと、ログファイルが極端に大きくなることがあります。マスターサーバーのすべての複製サーバーが更新されるまでは、マスター上にあるログはクリアされません。複製がダウンしている場合や、サービスが行われていない場合、複製サーバーと通信できない場合は、その複製サーバーに対応するマスターをクリアできません。このため複製がダウンしているか、または一定時間使用できない場合には、262ページの「ディレクトリを削除する」で説明するとおり、マスターから複製を削除する必要があります。ディレクトリ org_dir とサブディレクトリ groups_dir を最初に削除してから、ディレクトリ自体を削除してください。

ディスク容量の不足

十分なディスク容量が確保できない場合は、様々なエラーメッセージが表示されます (詳細は、656ページの「ディスク容量の不足」を参照)。

トランザクションログファイルを切り捨てることができない

まずはじめに、問題のファイルが存在するか、読み込み可能か、そのファイルに書き込み権が割り当てられているかどうかチェックしてください。

- トランザクションログは、ls /var/nis/trans.log で表示できます。

- ファイルの存在、アクセス権、読み取り権については、nisls –l と niscat で確認できます。
- 関係するメッセージは、syslog でチェックできます。

最も可能性のある原因は、適切なアクセス権を割り当てられているものの、ディスク容量が不足していることです。チェックポイント処理では、ログを切り捨て一時ファイルを削除する前に、ますログ一時ファイルのコピーを作成します。このため、一時ファイルを作成するためのディスク容量がない場合は、チェックポイント処理を進めることができません。使用可能なディスク容量をチェックし、必要であれば容量を確保してください。

ドメイン名の混同

NIS+ の多くのコマンドや操作にとって、ドメイン名は重要な役割を果たします。ルートサーバーを除いて、NIS+ のすべてのマスターと複製は、それ自身がサービスを提供するドメインより上にあるドメインのクライアントであるということに特に注意してください。サーバーまたは複製サーバーを、それ自身がサービスを提供するドメインのクライアントとして誤って扱った場合、
「Generic system error」や「Possible loop detected in namespace *directoryname*:*domainname*」というエラーメッセージが表示されます。

たとえば、altair というマシンが、subdoc.doc.com. ドメインのクライアントだとします。サブドメイン subdoc.doc.com. のマスターサーバーが sirius というマシンだとすると、sirius は doc.com. ドメインのクライアントになります。したがって、ドメインの指定や変更を行うときは、混同しないように次のルールに注意してください。

1. クライアントマシンは、特定のドメインかサブドメインに所属します。
2. 特定のサブドメインにサービスを提供するサーバーや複製サーバーは、そのドメインより上にあるサブドメインのクライアントです。
3. 2 の規則の唯一の例外は、ルートマスターサーバーとルート複製サーバーです。これらは、それ自身がサービスを提供するドメインのクライアントになります。つまり、ルートドメインのクライアントになるのは、ルートマスターとルート複製だけです。

したがって、上の例では、マシン altair の完全指定名
は、altair.subdoc.doc.com. です。マシン sirius の完全指定名
は、sirius.doc.com. です。sirius は doc.com. のクライアントであ

り、subdoc.doc.com. のクライアントではないの
で、sirius.subdoc.doc.com. は間違いであり、エラーになります。

org_dir や groups_dir を削除できない

親ディレクトリを削除する前に、必ず org_dir と groups_dir を削除してください。ドメインの groups_dir と org_dir を削除する前に、nisrmdir を使用してドメインを削除すると、これらの 2 つのサブディレクトリは、どちらも削除できなくなります。

複製の失敗からの NIS+ ディレクトリの削除または分離

複製サーバーからディレクトリを削除または分離する場合には、最初にディレクトリの org_dir と groups_dir のサブディレクトリを削除してから、ディレクトリ自体を削除します。各サブディレクトリが削除された後に、削除しようとするディレクトリの親ディレクトリで nisping を実行する必要があります (262ページの「ディレクトリを削除する」を参照)。

nisping の操作に失敗すると、ディレクトリは完全に削除または分離されません。

この状態が発生したら、次の手順を実行して、修正する必要があります。

1. 複製上の /var/nis/rep/org_dir を削除します。
2. 複製上の /var/nis/rep/serving_list で org_dir.*domain* が表示されないことを確認してください。
3. *domain* で nisping を実行します。
4. マスターサーバーから nisrmdir –f *replica_directory* を実行します。

分離しようとしている複製サーバーがダウンしているか、または通信不能である場合に nisrmdir –s コマンドは「Cannot remove replica *name* : attempt to remove a non-empty table」というエラーメッセージを戻します。

このような場合には、nisrmdir –f –s *replicaname* をマスターで実行して、分離を強制できます。しかし nisrmdir –f –s *replicaname* を使って通信不能な複製を分離する場合には、複製がオンライン状態に戻ったらすぐに、nisrmdir –f –s *replicaname* を再実行して、複製の /var/nis ファイルシステムをクリーンアップする必要があります。nisrmdir –f –s *replicaname* の再実行に失敗した場合には、複製がサービスを再開した時に複製上に残された古い情報によって問題が発生します。

NIS+ データベースの問題

この節では、名前空間のデータベースとテーブルに関連する問題を説明します。一般的な症状には次のようなものがあります。

- 処理内容に関係する次の表現が含まれているエラーメッセージが表示されます。
  - `Abort_transaction: Internal database error`
  - `Abort_transaction: Internal Error, log entry corrupt`
  - `Callback: - select failed`
  - `CALLBACK_SVC: bad argument`
- `rpc.nisd` が失敗します。

637ページの「NIS+ の所有権とアクセス権の問題」を参照してください。

rpc.nisd の複数の親プロセス

「症状」

次の表現が含まれている様々なエラーメッセージが表示されます。これらのメッセージは、データベースやトランザクションログが壊れていることを意味しています。

- `Log corrupted`
- `Log entry corrupt`
- `Corrupt database`
- `Database corrupted`

「考えられる原因」

複数の独立した `rpc.nisd` デーモンを実行させています。通常の動作では、`rpc.nisd` は他の `rpc.nisd` デーモンを子プロセスとして生成できます。このこと自体は問題はありません。しかし、1 台のマシン上で 2 つの `rpc.nisd` デーモンが親として動作している場合は、互いのデータを上書きし、ログとデータベースを壊してしまいます。このような現象が発生するのは、`rpc.nisd` が手作業で起動された場合です。

「診断」

`ps -ef | grep rpc.nisd` を実行します。親の `rpc.nisd` プロセスは、1 つしか動作していないことを確認します。

「対策」

「親」としての rpc.nisd のエントリが複数ある場合は、1 つを残して、他のデーモンの動作を終了させなければなりません。kill −9 *process-id* を実行し、もう一度 ps コマンドを実行して、他のデーモンが終了したかどうか確認します。

注 - −B オプションを指定して rpc.nisd を起動した場合は、rpc.nisd_resolv デーモンも終了させる必要があります。

NIS+ データベースが壊れている場合は、壊れていないデータベースを、最新のバックアップから復元します。次に、バックアップの時点よりあとで、名前空間に加えられた変更を反映します。しかし、ログも壊れている場合は、バックアップを行なったあとで名前空間に加えられた変更を、再度手作業で行わなければなりません。

rpc.nisd の失敗

NIS+ テーブルが大きすぎると、rpc.nisd は失敗します。

「診断」

nisls を使って、NIS+ テーブルの大きさをチェックします。7K バイト以上のテーブルでは、rpc.nisd は失敗します。

「対策」

NIS+ テーブルの大きさを小さくします。ネームサービスとして NIS+ は、オブジェクト自体ではなく、オブジェクトへのリファレンスを格納するために設計されています。

NIS+ と NIS の互換性の問題

この節では、NIS と NIS+ や以前のシステムとの間の互換性に関連する問題、またスイッチ構成ファイルに関連する問題を説明します。一般的な症状には次のようなものがあります。

- nsswitch.conf ファイルが正しく実行されない
- 処理内容に関係する次の表現が含まれているエラーメッセージが表示される
  - Unknown user
  - Permission denied
  - Invalid principal name

ユーザーがパスワードを変更したあと、ログインできない

「症状」

新しいユーザーや、最近パスワードを変更したユーザーが、ログインできません。または、特定のマシンからログインできますが、他のマシンからログインできません。そのようなユーザーに対して、次の表現が含まれているエラーメッセージが表示されることがあります。

- `Unknown user` *username*
- `Permission denied`
- `Invalid principal name`

「最初に考えられる原因」

NIS マシン上でパスワードが変更されました。

NIS+ の名前空間サーバーがサービスを提供しているドメインの中で、NIS を実行している Solaris 8 リリース のマシン上で、ユーザーまたはシステム管理者が `passwd` コマンドを使用してパスワードを変更した場合、そのユーザーのパスワードは、そのマシンの `/etc/passwd` ファイルの中だけで変更されています。そのユーザーが、ネットワーク上の他のマシンを使用してログインしても、そのマシン上では新しいパスワードは認識されません。NIS+ のパスワードテーブルに格納されている、古いパスワードを使用しなければなりません。

「診断」

そのユーザーの古いパスワードが、NIS+ の他のマシンでまだ有効かどうかチェックしてください。

「対策」

NIS+ を実行しているマシンで、`passwd` コマンドを使用して、ユーザーのパスワードを変更します。

「2 番目に考えられる原因」

パスワードの変更を行なっても、システム全体に反映されるまでに時間がかかります。

「診断」

名前空間の変更がドメインやシステム全体に反映されるまでに、ある程度の時間がかかります。ドメインのサイズや複製サーバーの台数により、数秒間で済むこともあれば、何十分もかかることがあります。

「対策」

変更結果がドメインに伝わるまで、常識的に受け入れられる程度の時間、待つだけです。または、`nisping org_dir` コマンドを使用して、システムを同期させることもできます。

nsswitch.conf ファイルが正しく実行されない

変更した (または、新しくインストールした) `nsswitch.conf` ファイルが正しく実行されません。

「症状」

新しい `nsswitch.conf` ファイルをインストールしたか、または既存のファイルを変更しましたが、システムがその変更結果を反映していません。

「考えられる原因」

`nsswitch.conf` ファイルのインストールや変更を行なった後は、必ずマシンを再起動して、変更結果を有効にしなければなりません。`nscd` が `nsswitch.conf` ファイルをキャッシュに書き込むためです。

「対策」

`nsswitch.conf(4)` のマニュアルページの情報を参照して、現在の `nsswitch.conf` ファイルをチェックしてください。必要に応じてファイルを修正し、マシンを再起動します。

NIS+ オブジェクトが見つからない問題

この節では、NIS+ がオブジェクトや主体を見つけることができない問題について説明します。一般的な症状には次のようなものがあります。

処理内容に関係する次の表現が含まれているエラーメッセージが表示されます。

- `Not found`
- `Not exist`
- `Can't find suitable transport for` *name*

- Cannot find
- Unable to find
- Unable to stat

構文やスペリングの誤り

NIS+ のオブジェクトが見つからない場合、最も可能性のある原因は、名前の入力を間違えたことです。構文をチェックし、正しい名前を使用しているかどうか確認してください。

正しくないパス名

「オブジェクト」が見つからない場合、考えられる原因として、正しくないパスを指定していることが挙げられます。また、NIS_PATH 環境変数が正しく設定されているかどうかも確認してください。

ドメインレベルが正しく指定されていない

すべてのサーバーは、それ自身がサービスを提供しているドメインではなく、その上にあるドメインのクライアントであることに注意してください。この規則には、2 つの例外があります。

- ルートマスターサーバーとルート複製サーバーは、ルートドメインのクライアントです。
- NIS+ のドメイン名は、ピリオドで終わります。完全指定名を使用する場合は、ドメイン名の終わりにピリオドをつけなければなりません。ドメイン名の終わりにピリオドをつけないと、NIS+ は、それが部分指定名であると想定します。この規則の例外としては、/etc/defaultdomain ファイルの中では、マシンのドメイン名の終わりにピリオドをつけません。/etc/defaultdomain ファイルの中で、マシンのドメイン名にピリオドをつけると、起動時に「Could not bind to server serving domain *name*」というエラーメッセージが表示され、ネットワークへの接続に問題が生じます。

オブジェクトが存在しない

NIS+ のオブジェクトが削除されたため、または作成されていないために、現在は存在していなくて、見つからないこともあります。nisls –l を使用して、そのドメインに目的のオブジェクトが存在するかどうかチェックしてください。

複製サーバーの同期遅延

NIS+ のオブジェクトの作成や変更を行うと、処理が完了して、特定の複製サーバーの情報が更新されるまでに、ある程度の遅れが発生します。通常の動作では、マスターかその複製から名前空間情報の照会が行われます。クライアントは、照会を複数のサーバー (マスターと複製) に自動的に分散し、システムの負荷のバランスを取ります。これは、どの時点でも、名前空間の情報をどのマシンが返してくるかわからないということを意味します。新しく作成されたオブジェクトや変更されたオブジェクトに関係するコマンドが、更新された情報をまだマスターから受け取っていない複製に送られた場合、「オブジェクトが見つかりません」というタイプのエラーか、同期していない古い情報を受け取ることになります。同様に、nisls のような、全体に関係するコマンドを使用して、まだ更新されていない複製サーバーに照会を行なった場合、新しく作成されたオブジェクトが含まれていないリストを受け取る可能性があります。

nisping を使用すると、同期していない状態にある複製サーバーを同期させることができます。

代わりに、NIS+ のコマンドで –M オプションを指定して、そのコマンドがドメインのマスターサーバーから名前空間の情報を受け取るように指定することも可能です。この方法を使用すると、確実に最新の情報を取得し、使用できます。ただし、–M オプションは、必要なときにだけ使用してください。複製サーバーを使用して名前空間のサービスを行う最大の理由は、負荷を分散して、ネットワークの効率を向上させることにあります。

ファイルが見つからないか壊れている

/var/nis/data ディレクトリの中の 1 つまたは複数のファイルが、壊れているか削除されています。最新のバックアップから、これらのファイルを復元してください。

旧バージョンの /var/nis について

Solaris 2.4 以前では、/var/nis ディレクトリに *hostname*.dict、*hostname*.log という 2 つのファイルが含まれていました。またサブディレクトリ /var/nis/ *hostname* もありました。Solaris 2.5 においては、2 つのディレクトリ名は trans.log、data.dict、サブディレクトリ名は /var/nis/data となります。

/var/nis、/var/nis/data といったディレクトリや、その中のファイルは、nisinit などの NIS+ 設定プロシージャーによって作成されますが、名前の変更をしないようにしてください。

Solaris 2.5 ではこれらのファイルの内容も変更されており、Solaris 2.4 以前との互換性はなくなっています。したがって、これらのファイルやディレクトリを Solaris 2.4 での名前にしてしまうと、Solaris 2.4、2.5 双方の rpc.nisd で機能しなくなりますので名前の変更をしないようにしてください。

名前の中の空白

「症状」

ある時にはオブジェクトが存在し、別の時には存在しないことがあります。NIS+ や UNIX の特定のコマンドが NIS+ のオブジェクトが存在しない、または見つからないと報告しますが、別のコマンドは同じオブジェクトを見つけることができます。

「診断」

nisls を使用して、オブジェクト名を探します。オブジェクト名を注意深くチェックして、その名前が実際は空白で始まっていないかどうか確認してください。NIS+ のコマンド行を使用して NIS+ のオブジェクトを作成するときに、フラグの後に誤って空白文字を 2 回入力すると、NIS+ のコマンドの中には、2 番目の空白文字をオブジェクト名の一部と解釈するものがあります。

「対策」

NIS+ のオブジェクト名が空白で始まっている場合は、名前を変更して空白を取り除くか、いったん削除して初めから作成し直します。

オートマウンタを使用できない問題

「症状」

他のホスト上のディレクトリに移動できません。

「考えられる原因」

NIS+ 環境では、NIS+ の必要条件に合わせて、オートマウンタの名前を変更しなければなりません。/etc/auto* テーブル名の一部としてピリオドが使用されている場合、NIS+ はそれらのテーブルをアクセスすることができません。たとえば、NIS+ は auto.direct というファイルにアクセスすることができません。

「診断」

nisls と niscat を使用して、オートマウンタのテーブル名が正しく割り当てられているかどうか確認します。

「対策」

ピリオドを下線 (_) に変更します。たとえば、auto.direct を auto_direct という名前に変更します。これらのテーブルを参照している可能性のある他のマップも変更してください。

テーブルエントリ間でのリンクが機能しない

nisln コマンド (またはその他のコマンド) を使って、テーブルエントリ間でのリンクは作成できません。NIS+ コマンドはエントリレベルでのリンクは追跡しません。

NIS+ の所有権とアクセス権の問題

この節では、ユーザーの所有権とアクセス権に関連する問題を説明します。

処理内容に関係する次の表現が含まれているエラーメッセージが表示されます。

- Unable to stat name
- Unable to stat NIS+ directory name
- Security exception on LOCAL system
- Unable to make request
- Insufficient permission to . . .
- You name do not have secure RPC credentials

一般的に観察されるその他の現象

- ユーザーまたはスーパーユーザーが、名前空間に関係する作業を行うことができない

アクセス権がない

アクセス権に関連して最も頻繁に発生する問題は、最も単純な問題です。行おうとしている業務に必要なアクセス権が、割り当てられていません。対象としているオブジェクトを指定して `niscat –o` を使用し、どのアクセス権が割り当てられているか確認します。他のアクセス権も必要な場合は、ユーザー自身、オブジェクトの所有者、システム管理者のうちの誰かが、そのオブジェクトのアクセス権の変更を行うことができます。詳細は、第 10 章と、第 12 章を参照してください。

資格がない

ユーザーやマシンに適切な資格がない場合は、ほとんどの操作を行なったときにエラーが発生します。ホームドメインの `cred` テーブルを対象にして `nismatch` を使用し、正しい資格を割り当てられているかどうか確認します (資格に関連した問題の詳細は、646ページの「無効になった資格」を参照)。

サーバーがセキュリティレベル **0** で動作している

セキュリティレベル 0 で動作しているサーバーは、NIS+ の主体の資格の作成や管理を行いません。

セキュリティレベル 0 で動作しているサーバーで `nispasswd` を試みると、次のエラーメッセージが表示されます。「`You` *name*
`do not have secure RPC credentials in NIS+ domain` *domainname*」

セキュリティレベル 0 は、管理者が名前空間の初期設定やテストを行う際にだけ使用されます。一般のユーザーがアクセスするような環境で使用すべきではありません。

ユーザーのログインがマシン名と同じ

マシン名と同じものを、ユーザーのログイン ID とすることはできません。ユーザー名と同じ名前をマシンに割り当てる (またはその逆) と、最初の主体は、セキュリティに関係するアクセス権を必要とする動作を行うことができなくなります。2 番目の主体の鍵が、`cred` テーブルの中にある、最初の主体の鍵を上書きするからです。さらに、2 番目の主体は、最初の主体に割り当てられていたアクセス権を持つようになります。

たとえば、`saladin` というログイン名を持つユーザーが、名前空間の中で読み込み専用のアクセス権を割り当てられていたとします。次に、`saladin` という名前を

持つマシンをドメインに追加します。ユーザー saladin は、何らかの種類のアクセス権を必要とする名前空間の操作を行うことができなくなります。そして、マシン saladin のスーパーユーザーは、名前空間の中で、読み込み専用のアクセス権だけを割り当てられます。

「症状」

- ユーザーやマシンが「permission denied」エラーメッセージを受け取ります。
- ユーザーまたはスーパーユーザーが、keylogin を正しく実行できなくなります。
- 「Security exception on LOCAL system. UNABLE TO MAKE REQUEST.」エラーメッセージが表示されます。
- 最初の主体が読み込みアクセスを割り当てられていない場合、2 番目の主体が、本来表示できるはずのオブジェクトを見ることができなくなります。

注 - nisclient や nisaddcred を実行したときに、「Adding Key」ではなく「Changing Key」というメッセージが表示された場合は、そのドメインの中で、ユーザー名またはホスト名がすでに重複しています。

「診断」

nismatch を実行して、hosts テーブルや passwd テーブル内のホストとユーザーを表示し、各テーブルの中に、同じホスト名やユーザー名が存在しないか確認します。以下に例を示します。

```
nismatch username passwd.org_dir
```

次に、ドメインの cred テーブルを対象にして nismatch を実行し、重複しているホスト名やユーザー名に、どのようなタイプの資格が割り当てられているかを調べます。LOCAL と DES 両方の資格が割り当てられている場合、cred テーブルのエントリはユーザーを表わしています。DES の資格だけが割り当てられている場合、エントリはマシンを表わしています。

「対策」

マシン名を変更します (ユーザー名を変更するより、マシン名を変更することをお勧めします)。次に、cred テーブルからそのマシンのエントリを削除し、nisclient を使用して、マシンを NIS+ のクライアントとして初期設定します (必要に応じて、nistbladm を使用し、そのマシンの別名を hosts テーブルの中に作成し、元のマシン名を別名として使用することもできます)。必要に応じて、cred テーブル内のユーザーの資格を変更します。

正しくない資格

646ページの「無効になった資格」を参照してください。

NIS+ のセキュリティの問題

この節では、パスワード、資格、暗号、その他セキュリティに関係した一般的な問題を取り上げます。

セキュリティ問題の症状

処理内容に関係する次の表現が含まれているエラーメッセージが表示されます。

- `Authentication error`
- `Authentication denied`
- `Cannot get public key`
- `Chkey failed`
- `Insufficient permission to`
- `Login incorrect`
- `Keyserv fails to encrypt`
- `No public key`
- `Permission denied`
- `Password [problems]`

ユーザーまたはスーパーユーザーは、名前空間に関係する作業を行うことができません (637ページの「NIS+ の所有権とアクセス権の問題」を参照)。

「Login Incorrect」というメッセージが表示された

原因としてもっとも多いのが、パスワードの誤入力です。もう一度入力するようユーザーに指示してください。「記憶しているパスワードが正しいか」、「パスワードでは大文字と小文字が区別されることを理解しているか」、「アルファベットの o と数字の 0、アルファベットの l と数字の 1 などを混同していないか」といったことについても確認してください。

「`login incorrect`」メッセージの原因としては他に以下のようなものが考えられます。

- パスワードが管理者によってロックされている (227ページの「パスワードのロック」、227ページの「パスワードロックの解除」を参照)。
- 指定の日数以上ログインが行われなかったため、パスワードがロックされた (235ページの「ログインの間隔の最大値の指定」を参照)。
- パスワードが有効期限を過ぎた (233ページの「パスワード使用権の有効期限」を参照)。

パスワードについては、第 11 章を参照してください。

パスワードがロック状態、期限切れ、または無効である

「`Permission denied, password expired`」といったタイプのメッセージは、「ユーザーのパスワードが有効期限を過ぎた」、「またはパスワード使用権が無効になった」といった理由で表示されることが最も多くなっています。詳細は第 11 章を参照してください。

- 229ページの「パスワードの有効期間の設定」
- 233ページの「パスワード使用権の有効期限」

資格に関する情報が古くなっている

資格、アクセス権ともに正しいにもかかわらず、クライアントの要求が拒否されるという場合もあります。これはおそらく、名前空間のどこかに古い情報が存在することが原因です。

資格情報の保存と更新

公開鍵など、資格に関する情報は、名前空間内の様々な場所に保存されています。NIS+ は、この情報を保存先のオブジェクトの生存期間の値にもとづいて定期的に更新しますが、更新と更新との間にときおり情報のずれが起こります。その結果、一部の操作が行えなくなります。表 A–2 は、資格に関する情報を保存するオブジェクト、テーブル、ファイルと、そのリセットの方法を示したものです。

表 A–2 資格に関する情報の保存場所

項目	保存対象	リセットおよび更新の方法
cred テーブル	NIS+ 主体の非公開鍵と公開鍵。これらの鍵のマスターコピーとなる	nisaddcred を使用して新しい資格を作成する。これによって既存の資格が更新される。chkey を使用しても同様のことが行える
ディレクトリオブジェクト	個々のサーバーの公開鍵のコピー	ディレクトリオブジェクトに対して /usr/lib/nis/ nisupdkeys コマンドを実行する
キーサーバー	その時点でログインされている NIS+ 主体の非公開鍵	主体ユーザーに対して keylogin を実行する。または主体ワークステーションに対して keylogin –r を実行する
NIS+ デーモン	ディレクトリオブジェクトのコピー (そのサーバーの公開鍵のコピーが含まれる)	デーモンおよびキャッシュマネージャを終了した後、再起動する
ディレクトリキャッシュ	ディレクトリオブジェクトのコピー (そのサーバーの公開鍵のコピーが含まれる)	NIS+ キャッシュマネージャを終了した後、nis_cachemgr –i コマンドを使用して再起動する。–i オプションを指定すると、コールドスタートファイルによってディレクトリキャッシュがリセットされた後、キャッシュマネージャが再起動される
コールドスタートファイル	ディレクトリオブジェクトのコピー (そのサーバーの公開鍵のコピーが含まれる)	ルートマスターで NIS+ デーモンを終了した後、再起動する。デーモンは、新しい情報を既存の NIS_COLD_START ファイルに再読み込みする。 まず、主体の /var/nis からコールドスタートファイルと共有ディレクトリファイルを削除し、キャッシュマネージャを終了する。次に nisinit –c を使用して主体を再初期化する。主体に対して「trusted」と指定したサーバーは、新しい情報を主体の既存のコールドスタートファイルに再読み込みする
passwd テーブル	ユーザーのパスワードまたはワークステーションのスーパーユーザーのパスワード	passwd -r nisplus コマンドを使用する。これによって、NIS+ passwd テーブル、cred テーブルの中でパスワードが更新される

表 A–2 資格に関する情報の保存場所 続く

項目	保存対象	リセットおよび更新の方法
passwd ファイル	ユーザーのパスワードまたはワークステーションのスーパーユーザーのパスワード	passwd -r nisplus コマンドを使用する。スーパーユーザー、一般ユーザーのどちらでログインしてもよい
passwd マップ (NIS)	ユーザーのパスワードまたはワークステーションのスーパーユーザーのパスワード	passwd -r nisplus を使用する

古くなったキャッシュに保存された鍵の更新

情報が古くなる原因として最も多いのが、サーバーの公開鍵のバージョンが古くなることです。一般に、この問題を解決するには、アクセスするドメインに対して nisupdkeys を実行します (nisupdkeys コマンドの使用方法の詳細は、第 7 章を参照)。

鍵の中にはファイルやキャッシュに保存されているものもあるため、nisupdkeys ですべての問題を解決できるわけではありません。鍵を手作業で更新しなければならない場合もあります。この場合は、「サーバーの公開鍵の内容は、公開鍵が作成された後どのように名前空間オブジェクトに伝えられるのか」ということについて理解する必要があります。サーバーの公開鍵の伝播には、一般に以下の 5 つの段階があります。それぞれの詳細について説明します。

1：サーバーの公開鍵が作成 される

NIS+ サーバーも初めは NIS+ クライアントなので、公開鍵の作成は他の NIS+ クライアントの公開鍵と同じ方法 (nisaddcred コマンドを使用する) で行います。公開鍵はその後、サーバーが実際にサポートするドメインではなく、サーバーのホームドメインの cred テーブルに保存されます。

2：公開鍵の内容がディレクトリオブジェクトに伝えられる

NIS+ ドメインと NIS+ サーバーの設定後は、サーバーとドメインを関係づけることができます。この「関係づけ」は、nismkdir コマンドで行います。nismkdir コマンドによってサーバーとディレクトリの関係づけが行われる際、サーバーの公開鍵も cred テーブルからドメインのディレクトリオブジェクトにコピーされます。たとえば、サーバーが doc.com. ルートドメインのクライアント

で、sales.doc.com. ドメインのマスターサーバーになっているという場合を考えてみましょう。

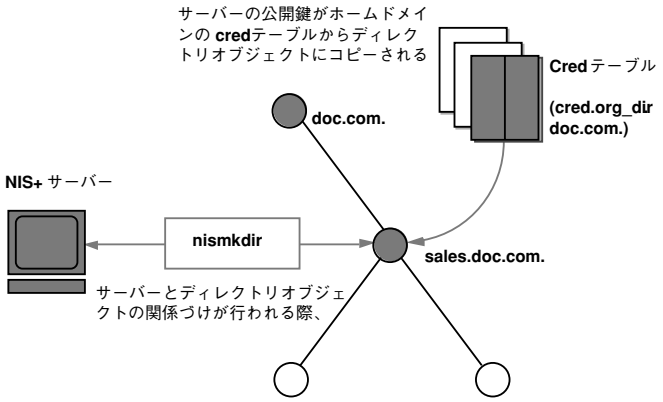


図 A–1　ディレクトリオブジェクトへの公開鍵の伝播

公開鍵は cred.org_dir.doc.com. ドメインから、ディレクトリオブジェクト sales.doc.com. にコピーされます。以上のことは、niscat -o sales.doc.com. というコマンドを使用して確認できます。

3：ディレクトリオブジェクトの内容がクライアントファイルに伝えられる

nisinit ユーティリティまたは nisclient スクリプトを使用すれば、すべての NIS+ クライアントを初期化できます。

他の類似のコマンドと同様、nisinit (または nisclient) では、コールドスタートファイル /var/nis/NIS_COLDSTART が作成されます。コールドスタートファイルは、クライアントのディレクトリキャッシュ /var/nis/NIS_SHARED_DIRCACHE の初期化に使用されます。コールドスタートファイルには、クライアントのドメイン中のディレクトリオブジェクトのコピーが含まれています。ディレクトリオブジェクトには、すでにサーバーの公開鍵のコピーが含まれているため、これで公開鍵の内容はクライアントのコールドスタートファイルに伝えられたことになります。

また、クライアントがホームドメインの外のサーバーに対して要求をした場合、リモートドメインのディレクトリオブジェクトのコピーが、クライアントの NIS_SHARED_DIRCACHE ファイルに保存されます。クライアントのキャッシュの内容は、nisshowcache コマンドを使用して調べることができます。

複製サーバーがドメインに追加されるか、サーバーの鍵が更新されるまでは、鍵の伝播はこの段階にとどまります。

4：複製サーバーがドメインに追加された場合の処理

複製サーバーがドメインに追加されると、nisping コマンドによって NIS+ テーブル (cred テーブルを含む) が新しい複製サーバーにダウンロードされます。これによって、元のサーバーの公開鍵も複製サーバーの cred テーブルに保存されます。

5：サーバーの公開鍵が更新 された場合の処理

サーバーの DES 資格 (サーバーのルート ID) を変更すると、公開鍵も変更されます。その結果、サーバー用に cred テーブルに保存される公開鍵が、以下の場所に保存されるものと矛盾します。

