LDAP の設定と構成

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94043-1100
U.S.A.

Part Number 806-7089-10
2001 年 2 月

原典: *LDAP Setup and Configuration Guide*

Part No: 806-5580-10

Revision A

Adobe PostScript

目次

表

図

はじめに

このマニュアルでは、LDAP クライアントシステムの設定、構成、管理について説明します。このマニュアルの内容は、今後のリリースで『*Solaris* ネーミングの管理』と『*Solaris* ネーミングの設定と構成』を一冊にまとめた『*System Administration Guide: Naming Services*』に記載される予定です。

対象読者

このマニュアルの情報は、経験のあるシステム/ネットワーク管理者を想定して書かれています。

このマニュアルでは、Solaris のネームサービスとしての LDAP に関係するネットワーク概念について説明していますが、LDAP の概念とネットワークの基礎について説明していません。読者は LDAP の概念についての知識を持っており、何らかの管理ツールを使用していると仮定しています。

お読みになる前に

Solaris のネームサービスについては、次のマニュアルを参照してください。

- 『*Solaris* ネーミングの管理』
- 『*Solaris* ネーミングの設定と構成』

iPlanet Directory Server 4.11 を使用している場合は、次のマニュアルを参照してください。

- `http://iPlanet.com` にある iPlanet Directory Server のインストール手順、リリースノート、技術情報。iPlanet Advantage Software, Volume I CD には、iPlanet Directory Server 4.11 のマニュアルと Solaris ディレクトリ拡張のマニュアルが含まれています。
- 『*Netscape Directory Server Schema Reference Guide*』
- 『*Netscape Server Deployment Manual*』
- 『*Managing Servers with Netscape Console 4.0*』
- 『*Directory Server Administrators Guide*』

内容の紹介

このマニュアルの構成は次のとおりです。

第 1 章では、LDAP モデルを紹介し、LDAP の処理について簡単に説明します。

第 2 章では、LDAP ディレクトリサーバーの設定方法に関する基本事項を説明します。

第 3 章では、LDAP クライアントの設定方法について説明します。

第 4 章では、iPlanet ディレクトリサーバーで Solaris LDAP ネームサービスクライアントをサポートするための構成方法を例を用いて説明します。

付録 A では、Solaris LDAP ネームサービスクライアントをサポートするために LDAP が必要とするスキーマについて説明します。

付録 B では、構成で発生する問題を解決する方法について簡単に説明します。

関連マニュアル

ディレクトリサービスの実装についての詳細は、次の資料を参照してください。

- Timothy A. Howes、Mark C. Smith、Gordon S. Good、『*Understanding And Deploying LDAP Directory Services*』MacMillan Technical Publishing、1999

Sun のマニュアルの注文方法

専門書を扱うインターネットの書店 Fatbrain.com から、米国 Sun Microsystems™, Inc. (以降、Sun™ とします) 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、`http://www1.fatbrain.com/documentation/sun` の Sun Documentation Center をご覧ください。

Sun のオンラインマニュアル

`http://docs.sun.com` では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索をおこなうこともできます。

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、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。
「 」	参照する章、節、ボタンやメニュー名、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超える場合に、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

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

- C シェルのスーパーユーザー

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

- Bourne シェルおよび Korn シェル

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

- Bourne シェルおよび Korn シェルのスーパーユーザー

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

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

| 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、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。
- 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。

概要

このマニュアルでは、iPlanet LDAP ディレクトリサーバーの設定方法とネームサービスをサポートするための Solaris クライアントの設定方法について説明します。

ネームサービス

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上でやり取りするために必要な情報を中央の 1 箇所に格納します。これには、次のような情報が含まれます。

マシン (ホスト) 名とアドレス
ユーザー名
パスワード
所属するグループなど

中央のネームサービスがないと、各ワークステーションがこれらの情報を保持しなければならないため、大規模なネットワークでは管理コストが非常に高くなります。ネームサービス情報は、ファイル、データベーステーブルなどの形式で格納されます。

Solaris のネームサービス

Solaris オペレーティング環境では、次のネームサービスを提供しています。
DNS: ドメインネームシステム
/etc ファイル: UNIX 元来のネームシステム
NIS: ネットワーク情報サービス
NIS+: ネットワーク情報サービスプラス
LDAP: 軽量ディレクトリアクセスプロトコル

LDAP 以外の上記の 4 つのネームサービスについての詳細は、『*Solaris* ネーミングの管理』を参照してください。

現行の多くのネットワークでは、上記のサービスを 2 つ以上組み合わせて使用し、それらをネームサービススイッチ (単にスイッチともいう) で調整しています。スイッチは、クライアントワークステーションまたはクライアントアプリケーションがネットワーク情報を取得する方法を制御します。また、スイッチはアプリケーションが名前情報を取得するために使用するネームサービスを決定します。Solaris のスイッチについての詳細は、`nsswitch.conf(4)` のマニュアルページを参照してください。

LDAP モデル

LDAP は、ディレクトリサーバーにアクセスするための新しい業界標準プロトコルです。LDAP は軽量プロトコルです。効率的かつ単純で、実装も容易ですが、同時に十分に実用的でもあります。システムに依存しない単純化されたエンコーディング方式を採用しており、TCP/IP のすぐ上で動作します。

LDAP ディレクトリは、ディレクトリエントリの集合に名前を付け、管理し、アクセスする方法を提供します。ディレクトリのエントリは、1 つの型と 1 つ以上の値を持つ属性で構成されます。各属性の構文は、取り得る値の種類 (ASCII 文字や jpeg の写真など) と、それらの値のディレクトリ処理時における解釈方法 (検索時や比較時に大文字と小文字を区別するかなど) を定義します。

ディレクトリエントリは、地域 (国)、組織 (会社) 境界、またはドメイン (dc) に基づいてツリー構造に編成されます。

エントリには、このツリー構造内での位置によって、識別名 (DN) が付けられます。識別名の各構成要素は相対識別名 (RDN) と呼ばれます。RDN は、そのエント