- 複製サーバーの cred テーブル (数分の間)
- サーバーがサポートするドメイン中の、メインディレクトリオブジェクト (生存期間終了まで)
- サーバーがサポートするドメイン中の、クライアントの NIS_COLDSTART ファイルおよび NIS_SHARED_DIRCACHE ファイル (生存期間終了まで、通常 12 時間)
- サーバーがサポートするドメインに要求をしたクライアントの、NIS_SHARED_DIRCACHE ファイル (生存期間終了まで)

サーバーの鍵は、ほとんどの場所において数分～ 12 時間で自動的に更新されます。すぐに更新するには、以下のコマンドを使用します。

表 A–3 サーバーの鍵の更新

保存場所	コマンド	参照ページ
複製サーバーの cred テーブル (nisping を使用しなくても、テーブルは数分で自動的に更新される)	nisping コマンド	271ページの「nisping コマンド」
サーバーがサポートするドメインのディレクトリオブジェクト	nisupdkeys コマンド	164ページの「nisupdkeys コマンド」
クライアントの NIS_COLDSTART ファイル	nisinit –c コマンド	266ページの「nisinit コマンド」
クライアントの NIS_SHARED_DIRCACHE ファイル	nis_cachemgr コマンド	268ページの「nis_cachemgr コマンド」

注 - nis_cachemgr の再起動は、既存の nis_cachemgr を終了してから行います。

無効になった資格

主体 (ユーザーかマシン) の資格が無効になっている場合は、その主体は nisls のようなコマンドも含め、名前空間の操作や処理を行うことができなくなってしまいます。資格が無効になると、未認証クラスに割り当てられるアクセス権も含め、すべてのアクセス権が失われるからです。

「症状」

ユーザーまたはスーパーユーザーが、名前空間に関係する作業を行うことができなくなります。名前空間のどのような操作を行なっても、「permission denied」というタイプのエラーメッセージが表示されます。ユーザーまたはスーパーユーザーは、nisls を実行することも不可能になります。

「考えられる原因」

鍵の破損、物理的な破損、古い資格、その他何らかの不適切な点
が、/etc/.rootkey ファイルの中にあります。

「診断」

snoop を使用して、不適切な資格を識別します。

または、その主体がリスト表示できる場合は、その主体としてログインを行い、主体が間違いなく承認されているはずのオブジェクトを対象として、NIS+ コマンドを実行します。たとえば、ほとんどの場合、未認証クラスにオブジェクトの読み込みは承認されているはずです。そこで、cred テーブルの中にリストされている主体は、nisls コマンドを正しく実行できるはずです。このコマンドを実行しても「permission denied」エラーが発生する場合は、おそらく、その主体の資格は無効になっています。

「対策」

- 「一般のユーザー」の場合、keylogout を実行してから、次に keylogin を実行して、その主体のログインを試みます。
- 「root」すなわち「スーパーユーザー」の場合、keylogout –f を実行してから、次に keylogin –r を実行します。

Keyserv のエラー

keyserv が、セッションを暗号化できません。このタイプの問題には、いくつかの原因が考えられます。

「考えられる原因と対策」

- クライアントが keylogin を実行していません。keylogin を実行したかどうか確認します。クライアントが正しく keylogin を実行したかどうかを確かめるには、(そのクライアントで) nisdefaults –v を実行します。Principal Name という行に「not authenticated」というメッセージが返れば、クライアントが正しく keylogin を実行していないことになります。
- クライアント (またはホスト) が、LOCAL や DES のような適切な資格を持っていません。クライアントの cred テーブルを対象にして niscat を実行し、クライアントが適切な資格を持っているかどうか確認します。必要に応じて、150 ページの「NIS+ 主体の資格情報を作成する方法」に従って資格を追加します。
- keyserv デーモンが動作していません。ps コマンドを使用して、keyserv が動作しているかどうか確認します。動作していない場合は、このデーモンを起動してから、keylogin を実行します。
- keyserv が動作している場合は、Secure RPC や NIS+ のコールを行う終始動作しているはずのプロセスが、まだ動作していません。たとえば、automountd、rpc.nisd、sendmail などが動作していません。これらのプロセスが正しく動作しているかどうか確認します。動作していない場合は再起動します。

マシンが以前は NIS+ のクライアントだった

このマシンが、同じドメインの中で NIS+ のクライアントとして初期設定されている場合は、試みに keylogin –r を実行し、Secure RPC パスワードプロンプトでスーパーユーザーのログインパスワードを入力します。

cred テーブルにエントリがない

cred テーブルの中に主体 (ユーザーまたはホスト) の NIS+ のパスワードが存在することを確認するために、主体のホームドメインの中で次のコマンドを実行します。

```
nisgrep -A cname=principal   cred.org_dir. domainname
```

nisgrep を別のドメインで実行する場合は、*domainname* には完全な名前を指定する必要があります。

ドメイン名が変更されている

ドメイン名は変更すべきではありません。

既存のドメイン名を変更すると、認証の問題が発生するはずです。ネットワーク全体で、オブジェクトの中に完全指定のドメイン名が埋め込まれているからです。ドメイン名を変更しないでください。

既にドメイン名を変更してしまい、認証の問題や、ドメイン名に関係する「malformed」や「illegal」などの表現が含まれているメッセージが表示されている場合は、ドメイン名を元の名前に戻します。ドメイン名を変更したい場合は、次の手順に従ってください。「新しい名前」を使用して「新しいドメイン」を作成し、新しいドメインでマシンをサーバーやクライアントとして設定し、これらが正しく動作していることを確認した上で、古いドメインを削除します。

マシンを新しいドメインに変更する

NIS+ のクライアントとなっているマシンを、他のドメインのクライアントに変更したい場合は、/etc/.rootkey ファイルを削除し、ネットワーク管理者から受け取ったパスワードか nisclient スクリプトから取り出したパスワードを使用して、nispopulate スクリプトを実行し直します。

/etc/passwd ファイルの中にある NIS+ のパスワードとログインパスワード

NIS+ のパスワードは、NIS+ の passwd テーブルの中に格納されています。ログインパスワードは、NIS+ の passwd テーブルか、各ユーザーの /etc/passwd ファイルの中に格納されています。ユーザーパスワードと NIS+ のパスワードは、同じでも違っていてもかまいません。/etc/passwd ファイル内のパスワードを変更するには、nsswitch.conf ファイルでの設定を「files」にするか、「-r files」というフラグを指定するかして passwd コマンドを実行する必要があります。

nsswitch.conf ファイルは、どの目的にどのファイルを使用するか指定します。nsswitch.conf ファイルが、システムの照会に対して誤った場所を指示している場合は、パスワードやアクセス権のエラーが発生するはずです。

Secure RPC パスワードとログインパスワードが異なる

主体のログインパスワードと Secure RPC パスワードが一致しないと、ログイン時に `keylogin` はパスワードを復号化できません。`keylogin` はデフォルトで主体のログインパスワードを使用することになっていて、また非公開鍵は主体の Secure RPC パスワードを使用して暗号化されているからです。

この場合、主体はシステムにログインすることはできますが、NIS+ においては未認証クラスとして扱われます。キーサーバーが、そのユーザーの非公開鍵を復号化されていない状態のまま持っているからです。NIS+ 環境では多くの場合、未認承クラスによる NIS+ オブジェクトへのアクセス (作成、削除、変更) が拒否されるため、この状態でユーザーが NIS+ オブジェクトにアクセスしようとしても「`permission denied`」といったタイプのエラーになってしまいます。

注 - 「ネットワークパスワード」と「Secure RPC パスワード」は、このコンテキストでは同義語として扱われる場合があります。ネットワークパスワードの入力を求められたら Secure RPC パスワードを入力します。

認証クラスを「未認承」以外にするには、ユーザーがこの状況で `keylogin` プログラムを実行し、パスワード入力を求める `keylogin` プロンプトが表示されたときに主体の Secure RPC パスワードを入力する必要があります (158ページの「キーログイン」を参照)。

しかし `keylogin` を実行しても、現在のログインセッション以外には無効で、一時的な解決にしかなりません。キーサーバーはこの方法によって、復号化された形でユーザーの非公開鍵を持つようになるのですが、ユーザーの `cred` テーブル中の非公開鍵が Secure RPC パスワード (ユーザーのログインパスワードとは異なっている) を使用して暗号化されているという点に変わりはないからです。ログインし直してしまえば、状況はまったく元どおりになってしまいます。根本的に問題を解決するためには、`cred` テーブルの非公開鍵を、Secure RPC パスワードではなくログイン ID に基づいたものに変更する必要があります。この作業は、`chkey` プログラムを使用して行います (159ページの「NIS+ 主体の鍵の変更」を参照)。

Secure RPC パスワードとログインパスワードが異なっているという問題を根本的に解決するには、具体的には以下の作業を行います。

1. ログインパスワードを使用してログインします。
2. `keylogin` プログラムを実行して、キーサーバーに保存される非公開鍵を一時的に復号し、一時的な NIS+ アクセス権を得ます。

3. chkey –p 実行して、cred テーブル中の暗号化された非公開鍵を、ユーザーのログインパスワードに基づいたものに固定的に変更します。

/etc/.rootkey ファイルがすでに存在している

「症状」

「insufficient permission」や、「permission denied」などの、様々なエラーメッセージ

「考えられる原因」

サーバーやクライアントの設定や初期設定を行なったときに、/etc/.rootkey ファイルがすでに存在していました。以前そのマシンに NIS+ をインストールしたことがあり、NIS+ を削除したとき、または NIS や /etc への変更を行なったときに、.rootkey ファイルを削除しなかったためにこのような状態が起こります。

「診断」

/etc ディレクトリで ls –l と nisls -l org_dir を実行し、/etc/.rootkey の日付を、cred テーブルの日付と比較します。/etc/.rootkey の日付が明らかに cred テーブルより古い場合は、ファイルがあらかじめ存在していたことが考えられます。

「対策」

問題のあるマシンで、ルートとして keylogin –r コマンドを実行し、そのマシンをもう一度クライアントとして設定し直します。

root のパスワードを変更したための問題

「症状」

マシンのルートのパスワードを変更した結果、変更結果が反映されなかったか、スーパーユーザーとしてログインできなくなりました。

「考えられる原因」

注 - セキュリティ上の理由から、passwd テーブルの中に、UserID 0 という項目を、設けるべきではありません。

ルートのパスワードを変更した際、ルートに対して chkey –p を実行していなかったり、何らかの問題が発生したことにより、変更が正しく行われませんでした。

「対策」

管理特権を持つユーザー (つまり、管理特権が割り当てられているグループに所属するメンバー) としてログインし、passwd を使用して、元のパスワードに戻します。元のパスワードが正しく機能するかどうか確かめてください。正しく機能すれば、passwd を使用してパスワードを変更した後、chkey –p を実行します。

注意 - NIS+ の名前空間の設定が終わり、すでに動作している状態でも、ルートマスターのマシンを使ってルートのパスワードを変更することは可能です。しかし、ルートマスターの鍵は変更しないでください。これらは、サブドメイン内のすべてのクライアント、複製サーバー、サーバーの中のすべてのディレクトリオブジェクトに埋め込まれているからです。chkey をルートで実行する際、必ず –p オプションを指定するようにすれば、ルートマスターの鍵を変更する必要はなくなります。

NIS+ の性能の低下とシステムのハングアップの問題

この節では、性能の低下とシステムのハングアップの問題を取り上げます。

性能の低下の症状

処理内容に関係する次の表現が含まれているエラーメッセージが表示されます。

- 「Busy try again later」
- 「Not responding」

次の一般的な現象が観察されることもあります。

- コマンドを実行しても、長時間、何も行われていないように見える
- システムやシェルが、キーボードやマウスの操作に全く反応しない
- NIS+ の動作が通常よりも遅い

チェックポイントの実行

誰かが nisping か nisping –C どちらかのコマンドを実行しました。または、rpc.nisd デーモンが、チェックポイントを実行しています。

注意 - 再起動しないでください。nisping を実行しないでください。

nisping やチェックポイントを実行すると、サーバーは反応が遅くなったり、他のコマンドにすぐに応答しなくなったりすることがあります。名前空間のサイズに応

じて、このようなコマンドが完了するまでに、かなりの時間を要します。これらのコマンドの実行中に、誰かが同様のコマンドを実行すると、所要時間は何倍にもなります。再起動も行わないでください。この種の問題は自然に解決します。サーバーが nisping やチェックポイントの実行を完了するまで待つだけです。

マスターサーバーと複製サーバーの同期が完全に行われている場合、同期が完了するまで、複製サーバーはサービスを停止します。再起動しないで待機してください。

変数 NIS_PATH

NIS_PATH 変数が、明快かつ単純な値に設定されているかどうか確認します。たとえば、デフォルトでは org_dir.$:$ に設定されています。複雑な NIS_PATH、特に他の変数を含んでいる変数は、システムの速度を低下させ、特定の操作にエラーを引き起こす可能性があります (詳細は、101ページの「環境変数 NIS_PATH」を参照)。

デフォルト以外のテーブルパスを指定するために nistbladm を使用することは避けてください。デフォルト以外のテーブルパスを指定すると、性能は低下します。

テーブルパス

テーブルパスは使用しないでください。性能を低下させることになります。

複製サーバーが多すぎる

1 つのドメインの中に複製サーバーが多すぎる場合は、複製作業を行なっている間はシステムの性能が低下します。1 つのドメインまたはサブドメインの中に、10 台より多くの複製サーバーを置くことは避けてください。ドメインの中の複製サーバーが 5 台より多い場合は、何台かを取り除いて性能が改善されるかどうか調べます。

再帰的なグループ

再帰的なグループとは、他のグループを包含するグループのことです。グループの中に他のグループを含めると、システム管理者として行う作業は減少しますが、システムの速度は低下します。再帰的なグループを使うべきではありません。

起動時の NIS+ データベースのログが大きい

rpc.nisd は、起動時に各ログを参照します。ログが大きい場合、rpc.nisd を起動する前に、nisping –C を使用して、チェックポイントを実行する方がよいかもしれません。

マスターの rpc.nisd デーモンが終了している

「症状」

–M オプションを指定して要求をマスターサーバーに送ったときに、マスターのマシンで rpc.nisd デーモンが終了していると、「server not responding」タイプのエラーメッセージが発生し、更新を行うことは認められません。–M オプションを指定しなかったときは、要求は機能している複製サーバーに自動的に転送されます。

「考えられる原因」

ホームディレクトリ名やホスト名の一部として大文字を使うと、rpc.nisd デーモンが終了することがあります。

「診断」

最初に、サーバー自体が起動されて、動作しているかどうか確認します。動作している場合は、ps -ef | grep rpc.nisd を実行し、デーモンが動作しているかどうか調べます。

「対策」

デーモンが終了している場合は、起動し直します。たびたびデーモンが終了する場合は、ご購入先にご連絡ください。

nis_cachemgr がない

「症状」

あるマシンが他のドメインにある名前空間オブジェクトを探すのに、非常に長い時間がかかっています。

「考えられる原因」

nis_cachemgr を実行していません。

「診断」

ps -ef | grep nis_cachemgr を実行して、nis_cachemgr が動作しているかどうか確認します。

「対策」

そのマシンで、nis_cachemgr を起動します。

NIS+ のインストール後、サーバーの起動が非常に遅い

「症状」

NIS+ のスクリプトを使用して NIS+ をインストールした後、サーバーの動作が非常に遅く、緩慢です。

「考えられる原因」

nispopulate スクリプトを動作させた後、nisping –C –a を実行していません。

「対策」

nisping –C –a を実行して、できるだけ早くチェックポイントを実行します。

niscat が エラーメッセージ「Server busy. Try Again」を返す

「症状」

niscat を実行すると、サーバーがビジーであるというエラーメッセージが表示されます。

「考えられる原因」

- 再同期を実行しているときのように重い負荷がかかっていて、サーバーがビジーです。
- サーバーがスワップ空間を使い果たしました。

「診断」

swap –s を実行して、サーバーのスワップ空間をチェックします。

「対策」

NIS+ を実行するには、適度のスワップ空間とディスク容量を用意すべきです。必要に応じて、空間を増やします。

ホスト名を変更した後で、NIS+ の照会がハングする

「症状」

NIS+ サーバーのホスト名を完全指定することは、推奨されていません。このような指定を行なった後、NIS+ の照会を実行すると、エラーメッセージを表示することなくハングアップが発生しました。次の可能性をチェックします。

「考えられる原因」

完全指定のホスト名は、次の条件を満たしていなければなりません。

- ホスト名のドメイン部は、`domainname` コマンドの結果と完全に一致している
- ホスト名を完全指定名に変更した後、`/etc` や `/etc/inet` 内のファイルに、新しいホスト名の情報を反映させる
- ホスト名の終わりにピリオドをつける

「対策」

ハングアップした NIS+ プロセスを終了させ、ホストやサーバーの `rpc.nisd` も終了させます。上の 2 つの条件に合わせて、ホスト名を変更します。`nisinit` を使用してサーバーを初期設定します。ホスト名が正しいことを確認した後もハングアップが発生する場合は、この節の他の考えられる原因をチェックしてください。

NIS+ のシステムリソースの問題

この節では、メモリーやディスク容量のようなシステムリソースが不足したときの問題を取り上げます。

リソースの問題

処理内容に関係する次の表現が含まれているエラーメッセージが表示されます。

- `No memory`
- `Out of disk space`
- 「`Cannot [do something] with log`」などのメッセージ
- `Unable to fork`

メモリーの不足

システムのメモリーやスワップ空間が不足すると、NIS+ の様々な問題やエラーメッセージに遭遇します。一時的な解決策として、不必要なウィンドウやプロセスを終了させることが考えられます。必要に応じて、ウィンドウシステムを終了し、端末のコマンド行を使用します。それでもメモリー不足を知らせるメッセージが表示されるときは、スワップ空間やメモリーを追加するか、十分なスワップ空間やメモリーを持つシステムに切り替えます。

特定の状況では、アプリケーションやプロセスがメモリーを無駄に消費し、使用メモリーサイズが極端に大きくなることがあります。次のコマンドを実行すると、アプリケーションやプロセスの現在のサイズを知ることができます。

```
ps -el
```

sz (サイズ) の列には、各プロセスの現在のメモリーサイズが表示されています。必要に応じて、サイズが同程度と思われるアプリケーションやプロセスを比較し、メモリーサイズが極端に大きいものが存在しないかどうか調べます。

ディスク容量の不足

ディスク容量が不足すると、様々なエラーメッセージが表示されます。ディスク領域の不足に共通する原因は、定期的に行われている tmp ファイルの削除やログファイルの切り捨てをしていないことです。切り捨てを行わないと、ログファイルと tmp ファイルは常時増加します。これらのファイルの増大速度は システムごとに異なり、同じシステムでも状態によって変化します。非効率的なシステムや、名前空間に問題を抱えているシステムでは、ログファイルは非常に急速に増大します。

注 - 多くの問題が発生しているときは、ログファイルと /tmp のファイルを頻繁にチェックします。ディスク容量が不足して他の問題が発生する前に、ログファイルを切り捨て、/tmp ファイルを削除します。また、ルートディレクトリとホームディレクトリで core ファイルを探して削除します。

ログファイルを切り捨てるには、システムのチェックポイントを設けます。チェックポイントのプロセスがある程度の時間を要し、完了するまではシステムの速度を低下させることに注意してください。また、ファイルの切り捨てを行う前に完全なコピーを作成するので、ある程度のディスク容量を必要とします。

システムのチェックポイントを実行するには、nisping –C を実行します。

プロセス数の不足

過度に負荷の大きいマシンでは、マシンの構成の中で指定されている、同時に実行できる最大のプロセス数に達する可能性があります。この結果、「unable to fork」のようなメッセージが表示されます。この問題を解決するために推奨されている方法は、不必要なプロセスを終了させることです。それでも問題が持続する場合は、システムの管理マニュアルの説明に従って、より多くのプロセスを扱えるようにシステムを構成し直してください。

NIS+ のユーザーの問題

この節では、一般ユーザーが遭遇する NIS+ の問題を取り上げます。

ユーザーの問題

- ユーザーがログインできない
- ユーザーが他のドメインにリモートログインできない

ユーザーがログインできない

ユーザーがログインできない原因としては、以下のように様々なものが考えられます。

- パスワードを忘れた

  新しいパスワードを設定します。別のマシンのユーザーの場合は、nispasswd を使用します。この作業は NIS+ 管理者だけが行えます。

- パスワードを間違えて入力した

  ユーザーが、正しいパスワードを覚えているかどうか、また「パスワードでは大文字と小文字が区別される」、「アルファベットの o 、l と数字の 0 、1 は、まったく別のものである」という点を理解しているかどうかを確認します。

- 「Login incorrect」というタイプのメッセージが表示される

  単純に「パスワードの入力を間違えた」という以外の原因については、640ページの「「Login Incorrect」というメッセージが表示された」を参照してください。

- ユーザーのパスワード使用権が有効期限を過ぎている (233ページの「パスワード使用権の有効期限」を参照)。

- 指定された期間を超えてログインが行われなかった (235ページの「ログインの間隔の最大値の指定」を参照)。
- nsswitch.conf ファイルの設定が正しくない

  passwd エントリの設定は以下の 5 つのうちのいずれかにします。

  - passwd: files
  - passwd: files nis
  - passwd: files nisplus
  - passwd: compat
  - passwd: compat

    passwd_compat: nisplus

これ以外の設定をすると、ユーザーがログインできなくなります (詳細は、214ページの「nsswitch.conf ファイルの必要条件」を参照)。

ユーザーが新しいパスワードを使ったときにログインできない

「症状」

最近パスワードを変更したユーザーが、ログインできません。または、特定のマシンからログインできますが、他のマシンからログインできません。

「考えられる原因」

- パスワードの変更を行なっても、システム全体に反映されるまでに時間がかかります。試しに、古いパスワードを使ってログインしてみてください。
- NIS+ が動作していないマシンで、パスワードを変更しました (658ページの「ユーザーが新しいパスワードを使ったときにログインできない」を参照) 。

ユーザーがリモートドメインにログインできない

「症状」

ユーザーが rlogin を使用して、他のドメインへのログインを試みましたが、「Permission denied」メッセージが表示されて、拒絶されました。

「考えられる原因」

他のドメインにリモートログイン (rlogin) するには、ユーザーはそのドメインで LOCAL の資格を持っていなければなりません。

「診断」

そのドメインで、nismatch *username.domainname*. cred.org_dir を実行し、LOCAL の資格を持っているかどうか調べます。

「対策」

リモートドメインから nisaddcred を使用し、そのドメインでの LOCAL の資格をユーザーに割り当てます。

ユーザーがパスワードを変更できない

原因として最も多いのが、古いパスワードの入力を間違えた (または忘れた) ということです。

他には以下のような原因が考えられます。

- パスワードの最小値に、最大値よりも大きい値が設定されている (220ページの「nistbladm と シャドウ列のフィールド」参照)
- パスワードがロックされている、あるいは有効期限を過ぎている (640ページの「「Login Incorrect」というメッセージが表示された」、641ページの「パスワードがロック状態、期限切れ、または無効である」を参照)

NIS+ に関するその他の問題

この節では、上記の問題に当てはまらない問題を取り上げます。

NIS+ の動作

特定のホストが NIS+ を実行しているかどうか知りたい場合があります。NIS+ が動作しているか知るには、スクリプトが必要です。

次のどれかに一致する場合は、NIS+ が動作していると考えられます。

- nis_cachemgr が動作している
- ホストに /var/nis/NIS_COLD_START ファイルがある
- nisls の実行に成功した

複製サーバーの更新のエラー

「症状」

更新が成功しなかったことを示すエラーメッセージが表示されます。次のメッセージが更新の成功を示すことに注意してください。「replica_update: *number* updates *number* errors 」

「考えられる原因」

次のエラーメッセージのどれかが表示された場合、サーバーがビジーであり、更新を延期すべきことがわかります。

- Master server busy,full dump rescheduled
- replica_update: error result was Master server busy,

  full dump rescheduled
- replica_update: master server busy, rescheduling the resync
- replica_update: master server busy, will try later
- replica_update:nis dump result Master server busy,

  full dump rescheduled
- nis_dump_svc: one replica is already resyncing

これらのメッセージは、NIS+ のエラーコード定数、NIS_DUMPLATER (ある複製サーバーが、すでに同期を実行している) により、または、一緒に表示されます。

次のメッセージは、他の問題が起こっていることを示します。

- replica_update: error result was ...
- replica_update: nis dump result nis_perror error string
- rootreplica_update: update failed nis dump result nis_perror

  string-variable: could not fetch object from master

–C (診断チャンネルのオープン) オプションを指定した rpc.nisd が動作している場合は、マスターサーバーか複製サーバーのシステムログに、詳細な情報が記録されることもあります。

これらのメッセージは、次のような潜在的な問題が起こっていることを示しています。

- サーバーが割り当てることができる子プロセスを使い果たした
- 読み込み専用の子プロセスがダンプを行うよう要求された
- 他の複製サーバーが、現在同期を実行している

「診断」

複製サーバーとマスターサーバー両方のシステムログから、詳細な情報を探します。情報が記録されている場合でも、詳細の程度は、システムのエラー報告レベルと、–C オプション (診断) を指定して rpc.nisd を実行したかどうかによって異なります。

「対策」

ほとんどの場合、これらのメッセージは、システムが修正することのできる、ソフトウェアの小さな問題が発生したことを意味しています。あるコマンドを実行した結果、これらのメッセージが表示された場合は、しばらく待って、もう一度同じコマンドを実行します。これらのメッセージが頻繁に表示される場合は、/etc/syslog.conf ファイル内のしきい値レベルを変更します。詳細は、syslog.conf(4) のマニュアルページを参照してください。

NIS の問題と対策

この節では、NIS を実行中のネットワーク上で発生する問題の解決方法を説明します。NIS クライアントで見うけられる問題と、NIS サーバーで見うけられるものを説明します。

NIS サーバーやクライアントをデバッグする前に、第 18 章を参照してください。その後で、この節で、問題を適切に解説する項を参照してください。

症状

一般的な NIS バインディングの問題には次のようなものがあります。

- 「ypbind がサーバーを見つけられない」、あるいは「サーバーと通信できない」というメッセージが表示される
- 「サーバーが応答していない」というメッセージが表示される
- 「NIS が使用できない」というメッセージが表示される
- クライアントのコマンドがバックグランドモードでゆっくりと処理されているか、通常よりも機能に時間がかかる
- クライアントのコマンドがハングする。システム全体は正常で、新しいコマンドを実行できる場合でも、コマンドがハングすることがある

- クライアントのコマンドがあいまいなメッセージと共に、またはまったくメッセージなしでクラッシュする

1 つのクライアントに影響する NIS の問題

NIS バインディングの問題を示す現象が、1 つ以上のクライアントで発生している場合には、問題はクライアントにあります。多くの NIS クライアントが、プロパティを正確にバインドできない場合には、問題は 1 つ以上の NIS サーバーにある可能性があります。666ページの「NIS の問題が多くのクライアントに影響している」を参照してください。

ypbind がクライアントで実行されていない

1 つのクライアントに問題があっても、同じサブネットの他のクライアントが正常に機能しています。問題のあるクライアント上で、`ls –l` を `/usr` のようなディレクトリで実行します。これは、多くのユーザーが所有するファイルを含み、ここにはクライアント `/etc/passwd` ファイルにはないものも含まれます。この結果の表示に、ローカルの `/etc/passwd` には、名前ではなく番号として入ってないファイルの所有者が含まれる場合には、NIS サービスがクライアントで機能していないことを示します。

通常これらの現象は、クライアント `ypbind` プロセスが実行していないこと示します。`ps –e` を実行して、`ypbind` をチェックします。`ypbind` が見つからなければ、スーパーユーザーとしてログインし、次のように入力して、`ypbind` を起動します。

```
client# /usr/lib/netsvc/yp/ypstart
```

ドメイン名がないか不正確である

あるクライアントに問題があり、他のクライアントは正常に機能していますが、`ypbind` は問題のあるクライアント上で実行しています。クライアントのドメインの設定が不正確な可能性があります。

クライアントで `domainname` コマンドを実行して、どのドメイン名が設定されているのかを調べます。

```
Client#7 domainname neverland.com
```

NIS のマスターサーバー上の /var/yp 内の実際のドメイン名と、出力を比較します。実際の NIS ドメインは、/var/yp ディレクトリ内のサブディレクトリとして表示されます。

```
Client#7 ls /var/yp...
-rwxr-xr-x 1 root Makefile
drwxr-xr-x 2 root binding
drwx------ 2 root doc.com
...
```

マシン上での domainname の実行によって得たドメイン名が、/var/yp 内のディレクトリとして示されたサーバードメインと同じではない場合には、マシンの /etc/defaultdomain ファイルで指定されたドメイン名が間違っています。スーパーユーザーとしてログインして、マシンの /etc/defaultdomain ファイル内のクライアントのドメイン名を修正します。これによって、マシンを起動するたびに、ドメイン名が正しいかどうかが確認されます。ここでマシンを再起動しましょう。

注 - ドメイン名では大文字と小文字を区別します。

クライアントがサーバーにバインドされない

ドメイン名が正しく設定されて、ypbind が実行中でも、コマンドがまだハングする場合には、ypbind コマンドを実行することによって、クライアントがサーバーにバインドされていることを確認してください。ypbind を起動したばかりの時は、ypwhich を数回実行します。特に最初のものは、ドメインがバインドされていないことを通知して、2 番目のものは成功します。

サーバーを使用できない

ドメイン名が正しく設定されていて、ypbind が実行中で、クライアントがサーバーと通信できないというメッセージを受け取った場合には、いくつかの問題があります。

- バインドするサーバーのリストを含む /var/yp/binding/*domainname*/ypservers ファイルがクライアントにあるかどうかを確認します。ない場合には、ypinit –c を実行して、設定の順番にクライアントのバインド先のサーバーを指定します。

- クライアントに /var/yp/binding/*domainname*/ypservers ファイルがあり、1つ以上のサーバーが使用できない場合には、十分な数のサーバーがあるかどうかを調べます。ない場合には、yppinit –c を実行して、リストにサーバーを追加します。
- クライアントの ypservers ファイルにリストされたサーバーのどれもが使用できない場合には、クライアントはブロードキャストモードで稼働中のサーバーを検索します。稼働中のサーバーがクライアントのサブネットにある場合には、クライアントはそれを見つけます (検索中はパフォーマンスが落ちる)。クライアントのサブネットに稼動中のサーバーがない場合には、次の方法で問題を解決できます。
  - クライアントがサブネットにサーバーや、それに対する経路を持たない場合には、そのサブネットに新しいスレーブサーバーをインストールできます。
  - ブロードキャストパケットをパスするようにルーターが設定されているかどうかを確認して、クライアントがブロードキャストを使って、別のサブネット上のサーバーを見つけることができるようにします。netstat –r コマンドを使って、経路を検証できます。
  - 経路はあるが、それが機能していない場合には、経路デーモン in.routed/in.rdisc が実行中かどうかを確認してください。実行していない場合には、それを起動します。