リの 1 つ以上の属性から成ります (識別名の正式な定義については、RFC2253 を参照してください)。

ディレクトリツリー構造の階層は、UNIX ファイルシステムの階層に似ています。つまり、RDN がファイル名、DN がそのファイルへの絶対パスに相当します。UNIX ファイルシステムと同様、同じ親を持つディレクトリエントリはそれぞれ固有の RDN を持たなければなりません。ただし、ディレクトリツリー内では、末端ノードでも非末端ノードでも中身または属性を持つことができます。

DNS 名前空間と同様に、LDAP ディレクトリエントリは「リトルエンディアン」方式でアクセスされます。つまり、LDAP の名前では、最も小さな構成要素が最初に、ルート直下の最も大きな構成要素が最後にきます。DN は、RDN の並びをツリーのルートまで連結することによって構成されます。たとえば、Joe Qwerty という人が米国にある Ultra Keyboards という会社で働いているとすると、Joe Qwerty という人物の俗名 (CN) 属性には「Joe Qwerty」という値が、DN 属性には「cn=Joseph Qwerty, o=Ultra Keyboards, c=US」という値が格納されます。

LDAP をネームサービスとして使う理由

LDAP を使うと、既存のアプリケーション固有のディレクトリを置き換え、情報を統合することができます。つまり、LDAP サーバー上で行なった変更は、その情報を使用するすべてのディレクトリ対応アプリケーションで有効になります。たとえば、新規ユーザーに関するさまざまな情報を 1 つのインタフェースで 1 回だけ更新すれば、そのユーザーの UNIX アカウント、メールアドレス、別名がすぐに作成され、部門メーリングリストのメンバーに登録され、部外秘の Web サーバーにアクセスできるようになり、職責限定のニュースグループの購読者になるという場合を想像してみてください。またそのユーザーは、会社の電話帳、メールアドレス帳、会議カレンダーシステムにもすぐに登録されます。そのユーザーが退社したときも、1 回の操作で上記のすべてのサービスを無効にすることができます。

ディレクトリは、汎用のデータベースとは使われ方が異なります。ディレクトリには、頻繁に検索されるが、滅多に変更されない情報が格納されます。たとえばホスト名やユーザー名などの情報は、1 回登録され、何千回も検索されます。LDAP サーバーは、このような用途に合わせて調整されています。一方リレーショナルデータベースは、絶えず変化するデータを保守するという用途に合わせて調整されています。

複製サーバーを作成することによって、ディレクトリデータは複数のサーバーから利用できるようになり、装置の故障などという不測の事態に備えることができます。また、ディレクトリデータをより多く複製し、それを使用するユーザーやアプリケーションの近くに置くことができるため、処理のパフォーマンスが向上します。

複製サーバーを使用する理由は、正式な (複製元の) サーバーの負荷を軽減することだけではありません。多くの UNIX ネットワークでは、サブネットにスレーブサーバーを置く NIS (YPともいう) を使用しています。NIS と同様に、サブネット上に複製を置くと、ルーター経由のネットワークトラフィックを避けることができるため、応答待ち時間を短縮できます。ただし NIS と違って、LDAP の同期方式では増分更新を採用しているため、すべてのデータを定期的に複製に転送するのではなく、更新データを随時複製に転送します。

正式な (複製元の) サーバーの情報を保守するには、読み取り、書き込み、検索、比較の各権利についてアクセス制御を設定する必要があります。アクセス制御の対象にできるのは、サブツリー、エントリ、属性タイプで、個人、グループ、または「自分自身」(これにより、認証されたユーザーが自分自身のエントリにアクセスできる) に権限を与えることができます。これは極めて柔軟性の高い方式です。たとえば、人事部門の社員だけに役職やマネージャ属性の変更を許可したいとか、管理アシスタントにオフィスの住所とポケットベルの番号の変更許可を与えるとか、各個人に自宅の電話番号、車のナンバーといった情報の変更を許可するなどということが可能になります。詳細は、iPlanet ディレクトリサーバーのマニュアルを参照してください。

UNIX のログイン情報を例に説明してみましょう。いったんユーザーの属性をディレクトリサーバーに格納すると、ディレクトリサーバーのインタフェースを介して更新することによって、複数のオペレーティングシステムプラットフォームで使用しているユーザー名とパスワードを一括して更新できます。これにより、ユーザー情報の変更が簡単になるだけでなく、稀にしか使用しないアカウントのパスワードを忘れるといったことも少なくなります。

LDAP を Solaris オペレーティング環境のネームサービスとして使用する

Solaris クライアントは、NIS や NIS+ と同じようにネームサービススイッチを介して LDAP を使用することによって、名前情報を取得できます。

Solaris で広く使用されている、プロトコル非依存のネームサービスインタフェースは、標準の getXbyY API です。アプリケーションから getXbyY() (たとえば gethostbyname(3NSL)) を呼び出すと、ネームサービススイッチを走査し、そこから適切なネームサービスプロトコルを見つけます。それが LDAP であれば、LDAP API を呼び出して LDAP サーバーから情報を取得します。ネームサービススイッチについての詳細は、nsswitch.conf(4) のマニュアルページを参照してください。