注 - セキュリティと管理の意味から、クライアントにブロードキャストを使ってサーバーを検索させるのではなく、クライアントの ypservers ファイルでクライアントのバインド先のサーバーを指定してください。ブロードキャストは、ネットワークを結合して、クライアントの速度を落とします。異なるクライアントに対して、異なるサーバーをリストすることによって、サーバー負荷の均衡がとれなくなります。

- クライアント ypservers ファイルにリストされたサーバーが、/etc/hosts ファイルにエントリを持っているかどうかを確認します。持っていない場合には、NIS マップホストの入力ファイルにサーバーを追加して、389ページの「NIS マップに関する作業」で説明するとおりに、yppinit –c または ypinit –s を実行してマップを再構築します。
- /etc/nsswitch.conf ファイルが設定されて、NIS の他にマシンのローカルの hosts ファイルを参照できるかどうかを確認します。スイッチについての詳細は第 2 章を参照してください。

- /etc/nsswitch.conf ファイルが設定されて、services と rpc に対して、files を参照できるかどうかを確認します。

ypwhich の表示に一貫性がない

ypwhich を同じクライアントで数回使うと、NIS サーバーが変わるので結果の表示が異なります。これは正常な状態です。NIS クライアントから NIS サーバーへのバインディングは、ネットワークや NIS サーバーを使用中の場合は時間の経過に伴って変化します。ネットワークは、すべてのクライアントが受け入れ可能な応答を NIS サーバーから受信した時点で安定します。クライアントのマシンが NIS サービスを得ているかぎりは、サービスの供給元は問題にはなりません。たとえば、NIS サーバーマシンがそれ自体の NIS サービスを、ネットワーク上の別の NIS サーバーから受けることがあります。

サーバーのバインディングが不可能な場合

サーバーのバインディングが不可能な場合には、ypset コマンドを使えます。別のネットワークまたはサブネットの別のサーバーが使用可能な場合には、そのサーバーへのバインディングが一時的に可能になります。ただし、–ypset オプションを使用するためには、ypbind を –ypset または –ypsetme オプションのどちらかを指定して、実行する必要があります。

注 - セキュリティの目的のために、–ypset と –ypsetme のオプションの使用は、制御された状態でのデバッグだけに限定してください。–ypset と –ypsetme のオプションの使用によって、セキュリティが侵害される恐れがあります。これらを実行中は、サーバーのバインディングをだれでも変更でき、他のユーザーを妨害したり、重要なデータへの未承認のアクセスが認められるためです。これらのオプションで ypbind を起動する場合には、いったん問題を確定したら、ypbind を消去して、これらのオプションを指定しないで再起動してください。

---

ypbind のクラッシュ

ypbind が、起動するたびに、すぐにクラッシュする場合には、システムの他の部分で問題を調べてください。以下を入力して、rpcbind デーモンの存在をチェックします。

```
% ps -ef | grep rpcbind
```

rpcbind が存在しないか、または安定せず、動作に異常がある場合には、RPC のマニュアルを参照してください。

正常に機能しているマシンから、問題のあるクライアント上の rpcbind と通信ができる場合があります。稼動中のマシンから、以下を入力します。

```
% rpcinfo client
```

問題のあるクライアント上の rpcbind に問題がない場合には、rpcinfo によって次の出力がされます。

```
program version netid address service owner
...
100007 2 udp 0.0.0.0.2.219 ypbind superuser
100007 1 udp 0.0.0.0.2.219 ypbind superuser
100007 1 tcp 0.0.0.0.2.220 ypbind superuser
100007 2 tcp 0.0.0.0.128.4 ypbind superuser
100007 2 ticotsord \000\000\020H ypbind superuser
100007 2 ticots \000\000\020K ypbind superuser ...
```

マシンは異なるアドレスを持ちます。それらが表示されない場合には、そのサービスを ypbind が登録できていません。マシンを再起動して、再度 rpcinfo を実行します。ypbind プロセスがある場合には /usr/lib/netsvc/yp/ypbind を再起動しようとするたびに変更します。

NIS の問題が多くのクライアントに影響している

1 つ以上のクライアントで、NIS バインディングの問題を示す現象が発生している場合には、それらクライアントに問題がある可能性があります (662ページの「1 つのクライアントに影響する NIS の問題」を参照)。多くのクライアントが正確にバインドできなくなっている場合には、1 つ以上の NIS サーバーに問題がある可能性があります。

ネットワークまたはサーバーが過負荷

ネットワークまたは NIS サーバーが過負荷状態で、クライアント ypbind プロセスに ypserv が時間以内に応答を戻せない場合には、NIS がハングする場合があります。

こういった状態では、ネットワーク上のすべてのクライアントで同じまたは類似した問題が発生します。ほとんどの場合に、この状態は一時的です。NIS サーバーが

再起動して ypserv を再起動するか、または NIS サーバーまたはネットワーク自体の負荷が減少すると、通常、メッセージは消えます。

サーバーの誤動作

サーバーが起動して実行中であることを確認してください。サーバーが物理的に近くにない場合には、ping コマンドを使ってください。

NIS デーモンを実行していない

サーバーが起動して実行中である場合には、クライアントマシンが正常に動作していることを調べて、ypwhich コマンドを実行します。ypwhich が応答しない場合には、それを消去します。NIS サーバーにスーパーユーザーになってログインし、次のように入力して NIS ypbind プロセスが実行中かどうかをチェックします。

```
# ps -e | grep yp
```

注 - –f オプションは ps と共に使わないでください。このオプションはユーザー ID を名前に変換しようとするため、より多くのネームサービスの検索が失敗するようになります。

---

ypbind または ypserv デーモンのどちらかが実行されていない場合には、それらを消去してから、次のように入力して再起動します。

```
# /usr/lib/netsvc/yp/ypstop
# /usr/lib/netsvc/yp/ypstart
```

ypbind または ypserv プロセスの両方が NIS サーバーで実行中の場合には、次のように入力します。

```
# ypwhich
```

ypwhich が応答しない場合には、ypserv がハングしていて、再起動が必要な状態である可能性があります。サーバーに root でログインして、ypserv を消去し、次のように入力して再起動します。

```
# /usr/lib/netsvc/yp/ypstop
# /usr/lib/netsvc/yp/ypstart
```

サーバーに別のバージョンの NIS マップが存在する

NIS はマップをサーバー間で伝播するので、ネットワーク上に異なる NIS サーバーに、同じマップの異なるバージョンが存在することがあります。相違点が長時間継続しない場合には、このバージョンの違いは、許容可能です。

マップの不一致のもっとも一般的な原因は、マップの正常な伝播を妨げる何かが存在するためです。たとえば、NIS サーバーまたはルーターが、NIS サーバー間でダウンしている場合です。すべての NIS サーバーとそれらの間に存在するルーターが実行中の場合には、ypxfr は成功します。

サーバーとルーターが正常に機能している場合には、以下をチェックします。

- ypxfr 出力のログをとります (668ページの「ypxfr 出力のログ」を参照)。
- 制御ファイルをチェックします (669ページの「crontab ファイルと ypxfr シェルスクリプトをチェックする」を参照)。
- マスターの ypservers マップをチェックします (669ページの「ypservers マップをチェックする」を参照)。

ypxfr 出力のログ

特定のスレーブサーバーで、マップの更新に問題がある場合には、そのサーバーにログインして、ypxfr を対話形式で実行します。ypxfr が失敗すると、ypxfr がその失敗を通知するので、問題の修正が可能になります。ypxfr が成功しても、時々失敗するような場合には、メッセージのログを取るためにログファイルを作成します。ログファイルを作成するには、次のように入力します。

```
ypslave# cd /var/yp
ypslave# touch ypxfr.log
```

これによって、ypxfr からのすべての出力を保存する ypxfr.log ファイルが作成されます。

対話形式で実行中の出力 ypxfr の表示に出力は類似しますが、ログファイルの各行にタイムスタンプが押されます。タイムスタンプは、通常とは異なる順番になりま

すが、問題はありません。タイムスタンプは、ypxfr が実行し始めたことを示します。ypxfr のコピーが同時に実行しても、作業時間が異なる場合には、起動元とは異なる順番でサマリーステータス行をログファイルに記述していることがあります。任意の型の失敗が、断続的にログに示されます。

注 - 問題を解決したら、ログファイルを削除してログを停止します。削除しないと、ログは制限なく大きくなります。

---

crontab ファイルと ***ypxfr*** シェルスクリプトをチェックする

root の crontab ファイルを調べて、それが起動した ypxfr シェルスクリプトをチェックします。これらファイルにタイプミスがあると、伝播の問題が発生します。/var/spool/cron/crontabs/root ファイル内でシェルスクリプトを参照できないか、または任意のシェルスクリプト内でマップを参照できない場合にも、エラーが発生します。

ypservers マップをチェックする

NIS スレーブサーバーが、ドメインに対するマスターサーバー上の ypservers マップにリストされていることも確認してください。リストされていない場合には、スレーブサーバーはサーバーとして正しく機能しますが、yppush はマップの変更をスレーブサーバーに伝播しません。

対策

NIS スレーブサーバーの問題が明白ではない場合には、その問題を回避できます。その一方で、rcp または ftp を使ってデバッグし、一貫性のないマップの最新バージョンを問題のない NIS サーバーからコピーできます。たとえば、次のように問題のあるマップを転送します。

```
ypslave# rcp ypmaster:/var/yp/ mydomain/map.\* /var/yp/ mydomain
```

この場合 * の文字は、コマンド行でエスケープされて、ypslave でローカルにではなく、ypmaster で拡張されます。

ypserv のクラッシュ

ypserv プロセスがほとんど即座にクラッシュして、何度再起動しても安定しないときは、デバッグプロセスは、665ページの「ypbind のクラッシュ」で説明するものと実質的に同じです。rpcbind デーモンの存在を次のようにチェックしてください。

```
ypserver% ps -e | grep rpcbind
```

デーモンが見つからない場合にはサーバーを再起動します。そうでない場合には、デーモンが実行中でないなら、以下を入力して、同様の出力を検索します。

```
% rpcinfo -p ypserver
program vers proto port service
1000004 tcp 111 portmapper
1000003 tcp 111 portmapper
1000682 udp 32813 cmsd
...
1000071 tcp 34900 ypbind
1000042 udp 731 ypserv
1000041 udp 731 ypserv
1000041 tcp 732 ypserv
1000042 tcp 32772 ypserv
```

マシンに異なるポート番号がある場合があります。ypserv プロセスを表わす 4 つのエントリは次のとおりです。

```
100004   2   udp   731   ypserv
100004   1   udp   731   ypserv
100004   1   tcp   732   ypserv
100004   2   tcp   32772   ypserv
```

このエントリがなく、ypserv がそのサービスを rpcbind で登録できない場合には、マシンを再起動してください。これらのエントリがある場合には、rpcbind からサービスの登録を解除してから、ypserv を再起動します。rpcbind からサービスの登録を解除するには、サーバーで次のように入力します。

```
# rpcinfo -d number 1
# rpcinfo -d number 2
```

この場合 *number* は、rpcinfo によって通知される ID 番号です (前述の例では、100004)。

DNS の問題と対策

この節では、一般的な DNS の問題とその対策を説明します。

クライアントはマシンを見つけられるが、サーバーはできない

「症状」

DNS クライアントは、IP アドレスかホスト名でマシンを見つけられますが、サーバーは IP アドレスでしか見つかりません。

「考えられる原因と対策」

サーバーの nsswitch.conf ファイルの hosts 行から DNS を省略したために発生する可能性があります。たとえば、不完全な hosts 行は、次のようになります。

```
host:files
```

DNS を使用中には、次のどちらかのように、すべてのマシンの nsswitch.conf ファイルの hosts レコード内に dns を含む必要があります。

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

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

変更に効果がないか不安定になる

「症状」

マシンまたはサーバーを追加または削除しても、変更が認識されず、その効果が現れません。またはある時は変更が認識され、別の時にはその効果が現れません。

「考えられる原因」

考えられる原因は、主マスターサーバー上の SOA のシリアル番号を増やすのを忘れた場合です。新しい SOA 番号がないので、主サーバーのものと一致させるためのデータ更新を副サーバーは行いません。このため、古い未変更のデータファイルで作業を行なっています。

この他に考えられる原因は、1 つ以上の主要なデータファイルの SOA のシリアル番号が、副サーバー上の対応するシリアル番号よりも小さい値に設定されたということです。たとえば、この状態は、主サーバー上のファイルを削除してから、ある種の入力ファイルを使って最初からそれを作成し直した場合に発生します。

考えられる 3 番目の原因は、主サーバーのデータファイルへの変更を行なった後に、HUP 信号を主サーバーに送信し忘れた場合です。

「診断と対策」

最初に、変更したデータファイルの SOA のシリアル番号と副サーバー上の対応するファイルをチェックします。

- 主ファイルの SOA のシリアル番号が、副ファイルのシリアル番号と同じか、またはそれ以下の場合には、主サーバーのファイルでシリアル番号を増加させて、副ファイルの番号よりも大きくなるようにします。たとえば、両方のファイルの SOA のシリアル番号が 37 の場合には、主サーバーのファイルの番号を 38 に変更します。次回、副サーバーが主サーバーをチェックすると、新しいデータがロードされます。主サーバーに、副サーバーへの転送を即座に強制するユーティリティがあります。これらユーティリティの 1 つがある場合には、主サーバーのチェックを待たずに副サーバーを更新できます。
- 最新の `named` *`nnnn`* `restarted` または、`named` *`nnn`* `reloading nameserver` エントリに対する `syslog` 出力をレビューします。そのエントリ用のタイムスタンプが、ファイルへの変更を終了した時間の前の場合には、サーバーを再起動するか、または、613ページの「in.named に DNS データを強制的に再読み込みさせる」で説明するとおりに新しいデータの読み取りを強制します。

DNS クライアントが短縮名を検索できない

「症状」

クライアントは完全指定名は検索できますが、短縮名は検索できません。

「考えられる原因と現象」

クライアントの `/etc/resolv.conf` ファイルで、ドメイン名の最後にスペースがないかをチェックします。スペースやタブはドメイン名の最後では許可されません。

リバースドメインデータが正確に副サーバーに転送されない

「症状」

ゾーンのドメイン名の付いたデータは、ゾーンの主マスターサーバーからゾーンの副サーバーに正確に転送される一方で、リバースドメインデータは転送されません。つまり、副サーバーの host.rev ファイルが主サーバーから正確に更新されていません。

「考えられる原因」

副サーバーのブートファイルの構文エラー

「診断と対策」

副サーバーのブートファイルをチェックします。主サーバーのブートファイルの IP アドレスが、ホストデータに対するのと同じようにリバースゾーンエントリに対してリストされていることを確認してください。

たとえば、主サーバーの IP アドレスが、secondary in-addr.arpa レコードからなくなっているので、次のブートファイルは正しくありません。

```
;
; /etc/named.boot file for dnssecondary
directory /var/named
secondary   doc.com   129.146.168.119        dnshosts.bakup
secondary   168.146.129.in-addr.arpa  doc.rev.bakup
```

正しいファイルは次のようになります。

```
;
; /etc/named.boot file for dnssecondary
directory /var/named
secondary   doc.com   129.146.168.119        dnshosts.bakup
secondary   168.146.129.in-addr.arpa   129.146.168.119  doc.rev.bakup
```

サーバーが失敗してゾーンが問題を期限切れにした

副サーバーがそのマスターから更新を得られないときは、「master unreachable」のメッセージをログに記録します。問題が修正されない場合には、副サーバーはゾーンを期限切れにして、クライアントからの要求への応答を停止します。これが発生すると、「server failed」のメッセージが表示されるようになります。

「症状」

- syslog 内の「Masters for secondary zone *domain* unreachable」のメッセージ
- syslog 内の「Secondary zone *domain* expired」のメッセージ
- ユーザーへの「*** *domain* Can't find *name*: server failed」のメッセージ

問題が副サーバーにある場合には、一部のユーザーは、マスターから DNS 情報を獲得でき、問題なく操作できます。

「考えられる原因」

これらの問題に対して考えられる主な 2 つの原因は、副サーバーのブートファイル内のマスターに対する誤った IP アドレスとネットワーク障害があります。

「診断と対策」

- 副サーバーのブートファイルに、マスターに対する正しい IP アドレスがあるかをチェックします。次の行をチェックしてください。

```
secondary domain    IPaddress  hostsfile
```

マスターの IP アドレスが、hosts ファイルで指定されたマスターに対するアドレスとマスターの実際の IP アドレスと一致することを確認してください。IP アドレスが誤っている場合には、それを修正してから副サーバーを再起動します。

- マスターの IP アドレスが正しい場合には、マスターの IP アドレスを ping することによって、マスターが起動して、正しく実行中であることを確認します。たとえば、マスターを IP アドレス 129.146.168.119 に ping するには、次のように入力します。

```
% ping 129.146.168.119 -n 10
```

- マスターが ping に応答しない場合には、それが起動して、正しく実行中であることを確認します。
- マスターが実行中の場合には、ps を使って、それが named を実行中であることを確認します。named を実行していない場合には、再起動します。
- マスターが named を正しく実行中の場合には、ネットワーク障害が発生している可能性があります。

rlogin、rsh、ftp の問題

「症状」

- インターネットで、別のドメインのマシンに rlogin を試みたときに、ユーザーがパスワード入力を求められます。
- インターネットで、別のドメインのマシンに ftp を試みたときに、ユーザーがアクセスを拒否されます。
- 自分自身のネットワーク上のマシンに対して rlogin や rsh を試みたときに、ユーザーがアクセスを拒否されます。

「考えられる原因」

- 主マスターサーバーの hosts.rev ファイルに PTR を持たないマシンでユーザーが作業をしています。
- hosts.rev ファイル内のサブドメインがないか、または不正確な委任が行われています。

「診断と対策」

適切な hosts.rev ファイルをチェックして、ユーザーのマシンに対して PTR レコードがあることを確認します。たとえば、129.146.168.46 の IP アドレスを持つマシン altair.doc.com で、ユーザーが作業をしている場合には、doc.com の主マスターサーバーの doc.rev ファイルは、次のようなエントリを持つ必要があります。

```
46  IN  PTR  altair.doc.com.
```

レコードがない場合には、それを hosts.rev ファイルに追加してから、サーバーを再起動するか、613ページの「in.named に DNS データを強制的に再読み込みさせる」で説明するとおりにデータを再ロードします。

hosts.rev ファイルの NS エントリをチェック、および修正してからサーバーを再起動するか、613ページの「in.named に DNS データを強制的に再読み込みさせる」で説明するとおりに、そのデータを再ロードします。

その他の DNS 構文エラー

「症状」

次のような言い回しを伴ったコンソールまたは syslog のエラーメッセージは、たいてい DNS データとブートファイルの構文エラーによって発生します。

- No such...
- Unknown field...
- Non-authoritative answer:
- Database format error...
- illegal または illegal
- error receiving zone transfer

関連ファイルでスペルと構文のエラーをチェックしてください。

一般的な構文エラーは、ドメイン名で後ろに付く点 (ドット) の誤用 (禁じられている場合に使い、必要な場合に使わないなど) に起因します。611ページの「ドメイン名の終わりにつけるドット」を参照してください。

FNS の問題と対策

この節では、問題と共に、考えられる原因、診断、対策を示します。

FNS エラーについての一般的な情報については、686ページの「FNS エラーメッセージ」と付録 B を参照してください。

初期コンテキストが取得できない

「症状」

「Cannot obtain initial context」というメッセージが表示されます。

「考えられる原因」

インストール時の問題により発生します。

「診断」

/usr/lib/fn/fn_ctx_initial.so というファイルを検索し、FNS が正しくインストールされているかどうかを確認します。

「対策」

fn_ctx_initial.so ライブラリをインストールします。

初期コンテキストが空になっている

「症状」

`fnlist` を実行して初期コンテキストの内容を確認すると何も表示されないという状態です。

「考えられる原因」

NIS+ の設定によって起こる問題です。`fn*` コマンドを実行するユーザーやマシンに関連した組織に、`ctx_dir` ディレクトリがないことが原因です。

「診断」

`nisls` コマンドを使用して `ctx_dir` ディレクトリの有無を確認します。

「対策」

診断の結果 `ctx_dir` ディレクトリがなければ、`fncreate -t org/`*nis+_domain_name*`/` を実行して作成します。

「no permission」というメッセージが表示される (FNS)

「症状」

「`no permission`」というメッセージが表示されます。

「考えられる原因」

このメッセージは、「コマンドを実行しようとしたが、アクセス権がない」ということを意味します。

「診断」

適切な NIS+ コマンドを使用してアクセス権を確認します (518ページの「FNS と NIS+ の詳細情報」を参照)。NIS+ 主体名は、`nisdefaults` コマンドで知ることができます。

また、使用している名前が正しいかどうかも確認する必要があります。たとえば、ルート組織のコンテキスト名は、`org//` を使用して決定します。ルート組織を操作する権限があるかどうか確認してください。`myorgunit/` を指定する場合もあります。

「対策」

アクセス権が正しく設定されているにもかかわらずこのメッセージが表示される場合は、適切な資格が与えられていない可能性があります。

これには以下の原因が考えられます。

- keylogin が実行されていない (NIS+ 主体はデフォルトでは「未認証」になる)
- NIS+ 以外のソースに対して keylogin が実行された
  - /etc/nsswitch.conf ファイルに publickey: nisplus エントリがあるかどうか確認する
  - これはおそらく認証エラーとなる

fnlist で下位組織のリストが表示されない

「症状」

fnlist に組織名を指定して実行しても何も表示されません。

「考えられる原因」

NIS+ の設定によって発生する問題です。下位組織は NIS+ ドメインでなければなりません。NIS+ ドメインには、org_dir というサブディレクトリが必要です。

「診断」

nisls コマンドを使用して、どんなサブディレクトリが存在するかを調べます。nisls をサブディレクトリごとに実行すれば、どのサブディレクトリに org_dir があるかを確かめることができます。org_dir のあるサブディレクトリが下位組織になります。

「対策」

NIS+ ドメインに、org_dir というサブディレクトリを作成します。

ホストコンテキストまたはユーザーコンテキストが作成できない

「症状」

fncreate −t を、user (または username)、または host (または hostname) を指定して実行しても何も起こりません。

「考えられる原因」

NIS_GROUP 環境変数が設定されていません。ユーザーコンテキストあるいはホストコンテキストを作成した際、所有権が名前空間を設定した管理者ではなく、ホスト

またはユーザーにあったようです。したがって、fncreate を実行するには、NIS_GROUP 変数を設定して、グループ内の管理者によってコンテキストの操作が行えるようにする必要があります。

「診断」

NIS_GROUP 環境変数をチェックします。

「対策」

NIS_GROUP 環境変数に、コンテキストの管理者のグループ名を設定します。

作成したコンテキストを削除できない

「症状」

ホストコンテキストまたはユーザーコンテキストに対して fndestroy を実行しても、コンテキストが削除されません。

「考えられる原因」

ホストコンテキストまたはユーザーコンテキストの所有権を持っていないことです。ユーザーコンテキストあるいはホストコンテキストを作成した際、所有権が名前空間を設定した管理者ではなく、ホストまたはユーザーにあったようです。

「診断」

NIS_GROUP 環境変数をチェックします。

「対策」

NIS_GROUP 環境変数に、コンテキストの管理者のグループ名を設定します。

fnunbind を実行すると「name in use」というメッセージが表示される

「症状」

バインデイングを削除しようとすると、「name in use」というメッセージが表示されます。指定する名前によってコマンドが正しく実行される場合と、そうでない場合があります。

「考えられる原因」

コンテキストの名前のバインドを解除できません。この制限は、名前のないコンテキスト (orphaned context) が残るような場合に適用されます。

「診断」

fnlist コマンドを実行し、fnunbind に指定しようとする名前がコンテキストのものかどうか確認します。

「対策」

名前がコンテキストのものであれば、fndestroy コマンドを使用してコンテキストを削除します。

fnbind/fncreate -s を実行すると「name in use」というメッセージが表示される

「症状」

–s オプションをつけて fnbind および fncreate を実行すると、指定する名前によっては「name in use」というメッセージが表示されます。

「考えられる原因」

fnbind –s、および fncreate –s を実行すると、既存のバインディングは上書きされます。ただしそれによって orphaned context ができるような場合、「name in use」というエラーメッセージが表示され、この操作は失敗します。

「診断」

fnlist コマンドを実行して、fnbind、fncreate に指定しようとする名前がコンテキストのものかどうか確認します。

「対策」

fndestroy コマンドを実行してコンテキストを削除してから、fnbind または fncreate を (先にエラーになった場合と同じ名前を指定して) 実行します。

実体のない名前を指定して fndestroy / fnunbind を実行しても「Operation Failed」が返らない

「症状」

実体のない名前を指定して fndestroy、fnunbind を実行しても、操作が失敗したことがまったく知らされない。

「考えられる原因」

fndestroy、fnunbind のセマンティクスが、「端末名がバインドされていなくても、操作が success を返す」というようになっているためだと考えられます。

「診断」

問題が発生した場合と同じ名前を指定して fnlookup コマンドを実行します。「name not found」というメッセージが表示されるはずです。

「対策」

考えられる現象を参照。

付録 **B**

エラーメッセージ

この節では、一般的な NIS+ のエラーメッセージをアルファベット順に説明します。メッセージごとに説明を行い、該当するものがある場合は、解決策や、このマニュアルの他の部分の参照箇所を示します。

付録 A は、様々な種類の問題と解決策を示しています。この付録の中のエラーメッセージで該当するものがある場合は、付録 A の参照箇所を示しています。

エラーメッセージについて

この付録の中で説明されているエラーメッセージのいくつかは、該当するマニュアルページで詳細に説明されています。

この章で説明したエラーメッセージの一部は、マニュアルページでさらに詳しく説明します。

エラーメッセージの内容

エラーメッセージは、ポップアップウィンドウ、シェルツールのコマンド行、ユーザーコンソールウィンドウや、各種のログファイルに表示または記録されます。`/etc/syslog.conf` ファイルに指定されているエラーの重大度の基準を上げたり下げたりすることもできます。

ほとんどの場合、入力したコマンドか、コマンドが送られたコンテナオブジェクト (ファイル、マップ、テーブル、またはディレクトリ) によって、エラーメッセージ

が生成されます。しかし時には、コマンドに応答したサーバーによって、エラーメッセージが生成されることもあります (メッセージは通常、syslog に記録される)。たとえば、「permission denied」メッセージは、ほとんどの場合ユーザーかマシンが原因となって発生しますが、コマンドやマシンから要求された機能を実行するのに必要なアクセス権を、サーバー上のソフトウェアが持っていない場合に発生することもあります。

同様に、一部のコマンドは、非常に多くの種類のオブジェクトに対して検索や照会を行います。オブジェクトによっては、原因が明確であるとは限りません。オブジェクトのアクセス権 (読み取り専用の状態、利用できないなどの状態) が原因で、エラーメッセージが返されることがあります。このような場合は、どのオブジェクトが原因となって問題が起こったのか、メッセージからわかる場合もあれば、わからない場合もあります。

通常の操作では、ネーミングなどのソフトウェアとサーバーは、関数ルーチン呼び出しを行います。時として、これらの呼び出しが障害を起こし、エラーメッセージが生成されることがあります。また、ユーザーが入力したコマンドをクライアントやサーバーが処理する前に、それ以外のコマンドによる呼び出しが障害を起こし、エラーメッセージが生成されることもあります。そのようなエラーメッセージは、あたかも今入力したコマンドに対する応答のように見えるかもしれませんが、実際はそれまでの操作に対する応答です。

注 - 名前空間で作業を行なっているときに、遠隔手続き呼び出し (RPC) によってエラーメッセージが生成される可能性もあります。これらの RPC エラーメッセージは、このマニュアルでは説明していません。ご使用のシステムのマニュアルを参照してください。

状況によって異なる意味

NIS+ のソフトウェアのどの部分がエラーメッセージを生成したかにより、同じエラーメッセージがやや異なる意味を持つことがあります。たとえば、nisls コマンドが「Not Found」というメッセージを生成した場合、指定された名前の NIS+ のオブジェクトが存在しないことを意味します。しかし、nismatch コマンドが同じメッセージを表示した場合、検索基準に一致するテーブル項目が見つからなかったことを意味します。

エラーメッセージのアルファベット順ソート規則

この付録のエラーメッセージは、次の規則に従ってアルファベット順に示します。

- 大文字と小文字を区別しません。つまり、「A」で始まるメッセージと「a」で始まるメッセージは同じ順になります。
- アルファベット以外の記号は無視します。つまり、`_svcauth_des` で始まるメッセージは、文字「S」で始まる他のメッセージと同じ順になります。
- 「NIS+」で始まる (または「NIS+」を含む) エラーメッセージは、アルファベット順で「NIS」で始まる (または「NIS+」を含む) メッセージの後になります。
- エラーメッセージの前に、エラーメッセージを生成したホスト、アプリケーション、プログラム、ルーチンの日付または名前が付き、その後にコロンが付く場合があります。この場合には、コマンドの最初の名前はメッセージをアルファベット順にするために使われます。
- メッセージの中には、ユーザー ID、プロセス番号、ドメイン名などの変数が含まれています。この章では、これら変数は斜体の文字で示されます。変数は任意のものにできるので、この章に示すメッセージの分類には含まれていません。たとえば、「`sales:is not a table`」(この場合は `sales` が名前) という実際のメッセージは、「*name*: `is not a table`」として示され、I で始まるメッセージの中で「`is not a table`」としてアルファベット順にされます。
- 「`** ERROR:` *domainname* `does not exist`」のように、アスタリスクで始まるエラーメッセージは、NIS+ のインストールスクリプトや設定スクリプトによって生成されたものです。これらのメッセージは、アスタリスクを無視し、最初の文字でソートされます。

エラーメッセージ内の番号

- DNS の多く、または他のメッセージは、IP アドレスを含みます。IP アドレスは *n.n.n.n.* で示されます。
- エラーメッセージにプロセス ID 番号、項目の数などのような番号が含まれる場合があります。エラーメッセージ内の番号は *nnnn* で示されます。

FNS エラーメッセージ

FNS エラーメッセージは、FN_status_t オブジェクトにステータスコードとしてカプセル化されます。対応するステータスコードについては、FN_status_t(3N) のマニュアルページを参照してください。

エラーが発生すると、FNS コマンドは、操作が失敗した名前の残りの部分を表示します。表示されなかった名前の部分の処理は成功しています。

たとえば、ユーザーが、org//service/trading/bb に対するコンテキストの作成を試みたとします。名前 org//service/ の解決には成功しましたが、trading は org//service/ によって名前を付けられたコンテキストで見つかりませんでした。このため trading/bb は、操作が失敗したときに残る名前の一部として表示されます。

```
Error in creating 'org//service/trading/bb': Name Not Found: 'trading/bb'
```

別の例では、ユーザーがコンテキスト org//service/dictionary/english の削除を試みましたが、名前を付けられたコンテキストが空であったので、操作を実行できませんでした。引用符 ('') の組み合わせは、与えられた完全な名前を FNS は解決できましたが、要求されたとおりに操作を完了できなかったことを示します。

```
Error in destroying 'org//service/dictionary/english': Context Not Empty: ''
```

NIS+、FNS に共通するエラーメッセージ

abort_transaction: Failed to action NIS+ *objectname*

abort_transaction ルーチンが、サーバーのクラッシュや他の回復不可能なエラーにより、不完全なトランザクションから抜けることに失敗しました。詳細は、630ページの「NIS+ データベースの問題」を参照してください。

abort_transaction: Internal database error abort_transaction: Internal error, log entry corrupt NIS+ *objectname*

これら 2 つのメッセージは、名前空間のデータベースやログが壊れていることを意味します。詳細は、630ページの「NIS+ データベースの問題」を参照してください。

add_cleanup: Cant allocate more rags.