図 1–1 に、ネームサービス、ネームサービススイッチ、および LDAP 実装各部の関係を示します。

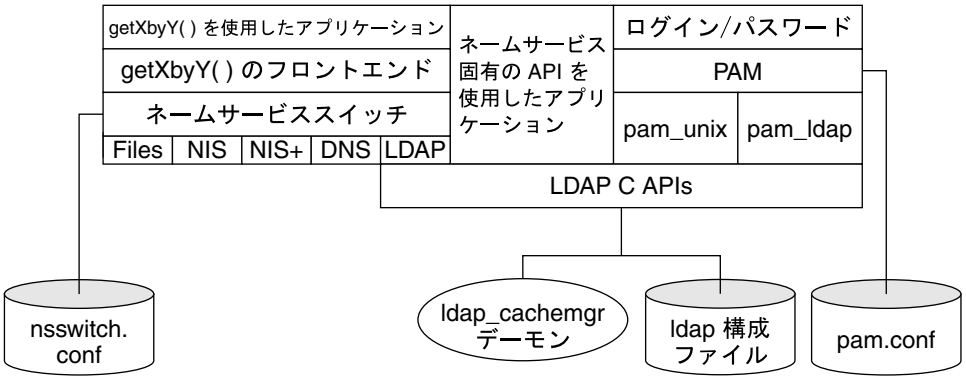

図 1–1　アーキテクチャの概要

前述の LDAP のすべての機能に加えて、ディレクトリ内にクライアントプロファイルを格納しておくと、Solaris クライアントの構成と保守が大幅に簡略化されます。この場合、各クライアントがデーモンを動作させ、このデーモンがディレクトリから最新のプロファイルをダウンロードしてクライアントの構成を更新します。これにより、クライアントの構成を変更する必要が生じても (新しい LDAP サーバーが追加されたり、セキュリティモデルが変更された場合など)、システム管理者は適切なプロファイルを変更するだけで、クライアントが最新の構成を自動的に取得します。詳細は、ldap_cachemgr(1M) のマニュアルページを参照してください。

LDAP の処理

LDAP では、次の 3 つの機能に関して 9 つの処理が定義されています。

- 問い合わせ

search 処理および compare 処理は、ディレクトリに対して問い合わせを実行し、情報を取得します。

- 更新

  add、delete、modify、modify RDN の各処理は、ディレクトリ情報を更新します。

- 認証

  bind 処理および unbind 処理は、ディレクトリ情報のセキュリティを保護するための基盤を提供します。abandon 処理を使用すれば、実行中の操作を取り消すことができます。

第 2 章

サーバーの設定

この章では、Solaris LDAP クライアントが名前情報を検索できるように、LDAP サーバーを設定する方法について説明します。この設定によって、Solaris LDAP クライアントは、`getXbyY` インタフェースまたは `ldaplist(1)` を使用して、LDAP サーバー上の名前情報を検索できるようになります。

この章の内容は次のとおりです。

前提条件

Solaris ネームサービスクライアントは LDAP v3 プロトコルだけで利用されている制御を使用するため、Solaris ネームサービスクライアントで名前情報の検索を行うには、サーバーが LDAP v3 プロトコルに対応している必要があります。

以下の制御は v3 だけで使用できます。

- 単純ページモード (RFC 2696)
- 仮想リスト表示制御

サーバーは次のいずれかの認証方式をサポートしていなければなりません。

- anonymous (匿名)
- SIMPLE (平文パスワード)
- SASL CRAM-MD5

▼ ディレクトリが単純ページモード制御をサポートしていることを確認する

1. **ldapsearch** コマンドを使用して、ディレクトリがページモード制御タイプ **1.2.840.113556.1.4.319**、ページモード制御値 **2.16.840.1.113730.3.4.2** の **OID** によって特定される単純ページモード制御をサポートしているかどうかを調べます。

```
# ldapsearch -b "" -s base objectclass=\*
```

たとえば、ldapsearch コマンドは次のような結果を返します。

```
objectclass=top
namingcontexts=dc=sun,dc=com,o=internet
subschemasubentry=cn=schema
supportedsaslmechanisms=CRAM-MD5
supportedextension=1.3.6.1.4.1.1466.20037
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=2.16.840.1.113730.3.4.2
supportedldapversion=2
supportedldapversion=3
```

▼ ディレクトリが仮想リスト表示をサポートしていることを確認する

1. ldapsearch コマンドを使用して、ディレクトリが **VLV** 制御タイプ **1.2.840.113556.1.4.473**、**VLV** 制御値 **2.16.840.1.113730.3.4.9** の **OID** によって特定される仮想リスト表示制御をサポートしているかどうかを調べます。

```
# ldapsearch -b "" -s base objectclass=\*
```

たとえば、ldapsearch コマンドは次のような結果を返します。

```
objectclass=top
namingcontexts=dc=sun,dc=com
namingcontexts=o=NetscapeRoot
subschemasubentry=cn=schema
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=2.16.840.1.113730.3.4.4
supportedcontrol=2.16.840.1.113730.3.4.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=2.16.840.1.113730.3.4.9
supportedcontrol=2.16.840.1.113730.3.4.12
supportedsaslmechanisms=EXTERNAL
supportedldapversion=2
supportedldapversion=3
dataversion=atitrain2.east.sun.com:389 020000605172910
netscapemdsuffix=cn=ldap://:389,dc=atitrain2,dc=east,dc=sun,dc=com
```

注 - ldapsearch コマンドについての詳細は、ldapsearch(1) のマニュアルページを参照してください。

スキーマ

Solaris LDAP ネームサービスクライアントをサポートするには、IETF によって定義されているスキーマといくつかの Solaris 固有のスキーマが必要です。

IETF によって定義されている LDAP スキーマで必要なものは、RFC 2307 の NIS スキーマと LDAP メールグループインターネットドラフトによるものの 2 つです。ネーム情報サービスをサポートするには、これらのスキーマ定義をディレクトリサーバーに追加する必要があります。IETF のスキーマと Solaris 固有のスキーマについての詳細は、付録 A を参照してください。各種の RFC は、IETF のページ (`http://www.ietf.org`) にあります。

ディレクトリ情報ツリー

Solaris LDAP クライアントは、デフォルトの ディレクトリ情報ツリー (DIT : Directory Information Tree) 内の情報を使用します。変更された DIT をプロファイルに指定すると、この DIT がデフォルトの DIT よりも優先されます。DIT は、特定の情報タイプのエントリを含むサブツリーであるコンテナに分割されます。

検索の起点となる識別名 (`baseDN`) には、そのクライアントのすべての情報が格納されている DIT 内の場所を指定します。検索の起点として指定されたノードには、`NisDomainObject` オブジェクトクラスが存在しなければなりません。検索の起点ノードのサブツリーには、各タイプの情報のすべてのコンテナを指定します。図 2–1 を見てください。

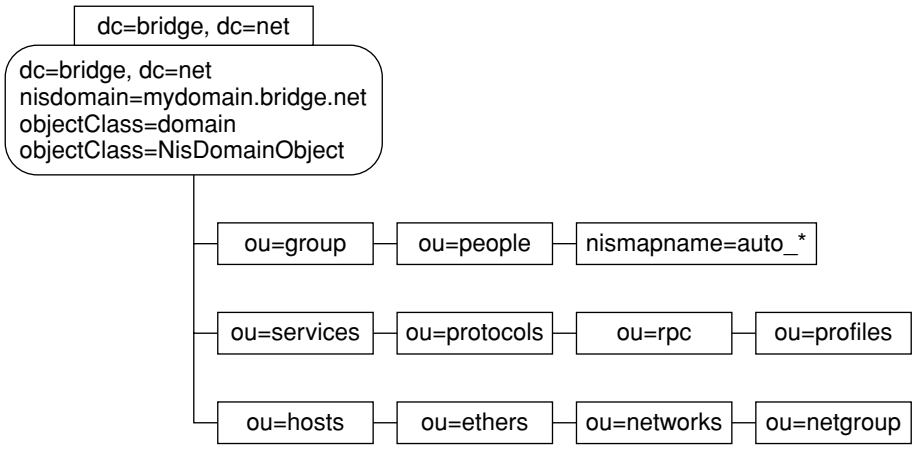

図 2–1　ディレクトリ情報ツリー (DIT) コンテナ

表 2–1 に、この DIT に格納されているコンテナと情報タイプを示します。

表 2–1　ディレクトリ情報ツリー (DIT)

コンテナ	情報タイプ
ou=Ethers	bootparams(4), ethers(4)
ou=Group	group(4)
ou=Hosts	hosts(4), ipnodes(4), publickey(4)
ou=Aliases	aliases(4)
ou=Netgroup	netgroup(4)
ou=Networks	networks(4), netmasks(4)
ou=People	passwd(1), shadow(4), user_attr(4), audit_user(4), ユーザーの公開鍵
ou=Protocols	protocols(4)
ou=Rpc	rpc(4)
ou=Services	services(4)
ou=SolarisAuthAttr	auth_attr(4)
ou=SolarisProfAttr	prof_attr(4), exec_attr(4)
ou=projects	project(4)
nismapname=auto_*	auto_*

DIT 内のデフォルトコンテナを無効にする

LDAP を使用する際にデフォルトのコンテナを無効にする必要がある場合は、プロファイル内に変更されたコンテナを指定します。その際、データベースごとに代替の検索起点識別名 (baseDN) を定義できます。

たとえば、ある組織で ou=People コンテナを、ou=employee および ou=contractor コンテナに置き換えたいとします。このプロファイルエントリ (DIT 内の任意の場所に指定可能) では、代替検索識別名 (DN) を指定する必要があります。代替検索 DN を指定するには、-B オプションを使用して LDAP クライアントプロファイルを作成します。詳細は、ldap_gen_profile(1M) のマニュアルページを参照してください。属性は次のようになります。

```
SolarisDataSearchDN="passwd:(ou=employee,dc=mkt,dc=mystore,dc=com),
                     (ou=contractor,dc=mkt,dc=mystore,dc=com)"
```

NIS ドメイン

Solaris クライアントが特定ドメインのサーバーを見つけるには、nisDomainObject オブジェクトクラスの nisDomain 属性が、目的のドメインを表す DIT のルート DN エントリに定義されている必要があります。クライアントはこの情報を使用して、システムを初期化し、クライアントプロファイルを更新します。初期化時に、クライアントは目的のドメインに一致する nisDomain 属性値を持つ LDAP サーバー上のエントリを検索します。そして、そのエントリの DN を、名前情報の起点識別名 (BaseDN) として使用します。

クライアントプロファイルを更新するとき、クライアントマシン上の ldap_cachemgr は、ルート DN エントリに定義された nisDomain が目的のドメインと一致することを確認します。

このマニュアルでは、例として次の nisDomain を使用することにします。

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
objectclass: top
objectclass: domain
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
```

クライアントプロファイル

Solaris クライアントの設定を簡略化するには、クライアントプロファイルを定義する必要があります。このプロファイルはサーバー上に作成する必要があります。クライアントは初期化時にプロファイル名とサーバーのアドレスを指定するだけで、簡単にシステムを設定できます。クライアントプロファイルによって、システム管理者は、Solaris クライアントが使用する LDAP 環境を定義できます。

プロファイルを使用することによる顕著な効果は、マシンのインストール作業が容易になることです。しかし、プロファイルの本当の利点がわかるのは、自分の環境に変更 (たとえばサーバーの追加や削除) を加え始めたときです。詳細は、ldap_gen_profile(1M) のマニュアルページを参照してください。

次に、クライアントプロファイルに定義できる属性をまとめておきます。

- SolarisLDAPServers

  コンマで区切られた LDAP サーバー IP アドレスのリストです。IP アドレスの後に、クライアントが使用するポート番号をコロンで区切って指定できます。このパラメータにデフォルト値はありません。少なくとも 1 つの LDAP サーバーを定義する必要があります。複数のサーバーを定義すると、最初のサーバーから応答がないときには、次のサーバーが使用されます。

- SolarisSearchBaseDN

  名前情報が格納される LDAP ネーム起点識別名です。

- SolarisBindDN

  クライアントが認証処理で使用する LDAP の識別情報です。通常は、プロキシエージェントの DN です。デフォルトは NULL 文字列です。

- SolarisBindPassword

  SIMPLE 認証および CRAM_MD5 認証を使用するときの SolarisBindDN のパスワードです。デフォルトは NULL 文字列です。

- SolarisAuthMethod

  クライアントが使用する認証方式の順序付きリストです。使用可能な認証方式として、NONE、SIMPLE、CRAM_MD5 があります。デフォルトは NONE です。複数の認証方式を指定すると、資格以外の理由で最初の方式による認証に失敗したとき、次の認証方式が試されます。

- SolarisTransportSecurity

  クライアントが使用する安全な通信方法です。デフォルトは NONE です。現在のところ、NONE しか指定できません。

- SolarisDataSearchDN

  名前情報を検索するときの代替の起点識別名 (baseDN) です。この属性を指定することによって、デフォルトの名前情報タイプを無効にできます。代替 baseDN の書式は次のとおりです。