このメッセージは、システムで使用できるメモリーが不足していることを示しています。

`add_pingitem: Couldn't add` *directoryname* `to pinglist (no memory)`

(メモリー不足が原因となる問題の詳細は、656ページの「メモリーの不足」を参照してください。)

```
add_update: Attempt add transaction from read only child.
add_update Warning: attempt add transaction from read only child
```

読み取り専用の子プロセス `rpc.nisd` が、ログに項目を追加しようとしました。ログの中にときどきこのメッセージが記録されても、重大な問題ではありません。しかし、頻繁に記録される場合は、ご購入先にご連絡ください。

`Attempting to free a free rag!`

このメッセージは、`rpc.nisd` にソフトウェア上の問題が起こったことを意味します。`rpc.nisd` は、異常終了したはずです。`ps -ef | grep rpc.nisd` を実行して、`rpc.nisd` が現在も動作しているかどうか確認します。まだ動作している場合は、`rpc.nisd` を終了させ、前回と同じオプションを指定して再起動します。動作していない場合は、前回と同じオプションで再起動します。`/var/nis` をチェックして、`core` ファイルのダンプが行われていないか確認します。`core` ファイルが存在する場合は、削除してください。

注 - `–YB` オプションを指定して `rpc.nisd` を起動した場合は、`rpc.nisd_reply` デーモンも終了しなければなりません。

`Attempt to remove a non-empty table`

`nistbladm` が，まだエントリを含んでいる NIS+ テーブルを削除しようとしました。または、`nisrmdir` が、ファイルあるいはサブディレクトリを含むディレクトリを削除しようとしました。

- テーブルを削除する場合には、`niscat` を使ってテーブルの内容をチェックし、`nistbladm` を使って既存の内容を削除します。
- ディレクトリを削除する場合には、`nisls –l –R` を使って既存のファイルまたはサブディレクトリをチェックして、それらを最初に削除します。
- `nisrmdir –s` を使って、ドメインから複製を分離するときに、複製がダウンしているか、またはマスターとの通信不能の状態にある場合に、このエラー

メッセージが表示されます。このような場合には、nisrmdir –f –s *replicaname* をマスターで実行して、分離を強制できます。しかし、nisrmdir –f –s *replicaname* を使って通信不能な複製を分離する場合には、複製がオンライン状態に戻ったらすぐに、nisrmdir –f –s *replicaname* を再実行して、複製の /var/nis ファイルシステムを消去する必要があります。nisrmdir –f –s *replicaname* を再実行しないと、複製がサービスを再開した時に複製上に残された古い情報によって問題が発生します。

このメッセージは、NIS+ のエラーコード定数 NIS_NOTEMPTY によって生成されます (詳細な情報は、nis_tables(3N) のマニュアルページを参照)。

`attribute no permission`

FNS のエラーメッセージです。「呼び出し側が属性に対して何らかの操作をしようとしたが、アクセス権がないため実行できなかった」ということを意味します。

`attribute value required`

FNS のエラーメッセージです。「値を指定しないで属性の作成が試みられたが、ネーミングシステムがそれを許可しなかった」ということを意味します。

`authdes_marshal: DES encryption failure`

認証データの DES 暗号化に失敗しました。考えられる原因は次のとおりです。

- ライブラリ関数か引数が壊れている
- DES 暗号化チップを使用している場合、そのチップに問題がある

ご購入先にご連絡ください。

`authdes_refresh: keyserv is unable to encrypt session key`

keyserv プロセスは、公開鍵を使用してセッション鍵の暗号化を行うことができませんでした (詳細は、647ページの「Keyserv のエラー」を参照)。

`authdes_refresh: unable to encrypt conversation key`

keyserv プロセスは、公開鍵を使用してセッション鍵の暗号化を行うことができませんでした。このメッセージが表示された場合、通常はユーザーの側で何らかの操作を行う必要があります。考えられる原因は次のとおりです。

- keyserv プロセスが終了しているか、応答しません。ps –ef を使用して、keyserv プロセスが keyserv のホスト上で動作しているかどうかチェッ

クします。動作していない場合は、keyserv を起動し、keylogin を実行します。

- クライアントが keylogin を実行していません。クライアント側で keylogin を実行し、問題が解決できるかどうか調べます。
- クライアントのホストに、資格が割り当てられていません。クライアントのホームドメインの cred テーブルに対して nismatch を使用して、クライアントのホストに適切な資格が割り当てられているかどうか調べます。資格がない場合は作成します。
- DES の暗号化に失敗しました (エラーメッセージ「authdes_marshal: DES encryption failure」を参照)。

セキュリティ鍵の問題の詳細は、640ページの「NIS+ のセキュリティの問題」を参照してください。

`authdes_refresh: unable to synchronize clock`

このエラーメッセージは、クライアントとサーバーそれぞれのクロックが同期していないことを示しています。通常は、この問題は自動的に訂正されます。しかし、このメッセージの後に、タイムスタンプに関係したエラーが発生した場合は、手作業でクロックを同期させなければなりません。この問題が再度発生した場合は、rpcbind が正しく機能しているかどうかチェックします。

`authdes_refresh: unable to synch up w/server`

クライアントとサーバーそれぞれのクロックを同期させることに失敗しました。サーバー上の rpcbind プロセスが応答しなかったことが原因となった可能性もあります。サーバー上で ps –ef を実行し、rpcbind が動作しているかどうか調べます。動作していない場合は、rpcbind を再度起動します。このメッセージの後に、タイムスタンプに関係したエラーが発生した場合は、rdate *servername* を使用して、クライアントのクロックを手作業でサーバーのクロックに同期させなければなりません。

`authdes_seccreate: keyserv is unable to generate session key`

このエラーメッセージは、keyserv がこのセッション用のランダムな DES キーを生成できなかったことを示しています。このメッセージが表示された場合、ユーザーの側で何らかの操作を行う必要があります。

- keyserv が正しく動作しているかどうかチェックします。動作していない場合は、automountd、rpc.nisd、sendmail などのように、secure RPC を使

用したり NIS+ のコールを行う、長時間にわたって動作するプロセスとともに、keyserv を再度起動します。次に、keylogin を実行します。

- keyserv が起動されていて正しく動作している場合は、このエラーをログに記録したプロセスをもう一度起動します。

`authdes_seccreate: no public key found for` *servername*

クライアント側で、サーバー名 *servername* の DES の資格を取得できませんでした。このメッセージが表示された場合、ユーザーの側で何らかの操作を行う必要があります。

- *servername* に DES の資格が割り当てられているかどうかチェックします。割り当てられていない場合は、DES の資格を作成します。
- スイッチ構成ファイルをチェックして、指定されているネームサービスを調べ、そのサービスが応答するかどうか確認します。応答しない場合は、ネームサービスを再度起動します。

`authdes_seccreate: out of memory`

メモリー不足が原因となる問題の詳細は、655ページの「NIS+ のシステムリソースの問題」を参照してください。

`authdes_seccreate: unable to gen conversation key`

keyserv プロセスは、ランダムな DES キーを生成できませんでした。最も可能性の高い原因は、keyserv プロセスが停止しているか、応答しないことです。ps –ef を使用して、keyserv プロセスが keyserv のホストで動作しているかどうかチェックします。動作していない場合は、keyserv を起動し、次に keylogin を実行します。

keyserv を起動し直しても問題が解決しない場合は、secure RPC を使用したり NIS+ をコールするプロセス (たとえば、automountd、rpc.nisd、sendmail) が動作していない可能性があります。これらのプロセスが動作しているかどうかチェックし、動作していない場合は再度起動します。

(セキュリティ鍵の問題に関する詳細は、640ページの「NIS+ のセキュリティの問題」を参照してください。)

`authdes_validate: DES decryption failure`

「authdes_marshal: DES encryption failure」を参照してください。

`authdes_validate: verifier mismatch`

クライアントがサーバーに送ったタイムスタンプが、サーバーから受け取ったタイムスタンプに一致していません (Secure RPC セッションの中で回復できません) 。考えられる原因は次のとおりです。

- クライアントかサーバーのキャッシュに格納されているセッション鍵またはタイムスタンプのデータが壊れています。
- サーバーがキャッシュから、まだ有効なセッションのセッション鍵を削除しました。
- ネットワークデータが壊れています。

コマンドを再度実行してください。

`authentication failure`

FNS のエラーメッセージです。「要求を作成した主体が、関連するネームサービスによって認証されなかったため、操作が完了しなかった」ということを意味します。NIS+ サービスを使用している場合は、(コマンド `nisdefaults` を使用して)「ユーザーが正しい主体として認識されているかどうか」を確認し、また「公開鍵のソースが、マシンに正しく指定されているか」を確認します (`/etc/nsswitch.conf` ファイルに "`publickey: nisplus`" というエントリがあるかどうかを確認する)。

`bad reference`

FNS のエラーメッセージです。「FNS がリファレンスの内容を理解できなかった」ということを意味します。「リファレンスの内容が壊れている」、「FNS のものであるとされているリファレンスが、FNS で復号化できない」といった場合に発生します。

`CacheBind: xdr_directory_obj failed.`

このエラーメッセージが表示された場合、最も可能性のある原因は次のとおりです。

- `xdr_directory_obj` ルーチンに渡されたパラメータが不適切です。直前に入力したコマンドの構文が正確かどうかチェックしてください。
- システムへのメモリー割り当てに失敗しました。メモリーの問題の詳細は、656ページの「メモリーの不足」を参照してください。

- コマンドの構文は正確で、システムもメモリー不足を起こしていないという場合は、ご購入先にご連絡ください。

`Cache expired`

オブジェクトのキャッシュから返されたエントリが、有効期限を過ぎています。つまり、持続時間の値が 0 になり、エントリが変更された可能性があることを示しています。フラグ –NO_CACHE が検索用関数に渡された場合は、その検索関数は動作を繰り返し行い、有効期限を過ぎていないオブジェクトのコピーの取得を試みます。

このメッセージは、NIS+ のエラーコード定数 NIS_CACHEEXPIRED によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

`Callback: - select failed` *`message nnnn`*

内部システムコールがエラーを起こしました。ほとんどの場合、この問題は自動的に修正されます。自動的に修正されない場合は、rpc.nisd が異常終了していないかどうか確認します。異常終了している場合は、再度起動します。この問題が頻繁に発生する場合は、ご購入先にご連絡ください。

`CALLBACK_SVC: bad argument`

内部システムコールがエラーを起こしました。ほとんどの場合、この問題は自動的に修正されます。自動的に修正されない場合は、rpc.nisd が異常終了していないかどうか確認します。異常終了している場合は、再度起動します。この問題が頻繁に発生する場合は、ご購入先にご連絡ください。

`Cannot grow transaction log error` *`string`*

システムは、ログファイルへの追加を行うことができません。理由は *string* が示すとおりです。最もよく考えられる原因は、ディスク容量の不足です (656ページの「ディスク容量の不足」を参照)。

`Cannot obtain Initial Context`

FNS のエラーメッセージです。インストールに問題があることを示します (676ページの「初期コンテキストが取得できない」を参照)。

`Cannot truncate transaction log file`

ログにチェックポイントを実行しようとしましたが、rpc.nisd デーモンは、チェックポイントを設定したエントリをログから削除した後で、ログファイルを圧縮しようとしました。様々な要因が、このルーチンに障害を引き起こす可能性がありますが、詳細は ftruncate(3C) のマニュアルページを参照してください (630ページの「NIS+ データベースの問題」も参照)。

Cannot write one character to transaction log, error*message*

rpc.nisd デーモンは、現在のトランザクションを使用して、トランザクションログに更新部分を追加しようとしましたが失敗しました。原因は、関数が返した *message* の中に示されています。書き込みルーチンのマニュアルページの中に詳細が記されていることもあります。

Can't compile regular expression *variable*

keypat 内の正規表現が不適切だったために、nisgrep コマンドによってエラーメッセージが返されました。

Can't get any map parameter information.

661ページの「NIS の問題と対策」を参照してください。

Can't find name service for passwd

「nsswitch.conf ファイルがない」、「nsswitch.conf ファイルに passwd のエントリがない」、「passwd のエントリが意味をなさない」、「passwd のエントリの形式が正しくない」のいずれかを意味するエラーメッセージです。

Can't find *name* 's secret key

考えられる原因は次のとおりです。

- パスワードの入力ミス
- cred テーブルの中に、*name* のエントリがない
- NIS+ が鍵を復号化できなかった (エントリが壊れている可能性がある)
- nsswitch.conf ファイルが、cred テーブルの中に記録されている NIS+ パスワードとは異なるパスワードを使って、/etc/passwd ファイルの中に格納されているローカルパスワードの照会を行なっている

このタイプの問題の診断と解決に関する情報は、640ページの「NIS+ のセキュリティの問題」を参照してください。

*** *servername.domainname* can't find *machinename*; Server failed.

DNS のエラーメッセージです。673ページの「サーバーが失敗してゾーンが問題を期限切れにした」と、675ページの「その他の DNS 構文エラー」を参照してください。

Can't find server name for address 127.0.0.1; server failed.

DNS のエラーメッセージです。通常このメッセージは、主マスターサーバーが不正な情報を持つ古くなった named.ca ファイルを使用していることを示します。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は、最上位のドメイン (たとえば、.com) を管理する当局から現在の named.ca ファイルを得る必要があります。.com、.edu、.gov、.mil、.org などに対する当局は、InterNIC です。ネットワークがインターネットに接続されていない場合には、named.ca ファイルのエラーをチェックしてください。

checkpoint_log: Called from read only child ignored.

このメッセージは、読み取り専用プロセスが、親プロセスに限定されている操作を試みて、その操作が異常終了したことを示す状態メッセージです。何か動作を行う必要はありません。

checkpoint_log: Unable to checkpoint, log unstable.

安定した状態にないログに対して、チェックポイントを実行しようとしました。つまり、ログは再同期、更新、チェックポイント、どれかの状態にありました。ログが安定した状態になるまで待って、nisping コマンドを再度実行します。

check_updaters: Starting resync.

システムの状態メッセージです。何か動作を行う必要はありません。

Child process requested to checkpoint!

このメッセージは、システムが修正することのできる、ソフトウェアの小さな問題が発生したことを意味しています。これらのメッセージが頻繁に表示される場合は、/etc/syslog.conf ファイル内の基準を変更できます。詳細は、syslog.conf(4) のマニュアルページを参照してください。

Column not found: *columnname*

指定した列名は、指定したテーブルの中に存在しません。

`communication failure`

FNS のエラーメッセージです。「FNS がネームサービスとコミュニケーションできず、操作を完了できなかった」ということを意味します。

`configuration error`

構成上の問題により発生するエラーです。以下に例を示します。

(1) バインディングテーブルが削除されました (FNS の問題ではありません)。

(2) ホストは NIS+ ホストディレクトリオブジェクトの中に存在しますが、ホストに対応する FNS ホストコンテキストがありません。

`context not empty`

FNS のエラーメッセージです。「中にバインディングの残っているコンテキストを削除しようとした」ということを意味します。

`continue operation using status values`

FNS のエラーメッセージです。「ステータスオブジェクトの中に残っている名前、名前との対応づけのできたリファレンスを使用して動作を続行する」ということを意味します。

`Could not find` *string* `'s secret key`

考えられる原因は次のとおりです。

- 正しくないパスワードを入力しました。
- `cred` テーブルの中に、`name` のエントリがありません。
- NIS+ が鍵を復号化できませんでした (おそらく、エントリが壊れています)。
- `nsswitch.conf` ファイルで公開鍵の指定方法を間違えています。`cred` テーブルの中に登録されている NIS+ パスワードとは異なるパスワードを使用して、`/etc/passwd` ファイルの中に格納されているローカル公開鍵の照会を行なっています。

このタイプの問題の診断と解決に関する情報は、640ページの「NIS+ のセキュリティの問題」を参照してください。

`Could not generate netname`

keyloginを実行する際、Secure RPC のソフトウェアが、ユーザーの UID に対応する Secure RPC のネット名を生成できませんでした。考えられる原因は次のとおりです。

- マシンのホームドメインにある NIS+ の cred テーブルで、LOCAL の資格が割り当てられていません。
- /etc/passwd ファイル内にあるローカルエントリの UID が、NIS+ の passwd テーブル内にある UID と違っています。

string: could not get secret key for '*string*

考えられる原因は次のとおりです。

- 正しくないパスワードを入力しました。
- cred テーブルの中に、name のエントリがありません。
- NIS+ が鍵を復号できませんでした (エントリが壊れている可能性があります)。
- nsswitch.conf ファイルで、公開鍵の指定方法を間違えています。cred テーブルの中に登録されている NIS+ パスワードとは異なるパスワードを使用して、/etc/passwd ファイルの中に格納されているローカル公開鍵の照会を行なっています。

このタイプの問題の診断と解決に関する情報は、640ページの「NIS+ のセキュリティの問題」を参照してください。

Couldn't fork a process!

サーバーは、コールバックの要求を満たす子プロセスをフォークできませんでした。おそらく、システムが最大のプロセス数に達したためです。不必要なプロセスのいくつかを終了させるか、システムが扱える最大プロセス数を増やすことができます (詳細は、657ページの「プロセス数の不足」を参照)。

Couldn't parse access rights for column *string*

このメッセージは、通常 nistbladm –u コマンドで、演算子として + (プラス記号)、- (マイナス記号)、= (等号) 以外の記号を入力したときに返されます。他に考えられる原因として、列の権利をコンマで区切っていないことや、アクセス権の種類として r、d、c、m のどれでもない文字を指定したことが挙げられます。エントリのタイプによるエラーの場合は、構文をチェックしてください。入力が正しくてもエラーが発生する場合は、テーブルが壊れていることも考えられます。

Database for table does not exist

テーブルの参照に失敗しました。考えられる原因については、633ページの「NIS+ オブジェクトが見つからない問題」を参照してください。

このメッセージは、NIS+ のエラーコード定数 NIS_NOSUCHTABLE によって生成されます (詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照)。

```
_db_add: child process attempting to add/modify _db_addib:
non-parent process attempting an add
```

これらのメッセージは、読み取り専用プロセスや、親プロセス以外のプロセスが、データベース内のオブジェクトを追加または変更しようとしたことを示します。ほとんどの場合、なんらかの操作を行う必要はありません。この問題が頻繁に発生する場合は、ご購入先にご連絡ください。

db_checkpoint: Unable to checkpoint *string*

このメッセージは、なんらかの理由で、NIS+ がディレクトリにチェックポイントを実行できなかったことを示します。最も可能性の高い原因は、ディスク容量の不足です (656ページの「ディスク容量の不足」を参照)。

```
_db_remib: non-parent process attempting an remove _db_remove:
non-parent process attempting a remove
```

これらのメッセージは、読み取り専用プロセスや、親プロセス以外のプロセスが、テーブルエントリを削除しようとしたことを示します。ほとんどの場合、なんらかの操作を行う必要はありません。この問題が頻繁に発生する場合は、ご購入先にご連絡ください。

```
Do you want to see more information on this command?
```

このメッセージは、スクリプトのコマンド行の中に、なんらかの構文エラーかスペルミスがあることを示します。

```
Entry/Table type mismatch
```

テーブルでエントリを追加または変更しようとして、他のタイプのテーブルからエントリを引き渡した場合に、このメッセージが表示されます。たとえば、列数が等しくないことが考えられます。更新に使っているテーブルのタイプが正確に一致しているかどうかチェックします。

このメッセージは、NIS+ のエラーコード定数 NIS_TYPEMISMATCH によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

error

FNS のエラーメッセージです。「要求処理中に、ここまでにとりあげたどのエラーにも該当しない状況が発生した」ということを意味します。関連のあるネームサービスの状態をチェックし、特殊な問題が発生していないことを確認します。

**ERROR: chkey failed again. Please contact your network administrator to verify your network password.

このメッセージは、ネットワークのパスワードの入力ミスを示します。

- マシンを初めて初期設定するときは、ネットワーク管理者に連絡を取り、ネットワークのパスワードを確認します。
- このマシンが、同じドメインの中で NIS+ のクライアントとして初期設定されたことがある場合は、Secure RPC のパスワードプロンプトで、root のログインパスワードを入力してみます。
- このマシンが現在 NIS+ のクライアントであり、他のドメインのクライアントに変更したい場合は、/etc/.rootkey ファイルを削除し、ネットワーク管理者から受け取ったパスワード (または nispopulate スクリプトから生成したパスワード) を使用して、nisclient スクリプトを再度実行します。

Error: Could not create a valid NIS+ coldstart file

このメッセージは、NIS+ の初期設定ルーチン nisinit によって生成されます。この後に、「lookup:..」で始まるメッセージが表示されます。2 番目のメッセージは、NIS+ の有効なコールドスタートファイルが作成できなかった理由を示します。

**ERROR: could not restore file *filename*

このメッセージは、NIS+ が、*filename*.no_nisplus を *filename* に変更できなかったことを示します。

システムコンソールを使用して、システムエラーメッセージの有無をチェックします。

- システムエラーメッセージがある場合は、そのエラーメッセージの示す問題を解決し、nisclient –i を再度実行します。
- システムエラーメッセージがない場合は、そのファイルを手作業で名前を変更し、nisclient –i を再度実行します。

**ERROR: Couldn't get the server NIS+_server's address.

このスクリプトは、指定されたドメインのサーバーの IP アドレスを取り出すことができませんでした。/etc/hosts ファイル内のサーバー *NIS+_server* に、手作業で IP アドレスを追加し、nisclient –i を再度実行します。

**ERROR: directory *directory-path* does not exist.

このメッセージは、正しくないディレクトリパス名を入力したことを示します。正しいディレクトリパス名を入力します。

**ERROR: *domainname* does not exist.

このメッセージは、存在しないドメインを複製しようとしたことを示します。

- *domainname* のスペルが正しくない場合は、正しいドメイン名を指定して、このスクリプトをもう一度実行します。
- *domainname* のドメインが存在しない場合は、そのドメインを作成します。次に、複製を行うことができます。

**ERROR: *parent-domain* does not exist.

このメッセージは、コマンド行で指定されたドメインの親ドメインが存在しないことを示します。このメッセージは、ルートマスター以外のサーバーの設定を行なっているときにだけ表示されるはずです。

- ドメイン名のスペルが正しくない場合は、正しいドメイン名を指定して、もう一度このスクリプトを実行します。
- このドメインの親ドメインが存在しない場合は、最初に親ドメインを作成し、次にこのドメインを作成します。

**ERROR: Don't know about the domain ''*domainname*''. Please check your *domainname*.

このメッセージは、認識できないドメイン名が入力されたことを示します。正しいドメイン名を指定して、もう一度このスクリプトを実行します。

**ERROR: failed dumping *tablename* table.

このスクリプトは、指定されたテーブルのダンプに失敗したため、cred テーブルを生成できませんでした。

- niscat *tablename* .org_dir の実行に失敗した場合は、すべてのサーバーが動作しているかどうか確認し、もう一度このスクリプトを実行して、tablename テーブルを生成します。
- niscat *tablename* .org_dir が実行できる場合は、NIS+ のサーバーが一時的にビジーだったために、エラーが発生した可能性があります。もう一度このスクリプトを実行して、tablename テーブルを生成します。

**ERROR: host *hostname* is not a valid NIS+ principal in domain *domainname*. This host name must be defined in the credential table in domain *domainname*. Use nisclient -c to create the host credential

あるマシンを NIS+ のサーバーにする前に、そのマシンを NIS+ のクライアントとし、適切な資格を割り当てる必要があります。マシンを NIS+ のルート複製サーバーに変換するには、最初にそのマシンをルートドメインの NIS+ クライアントにする必要があります。ドメインに新しいクライアントを追加するための指示に従い、nisserver –R を再度実行します。

マシンを NIS+ のルート以外のマスターサーバーや複製サーバーに変換するには、そのマシンを、サービスを提供するドメインの親ドメインの NIS+ クライアントにしておかなければなりません。ドメインに新しいクライアントを追加するための指示に従い、nisserver –M か nisserver –R を再度実行します。

ルートマスターサーバーを設定するときは、この問題は発生しないはずです。

Error in accessing NIS+ cold start file is NIS+ installed?

このメッセージは、NIS+ がマシンにインストールされていない場合や、何らかの理由で /var/nis/NIS_COLD_START ファイルが見つからないかアクセスできない場合に発生します。/var/nis/NIS_COLD_START ファイルが存在しているかどうかチェックします。ファイルが存在している場合は、パスが正しく設定されているかどうか、また NIS_COLD_START に正しいアクセス権が割り当てられているかどうか確認します。次に、古いコールドスタートファイル名の変更か削除を行い、nisclient スクリプトをもう一度実行して、NIS+ をマシンにインストールします。

このメッセージは、NIS+ のエラーコード定数 NIS_COLDSTART_ERR を送信するキャッシュマネージャによって生成されます (ファイルがアクセスできない理由の詳細は、write(1) と open(2) のマニュアルページを参照)。

Error in RPC subsystem

これは致命的なエラーで、RPC サブシステムになんらかの障害が発生したことを示します。通常は、クライアント側かサーバー側に、RPC 要求が失敗した理由を示す syslog メッセージがあります。

このメッセージは、NIS+ のエラーコード定数 NIS_RPCERROR によって生成されます。詳細は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

**ERROR: it failed to add the credential for root.

NIS+ のコマンド nisaddcred は、ルートマスターサーバーの設定を行なっているときに、root の資格を作成できませんでした。システムコンソールを使用して、システムエラーメッセージの有無をチェックします。

- システムエラーメッセージが発生している場合は、そのエラーメッセージの示す問題を解決し、nisserver を再度実行します。
- システムエラーメッセージが発生していない場合は、rpc.nisd プロセスが動作しているかどうかチェックします。動作していない場合は、rpc.nisd をもう一度起動し、nisserver を再度実行します。

**ERROR: it failed to create the tables.

NIS+ のコマンド nissetup は、ディレクトリやテーブルを作成できませんでした。システムコンソールを使用して、システムエラーメッセージの有無をチェックします。

- システムエラーメッセージが発生している場合は、そのエラーメッセージの示す問題を解決し、nisserver を再度実行します。
- システムエラーメッセージが発生していない場合は、rpc.nisd プロセスが動作しているかどうかチェックします。動作していない場合は、rpc.nisd をもう一度起動し、nisserver を再度実行します。

**ERROR: it failed to initialize the root server.

NIS+ のコマンド nisinit –r は、ルートマスターサーバーの初期設定に失敗しました。システムコンソールを使用して、システムエラーメッセージの有無を

チェックします。システムエラーメッセージが発生している場合は、そのエラーメッセージの示す問題を解決し、nisserver を再度実行します。

**ERROR: it failed to make the *domainname* directory

nisserver を実行して、ルート以外のマスターを作成しているときに、NIS+ のコマンド nismkdir は、*domainname* という新しいディレクトリの作成に失敗しました。親ドメインに、新しいドメインを作成するためのアクセス権が割り当てられていません。

- そのドメインの所有者ではなく、親ドメインのグループメンバーでもない場合は、所有者か親ドメインのグループメンバーとして、そのスクリプトをもう一度実行します。
- rpc.nisd が、作成中のドメインのサーバーとなるマシンで動作していない場合は、rpc.nisd をもう一度起動します。

**ERROR: it failed to promote new master for the *domainname* directory

nisserver スクリプトを使ってルート以外のマスターを作成しているときに、NIS+ のコマンド nismkdir は、ディレクトリ *domainname* の新しいマスターの変換に失敗しました。

- このドメインの親ドメインで変更権を割り当てられていない場合は、親ドメインの所有者かグループメンバーとして、そのスクリプトをもう一度実行します。
- rpc.nisd が、変換先のドメインのサーバーで動作していない場合は、これらのサーバー上で rpc.nisd をもう一度起動し、nisserver を再度実行します。

**ERROR: it failed to replicate the *directory-name* directory

NIS+ のコマンド nismkdir は、ディレクトリ *directory-name* の新しい複製サーバーを作成できませんでした。

- rpc.nisd が、複製サーバーを作成したいドメインのマスターサーバー上で動作していない場合は、マスターサーバーで rpc.nisd をもう一度起動し、nisserver を再度実行します。
- rpc.nisd が、新しい複製サーバー上で動作していない場合は、新しい複製サーバーで rpc.nisd をもう一度起動し、nisserver を再度実行します。

**ERROR: invalid group name. It must be a group in the *root-domain* domain.

このメッセージは、ルートマスターサーバーを構成しているときに、不適切なグループ名が使用されたことを示します。*root-domain* の適切なグループ名を指定して、nisserver –r をもう一度実行します。

**ERROR: invalid name ``*client-name*'' It is neither an host nor an user name.

このメッセージは、不適切な *client-name* が入力されたことを示します。

- *client-name* のスペルが正しくない場合は、正しい *client-name* を指定して、nisclient –c をもう一度実行します。
- *client-name* のスペルは正しくても、適切なテーブルの中に存在していない場合は、テーブルの中に *client-name* を記述して、nisclient –c をもう一度実行します。たとえば、ユーザーのクライアントは passwd テーブルの中に存在し、ホストのクライアントは hosts テーブルの中に存在します。

**ERROR: *hostname* is a master server for this domain. You cannot demote a master server to replica. If you really want to demote this master, you should promote a replica server to master using nisserver with the M option.

同じドメインの中で、マスターサーバーを直接複製サーバーに変換することはできません。しかし、複製サーバーのホスト名を指定して nisserver –M を実行することにより、複製サーバーを新しいマスターサーバーに変換できます。同時に、元のマスターは自動的に複製サーバーに変換されます。

**ERROR: missing hostnames or usernames.

このメッセージは、コマンド行でクライアント名が入力されなかったことを示します。クライアント名を指定して、nisclient –c をもう一度実行します。

**ERROR: NIS+ group name must end with a ``.''

このメッセージは、ピリオドで終わる完全指定のグループ名が指定されなかったことを示します。完全指定のグループ名を使用して、このスクリプトをもう一度実行します。

**ERROR: NIS+ server is not running on *remote-host*. You must do the following before becoming a NIS+ server: 1. become a NIS+ client

of the parent domain or any domain above the domain which you plan to serve. (nisclient) 2. start the NIS+ server. (rpc.nisd)

このメッセージは、NIS+ に変換しようとしているリモートマシン上で、rpc.nisd が動作していないことを示します。nisclient スクリプトを使用して、そのマシンを、サービスを提供したいドメインの親ドメイン、またはそれより上位のドメインの NIS+ クライアントにします。次に、*remote-host* 上で、rpc.nisd を起動します。

**ERROR: nisinit failed.

nisinit は、NIS_COLD_START ファイルを作成できませんでした。

次のことをチェックします。

- –H オプションで指定した NIS+ サーバーが動作しているかどうか (ping を使用して確認する)
- 正しいドメイン名を入力したかどうか
- rpc.nisd がサーバー上で動作しているかどうか
- 未認証クラスに、そのドメインでの読み取り権が割り当てられているかどうか

**ERROR: NIS map transfer failed. *tablename* table will not be loaded.

NIS+ は、このテーブルの NIS マップを、NIS+ データベースに転送できませんでした。

- NIS サーバーのホストが動作している場合は、もう一度このスクリプトを実行します。一時的な障害が原因で、エラーが発生した可能性があります。
- すべてのテーブルでこのエラーが発生した場合は、他の NIS サーバーを使用して、このスクリプトをもう一度実行します。

**ERROR: no permission to create directory *domainname*

親ドメインに、新しいドメインを作成するために必要な作成権が割り当てられていません。そのドメインの所有者ではなく、親ドメインのグループメンバーでもない場合は、所有者か親ドメインのグループメンバーとして、そのスクリプトをもう一度実行します。

**ERROR: no permission to replicate directory *domainname*.

このメッセージは、ドメインを複製するためのアクセス権が割り当てられていないことを示します。このドメインの所有者かグループメンバーとして、スクリプトをもう一度実行します。

`error receiving zone transfer`

DNS のエラーメッセージです。通常これは、主サーバーの DNS ファイルの 1 つの構文エラーを示します。675ページの「その他の DNS 構文エラー」を参照してください。

`**ERROR: table` *tablename* `.org_dir.`*domainname* `does not exist.''`
tablename `table will not be loaded.''`

スクリプトは、NIS+ のテーブル `tablename` を見つけることができませんでした。

- *tablename* のスペルが正しくない場合は、正しいテーブル名を指定して、スクリプトをもう一度実行します。
- *tablename* が NIS+ の標準テーブルの 1 つであり、かつ *tablename* テーブルが存在しない場合は、`nissetup` を使用して、そのテーブルを作成します。*tablename* が NIS+ の標準テーブルではなく、また存在しない場合は、`nistbladm` を使用して、独自のテーブル *tablename* を作成します。次に、スクリプトを再度実行して、このテーブルを生成します。
- *tablename* テーブルが存在する場合は、NIS+ のサーバーが一時的にビジーであったためにエラーが発生した可能性があります。このスクリプトをもう一度実行して、*tabelname* テーブルを生成します。

`**ERROR: this name ''`*clientname*`'' is in both the passwd and hosts tables. You cannot have an username same as the host name.`

client-name が、`passwd` テーブルと `hosts` テーブルの両方に記述されています。両方のテーブルに、共通の名前を記述することは認められていません。`passwd` と `hosts` どちらかのテーブルから、手作業でこのエントリを削除します。次に、`nisclient –c` を再度実行します。

`**ERROR: You cannot use the -u option as a root user.`

このメッセージは、スーパーユーザーが `nisclient –u` を実行しようとしたことを示します。`–u` オプションは、普通のユーザーだけを初期設定するためのものです。NIS+ クライアントとは異なり、スーパーユーザーは初期設定を行う必要がありません。

**ERROR: You have specified the *Z* option after having selected the *X* option. Please select only one of these options [*list*]. Do you want to see more information on this command?

いま実行しているスクリプトでは、リスト表示されているオプションのうち、どれか 1 つしか使用できません。

- y と入力すると、詳細な情報が表示されます。
- n と入力すると、スクリプトは停止して、元のプロンプトに戻ります。

スクリプトを終了した後で、オプションを 1 つだけ指定して、もう一度実行します。

**ERROR: you must specify a fully qualified groupname.

このメッセージは、ピリオドで終わる完全指定のグループ名が指定されなかったことを示します。完全指定のグループ名を使用して、このスクリプトをもう一度実行します。

**ERROR: you must specify both the NIS domainname (-y) and the NIS server host name (-h).

このメッセージは、NIS のドメイン名と NIS サーバーのホスト名の一方または両方を指定しなかったことを示します。プロンプトかコマンド行で、NIS のドメイン名と NIS サーバーのホスト名を入力します。

**ERROR: you must specify one of these options: -c, -i, -u, -r.

このメッセージは、これらのオプション –c、–i、–u、–r のどれかがコマンド行に指定されていなかったことを示します。正しいオプションを指定して、スクリプトをもう一度実行します。

**ERROR: you must specify one of these options: -r, -M or -R''

このメッセージは、–r、–M、–R のどのオプションも指定されなかったことを示します。正しいオプションを指定して、スクリプトをもう一度実行します。

**ERROR: you must specify one of these options: -C, -F, or -Y

このメッセージは、–Y、–F どちらのオプションも入力しなかったことを示します。正しいオプションを指定して、スクリプトをもう一度実行します。

**ERROR: You must be root to use -i option.

このメッセージは、普通のユーザーが nisclient –i を実行しようとしたことを示します。nisclient –i を実行できるアクセス権を割り当てられているのは、スーパーユーザーだけです。

`Error while talking to callback proc`

サーバーがクライアントからコールバックされているときに、サーバー上で RPC エラーが発生しました。この時点でトランザクションは異常終了し、まだ送信されていないデータは破棄されます。サーバー上の syslog で詳細をチェックします。

このメッセージは、NIS+ のエラーコード定数 NIS_CBERROR によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

`First/Next chain broken`

このメッセージは、コールバックルーチンが結果を返しているときに、クライアントとサーバーの接続が解除されたことを示します。プロセスの実行中にサーバーが終了すると、このようなメッセージが発生することがあります。

このメッセージは、NIS+ のエラーコード定数 NIS_CHAINBROKEN によって生成されます。

`getzone: print_update failed`

DNS のエラーメッセージです。通常これは、主サーバーのDNS ファイルの 1 つの構文エラーを示します。675ページの「その他の DNS 構文エラー」を参照してください。

`Generic system error`

要求をしているときに、何らかの一般的なシステムエラーが発生しました。システムの syslog の記録をチェックし、サーバーから返されたエラーメッセージを探します。

このメッセージは通常、サーバーがクラッシュしたか、データベースが壊れたことを示します。サーバーや複製サーバーは、自らサービスを提供するドメインより上位にあるドメインのクライアントとなっていますが、サービスを提供するドメインに所属しているかのように、これらのサーバーや複製サーバーの名前を指定すると、このメッセージが生成されます (詳細は、628ページの「ドメイン名の混同」を参照してください)。

このメッセージは、NIS+ のエラーコード定数 NIS_SYSTEMERROR によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

`illegal name`

FNS のエラーメッセージです。指定された名前が正しくないことを示します。

`Illegal object type for operation`

このタイプの問題の詳細は、625ページの「無効なオブジェクトの問題」を参照してください。

このメッセージは、NIS+ のエラーコード定数 DB_BADOBJECT によって生成されます。

`incompatible code sets`

FNS のエラーメッセージです。「操作中、互換性のないコードセットの文字列が使用された」、あるいは「サポートされていないコードセットが提供されている」ということを意味します。

`in.named [`*nnnn*`]: lame server on` *hostname*

DNS のエラーメッセージです。不完全な委託とは、親ドメインのサーバーの hosts ファイルにある NS レコードでは別のサーバーがサブドメインのゾーンに対して権限を持つとされているのに、実際にはそのサーバーはそのゾーンに対する権限を持っていないことをいいます。親のホストファイルの NS レコードは、委託されたすべてのサブゾーンについて、権限をもつサーバーを全部含んだスーパーセットである必要があります。

`insufficient permission to update credentials.`

このメッセージは、動作に必要なアクセス権が割り当てられていない状態で、nisaddcred コマンドを実行したときに発生します。テーブル、列、エントリどれかのレベルでアクセス権が不足している可能性があります。niscat -o cred.org_dir を使用して、cred テーブルでどのようなアクセス権が割り当てられているを確認します。他のアクセス権も必要な場合は、ユーザー自身、またはシステム管理者のどちらかが、そのオブジェクトのアクセス権の変更を行うことができます。詳細は、第 10 章と第 12 章を参照してください。

アクセス権の問題の詳細は、637ページの「NIS+ の所有権とアクセス権の問題」を参照してください。

`insufficient resources`

FNS のエラーメッセージです。「FNS が使用するネームサービスに十分なリソースがなく、要求された動作を実行できない」ということを意味します。ネームサービスのメモリーおよびディスク容量をチェックします。

`invalid attribute identifier`

FNS のエラーメッセージです。「属性識別子が、ネーミングシステムで使用できない形式になっている」、あるいは「属性識別子の内容が、指定された形式において有効なものではない」ということを意味します。

`invalid attribute value`

FNS のエラーメッセージです。「指定された属性の値の形式が正しくない」ということを意味します。

`invalid enumeration handle`

FNS のエラーメッセージです。「提供されている列挙ハンドルが正しくない」ということを意味します。「処理中に更新が発生した」などの理由が考えられます。

`Invalid Object for operation`

- 「Name context」

  関数に渡された名前は、NIS+ の有効な名前ではありません。

- 「Table context」

  指定されたオブジェクトは、このテーブルにとって有効な NIS+ エントリのオブジェクトではありません。列数が一致しない場合や、テーブルの中で関連する列とデータ型 (たとえば、バイナリかテキストか) が異なる場合に、このメッセージが発生する可能性があります。

このメッセージは、NIS+ のエラーコード定数 `NIS_INVALIDOBJ` によって生成されます (詳細な情報は、`nis_tables(3N)` と `nis_names(3N)` のマニュアルページを参照してください)。

`invalid syntax attributes`

FNS のエラーメッセージです。「指定された構文属性が正しくないために、構文を完全に決定できない」ということを意味します。

`invalid usecs` *Routine_name*`: invalid usecs`

このメッセージは、`struct time stamp` タイプの変数フィールド `tv_usecs` の値が、1 秒未満マイクロ秒単位の値より大きいときに発生します。通常は、何らかのタイプのソフトウェアエラーが原因です。

tablename `is not a table`

tablename という名前を持つオブジェクトは、テーブルオブジェクトではありません。たとえば、`nisgrep` や `nismatch` のコマンド行で、テーブルオブジェクト以外のオブジェクトを指定すると、このメッセージが発生します。

`link error`

FNS のエラーメッセージです。指定された名前を使用して XFN リンクを作成する際に発生するエラーです。

`link loop limit reached`

FNS のエラーメッセージです。「複数の名前を結びつける際、XFN リンクが原因で完了しないループが検出された」、または「一回の操作における XFN のリンクの数が、実装で定義された上限を超えた」ということを意味します。

`Link Points to illegal name`

渡された名前を LINK タイプのオブジェクトとして解決しましたが、そのオブジェクトの内容が不適切な名前を指しています。

テーブルエントリをリンクさせることができません。このエラーメッセージはエントリレベルでリンクを作成しようとすると発生します。

このメッセージは、NIS+ のエラーコード定数 `NIS_LINKNAMEERROR` によって生成されます (詳細な情報は、`nis_tables(3N)` と `nis_names(3N)` のマニュアルページを参照)。

`Load limit of` *number* `reached!`

サーバー上で子プロセスの数が既に上限に達しているときに、子プロセスを作成しようとしました。このメッセージは、メッセージをログに記録する基準のレベルが `LOG_WARNING` を含む場合に、サーバーのシステムログに記録されます。

`login and keylogin passwords differ.`

nispasswd を使用してパスワードを変更し、システムはパスワードを変更しましたが、cred テーブル内の資格のエントリを新しいパスワードに更新できず、さらに、passwd テーブルの中で元のパスワードに復元もできない場合に、このメッセージが発生します。このメッセージの後に、次の指示が表示されます。

```
Use NEW password for login and OLD password for
keylogin. Use ''chkey -p'' to reencrypt the credentials with
the new login password. You must keylogin explicitly after
your next login.
```

この指示の後に、なぜ元のパスワードに復元することができなかったのか、理由を示す状態メッセージが表示されます。これらのメッセージが表示された場合は、その指示に従ってください。

`Login incorrect`

このメッセージの原因として最も多いのは、パスワードのタイプミスです。もう一度入力し直してください。また覚えているパスワードが正しいかどうかも確認してください。特に、大文字と小文字、アルファベットの o と数字の 0、アルファベットの l と数字の 1 を間違えないよう注意してください。

このメッセージの原因については、640ページの「「Login Incorrect」というメッセージが表示された」にも例が挙げられています。

`log_resync: Cannot truncate transaction log file`

ログにチェックポイントを実行しようとしましたが、rpc.nisd デーモンは、チェックポイントが設定されたエントリをログから削除した後で、ログファイルの圧縮を試みました。様々な要因が、このルーチンに障害を引き起こす可能性がありますが、詳細は、ftruncate(3C) のマニュアルページを参照してください (630ページの「NIS+ データベースの問題」も参照)。

`malformed link`

FNS のエラーメッセージです。「fn_ctx_lookup_link() の動作中に、不正なリンク参照が検出された」ということを意味します。指定された名前が、リンクされていないリファレンスに結びつけられたということです。

`Malformed Name or illegal name`

関数に渡された名前は、NIS+ の有効な名前ではありません。

考えられる理由の一つは、誰かがドメイン名を変更したことです。既存のドメイン名は変更すべきではありません (648ページの「ドメイン名が変更されている」を参照)。

このメッセージは、NIS+ のエラーコード定数 NIS_BADNAME によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照。

_map_addr: RPC timed out.

プロセスやアプリケーションが、必要なデータの取得や NIS+ からのホスト名の解決を行うデフォルト時間内に、NIS+ と通信できませんでした。ほとんどの場合、短時間待つと、この問題は自動的に回復します (性能低下の問題に関する詳細な情報は、651ページの「NIS+ の性能の低下とシステムのハングアップの問題」を参照してください)。

Master server busy full dump rescheduled

このメッセージは、マスターサーバーがビジーであったため、複製サーバーがマスターサーバーから得られたフルダンプを使って自らを更新することができなかったことを示します。詳細な情報は、659ページの「複製サーバーの更新のエラー」を参照してください。

String Missing or malformed attribute

属性の名前が、テーブル内の指定された列に一致していません。または、属性に値が割り当てられていません。

コマンドの構文エラーが原因となっている可能性があります。*string* から、何が間違っているのか推測できます。一般的な原因は、スペルミス、等号 (=) を不適切な場所に置いたこと、列名やテーブル名の間違いなどです。

このメッセージは、NIS+ のエラーコード定数 NIS_BADATTRIBUTE によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

Modification failed

コマンドを実行しているときに、誰かがグループ名を変更したため、nisgrpadm がこのメッセージを返しました。誰かがグループの操作を行なっているかどうかチェックします。次に、このコマンドを再度実行します。

このメッセージは、NIS+ のエラーコード定数 NIS_IBMODERROR によって生成されます。

`Modify operation failed`

試みた変更が、何らかの理由で失敗しました。

このメッセージは、NIS+ のエラーコード定数 `NIS_MODFAIL` によって生成されます (詳細な情報は、`nis_tables(3N)` と `nis_names(3N)` のマニュアルページを参照してください)。

servername `named [`*nnnn*`]: directory` *directoryname*`: No such file or directory.`

DNS のエラーメッセージです。通常これは、DNS のブートまたはデータファイルの構文またはスペルのエラーを示します。

servername `named [`*nnnn*`]: /etc/named.boot: line` *n* `unknown field` *`'name'`*

DNS のエラーメッセージです。たいていこれは、DNS `named.boot` ファイルのスペルエラーを示します。たとえば、「`primary`」または「`secondary`」のスペルが間違っている場合があります。

servername `named [`*nnnn*`]:` *servername* `has CNAME and other data (illegal)`

DNSのエラーメッセージです。たいていこれは、`machine` *servername* に対する CNAME レコードの構文エラーまたは誤用を示します。

servername `named [`*nnnn*`]:` *domainname* `Line` *n*`: Database format error (`*n.n.n.n*`)`

DNS のエラーメッセージです。IP アドレスが *n.n.n.n* になっているドメイン *name1* 内のマシンに対するリソースレコードで、型がない (通常は `IN`) か、その他の構文エラーがあります。

servername `named [`*nnnn*`]: Line` *n* `Unknown type:` *n.n.n.n*`.`

DNS のエラーメッセージです。IP アドレスが *n.n.n.n* になっているマシンに対する DNS のホストファイルリソースレコードが、型 (通常は `IN`) を含んでいません。

servername `named [`*nnnn*`]: secondary zone` *zonename* `expired.`

DNS のエラーメッセージです。673ページの「サーバーが失敗してゾーンが問題を期限切れにした」を参照してください。

servername named [*nnnn*]: zoneref: Masters for secondary zone *zonename* unreachable

DNSのエラーメッセージです。673ページの「サーバーが失敗してゾーンが問題を期限切れにした」を参照してください。

name in use

FNS のエラーメッセージです。「指定された名前が、コンテキスト中ですでに使用されている」ということを意味します。

name not found

FNS のエラーメッセージです。指定された名前が見つからないということを意味します。

Name not served by this server

指定された名前へのサービスを提供していないサーバーに、要求を行いました。通常、このメッセージは発生しないはずです。しかし、内蔵のサーバー検索メカニズムを使用しない場合は、独自のメカニズムが壊れた結果、発生する可能性もあります。

他に考えられる原因は次のとおりです。

- コールドスタートファイルが壊れています。/var/nis/NIS_COLD_START を削除して、再起動します。
- ローカルキャッシュの期限切れのような、キャッシュの問題。nis_cachemgr と /var/nis/NIS_SHARED_DIRCACHE のプロセスを終了し、次に再起動します。ルートディレクトリ以外の場所で問題が発生した場合は、単純にドメインのキャッシュマネージャのプロセスを終了し、コマンドをもう一度実行します。
- 誰かが複製サーバーから、このディレクトリを削除しました。

このメッセージは、NIS+ のエラーコード定数 NIS_NOT_ME によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

Named object is not searchable

テーブル名を NIS+ のオブジェクトとして解決しましたが、オブジェクトが検索できませんでした。

このメッセージは、NIS+ のエラーコード定数 NIS_NOTSEARCHABLE によって生成されます (詳細な情報は、nis_tables(3N) のマニュアルページを参照)。

Name/entry isn't unique

複数のエントリを返す特定の検索基準に基づく動作が要求されました。たとえば、nistbladm –r を使用して、passwd テーブルからユーザーを削除しようとしました。しかし、このテーブルには、そのユーザーに対応するエントリが 2 つ存在したため、次のメッセージが表示されます。

```
mymachine# nistbladm -r [name=arnold],passwd.org_dir
Can't remove entry: Name/entry isn't unique
```

–r ではなく –R オプションを使用すると、コマンドを複数のエントリに適用できます。たとえば、arnold に対応するすべてのエントリを削除するには、次のように入力します。

```
mymachine#  nistbladm -R [name=arnold],passwd.org_dir
```

NIS make terminated

問題によって、操作が完了する前に、NIS の make 操作が中断しました。NIS の make ファイルで問題と構文エラーをチェックしてください。

NIS: server not responding for domain *domainname*. Still trying

661ページの「NIS の問題と対策」を参照してください。

NIS+ error

「NIS+ サーバーがエラーを返したが、passwd コマンドがエラーの種類を特定できなかった」という場合に使用されるエラーメッセージです。

NisDirCacheEntry:write: xdr_directory_obj failed

最も可能性のある原因は次のとおりです。システムへのメモリー割り当てに失敗しました。メモリーの問題の詳細は、656ページの「メモリーの不足」を参照してください。システムがメモリー不足を起こしていないと思われる場合は、ご購入先にご連絡ください。

NIS+ operation failed

この一般的なエラーメッセージは、めったに発生しないはずです。通常はシステムが修正することのできる、ソフトウェアの小さな問題が発生したことを意味しています。これらのメッセージが頻繁に表示される場合や、システムにより修正されないと思われる場合は、ご購入先にご連絡ください。

このメッセージは、NIS+ のエラーコード定数 NIS_FAIL によって生成されます。

string: NIS+ server busy try again later.

考えられる原因については、651ページの「NIS+ の性能の低下とシステムのハングアップの問題」を参照してください。

NIS+ server busy try again later.

サーバーは現在ビジーです。後でコマンドをもう一度実行してください。

考えられる原因については、651ページの「NIS+ の性能の低下とシステムのハングアップの問題」を参照してください。

NIS+ server for *string* not responding still trying

考えられる原因については、651ページの「NIS+ の性能の低下とシステムのハングアップの問題」を参照してください。

NIS+ server not responding

考えられる原因については、651ページの「NIS+ の性能の低下とシステムのハングアップの問題」を参照してください。

NIS+ server needs to be checkpointed. Use nisping -C*domainname*

注意 - すぐにチェックポイントを設定してください。

サーバーのシステムログが LOG_CRIT のレベルにある場合は、このメッセージが生成されます。これは、ログが大きくなりすぎたことを示します。nisping –C *domainname* を使用してチェックポイントを実行し、ログを切り捨てます。

ログのサイズに関する詳細な情報は、627ページの「ログが大きくなりすぎた」を参照してください。

NIS+ servers unreachable

このソフトエラーは、指定したテーブルオブジェクトが置かれているディレクトリにサービスを提供するサーバーに到達できなかったことを示します。ネッ

トワークの障害やサーバーのクラッシュが起こった場合に、このメッセージが発生する可能性があります。もう一度実行するだけで、成功する可能性もあります。nis_tables(3N) と nis_names(3N) のマニュアルページの –HARD_LOOKUP フラグの説明を参照してください。

このメッセージは、NIS+ のエラーコード定数 NIS_NaMEUNREACHABLE によって生成されます。

`NIS+ service is unavailable or not installed`

NIS+ のサービスが利用できないか、インストールされていません。このメッセージは、NIS+ のエラーコード定数 NIS_UNAVAIL によって生成されます。

`NIS+: write ColdStart File: xdr_directory_obj failed`

最も可能性のある原因は次のとおりです。

- パラメータが正しくありません。直前に入力したコマンドの構文が正確かどうかチェックします。
- システムへのメモリーの割り当てに失敗しました。メモリーの問題の詳細は、656ページの「メモリーの不足」を参照してください。
- コマンドの構文が正確で、システムもメモリー不足を起こしていないと思われる場合は、ご購入先にご連絡ください。

`nis_checkpoint_svc: readonly child instructed to checkpoint ignored.`

このメッセージは、読み取り専用プロセスが、親プロセスに限定されている操作を試みて、その操作が異常終了したことを示す状態メッセージです。何か操作を行う必要はありません。

`nis_dumplog_svc: readonly child called to dump log, ignore`

このメッセージは、読み取り専用プロセスが、親プロセスに限定されている操作を試みて、その操作が異常終了したことを示す状態メッセージです。何か操作を行う必要はありません。

`nis_dump_svc: load limit reached.`

システムで許可されている子プロセスの最大数に達しました。

`nis_dump_svc: one replica is already resyncing.`

マスターとの同期を実行できる複製は、1 度に 1 つだけです。後でコマンドをもう 1 度実行します。

nis_dump で始まる、これら 3 つのメッセージに関する詳細な情報は、659ページの「複製サーバーの更新のエラー」を参照してください。

`nis_dump_svc: Unable to fork a process.`

フォークを行うシステムコールが障害を起こしました。考えられる原因については、`fork(2)` のマニュアルページを参照してください。

`nis_mkdir_svc: readonly child called to mkdir, ignored`

このメッセージは、読み取り専用プロセスが、親プロセスに限定されている操作を試みて、その操作が異常終了したことを示す状態メッセージです。なんらかの操作を行う必要はありません。

`nis_ping_svc: readonly child was pung ignored.`

このメッセージは、読み取り専用プロセスが、親プロセスに限定されている操作を試みて、その操作が異常終了したことを示す状態メッセージです。なんらかの操作を行う必要はありません。

`nis_rmdir_svc: readonly child called to rmdir, ignored`

このメッセージは、読み取り専用プロセスが、親プロセスに限定されている操作を試みて、その操作が異常終了したことを示す状態メッセージです。なんらかの操作を行う必要はありません。

`nisaddcred: no password entry for uid` *userid* `nisaddcred: unable to create credential.`

これら 2 つのメッセージが発生するのは、`nispopulate` スクリプトを実行しているときです。NIS+ のコマンド `nisaddcred` は、リモートドメイン上で、ユーザー ID `userid` に LOCAL の資格を割り当てることに失敗しました。このメッセージは、リモートドメインで `passwd` テーブルを生成しているときにだけ発生します。

問題を解決するには、ローカルの `passwd` テーブルに、次のテーブルパスを追加します。

```
# nistbladm -u -p passwd.org_dir.  remote-domain passwd.org_dir
```

remote-domain は、nispopulate を実行したときに –d オプションで指定したドメインと同じものでなければなりません。このスクリプトをもう一度実行して、passwd テーブルを生成します。

No file space on server

サーバーにファイル領域がありません。

このメッセージは、NIS+ のエラーコード定数 NIS_NOFILESPACE によって生成されます。

No match

最も可能性の高い原因は、インデックスつきの名前を指定するために角括弧を使用し、そのためにシェルが解釈できなくなったことです。たとえば、インデックスつきの名前の両側を角括弧で囲む場合は、全体をさらに引用符で囲まない限り、シェルは角括弧を解釈することができません。そのため、次のメッセージが表示されます。

```
# nistbladm -m shell=/bin/csh [name=miyoko],passwd.org_dir
 No match
```

正しい構文は次のとおりです。

```
# nistbladm -m shell=/bin/csh `[name=miyoko],passwd.org_dir`
```

No memory

システムのメモリーが不足しているため、指定された操作を行うことができません。メモリー不足が原因となる問題の詳細は、655ページの「NIS+ のシステムリソースの問題」を参照してください。

Non NIS+ namespace encountered

名前を完全に解決できませんでした。通常は、関数に渡された名前を解決した結果、NIS+ のネームツリーの中にない名前空間が得られたことを示します。つまり、この名前は登録されていないディレクトリの中にあります。このメッセージが発生した場合、エラーとともに、DIRECTORY タイプの NIS+ オブジェクトが返されます。

このメッセージは、NIS+ のエラーコード定数 NIS_FOREIGNNS によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

No password entry for uid *userid* No password entry found for uid *userid*

これらのメッセージは両方とも、ユーザーの資格の作成や追加を試みたときに、passwd テーブルの中にそのユーザーが見つからなかったことを示します。資格の作成や追加を行う前に、passwd テーブルの中にユーザーを追加しておかなければなりません。

- 最も可能性のある原因は、コマンド行でユーザーの *userid* のスペルミスをしたことです。コマンド行の構文が正確か、またミススペルがないかチェックします。
- 正しいドメインにいるか、またはコマンド行で正しいドメインを指定しているかチェックします。
- コマンド行が正しい場合は、passwd テーブルをチェックして、入力した *userid* のユーザーが存在するかどうか確認します。次のように nismatch を使用します。

```
mymachine# nismatch uid=userid passwd.org_dir.
```

passwd テーブルの中にユーザーが含まれていない場合は、資格を作成する前に、nistbladm か nisaddent を使用して、passwd テーブルにユーザーを追加します。

no permission

FNS のエラーメッセージです。アクセス制御の問題により、動作が失敗したことを意味します。677ページの「「no permission」というメッセージが表示される (FNS)」、および 638ページの「アクセス権がない」を参照してください。

No shadow password information

「制御に必要な情報が欠けているため、パスワードの有効期間の設定が行えなかった」ということを意味します。

no such attribute

FNS のエラーメッセージです。オブジェクトが、指定の属性を持たないことを意味します。

`no supported address`

FNS のエラーメッセージです。「/usr/lib/fn ディレクトリに、(FNS の名前に結びつけられたリファレンス中にはいくつかのタイプのアドレスがあるが) どのタイプの共用ライブラリも存在しない」ということを意味します。共用ライブラリの名前は、`fn_ctx_`*address_type*`.so` という形になります。*address_type* の部分はアドレスのタイプによって異なります。リンクは通常、「`fn_ctx_`*address_type*`.so` から `fn_ctx_`*address_type*`.so.1` へ」という形で行われます。

たとえば、リファレンスのアドレスのタイプが `onc_fn_nisplus` だとすると、共用ライブラリの存在するパスは `/usr/lib/fn/fn_ctx_onc_fn_nisplus.so` ということになります。

`not a context`

FNS のエラーメッセージです。「リファレンスが、正しいコンテキストに対応していない」ということを意味します。

`Not found` *String* `Not found`

Names context - 名前空間の中に、指定された名前が存在しません。

Table context - 検索基準に一致するエントリが、テーブルの中に存在しません。検索基準が null (すべてのエントリを返す) の場合は、このテーブルが空であり、安全に削除できることを意味します。

`–FOLLOW_PATH` フラグが設定されている場合にこのエラーが発生するのは、検索基準に一致するエントリが、パス内のどのテーブルにも含まれていないことを意味します。

このメッセージは、NIS+ のエラーコード定数 `NIS_NOTFOUND` によって生成されます。詳細な情報は、`nis_tables(3N)` と `nis_names(3N)` のマニュアルページを参照してください。