```
database:alternate-baseDN-list
```

  database は nsswitch.conf ファイルに定義された情報タイプ、alternate-baseDN-list は代替 baseDN のリストです。リストの各要素はコンマで区切り、リスト全体を括弧で囲みます。各データベースは、このパラメータに指定された順序で検索されます。デフォルトは、どのコンテナでも NULL です。

- SolarisSearchScope

  名前情報を検索するときに適用する検索範囲です。指定可能な値は、Base、One level、Subtree のいずれかです。デフォルトは One level です。

- SolarisSearchTimeLimit

  名前情報を検索するときの LDAP 検索時間の上限 (秒) です。デフォルトは 30 秒です。

- SolarisCacheTTL

  クライアントが自身のプロファイル情報をサーバーから取得して更新するまでの有効期限です。ldap_cachemgr による自動更新を行いたくないときは、client_TTL に 0 (ゼロ) を設定します。指定できる値は、ゼロ (期限切れなし)、または正の整数 (秒) です。デフォルトは 43200 (12 時間) です。

- SolarisSearchReferral

  名前情報を検索するときの参照の扱いを指定します。参照に従うまたは参照に従わないのいずれかを指定できます。デフォルトでは常に参照に従います。

クライアントプロファイルを作成するための ldap_gen_profile(1M) コマンドは、Solaris クライアントツールの一部として提供されています。このツールは LDIF ファイルを生成します。生成された LDIF ファイルを LDAP サーバーに格納するには、ldapadd(1) コマンドを使用します。次に、クライアントプロファイルの作成方法を示します。

▼ クライアントプロファイルを作成する

1. ldap_gen_profile(1M) を使用してクライアントプロファイルを作成します。

```
# /usr/sbin/ldap_gen_profile \
-P myprofile \
-b dc=mkt,dc=mainstore,dc=com \
-a simple -w mypasswd \
-D cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com \
100.100.100.100
```

作成されたプロファイルの例を次に示します。

```
dn: cn=myprofile,ou=profile,dc=mkt,dc=mainstore,dc=com
SolarisBindDN: cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com
SolarisBindPassword: {NS1}xxxxxxxxxxxxx
SolarisLDAPServers: 100.100.100.100
SolarisSearchBaseDN: dc=mkt,dc=mainstore,dc=com
SolarisAuthMethod: NS_LDAP_AUTH_SIMPLE
SolarisTransportSecurity: NS_LDAP_SEC_NONE
SolarisSearchReferral: NS_LDAP_FOLLOWREF
SolarisSearchScope: NS_LDAP_SCOPE_ONELEVEL
SolarisSearchTimeLimit: 30
SolarisCacheTTL: 43200
cn: myprofile
ObjectClass: top
ObjectClass: SolarisNamingProfile
```

2. 作成されたプロファイルを (`profile.ldif` などの) ファイルに保存し、`ldapadd(1)` を使用して **LDAP** サーバーに格納します。