このタイプの問題に関する一般的な情報は、633ページの「NIS+ オブジェクトが見つからない問題」を参照してください。

`Not Found no such name`

このエラーメッセージは、テーブルオブジェクトが置かれているはずの、指定されたディレクトリが存在しないことを示します。サーバーは、自らサービスを提供するテーブルの親サーバーであるはずですが、このテーブルがどのディレクトリに置かれているか認識していません。

このメッセージは、NIS+ のエラーコード定数 NIS_NOSUCHNAME によって生成されます (詳細な情報は、nis_names(3N) と nis_names(3N) のマニュアルページを参照してください)。

このタイプの問題に関する一般的な情報は、633ページの「NIS+ オブジェクトが見つからない問題」を参照してください。

Not master server for this domain

このメッセージは、複製サーバー上のデータベースを直接更新しようとしたことを意味します。

また、ネームサービスを提供するサーバーに変更の要求が行われたものの、そのサーバーがマスターサーバーでなかったことを意味することもあります。ディレクトリオブジェクトが変更され、その結果新しいマスターサーバーが指定された場合に、これが発生します。/var/nis/NIS_SHARED_DIRCACHE ファイル内に、ディレクトリオブジェクトのキャッシュ済みコピーを保持しているクライアントは、ps を実行して nis_cachemgr のプロセス ID を確認し、nis_cachemgr のプロセスを終了させ、/var/nis/NIS_SHARED_DICACHE ファイルを削除して、nis_cachemgr を再度起動します。

このメッセージは、NIS+ のエラーコード定数 NIS_NOTMASTER によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

Not owner

オブジェクトの所有者だけに認められている操作を試みましたが、あなたは所有者ではありません。

このメッセージは、NIS+ のエラーコード定数 NIS_NOTOWNER によって生成されます。

operation not supported

FNS のエラーメッセージです。「行われた操作が、コンテキストによってサポートされていない」ということを意味します。サポートされていない操作の例としては、組織の破壊などがあります。

Object with same name exists

既に存在する名前を追加しようとしました。この名前を追加するには、最初に既存の名前を削除して新しい名前を追加するか、既存のオブジェクトの名前を変更します。

このメッセージは、NIS+ のエラーコード定数 NIS_NAMEEXISTS によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

parse error: *string* (key *variable*)

nisaddent コマンドが /etc ディレクトリ内のデータベースファイルを使用しているときに、ファイルのエントリにエラーが見つかると、このメッセージが表示されます。最初の *string* は問題について説明し、key の右側の *variable* は、障害のあるエントリを示します。/etc/passwd ファイルで問題が起こった場合は、/usr/sbin/pwck を使用して、このファイルをチェックします。

partial result returned

FNS のエラーメッセージです。操作によって得られた結果が不完全であることを意味します。

Partial Success

「要求の受信は正しく行われたが、対応するエントリがなかった」という点を除き、NIS_NOTFOUND とほぼ同じです。

このエラーが発生すると、サーバーは、エントリではなくテーブルオブジェクトのコピーを返します。パスの処理や、その他のローカルポリシーの実装をクライアントが行えるようにするためです。

このエラーメッセージは、NIS+ のエラーコード定数 NIS_PARTIAL によって生成されます。詳細は、nis_tables(3N) のマニュアルページを参照してください。

Passed object is not the same object on server

「オブジェクトを名前空間から削除しようとしたが、削除の対象となるオブジェクトとは別のオブジェクトが要求の中で渡されたために処理が異常終了した」ということを意味します。

このエラーメッセージは、NIS+ のエラーコード定数 NIS_NOTSAMEOBJ によって生成されます。詳細は、nis_tables(3N)、nis_names(3N) のマニュアルページを参照してください。

Password does not decrypt secret key for *name*

考えられる原因は次のとおりです。

- 正しくないパスワードを入力しました。
- cred テーブルの中に、*name* のエントリがありません。
- NIS+ が鍵を復号できませんでした (エントリが壊れている可能性があります)。
- Secure RPC パスワードとログインパスワードが一致していません。
- nsswitch.conf ファイルが、cred テーブルの中に記録されている NIS+ パスワードとは異なるパスワードを使用して、/etc/passwd ファイルの中に格納されているローカルパスワードの照会を行なっています。実際に暗号化されているパスワードは、ローカルの /etc/shadow ファイルに格納されています。

このタイプの問題の診断と解決に関する情報は、640ページの「NIS+ のセキュリティの問題」を参照してください。

`Password has not aged enough`

「パスワードが新しすぎるため、変更ができない」ということを意味します。パスワードは、作成されてから一定の日数 (*N* 日) が経過するまでは変更ができません。詳細は、210ページの「パスワードの変更」を参照してください。

`Permission denied`

ある操作を試みましたが、必要なアクセス権が割り当てられていなかったため、このエラーが発生しました。アクセス権の問題に関する詳細は、637ページの「NIS+ の所有権とアクセス権の問題」を参照してください。

このエラーメッセージは、ログイン、パスワードの問題、あるいは NIS+ のセキュリティの問題に関係があります。発生原因として最も一般的なのは、「ユーザーのパスワードが管理者によってロックされている」あるいは「ユーザーのアカウントが終了している」ということです。第 11 章、および付録 A を参照してください。

`Permissions on the password database may be too restrictive`

「NIS+ テーブル中のパスワードフィールドの内容を参照する (あるいは使用する) 権限を、ユーザーが持っていない」ということを意味します。詳細は第 10 章を参照してください。

`Please notify your System Administrator`

passwd コマンドでパスワード情報を更新しようとして表示された場合は、「何らかの理由 (数多く考えられる) で成功しなかった」ということを意味します。成功しない理由としては、「サービスが利用できない」、「必要なサーバーが停止している」、「アクセス権がないといった類の問題がある」などがあげられます。セキュリティの問題については、640ページの「NIS+ のセキュリティの問題」に様々なタイプの例があげられています。

`Please check your /etc/nsswitch.conf file`

「nsswitch.conf ファイルで、パスワード更新についてサポートされていない設定が行われている」ということを意味します。どのような設定がサポートされているかは、214ページの「nsswitch.conf ファイルの必要条件」を参照してください。

`Probable success`

Name context - 要求は成功しました。しかし、返されたオブジェクトはオブジェクトキャッシュからのものであり、サーバーから直接返されたものではありません。オブジェクトキャッシュからオブジェクトを取り出したくない場合は、検索用関数をコールするときに、–NO_CACHE フラグを指定しなければなりません。

Table context - 要求は成功しました。しかし、検索パスの中でテーブルを見つけることができなかったので、テーブルがアクセス可能だった場合と比べると、得られた結果は違っている可能性があります。

このメッセージは、NIS+ のエラーコード定数 NIS_S_SUCCESS によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

`Probably not found`

指定されたエントリは、このテーブルの中に存在しません。しかし、パス内のすべてのテーブルが検索されたわけではないので、それらのテーブルの中に、このエントリが含まれている可能性もあります。

このメッセージは、NIS+ のエラーコード定数 NIS_S_NOTFOUND によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

`Query illegal for named table`

要求された構造体をクライアントのライブラリに渡すときに、問題が検出されました。

このメッセージは、NIS+ のエラーコード定数 NIS_BADREQUEST によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

Reason: can't communicate with ypbind.

661ページの「NIS の問題と対策」を参照してください。

replica_update: Child process attempting update, aborted

このメッセージは、読み取り専用プロセスが更新を試みて、それが失敗したことを示す状態メッセージです。

replica_update: error result was string

このメッセージは、複製サーバーへのダンプを行う際に問題が起こったことを示します (*string* は理由を示します)。詳細な情報は、659ページの「複製サーバーの更新のエラー」を参照してください。

replica_update: error result was Master server busy, full dump rescheduled replica_update: master server busy rescheduling the resync. replica_update: master server is busy will try later. replica_update: nis dump result Master server busy, full dump rescheduled

これらのメッセージはどれも、サーバーがビジーなので、ダンプが後で行われることを示します。

replica_update: nis dump result nis_perror *errorstring*

このメッセージは、複製サーバーへのダンプを行う際に問題が起こったことを示します (*error string* は理由を示します)。詳細な情報は、659ページの「複製サーバーの更新のエラー」を参照してください。

replica_update: *nnnn* updates *nnnn* errors

更新に成功したことを示す状態メッセージです。

replica_update: WARNING: last_update (*directoryname*) returned 0!

NIS+ のプロセスは、このディレクトリに対応するトランザクションログが最後に更新されたタイムスタンプを見つけることができませんでした。この結果、シ

ステムは、問題の起こったディレクトリについて完全な同期を実行しなければなりません。

`Results Sent to callback proc`

これは状態メッセージです。なんらかの操作を行う必要はありません。

このメッセージは、NIS+ のエラーコード定数 `NIS_CBRESULTS` によって生成されます。詳細な情報は、`nis_tables(3N)` のマニュアルページを参照してください。

`root_replica_update: update failed` *string*`: could not fetch object from master.`

このメッセージは、複製サーバーへのダンプを行う際に問題が起こったことを示します。詳細な情報は、659ページの「複製サーバーの更新のエラー」を参照してください。

`RPC failure: ``RPC failure on yp operation.`

このメッセージは、NIS クライアントの `nsswitch.conf` ファイルが、`nis` ではなく `files` に設定され、サーバーが `/etc/hosts` ファイルまたは `/etc/inet/ipnodes` ファイルに含まれていないときに、`ypcat` によって戻されます。

`Security exception on local system. UNABLE TO MAKE REQUEST.`

ユーザーのログイン ID がマシン名と重複したときに、このメッセージが表示されることがあります。詳細な情報は、638ページの「ユーザーのログインがマシン名と同じ」を参照してください。

date: hostname: `sendmail (`*nnnn*`) : gethostbyaddr failed`

この問題の一般的な原因の 1 つに、0 を前に付けた状態での NIS+、NIS、ファイル、DNS データセットの IP アドレスへの入力があります。たとえば、`151.029.066.001` のような IP アドレスは入力しないでください。このアドレスの正確な入力方法は、`151.29.66.1` です。

`Server busy, try again`

サーバーがビジーのため、要求をすぐに処理できません。

- 追加、削除、変更の操作を行なったときにこのメッセージが返されるのは、あるディレクトリのサービスを提供しているマスターサーバーが使用できないか、データベースのチェックポイントが実行中であることを意味します。
- サーバーが内部の状態を更新しているときに、このメッセージが返されることもあります。
- nis_list を実行した場合、クライアントがコールバックを指定し、サーバーがリソース不足でコールバックを行えないことも考えられます。

サーバーが使用可能になるのを待って、コマンドをもう一度実行します。

このメッセージは、NIS+ のエラーコード定数 NIS_TRYAGAIN によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

`Server out of memory`

ほとんどの場合、このメッセージは致命的な結果が発生したことを示します。サーバーがヒープ領域を使い果たしたことを意味します。

このメッセージは、NIS+ のエラーコード定数 NIS_NOMEMORY によって生成されます。詳細な情報は、nis_tables(3N) と nis_names(3N) のマニュアルページを参照してください。

`Sorry`

ログイン、またはパスワードの変更が拒否されたときに表示されるメッセージです。拒否の理由が表示されないのは、「表示された理由を読むことで、権利のない人がシステムに不正にアクセスできるようになる可能性がある」というセキュリティ上の理由からです。

`Sorry: less than` *nn* `days since the last change`

「パスワードが新しすぎるため、変更ができない」ということを意味します。パスワードは、作成されてから一定の日数 (*N* 日) が経過するまでは変更ができません。詳細は、210ページの「パスワードの変更」を参照してください。

`Success`

(1) 要求は成功しました。このメッセージは、NIS+ のエラーコード定数 NIS_SUCCESS によって生成されます。詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

(2) FNS のエラーメッセージ。操作が成功しました。

`_svcauth_des: bad nickname`

クライアントから受け取ったニックネームが不適切か、壊れています。原因は、おそらくネットワークの混雑です。このメッセージの重大度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは情報を示すだけです。レベルが高い場合は、このコマンドをもう一度実行しなければならない可能性があります。

`_svcauth_des: corrupted window from` *principalname*

送信されたウィンドウが、ベリファイアによって送信されたウィンドウと一致しません。

このメッセージの重要度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは主に情報を示すために表示されます。レベルが高い場合は、このコマンドをもう一度実行しなければならない可能性もありますし、次の説明に従って、問題を解決するための操作をしてください。

考えられる原因は次のとおりです。

- サーバーの鍵の組が変更されています。クライアントはサーバーの古い公開鍵を使いましたが、サーバーは `keyserv` によってキャッシュされている新しい秘密鍵を使いました。クライアントとサーバーの両方で `keylogin` を実行してください。
- クライアントの鍵の組が変更されています。クライアントシステム上で `keylogin` が実行されていないので、システムは依然としてクライアントの古い秘密鍵をサーバーに送信しています。しかし、サーバーはクライアントの新しい公開鍵を使っています。その結果、2 つの鍵は一致しません。クライアントとサーバーの両方で、`keylogin` をもう一度実行してください。
- ネットワークでデータが壊れました。コマンドをもう一度実行します。それでも問題が解決しない場合は、`snoop` コマンドを使用して、ネットワークに関係する問題の調査と解決を行います。次に、クライアントとサーバーの両方で、`keylogin` をもう一度実行します。

`_svcauth_des: decryption failure`

特定の認証データの DES 復号に失敗しました。考えられる原因は次のとおりです。

- ライブラリ関数か引数が壊れています。

- DES 暗号化チップを使用している場合、そのチップに問題があります。

このメッセージの重要度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは単に情報を示すために表示されます。レベルが高い場合は、ご購入先にご連絡ください。DES 暗号化チップに問題があると思われる場合も、ご購入先にご連絡ください。

`_svcauth_des: corrupted window from` *principalname*

送られたウィンドウが、ベリファイアで送られたものと一致しません。

このメッセージの重要度は、実行しているセキュリティのレベルによって異なります。低いセキュリティレベルでは、このメッセージは主に参考のための情報です。高いセキュリティレベルでは、しばらくたってからコマンドを再実行するか、もしくは次に説明するような修正作業を行う必要があります。

考えられる原因

- サーバーの鍵の組が変更されています。クライアントが、サーバーの古い公開鍵を使って、その一方でサーバーが `keyserv` でキャッシュされた新しい非公開鍵を持っています。`keylogin` をクライアントとサーバーの両方で実行してください。
- クライアントの鍵の組が変更されて、クライアントがクライアントシステムで `keylogin` を実行していません。このためシステムはまだ、クライアントの古い非公開鍵をサーバーに送付しており、そのサーバーはクライアントの新しい公開鍵を使用しています。このため、2 つの鍵は一致しません。`keylogin` を再度クライアントとサーバーの両方で実行してください。
- ネットワークでデータが壊れました。コマンドをもう一度実行してください。これで機能しない場合には、`snoop` コマンドを使って、ネットワークの問題を調査、修正してください。その後で、再度クライアントとサーバーの両方で `keylogin` を実行してください。

`_svcauth_des: decryption failure for` *principalname*

特定の認証データの DES 復号化に失敗しました。考えられる原因は次のとおりです。

- ライブラリ関数か引数が壊れています。
- DES 暗号化チップを使用している場合、そのチップに問題があります。

このメッセージの重大度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは単に情報を示すために

表示されます。レベルが高い場合は、ご購入先にご連絡ください。問題が、DES 暗号化チップに関連するように思われる場合は、ご購入先に連絡してください。

`_svcauth_des: invalid timestamp received from` *principalname*

クライアントから受け取ったタイムスタンプが壊れています。または、サーバーが間違った鍵を使って、復号しようとしています。考えられる原因は次のとおりです。

- ネットワークの混雑。コマンドをもう一度実行します。
- サーバーのキャッシュから、このクライアント用のエントリが削除されました。ネットワークの負荷をチェックします。

`_svcauth_des: key_decryptsessionkey failed for` *principalname*

`keyserv` プロセスは、特定の公開鍵を使用してセッションキーを復号化することに失敗しました。考えられる原因は次のとおりです。

- `keyserv` プロセスが終了しているか、応答しません。`ps –ef` を使用して、ホスト上で `keyserv` プロセスが動作しているかどうか確認します。動作していない場合は、`keyserv` を再度起動し、`keylogin` を実行します。
- サーバー主体のキーログインが行われていません。サーバー主体のキーログインを実行します。
- サーバー主体 (ホスト) に資格が割り当てられていません。クライアントのホームドメインにある `cred` テーブルを対象にして、`nismatch` *hostname*`.`*domainname*`. cred.org_dir` を実行します。必要に応じて、新しい資格を作成します。
- `keyserv` は再度起動されているようですが、長時間にわたって動作するアプリケーション `rpc.nisd`、`sendmail`、`automountd` なども再度起動する必要があります。
- DES 暗号化に失敗しました。ご購入先にご連絡ください。

`_svcauth_des: no public key for` *principalname*

サーバーがクライアントの公開鍵を取得できません。考えられる原因は次のとおりです。

- 主体に公開鍵がありません。主体のホームドメインにある `cred` テーブルを対象に、`niscat` を実行します。テーブルの中で、この主体に DES の資格が割り当てられていない場合は、`nisaddcred` を使用して資格を作成し、その主体で `keylogin` を実行します。

- nsswitch.conf ファイルで指定されたネームサービスが応答しません。

_svcauth_des: replayed credential from *principalname*

サーバーは要求を受け取り、キャッシュの中でそのクライアント名に一致するエントリを見つけましたが、受け取った要求の会話鍵のタイムスタンプが、キャッシュの中に格納されているタイムスタンプより古いものでした。

このメッセージの重大度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは主に情報を示すために表示されます。レベルが高い場合は、次の説明に従って問題を解決するための操作を行なってください。

考えられる原因は次のとおりです。

- クライアントとサーバーそれぞれのクロックが同期していません。rdate を使用して、クライアントのクロックをサーバーのクロックに同期させます。
- サーバーがランダムな順序で要求を受け取っています。マルチスレッドアプリケーションを使用している場合に、これが発生します。アプリケーションが tcp をサポートしている場合、/etc/netconfig (または NETPATH 環境変数) を tcp に設定します。

_svcauth_des: timestamp is earlier than the one previously seen from *principalname*

クライアントからいくつかのコールを受け取りましたが、後から受け取ったコールが、最初のコールより古いタイムスタンプを示しています。このメッセージの重大度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは主に情報を示すために表示されます。レベルが高い場合は、次の説明に従って問題を解決するための操作を行なってください。

考えられる原因は次のとおりです。

- クライアントとサーバーそれぞれのクロックが同期していません。rdate を使用して、クライアントのクロックをサーバーのクロックに同期させます。
- サーバーのキャッシュから、このクライアント用のエントリが削除されました。サーバーはキャッシュの中に、現在のクライアントの情報を維持しています。キャッシュのサイズは、64 個のクライアントに相当します。

_svcauth_des: timestamp expired for *principalname*

ウィンドウを受信する際、デフォルトでは 35 秒以内の間隔が規定されていますが、クライアントから受け取ったタイムスタンプが、この範囲に収まっていません。このメッセージの重大度は、有効にしているセキュリティのレベルに依存します。セキュリティレベルが低い場合は、このメッセージは主に情報を示すために表示されます。レベルが高い場合は、次の説明に従って問題を解決するための操作を行なってください。

考えられる原因は次のとおりです。

- サーバーやネットワークが低速なので、35 秒のウィンドウでは間隔が不足しています。
- クライアントとサーバーそれぞれのクロックがかなりずれているため、ウィンドウの間隔に収まりません。`rdate` を使用して、クライアントのクロックをサーバーのクロックに同期させます。
- サーバーのキャッシュから、このクライアント用のエントリが削除されました。操作をもう一度実行します。

`syntax not supported`

FNS のエラーメッセージです。「サポートされていないタイプの構文が使用された」ということを意味します。

`Too Many Attributes`

サーバーに渡された検索基準が、検索可能なテーブルの列より多くの属性を持っています。

このメッセージは、NIS+ のエラーコード定数 `NIS_TOOMANYATTRS` によって生成されます。詳細な情報は、`nis_tables(3N)` のマニュアルページを参照してください。

`too many attribute values`

FNS のエラーメッセージです。「1 つの属性に、ネーミングシステムでサポートされている範囲を超える数の値を持たせようとした」ということを意味します。

`Too many failures - try later`

`Too many tries; try again later`

どちらもログインあるいはパスワードの変更に関連するメッセージです。「ログイン、またはパスワードの変更のやり直しの回数が多すぎる、あるいは時間が

かかりすぎる」ということを意味します。詳細は、209ページの「Login incorrect メッセージ」または、211ページの「パスワード変更の失敗」を参照してください。

`Unable to authenticate NIS+ client`

サーバーがクライアントのコールバック要求を実行したときに、RPC の `clnt_call()` から `RPC_AUTHERR` という状態を受け取ると、このメッセージが生成されます。通常これは、古い認証情報が原因となっています。システムが、更新されていないキャッシュから取り出したデータを使用した場合や、認証情報が最近変更され、その変更結果がサーバーに伝わっていない場合に、このような古い認証情報が発生します。ほとんどの場合、この問題は短時間のうちに自動的に解決されます。

この問題が自動的に解決されない場合は、次のいずれかの問題があると考えられます。

- `/var/nis/NIS_SHARED_DIRCACHE` ファイルが壊れています。キャッシュマネージャのプロセスを終了させ、このファイルを削除し、キャッシュマネージャを再度起動します。
- `/var/nis/NIS_COLD_START` ファイルが壊れています。このファイルを削除し、`nisinit` を実行して、このファイルを再度作成します。
- `/etc/.rootkey` ファイルが壊れています。`keylogin –r` を実行します。

このメッセージは、NIS+ のエラーコード定数 `NIS_CLNTAUTH` によって生成されます。

`Unable to authenticate NIS+ server`

ほとんどの場合、システムが容易に修正することのできる、ソフトウェアの小さなエラーが発生したことを意味しています。サーバーが RPC の `clnt_call` から `RPC_AUTHERR` という状態を受け取ると、このメッセージが生成されます。

この問題が自動的に解決されない場合は、`/var/nis/NIS_COLD_START` ファイル、`/var/nis/NIS_SHARED_DIRCACHE` ファイル、`/etc/.rootkey` ファイルのどれかが壊れていることが考えられます。

このメッセージは、NIS+ のエラーコード定数 `NIS_SRVAUTH` によって生成されます。

`Unable to bind to master server for name '`*`string`*`'`

このタイプの問題に関する情報は、633ページの「NIS+ オブジェクトが見つからない問題」の「オブジェクトが見つからない問題」を参照してください。この特別なメッセージは、/etc/defaultdomain ファイルの中にあるサーバーのドメイン名の右端にピリオドを追加したときに発生することがあります。

`Unable to create callback.`

サーバーは、ユーザーのマシンのコールバックサービスと通信できませんでした。この結果、データを返すことができませんでした。

詳細な情報は、nis_tables(3N) のマニュアルページを参照してください。

`Unable to create process on server`

NIS+ のサービスルーチンが、サポートしていない手続き番号を指定する要求を受け取ると、このエラーが生成されます。

このメッセージは、NIS+ のエラーコード定数 NIS_NOPROC によって生成されます。

string: `Unable to decrypt secret key for` *string*.

考えられる原因は次のとおりです。

- 正しくないパスワードを入力しました。
- cred テーブルの中に、*name* のエントリがありません。
- NIS+ が鍵を復号化できませんでした。おそらく、エントリが壊れている可能性があります。
- nsswitch.conf ファイルが、cred テーブルの中に記録されている NIS+ パスワードとは異なるパスワードを使用して、/etc/passwd ファイルの中に格納されているローカルパスワードの照会を行なっています。

このタイプの問題の診断と解決に関する情報は、640ページの「NIS+ のセキュリティの問題」を参照してください。

`unavailable`

FNS のエラーメッセージです。操作に必要なネームサービスが利用できないということを意味します。

`Unknown error`

NIS+ のエラー処理ルーチンが、未登録のタイプのエラーを受け取ったときに、このメッセージが表示されます。

`Unknown object`

返されたオブジェクトのタイプが、未登録のタイプです。

このメッセージは、NIS+ のエラーコード定数 `NIS_UNKNOWNOBJ` によって生成されます。詳細な情報は、`nis_tables(3N)` と `nis_names(3N)` のマニュアルページを参照してください。

`update_directory: nnnn objects still running.`

複製を更新するときに、サーバーがディレクトリを更新している間、表示されるメッセージです。なんらかの操作を行う必要はありません。

`User principalname needs Secure RPC credentials to login but has none.`

キーログインが成功しなかったことを示します。この問題は一般に、`/etc/shadow` とリモートの NIS+ パスワードテーブルとで記憶しているパスワードが異なっているために発生します。

`Warning: couldn't reencrypt secret key for principalname`

この問題の原因として最も可能性が高いのは「Secure RPC パスワードがログインパスワードと異なっていて (またはローカルの `/etc/shadow` とリモートの NIS+ パスワードテーブルとで記憶しているパスワードが異なっていて)、まだ明示的に `keylogin` を実行していない」ということです。詳細は、648ページの「/etc/passwd ファイルの中にある NIS+ のパスワードとログインパスワード」、および、649ページの「Secure RPC パスワードとログインパスワードが異なる」を参照してください。

`WARNING: db::checkpoint: could not dump database: No such file or directory`

このメッセージは、チェックポイント実行の際に、システムがデータベースファイルをオープンできなかったことを示しています。

考えられる原因は次のとおりです。

- データベースファイルが削除されています。
- サーバーがファイル記述子を使い果たしました。

- ディスクに問題があります。
- ユーザーまたはホストに、必要なアクセス権が割り当てられていません。

`WARNING: db_dictionary::add_table: could not initialize database from scheme`

データベーステーブルを初期設定できませんでした。考えられる原因は次のとおりです。

- システムのリソースに問題があります。655ページの「NIS+ のシステムリソースの問題」を参照してください。
- コマンド構文での新しいテーブルの指定が正しくありませんでした。
- データベースが壊れています。

`WARNING: db_query::db_query:bad index`

ほとんどの場合、このメッセージは、インデックス付きの名前の指定が正しくないことを示しています。指定されたテーブルの中に、インデックス付きの名前が存在するかどうか確認します。コマンドのスペルと構文にエラーがないかチェックします。

`**WARNING: domain` *domainname* `already exists.`

このメッセージは、作成しようとしたドメインがすでに存在することを意味しています。

- ルート以外のマスターサーバーを新しく変換しようとしている場合や、直前に起こった `nisserver` の問題を回復しようとしている場合は、このスクリプトを続けて実行します。
- *domainname* のスペルが正しくない場合は、正しいドメイン名を指定して、このスクリプトをもう一度実行します。

`**WARNING: failed to add new member` *NIS+_principle* `into the` *groupname* `group. You will need to add this member manually: 1. /usr/sbin/nisgrpadm -a groupname NIS+_principal`

NIS+ のコマンド `nisgrpadm` が、NIS+ のグループ *groupname* に新しいメンバーを追加することに失敗しました。次のように入力して、NIS+ のこの主体を追加します。

```
# /usr/sbin/nisgrpadm -a      groupname  NIS+_principal
```

**WARNING: failed to populate *tablename* table.

nisaddent コマンドは、NIS+ の tablename テーブルをロードできませんでした。通常、この警告メッセージの前に、詳細なエラーメッセージが表示されます。

**WARNING: hostname specified will not be used. It will use the local hostname instead.

このメッセージは、–H オプションを指定して、リモートホスト名を入力したことを示します。nisserver –r スクリプトを使って、ルートマスターサーバーと同様の方法でリモートマシンを構成できません。

- ローカルマシンを NIS+ のルートマスターサーバーに変換したい場合は、別に操作を行う必要はありません。nisserver –r スクリプトは、入力されたホスト名を無視します。
- リモートホスト (ローカルマシンではなく) を NIS+ のルートマスターサーバーに変換したい場合は、このスクリプトを終了します。リモートホストを対象にして、nisserver –r を再度実行します。

**WARNING: *hostname* is already a server for this domain. If you choose to continue with the script, it will try to replicate the groups_dir and org_dir directories for this domain.

このメッセージは、複製サーバーを作成しようとしたドメインの中で、すでに *hostname* が複製サーバーとなっていることを警告します。

- nisserver によって以前に発生した問題を解決するためにこのスクリプトを実行している場合は、スクリプトを続けて実行します。
- *hostname* を間違って入力した場合は、正しいホスト名を指定して、このスクリプトをもう一度実行します。

**WARNING: *alias-hostname* is an alias name for host *canonical_hostname*. You cannot create credential for host alias.

このメッセージは、nisclient –c の名前リストに、ホストの別名を入力したことを示します。ホストの別名に資格を作成すべきではないので、標準的なホスト名にその資格を作成するかどうか、スクリプトが尋ねています。

**WARNING: file *directory-path/tablename* does not exist! *tablename* table will not be loaded.

このスクリプトは、*tablename* の入力ファイルを見つけることができませんでした。

- *directory-path*/*tablename* が間違って入力されている場合は、正しいテーブル名を指定して、このスクリプトをもう一度実行します。
- *directory-path*/*tablename* が存在しない場合は、適切なデータを使ってファイルを作成し更新します。次に、このスクリプトを再度実行して、このテーブルを生成します。

**WARNING: NIS auto.master map conversion failed. auto.master table will not be loaded.

auto.master テーブル内のすべてのピリオドを下線に変換しているときに、auto.master マップの変換に失敗しました。別の NIS サーバーを使って、このスクリプトをもう一度実行します。

**WARNING: NIS netgroup map conversion failed. netgroup table will not be loaded.

netgroup マップの中で NIS のドメイン名を NIS+ のドメイン名に変換しているときに、netgroup マップの変換に失敗しました。別の NIS サーバーを使って、このスクリプトをもう一度実行します。

**WARNING: nisupdkeys failed on directory *domainname*. This script will not be able to continue. Please remove the *domainname* directory using 'nisrmdir'.

NIS+ のコマンド nisupdkeys は、リスト指定されたディレクトリオブジェクト内の鍵を更新することに失敗しました。新しいドメインにサービスを提供することになっている新しいマスターサーバー上で rpc.nisd が動作していない場合は、rpc.nisd を再度起動します。次に、nisrmdir を使用して、*domainname* ディレクトリを削除します。最後に、nisserver を再度実行します。

WARNING: nisupdkeys failed on directory *directory-name* You will need to run nisupdkeys manually: 1. /usr/lib/nis/nisupdkeys *directory-name*

NIS+ のコマンド nisupdkeys は、リスト指定されたディレクトリオブジェクト内の鍵を更新することに失敗しました。次のように入力して、ディレクトリオブジェクト内の鍵を手作業で更新します。

```
# /usr/lib/nis/nisupdkeys    directory-name
```

**WARNING: once this script is executed, you will not be able to restore the existing NIS+ server environment. However, you can restore your NIS+ client environment using ‘‘nisclient -r’’ with the proper domainname and server information. Use ‘‘nisclient -r’’ to restore your NIS+ client environment.

少なくとも 1 台の NIS+ サーバーを設定する前にこのスクリプトを実行すると、これらのメッセージが表示されます。これらのメッセージは、スクリプトの実行を続行すると、NIS+ に関係したファイルが削除され、必要に応じて再度作成されることを示しています。

- NIS+ のファイルを削除してもよい場合は、このスクリプトを続けて実行します。
- NIS+ のファイルを保存するには、Do you want to continue? プロンプトで n と入力し、このスクリプトを終了します。次に、NIS+ のファイルを別のディレクトリに保存し、このスクリプトをもう一度実行します。

**WARNING: this script removes directories and files related to NIS+ under /var/nis directory with the exception of the NIS_COLD_START and NIS_SHARED_DIRCACHE files which will be renamed to <file>.no_nisplus. If you want to save these files, you should abort from this script now to save these files first.

すぐ上の「WARNING: once this script is executed,...」のメッセージを参照してください。

**WARNING: you must specify the NIS domainname.

このメッセージは、プロンプトで NIS のドメイン名を入力しなかったことを示しています。プロンプトで、NIS サーバーのドメイン名を入力します。

**WARNING: you must specify the NIS server *hostname*. Please try again.

このメッセージは、プロンプトで NIS のサーバーのホスト名を入力しなかったことを示しています。プロンプトで、NIS サーバーのホスト名を入力します。

Window verifier mismatch

これは、_svcauth_des コードによって生成されたデバッグ用メッセージです。鍵がキャッシュから削除されたため、ベリファイアが無効になっている可能性

があります。このメッセージが生成されると、_svcauth_des は、AUTH_BADCRED の状態を返します。

`You (`*string*`) do not have Secure RPC credentials in NIS+ domain '`*string*`'`

コマンドを実行するために必要とされる資格が割り当てられていないサーバーで nispasswd を実行すると、このメッセージが生成されることがあります。セキュリティレベル 0 で動作しているサーバーは、資格の作成や維持を行わないことに注意してください。

資格、所有権、アクセス権の問題に関する詳細は、637ページの「NIS+ の所有権とアクセス権の問題」を参照してください。

`You may not change this password`

パスワードの変更が管理者によって禁止されていることを意味します。

`You may not use nisplus repository`

「コマンド行で –r nisplus が使用されたが、NIS+ パスワードテーブルに適切なエントリがない」ということを意味します。パスワードテーブルに必要なエントリがあるかどうか確認してください。また nsswitch.conf ファイルに nisplus というエントリを追加してみてください。

`Your password has been expired for too long`

`Your password is expired`

どちらもパスワードが作成されてからの時間に関連のあるメッセージです。「パスワードが古すぎるので、すぐに変更する必要がある」ということを意味します。詳細は、209ページの「password expired メッセージ」を参照してください。

`Your password will expire in `*nn*` days`

`Your password will expire within 24 hours`

どちらもパスワードが作成されてからの時間に関連のあるメッセージです。「パスワードがまもなく無効になるので、すぐに変更する必要がある」ということを意味します。詳細は、210ページの「will expire メッセージ」を参照してください。

`Your specified repository is not defined in the nsswitch file!`

「-r オプションでリポジトリのパスワード情報を更新しようとしたが、該当するパスワードのリポジトリが nsswitch.conf ファイルのパスワードエントリにない」という意味の警告メッセージです。使用したコマンド自体は機能し、-r フラグで指定したリポジトリへの変更はすべて行われます。しかし変更の対象となっているリポジトリが nsswitch.conf ファイルで指定されているものではないため、該当するリポジトリを nsswitch.conf ファイルで指定するまでこの変更の影響はまったく表れません。

たとえば nsswitch.conf ファイルのパスワードエントリに files nis という指定があるとき、以下のコマンドを実行してパスワードの有効期限を決定するとします。

```
passwd -r nisplus
```

この場合は、files nis という指定のある nsswitch.conf ファイルを使用している限り、コマンドによる影響はまったく表れません。

verify_table_exists: cannot create table for *string* nis_perror *message*.

テーブルを対象にして操作を行う場合、NIS+ は最初にテーブルが存在するかどうかを確認します。テーブルが存在しない場合は、NIS+ はそのテーブルを作成しようとします。作成できない場合は、このエラーメッセージを返します。*string* は、検索や作成ができなかったテーブルを示します。nis_perror *message* の部分は、問題の原因に関係する情報を示します。メッセージのこの部分を、独立したメッセージと考えて、この付録にある説明を参照できます。このタイプの問題の考えられる原因は次のとおりです。

- サーバーはディレクトリの複製として追加されるだけであり、ディレクトリオブジェクトを持つことはできません。nisping -C により、チェックポイントを実行します。
- ディスク容量を使い果たしました。656ページの「ディスク容量の不足」を参照してください。
- データベースが壊れています。
- その他のタイプのソフトウェアエラー。ご購入先にご連絡ください。

ypcat: can't bind to NIS server for domain *domainname*. Reason: can't communicate with ypbind.

661ページの「NIS の問題と対策」を参照してください。

`yppoll: can't get any map parameter.`

661ページの「NIS の問題と対策」を参照してください。

付録 C

NIS+ テーブルの情報

この付録には Solaris 8 リリース内のデフォルトの NIS+ テーブルに格納されている情報がまとめられています。NIS+ テーブルとそれを管理するコマンドの概要については第 14 章を参照してください。

NIS+ テーブル

NIS+ 環境では、ほとんどの名前空間の情報は、NIS+ テーブルに格納されます。

ネームサービスがないと、ほとんどのネットワーク情報は /etc ファイルに格納され、ほとんどすべての NIS+ テーブルが対応する /etc ファイルを持ちます。NIS サービスでは、/etc ファイルにほとんど対応する NIS マップにネットワーク情報を格納しました。

注 - この章では、NIS+ の一部として配布されたものだけを説明します。ユーザーとアプリケーション開発者は、目的に応じて NIS+ との互換性があるテーブルを頻繁に作成します。ユーザーと開発者によって作成されたテーブルの詳細は、マニュアルを参照してください。

すべての NIS+ テーブルは、groups_dir ディレクトリオブジェクトに格納される admin とグループテーブルを除いて、ドメインの org_dir NIS+ ディレクトリオブジェクトに格納されます。

注 - テーブルエントリはリンクしないでください。テーブルは他のテーブルにリンクされますが、あるテーブルのエントリを別のテーブルのエントリにリンクしないでください。

NIS+ テーブルと他のネームサービス

Solaris の環境では、ネームサービスのスイッチファイル (nsswitch.conf) によって、1 つ以上のソースを異なる名前空間の情報に対して指定できます。NIS+ テーブルの他に、ソースは NIS マップ、DNS ゾーンファイル、/etc テーブルにすることができます。これらをスイッチファイルで指定する順番によって、異なるソースから情報が組み合わされる方法が決まります。スイッチファイルの詳細は 第 2 章を参照してください。

NIS+ テーブル入力ファイルフォーマット

これらテーブルのどれかに対して入力ファイルを作成している場合、ほとんどのテーブルは、次の 2 つのフォーマットの条件を共有します。

- エントリごとに 1 行を使う
- 1 つ以上のスペースまたはタブで列を分ける

特定のテーブルに、異なるまたは追加のフォーマットの条件がある場合には、「入力ファイルフォーマット」の項で説明します。

auto_home テーブル

auto_home テーブルは間接オートマウンタマップです。このマップによって NIS+ クライアントは、ドメイン内の任意のユーザーのホームディレクトリをマウントできます。このテーブルは、各ユーザーのホームディレクトリのマウントポイント、各ホームディレクトリの位置と、マウントオプションがあればそれを指定してマウントします。これは間接マップであるため、マウントポイントの最初の部分は auto_master テーブルに指定され、デフォルトでは /home となります。マウントポイントの 2 番目の部分 (つまり、/home の下にあるサブディレクトリ) は auto_home マップ内のエントリによって指定され、ユーザーごとに異なります。

auto_home テーブルには 2 つの列があります。

表 C–1 auto_home テーブル

列	内容	説明
キー	マウント先	ドメイン内のユーザーのログイン名
値	オプションと位置	ユーザーごとのマウントオプションがあればそれと、ユーザーのホームディレクトリの位置

たとえば、次のようになります。

```
costas barcelona:/export/partition2/costas
```

ユーザー costas のホームディレクトリは、サーバー barcelona 上にあり、ディレクトリ /export/partition2/costas 内にあります。またこのホームディレクトリは、クライアントの /home/costas ディレクトリの下にマウントされます。このエントリにはマウントオプションはありません。

auto_master テーブル

auto_master テーブルは、ドメイン内のすべてのオートマウンタマップを含みます。直接マップの場合、auto_master テーブルは、単にマップ名を提供するだけです。間接マップの場合、このテーブルは、そのマウントポイントの先頭ディレクトリとマップ名の両方を提供します。auto_master テーブルには次の 2 つの列があります。

表 C–2 auto_master テーブル

列	内容	説明
キー	マウント先	マップがマウントされる先頭ディレクトリ。マップが直接マップの場合、これは /--- で表されるダミーディレクトリである
値	マップ名	オートマウンタマップの名前

たとえば、auto_master テーブルに次のエントリがあるとします。

```
/home auto_home
/-auto_man
/programs auto_programs
```

最初のエントリは auto_home マップを指定します。つまり、auto_homeマップ内の全エントリに対するマウントポイントの先頭ディレクトリ /home を指定します (auto_home マップは間接マップ)。2 番目のエントリは auto_man マップを指定します。このマップは直接マップであるため、このエントリはマップ名だけを与えます。auto_man マップは、それ自体で、その各エントリに対するマウントポイントの完全パス名だけではなく、最上位ディレクトリも提供します。3 番目のエントリは auto_programs マップを指定します。このエントリはマウントポイントの先頭ディレクトリを与えるため、auto_programs マップは間接マップです。

すべてのオートマウンタマップは NIS+ テーブルとして格納されます。デフォルトでは、Solaris 環境は必須の auto_master マップ、および非常に便利な auto_home マップを提供します。

1 つのドメインに対してさらに多くのオートマウンタマップを作成できますが、これらは必ず NIS+ テーブルとして格納し、auto_master テーブルに登録してくだ

さい。その他のオートマウントマップを作成して、auto_master (ユーザーに対して作成された) に追加するときは、列名は「キー」と「値」にする必要があります。オートマウンタの詳細は、オートマウンタ、または NFS ファイルシステムの関連マニュアルを参照してください。

bootparams テーブル

bootparams テーブルは、ドメイン内のすべてのディスクレスワークステーションに関する構成情報をもっています。ディスクレスワークステーションは、ネットワークに接続されていてもハードディスクをもたないワークステーションです。ディスクレスワークステーションにはディスク装置がないため、自分のファイルとプログラムをネットワーク上のサーバーのファイルシステムに持っています。また、自分の構成情報 (つまり「ブートパラメタ」) をサーバーに持っています。

このような構成になっているため、すべてのディスクレスワークステーションには、この情報がどこに格納されているかを知っている初期設定プログラムがあります。ネットワークにネームサービスがない場合、このプログラムはサーバーの /etc/bootparams ファイル内でこの情報を探します。ネットワークが NIS+ ネームサービスを使う場合、このプログラムは、代わりに bootparams テーブル内でこの情報を探します。

bootparams テーブルは、ディスクレスワークステーションに関するどんな構成情報でも格納できます。このテーブルには 2 つの列があり、構成キーとその値を格納します。デフォルトでは、このテーブルは各ワークステーションのルート、スワップ、およびダンプの各パーティションの位置を格納するように設定されます。

デフォルトの bootparams テーブルには 2 つの列しかありませんが、列を使用して、次に示す情報を提供します。

表 C–3 bootparams テーブル

列	内容	説明
キー	ホスト名	ディスクレスワークステーションの正式ホスト名、hosts テーブルで指定
値	構成	ルートパーティション : ワークステーションのルートパーティションの位置 (サーバー名とパス)

表 C–3 bootparams テーブル 続く

列	内容	説明
		スワップパーティション : ワークステーションのスワップパーティションの位置 (サーバー名とパス)
		ダンプパーティション : ワークステーションのダンプパーティションの位置 (サーバー名とパス)
		インストールパーティション
		ドメイン

「入力ファイルのフォーマット」

列はタブ文字で区切ります。バックスラッシュ (\) は、1 つのエントリを複数の行で指定するのに使用します。ルート、スワップ、およびダンプの各パーティション用のエントリのフォーマットを次に示します。

```
client-name root= server:path \
swap=server:path \
dump=server:path \
install= server:path \
domain=domainname
```

次に例を示します。

```
buckarooroot=bigriver:/export/root1/buckaroo \
 swap=bigriver:/export/swap1/buckaroo \
 dump=bigriver:/export/dump/buckaroo \
 install=bigriver:/export/install/buckaroo \
 domain=sales.doc.com
```

x86 ベースのワークステーションでは、使用できるパラメータが増えます。詳細は、bootparams(4) のマニュアルページを参照してください。

client_info テーブル

client_info テーブルは、それが存在するドメインのサーバー設定を格納するために使われるオプションの内部 NIS+ テーブルです。このテーブルは、nisprefadm コマンドで作成および保管されます。

注意 - nisprefadm だけを使って、このテーブルで作業してください。このテーブルでは、他の NIS+ コマンドは使わないでください。

cred テーブル

NIS+ 主体の資格に関する情報を格納するテーブルです。ドメインにつき 1 つ存在し、ドメインに属するクライアントワークステーションおよびワークステーションにログインできるユーザー (つまり、ドメイン中の主体) の資格に関する情報を持っています。cred テーブルはドメイン中の org_dir サブディレクトリにあります。

注 - cred テーブルはリンクしないでください。org_dir ディレクトリにはそれぞれ固有の cred テーブルがあり、他の org_dir ディレクトリの cred テーブルへのリンクを使用できません。

cred テーブルには以下の 5 つの列があります。

表 C–4 cred テーブル

NIS+ 主体名	認証タイプ	認証名	公開データ	非公開データ
主体ユーザー名	LOCAL	UID	グループ ID リスト	
主体ユーザー名またはワークステーション名	DES	Secure RPC ネット名	公開鍵	暗号化非公開鍵

2 番目の「認証タイプ」という列の値により、他の 4 つの列の値のタイプが決定されます。

- 「LOCAL」

  認証タイプが LOCAL の場合、残りの列には主体ユーザー名、UID、GID が入ります (最後の列は空白)。

- 「DES」

  認証タイプが DES の場合、残りの列には主体名、Secure RPC ネット名、公開鍵、暗号化非公開鍵が入ります。これらの鍵は他の情報と組み合わされ、DES 資格の暗号化および復号化に使用されます。

資格および `cred` テーブルの詳細は、第 7 章を参照してください。

ethers テーブル

`ethers` テーブルは、インターネット上のワークステーションの 48 ビット Ethernet アドレスに関する情報を格納しています。次の 3 列があります。

表 C–5 `ethers` テーブル

列	内容	説明
アドレス	Ethernet アドレス	ワークステーションの 48 ビット Ethernet アドレス
名前	公式なホスト名	ワークステーションの名前、`hosts` テーブルで指定
コメント	コメント	エントリに関するコメント (必要に応じて)

Ethernet アドレスの形式は次のとおりです。

n:*n*:*n*:*n*:*n*:*n* *hostname*

ここで *n* は 0 ～ FF の 16 進数で、1 バイトを表します。このアドレスのバイト列は常に最上位のバイトから始まるネットワーク順になっています。

group テーブル

group テーブルは、UNIX ユーザーグループについての情報を格納します。group テーブルには 4 つの列があります。

表 C–6　group テーブル

列	説明
名前	グループ名
パスワード	グループのパスワード
グループ ID	グループの数値 ID
メンバー	グループメンバー名、コンマで区切る

Solaris の以前のリリースでは、ローカル /etc/group ファイルで +/- 構文を使用して、NIS グループマップ内のエントリを結合または上書きしました。Solaris 環境ではネームサービススイッチファイルを使用してワークステーションの情報ソースを指定するため、これはもはや不要です。Solaris 2.x システムで必要なのは、クライアントの /etc/nsswitch.conf ファイルを編集して files を指定し、そのあとグループ情報のソースとして nisplus を指定することです。これによって、group テーブルの内容がクライアントの /etc/group ファイルの内容に効果的に追加されます。

hosts テーブル

hosts テーブルは、ドメイン内の全ワークステーション名とそれらの IP アドレスを関連付けます。ワークステーションは、通常は NIS+ クライアントでもありますが、必ずしもその必要はありません。bootparams、group、および netgroup などのほかのテーブルは、このテーブルに収められたネットワーク名に依存しています。これらのテーブルは、ネットワーク名を使用して、ホームディレクトリやグループメンバー情報などのほかの属性を個々のワークステーションに割り当てます。hosts テーブルには 4 つの列があります。

表 C–7 hosts テーブル

列	説明
アドレス	ワークステーションの IP アドレス (ネットワーク番号とワークステーションの ID 番号)
正式名	ワークステーションの正式名
名前	ワークステーションを識別するために、ホスト名の代わりに使われる任意の名前
コメント	エントリに関するコメント (必要に応じて)

mail_aliases テーブル

mail_aliasesテーブルは、sendmail によって認識されるドメインのメール別名を含んでいます。これには 4 つの列があります。

表 C–8 mail_aliases テーブル

列	説明
別名	別名の名前
メンバーリスト	この別名に送信されたメールを受信するメンバーが収められているリスト。メンバーはユーザー、ワークステーション、またはほかの別名とすることができる
コメント	エントリに関するコメント (必要に応じて)
オプション	(マニュアルページを参照)

「入力ファイルのフォーマット」

各エントリのフォーマットは次のとおりです。

```
alias-name:member[, member]...
```

1 つのエントリが複数の行にまたがるときには、バックスラッシュを使用します。

netgroup テーブル

netgroup テーブルは、リモートマウント、ログインとシェルのアクセス権をチェックするために使用される、ネットワーク全域にわたるグループを定義します。リモートマウントに使用されるネットグループのメンバーはワークステーションです。リモートログインとシェルの場合、これらのメンバーはユーザーです。

注 - サービスを提供している NIS+ サーバーが互換モードで動作している場合、クライアントマシンのユーザーは netgroup テーブルに対して ypcat を実行できません。実行すると、エントリの有無に関わらず「テーブルが空である」という結果が出ます。

---

netgroup テーブルには 6 つの列があります。

表 C–9 netgroup テーブル

列	コメント	説明
名前	グループ名	ネットワークグループ名
グループ	グループ名	このグループを構成するグループ名
ホスト	ホスト名	ホスト名
ユーザー	ユーザー名	ユーザーのログイン名
ドメイン	ドメイン名	ドメイン名
コメント	コメント	エントリに関するコメント (必要に応じて)

「入力ファイルのフォーマット」

入力ファイルは、1 つのグループ名と任意の数のメンバーから構成されます。

```
groupname  member-list  . . .
```

メンバー指定は、他のネットグループ名、または 3 つのフィールドを順序正しく並べたものとすることができます。ネットグループ名、3 つのフィールドを両方指定することもできます。

```
member-list::=groupname  | (hostname, username , domainname)
```

最初のフィールドには、グループに属するワークステーション名を指定します。2 番目のフィールドに、グループに属するユーザー名を指定します。3 番目のフィールドには、メンバー指定が有効となっているドメインを指定します。

指定のないフィールドは、ワイルドカードを示します。たとえば、次の netgroup には、すべてのドメインのユーザーとすべてのワークステーションが含まれます。

```
everybody ( , , )
```

フィールド内のダッシュは、ワイルドカードの反対で、このグループに所属するワークステーションやユーザーがないことを示します。次に 2 つの例を示します。

```
(host1, -,doc.com.) (-,joe,doc.com.)
```

最初の指定では、1 台のワークステーション host1 を doc.com. ドメインに入れますが、すべてのユーザーを排除します。2 番目の指定では、1 人のユーザーを doc.com. ドメインに入れますが、すべてのワークステーションを排除します。

netmasks テーブル

netmasks テーブルには、標準のインターネットサブネットに使われるネットワークマスクが収められています。このテーブルには 3 つの列があります。

表 C–10 netmasks テーブル

列	説明
アドレス	ネットワークの IP 番号
マスク	ネットワーク上で使うネットワークマスク
コメント	エントリに関するコメント (必要に応じて)

ネットワーク番号に関しては、ワークステーションアドレスで使用される通常の IP ドット表記を使用できますが、ワークステーションアドレスの代わりにゼロを残します。たとえば、

```
128.32.0.0 255.255.255.0
```

というエントリでは、クラス B ネットワーク 128.32.0.0 が、サブネットフィールドに 24 ビット、ホストフィールドに 8 ビットを持つことを意味します。

networks テーブル

networks テーブルにはインターネットのネットワークが含まれます。このテーブルは、Network Information Control Center (NIC) で管理される公式のネットワークテーブルから作成されるのが普通ですが、これにローカルネットワークを追加しなければならないこともあります。このテーブルには 3 つの列があります。

表 C–11 networks テーブル

列	説明
正式名	ネットワークの正式名、インターネットが提供
アドレス	ネットワークの公式 IP 番号
名前	ネットワークの非公式名
コメント	エントリに関するコメント (必要に応じて)

passwd テーブル

passwd テーブルには、ドメイン内のユーザーのアカウントに関する情報が入っています。これらのユーザーは、一般的には NIS+ 主体ですが、必ずしもその必要はありません。ただし、ユーザーが NIS+ 主体である場合、それらの資格はここには収められず、ドメインの cred テーブルに格納されることに注意してください。passwd テーブルは、その他 (または未認証) に対して通常読み取り権を与えます。

注 - スーパーユーザー (ユーザー ID = 0) のエントリをこのテーブルに入れることはできません。ルートのパスワード情報は、/etc ディレクトリ上のファイルに格納してください。

passwd テーブル内の情報は、ユーザーのアカウントが作成されたときに追加されます。

passwd テーブルには次の列があります。

表 C–12 passwd テーブル

列	説明
名前	ユーザーのログイン名であり、ユーザーのアカウントが作成されたときに割り当てられる。この名前は大文字を含む必要はない。最大 8 文字まで
パスワード	ユーザーの暗号化されたパスワード
ユーザー ID	ユーザーの ID 番号であり、ユーザーのアカウントが作成されたときに割り当てられる
グループ ID	ユーザーのデフォルトグループの ID 番号
GCOS	ユーザーの実名と、ユーザーがメールメッセージ見出しの「From:」フィールドに入れたい情報。この列が「&」の場合、単にユーザーのログイン名を使用する
ホーム	ユーザーのホームディレクトリのパス名
シェル	ユーザーの初期シェルプログラム。デフォルトは B シェル : /usr/bin/sh
シャドウ	(表 C–13 を参照)

passwd テーブルには、さらにシャドウ列があります。この列には、ユーザーアカウントに関して、次に示すような制限情報が格納されています。

表 C–13　passwd テーブルのシャドウ列

項目	項目
最終変更日	1970 年 1 月 1 日からパスワードの最終変更日までの日数
最小値	推奨されるパスワード変更間隔の最小日数
最大値	パスワードが有効な最大日数
警告	ユーザーパスワードが期限切れになる前にユーザーが警告を受ける日数
間隔	ユーザーに許される休止日数
期限	ユーザーのアカウントが無効となる絶対日付
未使用	将来のために獲保。現在は 0 に設定されている

Solaris の以前のリリースでは、ローカル /etc/passwd ファイル内で +/- 構文を使って、NIS パスワードマップ内のエントリを統合または上書きしました。Solaris 2.x 環境ではネームサービススイッチを使ってワークステーションの情報ソースを指定するため、これはもはや不要です。Solaris 2.x システムで必要なのは、クライアントの /etc/nsswitch.conf ファイルを編集して files を指定し、それに続けて passwd 情報のソースとして nisplus を指定することです。これによって、passwd テーブルの内容が /etc/passwd ファイルの内容に効果的に追加されます。

しかし、それでも +/- 方式を使用したい場合は、クライアントの nsswitch.conf ファイルを編集します。NIS を使用しているなら passwd ソースに compat を指定し、NIS+ を使用しているなら passwd_compat: nisplus を追加してください。

protocols テーブル

protocols テーブルにはインターネットで使われるプロトコルが含まれます。これには 4 つの列があります。

表 C–14 protocols テーブル

列	説明
正式名	プロトコル名
名前	プロトコルの識別に使われる非公式な別名
番号	プロトコルの番号
コメント	プロトコルに関するコメント

rpc テーブル

rpc テーブルには、RPC プログラム名が含まれます。これには 4 つの列があります。

表 C–15 rpc テーブル

列	説明
正式名	プログラム名
名前	プログラムの起動に使用できる別の名前
番号	プログラムの番号
コメント	RPC プログラムに関するコメント

rpc テーブルの入力ファイルの例を次に示します。

```
#
# rpc file
#
rpcbind 100000 portmap sunrpc portmapper
rusersd 100002 rusers
nfs 100003 nfsprog
mountd 100005 mount showmount
walld 100008 rwall shutdown
```

(続く)

```
sprayd 100012 spray
llockmgr 100020
nlockmgr 100021
status 100024
bootparam 100026
keyserv 100029 keyserver
nisd 100300 rpc.nisd
#
```

services テーブル

services テーブルは、インターネット上で使用できるインターネットサービスに関する情報を格納しています。これには 5 つの列があります。

表 C–16 services テーブル

列	説明
正式名	サービスの公式インターネット名
名前	サービスを要求できる代替名のリスト
プロトコル	サービスを提供するためのプロトコル (たとえば 512/tcp)
ポート	ポート番号
コメント	サービスに関するコメント

timezone テーブル

timezone テーブルは、ドメイン内の全ワークステーションのデフォルトの時間帯を含んでいます。デフォルトの時間帯はインストール中に使用されますが、インストーラがこれを無効にすることができます。このテーブルには 3 つの列があります。

表 C–17　timezone テーブル

列	説明
ドメイン名	ドメイン名
タイムゾーン名	時間帯の名前 (たとえば US/Pacific)
コメント	時間帯に関するコメント

付録 **D**

FNS リファレンスの書式と構文

この付録では、XFN リファレンスにおける、DNS 文書 (TXT) レコードと X.500 属性の使用方法についての補足情報を述べます。

XFN リファレンス用 DNS 文書レコードの書式

Solaris の環境は、DNS 内で広域ネーミングシステムをフェデレートさせるための XFN 規格に準拠しています。DNS でネーミングシステムをフェデレートさせるには、DNS TXT リソースレコードに情報を入力する必要があります。この情報は、従属ネーミングシステム用に XFN リファレンスを作成するために使用されます。この付録では、これらの DNS TXT レコードの書式について説明します。

- DNS をフェデレートさせるのに必要な手順については、第 26 章を参照してください。
- DNS 内のレコードを操作する一般的な方法の詳細は、『*DNS and BIND*』(Paul Albitz & Cricket Liu 著 浅羽登志也／上水流由香 監訳、アスキー出版局、1995年) を参照してください。

XFN リファレンスのリファレンスタイプは、`XFNREF` というタグで始まる TXT レコードから作成します。書式は以下の通りです。

```
TXT "XFNREF rformat   reftype"
```

TXT の後に続く文字列の中にスペースが存在する場合、そのスペースを削除するか、文字列全体を引用符 (“ ”)で囲む必要があります。XFNREF、*rformat*、*reftype* という 3 つのフィールドは、スペース (スペースとタブ) で区切ります。*rformat* は、リファレンスタイプの識別子の書式を指定します。種類は次のとおりです。

- STRING – FN_ID_STRING 書式のマップ
- OID – FN_ID_ISO_OID_STRING 書式のマップ
- UUID – FN_ID_DCE_UUID 書式のマップ

reftype は、リファレンスタイプの識別子の内容を指定します。

XFNREF レコードが存在しない場合、リファレンスタイプはデフォルト設定により FN_ID_STRING を持つ XFN_SERVICE 識別子になります。2 つ以上の XFNREF TXT レコードが存在する場合、どのレコードが処理されるかは不定です。以下の TXT レコードはデフォルト設定の XFNREF と同じ働きをします。

```
TXT "XFNREF STRING XFN_SERVICE"
```

XFN リファレンスのアドレス情報は、XFN 文字列を接頭語にしたタグを持つ TXT レコードを使用して作成します。1 つのリファレンスに複数のアドレスを指定することもできます。同じタグを持つレコードはグループにまとめられ、グループごとにハンドラに渡されます。各ハンドラは渡された TXT レコードからアドレス (複数あるいは、ない場合もある) を作成し、リファレンスに付加します。XFNREF タグの場合は特別で、リファレンスタイプを作成するためにだけ使用されるため、アドレス作成の過程からは除外されます。

TXT レコードのアドレスを指定する構文は次のとおりです。

```
XFNaddress_type_tag      address_specific_data
```

XFN_address_type_tag と *address_specific_data* という 2 つのフィールドは、スペース (スペースとタブ) で区切ります。*address_type_tag* は、*address_specific_data* に使用するハンドラを指定します。

TXT レコードには 1 レコードにつき 2 K バイトという文字の制限があります。特定のアドレスのデータが長すぎて 1 つの TXT レコードに格納できない場合は、以下のように複数の TXT レコードを使用できます。

```
TXT "XFNaddress_type_tag  address_specific_data1"
TXT "XFNaddress_type_tag  address_specific_data2"
```

特定のタグのハンドラが呼び出され、両方のデータが渡されます。ハンドラはこれら 2 行を解釈する順番を決定します。

TXT レコードの順番はあまり重要ではありません。異なるタグを持つ行がある場合、特定のタグのハンドラが呼び出される前に、同じタグを持つ行はグループにまとめられます。以下の例では、*tag1* のハンドラは 2 つの文書行で呼び出され、*tag2* のハンドラは 3 つの文書行で呼出されます。

```
TXT "XFNtag1 address_specific_data1"
TXT "XFNtag2 address_specific_data2"
TXT "XFNtag1 address_specific_data3"
TXT "XFNtag2 address_specific_data4"
TXT "XFNtag2 address_specific_data5"
```

XFN リファレンスに使用できる TXT レコードの例を以下に示します。

「例 1」

```
TXT "XFNREF STRING XFN_SERVICE"
TXT "XFNNISPLUS doc.com. nismaster 129.144.40.23"
```

「例 2」

```
TXT "XFNREF OID 1.3.22.1.6.1.3"
TXT "XFNDCE (1 fd33328c4-2a4b-11ca-af85-09002b1c89bb...)"
```