```
# ldapadd -h ldap_server_hostname -D "cn=Directory Manager" \
-w nssecret -f profile.ldif
```

各クライアントマシン上の `ldap_cachemgr(1M)` は、LDAP 構成ファイルの内容を自動的に更新します。つまり、サーバー上のプロファイル情報を変更すると、変更内容が自動的に名前空間内のすべてのクライアントに反映されます。定期的な更新は、プロファイルの `SolarisCacheTTL` 属性に指定された TTL (time to live: 有効期限) の値に基づいて行われます。

セキュリティモデル

ディレクトリに格納された情報にアクセスするとき、まずクライアントはディレクトリに対して認証される必要があります。認証されると、ディレクトリにアクセス制御情報 (ACI: Access Control Information) として格納された承認情報に基づいて、ディレクトリ内の一部またはすべての情報にアクセスできるようになります。この節では、クライアントの識別情報、認証方式、PAM モジュールについて説明しま

す。ACI についての詳細は、『*iPlanet Directory Server Administrator's Guide*』を参照してください。

認証の対象となる識別情報

LDAP ネームサービスは、クライアントをディレクトリに認証してもらうとき、匿名またはプロキシエージェントのいずれかの識別情報を使用するように構成できます。

- 匿名 (anonymous)

  認証とは識別情報を確認する作業全般を意味し、anonymous はその特殊な場合と考えることができます。anonymous はどのレベルのセキュリティも提供しません。つまり、ディレクトリに対するすべての非認証接続による、すべてのネットワーク情報レコード (パスワードやシャドウの情報も含む) の参照/読み取りが可能です。セキュリティはまったく提供されませんが、状況によってはこれが適切になる場合があります。

- プロキシエージェント

  プロキシエージェントを識別情報とする場合、クライアントは、ディレクトリ内のプロキシアカウントを使用してディレクトリに対して認証を行います。プロキシアカウントとしては、ディレクトリにバインドすることが許可されているすべてのエントリ (iPlanet Directory Server の場合は、`userPassword` 属性を持っているすべてのエントリ) を使用できます。

  ディレクトリ内の部分的な情報に対するアクセス制御は、プロキシの識別情報に基づき、各種の権限を制限または許可する適切な ACI (アクセス制御情報) を設定することによって行います。プロキシエージェントの数とクライアントの数には何の関係もないため、2 つを自由に組み合わせることができます。2 つの例を考えてみます。1 つは、すべてのクライアントに 1 つのプロキシエージェントを対応させ、そのプロキシに、名前情報を持っている DIT 全体に対する読み取り権を与えることができます。もう 1 つは、各クライアントが異なるプロキシエージェントを使用して認証を行うようにサーバーを設定し、クライアントごとに異なるアクセス制限を行う ACI を設定することもできます。この 2 つは、プロキシエージェントを使用した認証の両極端な例です。通常は、この 2 つの中間的な設定になります。アクセス制御の細かさはディレクトリ設計者が決定することで、慎重に考慮する必要があります。プロキシエージェントが少なすぎると、システム管理者はリソースへのアクセスを制御する力が制限されてしまいます。エージェントの数が多すぎると、必要なプロファイルの数も多くなり、システムの設定と保守が複雑になります。

注 - クライアント構成はプロファイル内に格納されるため、使用するプロキシの数と定義する必要のあるプロファイル数には直接的な関係があります。

認証方式

プロキシエージェントを使用する場合、システム管理者は、ディレクトリに対して認証するときの認証方式も選択する必要があります。現時点で、Solaris 8 によってサポートされている認証方式は、`SIMPLE` と `CRAM-MD5` の 2 つです。

- `SIMPLE`

  `SIMPLE` を選択した場合、クライアントは、LDAP サーバーに単純なバインド要求を送信し、そのサーバーに対して認証を行います。この認証方式ではパスワードがネットワーク上を平文で流れるため、パスワードが漏洩しやすくなります。一方、LDAP 標準で定義されている必須の認証方式なので、すべてのディレクトリサーバーでサポートされているという利点があります。

- `CRAM-MD5`

  一部のディレクトリサーバーは、SASL (Simple Authentication and Security Layer) を介してチャレンジ-応答認証方式 (CRAM-MD5) もサポートしています。`CRAM-MD5` の主な利点は、認証のときパスワードが平文のままネットワーク上を流れないため、`SIMPLE` よりも安全であるという点です。`CRAM-MD5` についての詳細は、RFC 2195 を参照してください。`SASL` については、RFC 2222 を参照してください。

注 - iPlanet Directory Server version 4.11 は、現時点では `CRAM-MD5` をサポートしていません。

PAM

PAM (Pluggable Authentication Module, 接続可能な認証モジュール) を使用すれば、アプリケーションを Solaris オペレーティング環境で使用される認証方式から独立させておくことができます。PAM 層を使用すれば、アプリケーションは、システム管理者がそのクライアントにどの認証方式を定義しているかに関係なく、認証を実行できます。LDAP ネームサービスを使用するには、`pam_unix(5)` または `pam_ldap(5)` のどちらかの PAM モジュールを `pam.conf` に指定します。

- pam_unix

  pam_unix を使用すると、従来の UNIX 認証モデルが使用されます。このモデルでは、ユーザーの暗号化されたパスワードをローカルマシンでディレクトリから取得し、ユーザーにパスワードの入力を求めます。そして入力されたパスワードを暗号化し、それをディレクトリから取得した暗号化されたパスワードと比較することによって、ユーザーの認証を行います。LDAP を使用しているクライアントがこのモジュールを使用するように構成されている場合は、そのクライアントが使用している ID (anonymous または構成されたプロキシエージェント) で userPassword 属性を読み取ることができなければなりません。また、この認証モデルには次の 2 つの制限があります。

  1. パスワードを userPassword 属性に格納する必要がある。
  2. パスワードを UNIX crypt 形式で格納する必要がある (平文、他の暗号方式によって暗号化されたパスワードは使えない)。

- pam_ldap

  LDAP ディレクトリを使用する場合、pam_unix による従来の認証方式は必ずしも最善の方法ではないため、Solaris 8 では、ディレクトリに対して直接ユーザーの認証を実行する新しい PAM モジュールが追加されました。このモジュールを使用すると、Solaris クライアントは、ディレクトリサーバーがサポートしている新しい高度な認証方式を使用できます。当然、pam_ldap を使用しているクライアントは、パスワード属性に対する読み取り権を必要としません。また、パスワードを特定の形式でディレクトリに格納する必要もありません。

  さらに、pam_ldap はディレクトリサーバーに対して直接ユーザーの認証を実行するため、ユーザーレベルのアクセス制御を適切に設定しておけば、ACI (アクセス制御情報) を使用して各ユーザーの認証を制御できます。

  pam_unix を使用するときと同様に、パスワードを変更するには passwd コマンドを使用します。

インデックス

ほとんどの LDAP サーバーでは、インデックスを使用して検索パフォーマンスの向上を図ります。インデックスの使用法については、ディレクトリサーバーのマニュアルを参照してください。

Solaris クライアントから妥当な時間で名前情報を検索できるようにするには、基本的なインデックス付き属性に加えて、次の属性にもインデックスを付ける必要があります。

```
membernisnetgroup    pres,eq,sub
nisnetgrouptriple    pres,eq,sub
memberuid            pres,eq
macAddress           pres,eq
uid                  pres,eq
uidNumber            pres,eq
gidNumber            pres,eq
ipHostNumber         pres,eq
ipNetworkNumber      pres,eq
ipProtocolNumber     pres,eq
oncRpcNumber         pres,eq
ipServiceProtocol    pres,eq
ipServicePort        pres,eq
nisDomain            pres,eq
nisMapName           pres,eq
mail                 pres,eq
```

注 - 属性リストで使用される略語の意味は、`pres` が存在 (presence)、`eq` が等価 (equality)、`sub` が部分文字列 (substring) です。

さらに、サーバーで仮想リスト表示制御 (vlv) がサポートされている場合は、vlv のインデックスも作成する必要があります。ツリー内の多数のオブジェクトを含むすべてのコンテンナにインデックスを作成してください (多数であるかどうかは、ツリー内の他のオブジェクトの数との相対で判断されます)。インデックスの作成方法については、ディレクトリサーバーのマニュアルを参照してください。vlv のソート値は `cn uid` に設定し、vlv フィルタと範囲は次のリストのように定義します。