以下は、従属ネーミングシステムがバインドされた DNS テーブルの例です。

```
$ORIGIN test.doc.com
@       IN SOA foo root.eng.doc.com          (
                      100    ;; Serial
                      3600   ;; Refresh
                      3600   ;; Retry
                      3600   ;; Expire
                      3600   ;; Minimum
              )
          NS    nshost
          TXT   "XFNREF STRING XFN_SERVICE"
          TXT   "XFNNISPLUS doc.com. nismaster 129.144.40.23"
nshost IN  A 129.144.40.21
```

XFN リファレンス用 X.500 属性の構文

この節では、XFN リファレンス用の X.500 属性の使用についての補足情報を述べます。XFN リファレンスを X.500 における属性として保存できるようにするためには、ディレクトリスキーマを、この付録で定義されているオブジェクトクラスと属性をサポートするように変更する必要があります。

- X.500 を連合させるのに必要な手順については、第 26 章を参照してください。
- X.500 のディレクトリスキーマの変更については、『 *Managing the X.500 Client Toolkit* 』を参照してください。

オブジェクトクラス

XFN リファレンスをサポートするために、XFN と XFN 補助の 2 つの新しいオブジェクトクラスが導入されています。XFN オブジェクトクラスは FNS に適切ではありません。これは、米国 Sun Microsystems, Inc. の X.500 ディレクトリ製品が、新しく導入された複合 ASN.1 構文をサポートしていないためです。その代わりに、FNS では XFN 補助オブジェクトクラスを使用します。

ASN.1 では、次のような 2 つの新しいオブジェクトクラスが定義されています。

```
xFN OBJECT-CLASS ::= {
         SUBCLASS OF          { top }
         KIND               auxiliary
         MAY CONTAIN        { objectReferenceId |
                              objectReference |
                              nNSReferenceId |
                              nNSReference }
          ID                  id-oc-xFN
     }
     id-oc-xFN OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) ansi(840) sun(113536)
        ds-oc-xFN(24)
     }
xFNSupplement OBJECT-CLASS ::= {
         SUBCLASS OF            { top }
         KIND                   auxiliary
         MAY CONTAIN            { objectReferenceString |
                                  nNSReferenceString }
            ID                  id-oc-xFNSupplement
      }
      id-oc-xFNSupplement OBJECT IDENTIFIER ::= {
          iso(1) member-body(2) ansi(840) sun(113536)
```

(続く)

```
        ds-oc-xFNSupplement(25)
    }
```

XFN 補助オブジェクトクラスは、補助オブジェクトクラスとして定義されているため、X.500 のオブジェクトクラスすべてに引き継がれる可能性があります。XFN 補助オブジェクトクラスは、以下の 2 つの任意属性によって定義されます。

- objectReferenceString は、XFN リファレンスをオブジェクト自体で保存するときに使用される
- nNSReferenceString は、XFN リファレンスを次のネーミングシステムで保存するときに使用される

ASN.1 では、この 2 つの属性を以下のように定義しています。

```
    objectReferenceString ATTRIBUTE ::= {
        WITH SYNTAX              OCTET STRING
        EQUALITY MATCHING RULE   octetStringMatch
        SINGLE VALUE             TRUE
        ID                       { id-at-objectReferenceString }
    }
    id-at-objectReferenceString OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) ansi(840) sun(113536)
        ds-at-objectReferenceString(30)
    }
    nNSReferenceString ATTRIBUTE ::= {
        WITH SYNTAX              OCTET STRING
        EQUALITY MATCHING RULE   octetStringMatch
        SINGLE VALUE             TRUE
        ID                       { id-at-nNSReferenceString }
    }
    id-at-nNSReferenceString OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) ansi(840) sun(113536)
        ds-at-nNSReferenceString(31)
    }
```

objectReferenceString と nNSReferenceString は、共に XFN リファレンスを文字列形式で保存します。それぞれのオクテット列構文は、さらに以下の BNF 定義に準拠するように制約を加えられます。

```
<ref>          ::= <id> '$' <ref-addr-set>
<ref-addr-set> ::= <ref-addr> | <ref-addr> '$' <ref-addr-set>
<ref-addr>     ::= <id> '$' <addr-set>
<addr>         ::= <hex-string>
<id>           ::= 'id'   '$' <string> |
                   'uuid' '$' <uuid-string> |
                   'oid'  '$' <oid-string>
<string>       ::= <char> | <char> <string>
<char>         ::= <PCS> | '\' <PCS>
<PCS>          ::= // Portable Character Set:
                   // !"#$%&'()*+,-./0123456789:;<=>?
                   // @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
                   // `abcdefghijklmnopqrstuvwxyz{|}~
<uuid-string>  ::= <uuid-char>  | <uuid-char> <uuid-string>
<uuid-char>    ::= <hex-digit> | '-'
<oid-string>   ::= <oid-char>  | <oid-char> <oid-string>
<oid-char>     ::= <digit> | '.'
<hex-string>   ::= <hex-octet> | <hex-octet> <hex-string>
<hex-octet>    ::= <hex-digit> <hex-digit>
<hex-digit>    ::= <digit> |
                   'a' | 'b' | 'c' | 'd' | 'e' | 'f' |
                   'A' | 'B' | 'C' | 'D' | 'E' | 'F'
<digit>        ::= '0' | '1' | '2' | '3' | '4' | '5' |
                   '6' | '7' | '8' | '9'
```

以下は、文字列形式の XFN リファレンスの例です。

```
id$onc_fn_enterprise$id$onc_fn_nisplus_root$0000000f77697a2e636fd2e2062696762696700
```

この例では、onc_fn_enterprise というタイプの XFN リファレンスを使用しています。この中には、onc_fn_nisplus_root というアドレスタイプと 1 つのアドレス値があります。アドレス値は XDR によって符号化された文字列で、ドメイン名、doc.com の後にホスト名 cygnus が続く形で構成されています。

XFN リファレンスは、fnattr という FNS コマンドを使用して X.500 のエントリに加えることができます。この例では、次のように入力すると、c=us/o=doc という新しいエントリが作成され、オブジェクトクラスの属性と最上位、組織、XFN 補助の値が加えられます。

```
# fnattr -a .../c=us/o=doc object-class top organization xfn-supplement
```

fnbind という FNS コマンドは、NIS+ リファレンスを命名されたエントリにバインドし、X.500 を NIS+ 名前空間のルートにリンクします。fnbind の名前因数内の文字の後にスラッシュ (/) が使用されていることに注意してください。

```
# fnbind -r .../c=us/o=doc/ onc_fn_enterprise onc_fn_nisplus_root "doc.com. cygnus"
```

用語集

BNF
FNS (XFN) の用語。Backus-Naur の頭文字

data encryption standard (DES) (データ暗号化規格**)**
アメリカ商務省標準局によって開発された、データの暗号化と復号化のために一般的に使用される高度なアルゴリズム。

DES
「data encryption standard (DES) (データ暗号化規格)」の項を参照。

DNS
「ドメインネームシステム (DNS)」の項を参照。

DNS ゾーン
ネットワークドメイン内の管理境界であり、1 つまたは複数のサブドメインから構成されるのが普通。

DNS ゾーンファイル
DNS ソフトウェアがドメイン内の全ワークステーションの名前と IP アドレスを格納する一連のマップ。

DNS 転送
NIS サーバーまたは NIS 互換設定の NIS+ サーバーは、自分で応答できない要求を DNS サーバーに転送する。

FNS
「フェデレーテッド・ネーミング・サービス」の項を参照。

GID
「グループ ID」の項を参照。

IP
インターネットプロトコル。インターネットプロトコル体系の「ネットワーク層」プロトコル。

IP アドレス
ネットワーク内の各ホストを識別する一意な番号。

MIS (management information system or service)
経営情報システム

NIS
ネットワーク上のシステムとユーザーについての重要な情報が収められている分散型ネットワーク情報サービス。NIS データベースは、「マスターサーバー」とすべての「スレーブサーバー」に格納されている。

NIS+
ネットワーク上のシステムとユーザーについての階層情報が収められている分散型ネットワーク情報サービス。NIS+ データベースは、「マスターサーバー」とすべての「複製サーバー」に格納されている。

NIS 互換モード
NIS+ の構成の 1 つであり、このモードでは、NIS クライアントは NIS+ テーブルに格納されたデータにアクセスできる。また、NIS+ サーバーは NIS クライアントと NIS+ クライアントからの情報要求に応答できる。

NIS マップ
NIS によって使用されるファイルであり、たとえば、ネットワーク上の全ユーザーのパスワードエントリや、ネットワーク上の全ホストマシンの名前など、特定種類の情報を格納する。NIS サービスの一部であるプログラムは、これらのマップを参照する。「NIS」の項を参照。

NIS+ オブジェクト
NIS+ のドメイン、ディレクトリ、テーブル、グループのこと。「ドメイン」、「ディレクトリ」、「グループ」、「テーブル」の項を参照。

NIS+ 環境
管理上使用する用語。「該当する `nsswitch.conf` ファイルで、`nisplus` が情報源として指定されている」という状況を指す。また「コマンドが、NIS+ 名前空間のオブジェクトに操作をするようなオプションを付けて実行されている」という状況 (例：`passwd -r nisplus`) を指すこともある。

NIS+ 主体
「主体」の項を参照。

NIS+ トランザクションログ
名前空間内のオブジェクトに関する NIS+ テーブル宛のデータ更新が収められたファイル。名前空間内の変更は、複製サーバーに伝達されるまでは、トランザクションログに格納される。トランザク

ションログは、マスターサーバーの複製サーバーがすべて更新された後で、クリアされる。

NNSP (Next naming system pointer) 「次のネーミングシステム参照」の項を参照。

pinging NIS+ データに加えられた変更の内容が、NIS+マスターサーバーから当該ドメインの複製サーバーに送られるプロセス。

RPC 「遠隔手続き呼び出し (RPC)」の項を参照。

Secure RPC パスワード Secure RPC プロトコルにおいて必要となるパスワードのこと。秘密鍵の暗号化に使用されるユーザーのログインパスワードと同じでなければならない。

TCP 「Transport Control Protocol (TCP)」の項を参照。

TCP/IP Transport Control Protocol/Internet Protocol の頭文字。このプロトコル体系は、最初はインターネット用に開発された。「インターネット」プロトコル体系とも呼ばれる。Solaris 8 リリース ネットワークは、デフォルトでは TCP/IP 上で動作する。

Transport Control Protocol (TCP) インターネットプロトコル体系での主要な転送プロトコルであり、高信頼性でコネクション型の全二重ストリームを提供する。配送には IP を使用する。「TCP/IP」の項を参照。

X.500 開放型システム間相互接続 (OSI) 規格で定義されたグローバルディレクトリサービス。

XFN リンク 複合名でアドレスを示す特殊な形式のリファレンス。他の種類のリファレンスと同様に、XFNリンクはコンテキストの原子名と結合される。

アクセス権 NIS+ 主体 (principal) のクラスに割り当てられるアクセス権であり、NIS+ オブジェクト上でこれらのクラスが実行できる動作を決定する。読み取り、変更、作成、または 削除の 4 種類がある。

アプリケーションレベルのネームサービス ファイル、メール、印刷などのサービスを提供するアプリケーションに組み込まれているネームサービスのこと。アプリケーションレベルのネームサービスは、エンタープライズレベルネームサービスの下に位置する。エンタープライズレベルのネームサービスが提供

するコンテキストの中に、アプリケーションレベルネームサービスのコンテキストを組み込むことができる。

暗号化鍵　「データ暗号化鍵」の項を参照。

暗黙的なネームシステムポインタ　FNS (XFN) の用語。他のネームシステムのコンテキストを指す、名前のないリファレンスのこと。

インターネット　一連のルーターによって相互接続された世界規模のネットワークの集まり。これらのネットワークは 1 つの大きな仮想ネットワークとして機能できる。

インターネットアドレス　「TCP/IP」を使用するホストに割り当てられた 32 ビットのアドレス。「ドット表記」の項を参照。

インデックス付き名前　テーブル内のエントリの識別に使用されるネーミング形式。

遠隔手続き呼び出し **(RPC)**　分散コンピューティングのクライアントサーバーモデルを実現する、簡単で一般的なパラダイム。与えられた引数を使用することによって、要求がリモートシステムに送信され、指定された手順が実行される。そして、その結果が呼び出し側に返される。

エンタープライズのルート　エンタープライズのルートコンテキストのこと。エンタープライズの名前空間のルートにあるオブジェクトの名前を管理する。

エンタープライズレベルのネットワーク　「エンタープライズレベル」のネットワークといっても、ケーブルや赤外線ビーム、無線などを利用した LAN (Local Area Network) の場合もあれば、ケーブルや直通電話接続を利用して複数の LAN を結んだものもある。エンタープライズレベルのネットワーク内では、DNS や X.500/LDAP などのグローバルネームサービスを使用せずに、どのマシンからでも任意のマシンにアクセスできる。

エンタープライズレベルのネームサービス　エンタープライズレベルネットワーク内のマシン (ホスト) 名、ユーザー名、ファイル名を管理するサービスのこと。FNS では、組織単位、地理的サイト、アプリケーションサービスにも名前を付けて管理できる。

エントリ　データベーステーブルの中の、一列のデータのこと。

親コンテキスト　あるコンテキストとその兄弟的なコンテキストが存在するコンテキスト。

親ドメイン　「ドメイン」の項を参照。

鍵 (暗号化)　鍵の管理および配布システムの一部として、他の鍵の暗号化と復号化に使用される鍵。「データ暗号化鍵」の項も参照。

キー (列)　NIS+ テーブルエントリのデータは、そのテーブルのキーとは無関係に、どの列からでもアクセスできる。

キーサーバー　秘密鍵を格納する Solaris 8 リリースのプロセス。

キャッシュマネージャ　NIS+ クライアントのローカルキャッシュ (NIS_SHARED_DIRCACHE) を管理するプログラム。これらのクライアントによって最も頻繁に使用されるディレクトリをサポートする NIS+ サーバーについての位置情報 (トランスポートアドレス、認証情報、生存期間など) を格納するために使用される。

逆解決　DNS ソフトウェアを使用して、ワークステーションの IP アドレスをワークステーション名に変換するプロセス。

クライアント　(1) NIS+ では、NIS+ サーバーに対して NIS+ サービスをリクエストする主体 (マシンまたはユーザー) がクライアント。

(2) ファイルシステムのクライアントサーバーモデルでは、計算パワーや大きな記憶容量などの計算サーバーの資源にリモートアクセスするマシンがクライアント。

(3) ウィンドウシステムのクライアントサーバーモデルでは、「サーバープロセス」からウィンドウサービスにアクセスする「アプリケーション」がクライアント。このモデルでは、クライアントとサーバーは同じマシン上または別のマシン上で動作できる。

クライアントサーバーモデル　ネットワークサービスおよびこれらのモデルユーザープロセス (プログラム) を説明する一般的な方法の 1 つ。これらの例としては、「ドメイン名システム (DNS)」のネームサーバー / ネームリゾルバパラダイム、および「NFS」やディスクレスホストなどのファイルサーバー / ファイルクライアント関係がある。「クライアント」の項も参照。

グループ　(1) 共通の名前で参照されるユーザーの集り。

(2) NIS+ では、NIS+ オブジェクトに共通のアクセス権を与えられたユーザーの集まり。NIS+ グループの情報は、NIS+ グループテーブルに格納されている。

(3) UNIX では、ユーザーのファイルへのアクセスを決定する。デフォルトユーザーグループと標準ユーザーグループの 2 つがある。

グループ ID ユーザーのデフォルト「グループ」を識別する番号。

グローバルコンテキスト FNS (XFN) で、オブジェクトのグローバル名を管理するコンテキスト (現在、XFNで指定されているグローバルコンテキストは、DNS と X.500 のみ)。

グローバルネームサービス 電話回線、衛星回線、その他の通信システムにより連結された世界中のエンタープライズレベルネットワークの名前を管理するサービスのこと。この世界中のネットワークの集合体がいわゆる「インターネット」である。グローバルネームサービスでは、ネットワーク名だけでなく、任意のネットワーク内の個々のマシンやユーザーをも識別することができる。

原子名 名前表記規則によって決められた最小かつ不可分の名前要素のこと。FNS (XFN) の用語。

広域ネットワーク (WAN) 地理的に離れた複数の LAN (Local Area Network) またはシステムを、電話回線、光ファイバ、衛星などを使用して接続したネットワークのこと。

公開鍵 数学的に生成された 1 対の番号の公開構成要素であり、秘密鍵と組み合わせれば DES 鍵が生成される。この DES 鍵を使用すれば、情報の暗号化と復号化を行える。公開鍵は、すべてのユーザーとマシンが使用できる。どのユーザーやマシンにも、固有の公開鍵と秘密鍵が 1 対ある。

コールドスタートファイル クライアントが初期設定されたときにクライアントに与えられる NIS+ ファイル。クライアントがホームドメイン内のマスターサーバーと通信を開始できるだけの情報が収められている。

子ドメイン 「ドメイン」の項を参照。

コンテキスト 異なる複数の原子名とのバインディングを状態として持つオブジェクトのこと。どのコンテキストもそれに対応する名前表記規則を

持つ。各コンテキストでは、ルックアップ (名前解決) 操作でリファレンスを返す。その他、名前の結合、切り離し、結合された名前を一覧表示するなどの操作も実行できる場合もある。

合成名 **(compound name)**
ネームシステムの名前表記規則に従って複数の原子名を並べて構成した名前。

サーバー
(1) NIS+ 、NIS 、DNS 、FNS (XFN) では、ネットワークに NIS+ サービスを提供するホストマシンのこと。

(2) ファイルシステムの「クライアントサーバーモデル」では、サーバーとは計算資源 (計算サーバーとも呼ばれる) と大きな記憶容量を備えたマシン。クライアントマシンはリモートアクセスが可能であり、これらの資源を使用できる。ウィンドウシステムのクライアントサーバーモデルでは、サーバーとはアプリケーションまたは「クライアントプロセス」にウィンドウサービスを提供するプロセス。このモデルでは、クライアントとサーバーは同じマシン上でも別のマシン上でも実行できる。

(3) ファイルの提供を実際に処理する「デーモン」。

サーバーリスト
「優先サーバーリスト」の項を参照。

サービスコンテキスト
FNS (XFN) では、サービスを提供するオブジェクトの名前を管理するコンテキストのこと。

サイトコンテキスト
物理的なサイトに関連するオブジェクトの名前を管理するコンテキスト。

サブコンテキスト
他のコンテキストに属するコンテキスト。

サブネット
ルーティングを簡単にするため、1 つの論理ネットワークを小さな物理ネットワークに分割する方式。

資格
NIS+ 主体 (principal) についての認証情報。クライアントソフトウェアは、各要求と一緒にこの情報を NIS+ サーバーに送信する。この情報によって、ユーザーまたはマシンの確認検査が行われる。

識別名
X.500 ディレクトリ情報ベース (DIB) のエントリ。ルートから指定エントリまでのパスに沿ったツリーの各エントリから選択した属性で構成される。

主体

NIS+ 情報のユーザーのうち、名前空間に資格が格納されているものを指す。また NIS+ サーバーに対してなんらかのリクエストを行うユーザー、マシンを指す。クライアントユーザー、クライアントマシンの 2 種類がある。

- 「ルート主体」

  マシンのルートユーザー (ユーザー ID = 0) のこと。DES資格だけ必要。

- 「ユーザー主体」

  ルートを除くすべてのユーザー (ユーザー ID > 0)。ローカル資格、および DES 資格が必要。

初期コンテキスト

FNS (XFN) では、すべての XFN 名が何らかのコンテキストに関連付けられる形で解釈され、どの XFN 操作もコンテキストオブジェクト上で行われる。XFN インタフェースには、複合名解決の出発点となる初期コンテキストオブジェクトをクライアント側で得るための関数が用意されている。

初期コンテキスト関数

FNS 関数の 1 つ、`fn_ctx_handle_from_initial()` のこと。これを実行すると、名前解決の出発点となる初期コンテキストをクライアント側で得ることができる。

スレーブサーバー

(1) ネットワーク情報サービス (NIS) データベースのコピーを管理するサーバーシステム。このシステムには、ディスクと動作環境の完全なコピーがある。

(2) スレーブサーバーは、NIS+ では「複製サーバー」と呼ばれる。

接続

FNS (XFN) の用語。ある名前空間にあって、別のネーミングシステムのコンテキストに結合されている名前。

属性

FNS (XFN) では、名前の付いている個々のオブジェクトに対して、ゼロないし複数の属性が割り当てられる。個々の属性は一意の識別子、構文、ゼロないし複数の個別の値を持つ。

組織単位

エンタープライズは本部、研究所、部、課などで構成される。エンタープライズのサブユニットが組織単位である。

組織単位のコンテキスト

エンタープライズ内の組織単位に関連するオブジェクト名を管理するコンテキスト。

チェックポイント設定
: NIS+ データセットの変更 (サーバーのメモリに格納され、トランザクションログに記録される) をディスク上の NIS+ テーブルに書き込む一連のプロセス。つまり、NIS+ データセットの変更を反映して NIS+ テーブルを更新すること。

次のネーミングシステム参照 **(NNSP)**
: 下位のネーミングシステムで生成された複合名を解決するコンテキスト。

強い分離
: XFN コンテキストが XFN コンポーネントセパレータをネーミングシステムの境界として扱うこと。

テーブル
: NIS+ においては、NIS+ データを行および列の中に持つ 2 次元的な (リレーショナルでない) データベースオブジェクトのこと (NIS における「NIS マップ」は、「列を 2 つ持つ NIS+ テーブルに似ている)。NIS+ データは、テーブルの形で保存される。NIS+ では、定義済み (システム) テーブルが 16 個提供される。保存される情報のタイプはテーブルごとに異なる。

テーブルの生成 **(populate)**
: ファイルまたは NIS マップから NIS+ テーブルにデータを入れること。

ディレクトリ
: (1) NIS+ においては、NIS+ オブジェクトのコンテナのこと。NIS+ テーブル、グループ、サブディレクトリなどを指す。

  (2) UNIX においては、ファイルまたはサブディレクトリのコンテナのこと。

ディレクトリキャッシュ
: ディレクトリオブジェクトに関連付けられたデータの格納に使用されるローカルファイル。

データ暗号化鍵
: 暗号化を行うプログラムに使用されるデータを暗号化および復号化するための鍵。「鍵 (暗号化)」の項も参照。

ドット表記
: 32 ビット整数用の構文表現であり、10 進表記された 4 つの 8 ビット数が小数点 (ドット) で区切って表現される。192.67.67.20 のように、インターネットでの IP アドレスを表現するために使用される。

ドメイン
: (1) NIS+ では、NIS+ によって管理されるオブジェクト (階層構造になっている) のグループ。最上位のドメイン (ルートドメイン) 1 つ

と、サブドメイン 0 個以上からなる。ドメインおよびサブドメインは、地理的、組織的、機能的な基準によって編成される。

- 「親ドメイン」

  階層構造の中で、現在のドメインのすぐ上のドメインを表す相対的な名称。

- 「子ドメイン」

  階層構造の中で、現在のドメインのすぐ下のドメインを表す相対的な名称。

- 「ルートドメイン」

  現在の NIS+ 階層の最上位のドメイン。

(2) インターネットではネーミング階層の一部で、通常、Local Area Network (LAN)、Wide Area Network (WAN)、またはその一部に相当する。構文上、インターネットドメイン名は小数点 (ドット) によって区切られた一連の名前 (ラベル) から構成される。たとえば、`sales.doc.com`。

(3) ISO の開放型システム間相互接続 (OSI) では、「ドメイン」は、MHS プライベート管理ドメイン (PRMD) やディレクトリ管理ドメイン (DMD) などのように、複雑な分散システムの管理パーティションとして使用されるのが普通。

ドメイン名 ローカルネットワーク上のシステムグループに割り当てられた名前であり、管理ファイルを共有する。ネットワーク情報サービスのデータベースが正常に動作するためにはドメイン名が必要。「ドメイン」の項を参照。

ドメイン名システム (DNS) インターネットで使用されるネットワーク情報サービスのこと。すなわち DNS は、ドメイン名とマシン名をインターネットなどのエンタープライズ外部のアドレスにマッピングする場合のネーミングポリシーとメカニズムを提供する。

名前解決 ワークステーションやユーザーの名前をアドレスに変換するプロセス。

名前空間 (1) 名前空間はユーザー、ワークステーション、そしてアプリケーションがネットワーク上で必ず必要とする情報を格納している。

(2) ネーミングシステムで使用される名前セット。

(3) NIS+ 名前空間 - NIS+ ソフトウェアによって使用される階層型ネットワーク情報の集まり。

(4) NIS 名前空間 - NIS ソフトウェアによって使用される非階層型ネットワーク情報の集まり。

(5) DNS 名前空間 - DNS ソフトウェアを使用するネットワークワークステーションの集まり。

名前空間識別子　FNS (XFN) の用語。名前空間のルートを表す特殊な原子名のこと。

認証　ある NIS+ サーバーが、NIS+ 名前空間に対するアクセス要求の送信者を識別できるかどうかの判定。認証された要求は、所有者、グループ、およびその他という承認カテゴリに分類される。送信者が識別できない未認証要求は、未承認カテゴリに入れられる。

ネーミング規則　FNS (XFN) では、すべての名前が構文ルールによって生成される。「ネーミング規則」は、これら構文ルールの総称。

ネーミングシステム　接続された同種の複数のコンテキスト (同じネーミング規則を持ち、同じ意味に対して同じ操作を行う) の総称。たとえば、UNIX では、ファイルシステム内のディレクトリ群 (ディレクトリ上での名前管理操作を含む) がネーミングシステムを構成する。

ネームサーバー　1 つ以上のネットワークネームサービスを実行するサーバー。

ネームサービス　マシン、ユーザー、プリンタ、ドメイン、ルーターなどの、ネットワーク上の名前とアドレスを管理するネットワークサービスのこと。

ネームサービススイッチ　NIS+ クライアントがそのネットワーク情報を獲得できるソースを定義する構成ファイル (`/etc/nsswitch.conf`)。

ネットワークパスワード　「Secure RPC パスワード」を参照。

ネットワークマスク　ソフトウェアが、ローカルサブネットアドレスをそれ以外のインターネットプロトコルアドレスから分離するために使用する番号。

バインディング　原子名とオブジェクトリファレンス (オブジェクト) を関連付けること。簡素化のため、本書では「オブジェクトリファレンス」とそれが表す「オブジェクト」を同じ意味で使うことがある。

汎用コンテキスト　FNS (XFN) の用語。アプリケーションで使われる名前を結合させるコンテキスト。

秘密鍵　数学的に生成された 1 対の番号の非公開構成要素であり、公開鍵と組み合わせれば DES 鍵が生成される。この DES 鍵を使用すれば、情報の暗号化と復号化を行える。送信側の公開鍵は、その鍵の所有者だけが使用できる。どのユーザーやマシンにも、固有の公開鍵と秘密鍵が 1 対ある。

フェデレーテッド・ネーミング・サービス　フェデレーテッド・ネーミング・システムが提供するサービスのこと。

フェデレーテッド・ネーミング・システム　複合名を標準のインタフェースで解決できるようにするため、複数の自律的なネーミングシステムを集めてネーミングシステム同士を連係させたもの。連合内の各ネーミングシステムは、名前解決以外の操作については、自律的に選択できる。

フェデレートされた名前空間　FNS (XFN) の用語。連合を作っているネーミングシステムと名前空間との関係に関する管理方針に従って生成され得るすべての名前を集めたものの総称。

複合名 **(composite name)**　複数のネーミングシステムが使われている名前のこと。ゼロないし複数の要素が順番に並んだ構成になっている。各ネーミングシステムの名前空間にある名前が個々の構成要素になる。「複合名の解決」といえば、複数のネーミングシステムが使われている名前を解決することを指す。

複製サーバー　NIS+ サーバーのうち、ドメインのマスター NIS+ サーバーデータベースの複製を持つもの。複製サーバーは、NIS+ サーバーソフトウェアを実行し、NIS+ テーブルのコピーを管理する。複製サーバーによって、NIS+ サービスの可用性が向上する。NIS+ ドメインには個々に 1 つ以上の複製サーバーが必要である (「複製サーバー」は、NIS 名前空間では「スレーブサーバー」と呼ばれる)。

ホストコンテキスト　あるコンピュータに関連するオブジェクトの名前を管理するコンテキスト。

マスターサーバー　ドメイン内の NIS データベースのマスターコピーを保持しているサーバーのこと。名前空間に対する変更は、必ずマスターサーバー

のデータベース上で行う。ドメイン中に複数のマスターサーバーを作成できない。

メール交換レコード　DNS ドメイン名、およびこれらに対応するメールホストのリストが収められているファイル。

メールホスト　サイトの電子メールのルーターおよびレシーバとして機能するワークステーション。

ユーザーコンテキスト　ユーザーに関連するオブジェクトの名前を管理するコンテキスト。

優先サーバー　クライアントマシンから見て、優先 NIS+ サーバーとは、そのクライアントが名前空間に関する情報を得る際に、他のサーバーより先に照会を試みるサーバーのことをいう。あるクライアントまたはドメインの優先サーバーリストにあるサーバーは、そのクライアントまたはドメインの優先サーバーであると考えられる。

優先サーバーリスト　`client_info` テーブルまたは `client_info` ファイルのこと。優先サーバーリストには、あるクライアントマシンまたはドメインから見た優先サーバーが指定される。

優先順位の番号　クライアントマシンが名前空間に関する情報を取得する際に、どの NIS+ サーバーから順に照会するかを示す番号。マシンは、ある優先順位番号を持つすべてのサーバーに要求を送ったあと、その次に高いランク番号を持つサーバーに要求を送る。たとえば、ランク番号 0 の NIS+ サーバーに要求を送った後、ランク番号 1 のサーバーに要求を送る。

弱い分離　XFN コンテキストが XFN コンポーネントセパレータをネーミングシステムの境界としては扱わないこと。

リファレンス　ある名前を指し示すもの。具体的には、オブジェクト通信の終端を示すアドレスなど。

ルートコンテキスト　名前空間のルートにあってオブジェクト名を管理するコンテキスト。

ルートドメイン　「ドメイン」の項を参照。

ルート複製サーバー　NIS+ サーバーのうち、ルートドメインのマスター NIS+ サーバーデータベースの複製を持つものを指す。

ルートマスターサーバー	NIS+ ルートドメインのマスターサーバーのこと。
レコード	「エントリ」の項を参照。
ローカルエリアネットワーク **(LAN)**	1 つの地理的なサイトの中にある複数のシステムを、データやソフトウェアの共有、交換の目的で接続したもの。

索引

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は