```
getpwent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getspent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getgrent:      vlvFilter: (objectclass=posixGroup),       vlvScope: 1
gethostent:    vlvFilter: (objectclass=ipHost),           vlvScope: 1
getnetent:     vlvFilter: (objectclass=ipNetwork),        vlvScope: 1
getprotoent:   vlvFilter: (objectclass=ipProtocol),       vlvScope: 1
getrpcent:     vlvFilter: (objectclass=oncRpc),           vlvScope: 1
getaliasent:   vlvFilter: (objectclass=rfc822MailGroup),  vlvScope: 1
getserviceent: vlvFilter: (objectclass=ipService),        vlvScope: 1
```

インデックスを作成した場合の代価

インデックスを作成すると、検索パフォーマンスが向上する反面、次のような代価も発生します。

- データベースの変更が遅くなる

  保守するインデックスが増えるほど、データベースの更新に時間がかかります。特に部分文字列 (サブストリング) インデックスでは、属性値が作成または変更されるたびにディレクトリサーバーが複数のインデックスファイルを生成するので、更新に時間がかかります。部分文字列インデックスでは、作成されるインデックスエントリの数は、インデックス付けされる文字列の長さに比例します。

- より多くのシステム資源が必要になる

  - より多くのディスク領域が必要になる

    インデックス付けする属性が増えると、ディレクトリサーバーが作成するファイル数も増えます。

  - より多くのメモリーが必要になる

    動作効率を上げるため、ディレクトリサーバーはできるだけ多くのインデックスをメモリー上に置きます。このため、メモリーの消費量が非常に大きくなります。

  - ディスクアクセスが増える

    あまり使用されないインデックスを持っていると、利用頻度の低いインデックスファイルがディスクに書き出されてメモリーから削除された後に、より利用頻度の高いインデックスファイルがディスクから読み込まれます。

データの読み込み

Solaris ディレクトリ拡張パッケージには、NIS から LDAP に移行するために必要なツールとマニュアルが含まれています。これらのツールは、iPlanet Advantage Software CD (Volume 1) に収録されており、iPlanet の Web サイトからも入手できます。`dsimport` ツールは、NIS データを ldap 形式に変換するとき使用します。このツールは、入力データ形式、環境変数などに従ってカスタマイズされたマップファイル `nis.mapping` を使用します。詳細は、上記の CD または iPlanet の Web サイトを参照してください。

コマンド行ツール

LDAP は、LDAP API によって実行される処理に対応するコマンド行ツールを提供しています。これらのツールでは、認証やバインド関連のパラメータなど、共通のオプションを使用できます。

- `ldapsearch`

  ディレクトリエントリを検索し、見つかった属性と値を表示します。

- `ldapmodify`

  ディレクトリエントリの変更、追加、削除、名前変更をします。

- `ldapadd`

  新規のエントリを追加します。

- `ldapdelete`

  既存のディレクトリエントリを削除します。

- `ldapmodrdn`

  既存のディレクトリエントリの名前を変更します。

`ldapsearch`、`ldapadd`、`ldapmodify` の各ツールは、LDAP データ交換書式 (LDIF) と呼ばれる共通の書式をサポートしています。LDIF は、ディレクトリ情報を表現するテキストベースの書式です。

LDAP データ交換書式 (LDIF)

`ldapsearch` ツールは、LDIF の書式で出力します。`ldapadd` ツールはこの書式のデータを処理します。また、`ldapmodify` ツールはこの LDIF を基本とした変更情報を使用します。

LDIF ファイルには、1 つ以上のエントリが含まれています。各エントリは空白行で区切られます。基本的なエントリ形式は次のとおりです。

```
[id]
dn: entryDN
attrtype: attrvalue
```

(続く)

```
...
```

- *id*

  数字のエントリ識別子 (省略可能。LDAP ツールでは使用されない)。

- *entryDN*

  ディレクトリエントリの LDAP 識別名 (DN)。

- *attrtype*

  LDAP 属性タイプ。`cn` や `telephoneNumber` など。

- *attrvalue*

  `attrtype` の値。

`attrtype: attrvalue` 行は、必要なだけ繰り返すことによって、1 つのエントリのすべての属性値を指定できます。行を継続するには、次の行の先頭に 1 文字の空白または水平タブ文字を挿入します。

たとえば、5 つの属性を持つ Joe Qwerty のエントリを含む LDIF ファイルは、次のようになります (`cn` と `objectclass` は 2 つの値を持ちます)。

```
dn: cn=Joseph Qwerty, o=Ultra Keyboards Inc., c=US
cn: Joseph Qwerty
cn: Joe Qwerty
sn: Qwerty
mail: jqwerty@ultra.com
seeAlso: cn=Joe Qwerty, ou=Engineering Division, o=Peo
 ple, o=IEEE, c=US
objectClass: top
objectClass: person
```

注 - `seeAlso` の値は、「`ple, ...`」で始まる行の先頭に 1 文字の空白を挿入することによって、2 行に分割されています。

▼ ディレクトリを検索する

ディレクトリエントリを検索するには、ldapsearch(1) を使用します。ldapsearch は、LDAP ディレクトリサーバーへの接続を開き、そのディレクトリサーバーにバインドして、ディレクトリの検索を実行します。

1. 米国の **Ultra Keyboards** 社に勤務する、**IEEE** のメンバーを検索します。

```
% ldapsearch -L -b "o=IEEE, o=Ultra Keyboards Inc., c=US" uid=\*
```

検索結果は次のようになります。

```
dn: uid=jqwerty, o=IEEE, o=Ultra keyboards Inc., c=US
uid: jqwerty
cn: jqwerty
userpassword: {crypt}somecryptedtext
uidnumber: 12345
gidnumber: 123
gecos: Joseph Qwerty
homedirectory: /home/jqwerty
loginshell: /bin/csh
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
shadowlastchange: 3455

dn: uid=bhand, o=IEEE, o=Ultra keyboards Inc., c=US
uid: bhand
cn: bhand
userpassword: {crypt}somecryptedtext
uidnumber: 12347
gidnumber: 123
gecos: William Handset
homedirectory: /home/bhand
loginshell: /bin/csh
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
shadowlastchange: 3440
```

▼ ディレクトリエントリを変更する

ディレクトリエントリを変更するには、ldapmodify(1) を使用します。ldapmodify は、LDAP ディレクトリサーバーとの接続を開き、そのディレク

トリサーバーにバインドして、ディレクトリに対して一連の LDAP 変更処理を実行します。

1. ディレクトリマネージャ **(**パスワード **enigma)** としてバインドし、メールアドレス **eng@ultra.com** を **Joe Qwerty** のエントリに追加します。

```
% ldapmodify -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma < modfile
```

`modfile` の内容は次のとおりです。

```
dn: cn=carol,ou=People,o=Ultra Keyboards Inc.,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}mgq25KV6CE0p6
-
replace: objectclass
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
-
add: shadowlastchange
shadowlastchange: 6447
-

dn: cn=stephen,ou=People,o=Ultra Keyboards Inc.,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}w.4P1JPV3w.Zs
-
replace: objectclass
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
-
add: shadowlastchange
shadowlastchange: 6447
-

dn: cn=frank,ou=People,o=Ultra Keyboards Inc.,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}mMBEaHRlf5rJQ
-
replace: objectclass
```

(続く**)**

```
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
-
add: shadowlastchange
shadowlastchange: 9712
-
```

注 - ハイフンだけの行は、同一ディレクトリエントリに対する一連の変更コマンドを区切ります。空白行は、異なるディレクトリエントリを区切ります。

正常に終了すると、`ldapmodify` は次のようなメッセージを表示します。

```
# ldapmodify -D "cn=Directory Manager" -w nssecret -f domain.ldif
modifying entry dc=sun,dc=com
```

処理が正常に終了しなかった場合は、エラーメッセージが表示されます。

▼ ディレクトリにエントリを追加する

ディレクトリにエントリを追加するには、`ldapadd(1)` を使用します。`ldapadd` は、LDAP ディレクトリサーバーへの接続を開き、そのディレクトリサーバーにバインドして、ディレクトリに対して一連の LDAP 追加処理を実行します。

1. ディレクトリマネージャ **(**パスワード **enigma)** としてバインドし、**Penny Gold** と **Amy Lamb** のエントリを追加します。

```
% ldapadd -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma < addfile
```

`addfile` の内容は次のとおりです。

```
dn: cn=Penny Gold, o=Ultra Keyboards Inc., c=US
changetype: add
objectclass: top
objectclass: person
objectclass: inetOrgPerson
cn: Penny Gold
sn: Gold
mail: pgold@ultra.com

dn: cn=Amy Lamb, o=Ultra Keyboards Inc., c=US
changetype: add
objectclass: top
objectclass: person
objectclass: inetOrgPerson
cn: Amy Lamb
sn: Lamb
mail: alamb@ultra.com
```

▼ ディレクトリからエントリを削除する

ディレクトリからエントリを削除するには、ldapdelete(1) を使用します。ldapdelete は、LDAP ディレクトリサーバーへの接続を開き、そのディレクトリサーバーにバインドして、ディレクトリに対して 1 つ以上の LDAP エントリ削除処理を実行します。

1. ディレクトリマネージャ **(**パスワード **enigma)** としてバインドし、**Penny Gold** のエントリを削除します。

```
% ldapdelete -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma "cn=Penny Gold, o=Ultra Keyboards Inc., c=US"
```

ldapdelete は、正常終了時には何も表示しません。異常終了時にはエラーメッセージを表示します。

▼ ディレクトリエントリの名前を変更する

既存のディレクトリエントリの名前を変更するには、ldapmodrdn(1) を使用します。ldapmodrdn は、LDAP ディレクトリサーバーへの接続を開き、そのディレク

トリサーバーにバインドして、ディレクトリに対して 1 つ以上の LDAP RDN 変更 (名前変更) 処理を実行します。

1. ディレクトリマネージャ **(**パスワード **enigma)** としてバインドし、**RDN cn** 値を、**User Interface** から **Ergonomic** に変更します。

```
% ldapmodrdn -r -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma "cn=User Interface, o=Ultra Keyboards Inc., \
c=US" "cn=Ergonomic"
```

`ldapmodrdn` は、正常終了時には何も表示しません。異常終了時にはエラーメッセージを表示します。

第 3 章

iPlanet Directory Server の設定

この章では、Solaris LDAP クライアントが名前情報を検索できるように、iPlanet Directory Server を設定する方法について説明します。この章の説明は、iPlanet Directory Server バージョン 4.11 に基づいています。

iPlanet Directory Server を使用する場合は、iPlanet に関する次のマニュアルを照してください。

- 『*Netscape Directory Server Schema Reference Guide*』
- 『*Netscape Server Deployment Manual*』
- 『*Managing Servers with Netscape Console 4.0*』
- 『*Directory Server Administrators Guide*』

この章の内容は次のとおりです。

構成ディレクトリへのオブジェクトクラス定義の追加

▼ 環境を準備する

1. ディレクトリサーバーを停止します。

▼ slapd.oc.conf ファイルを変更する

1. **ipNetwork** オブジェクトクラスを変更し、**cn** を必須ではなくしますが、メンバーのままにしておきます。

   変更前の ipNetwork は次のとおりです。

```
objectclass ipNetwork
    oid
         1.3.6.1.1.1.2.7
    requires
         objectClass,
         ipNetworkNumber,
         cn
    allows
         ipNetmaskNumber,
         manager,
         l,
         description
```

   cn 行を requires から削除し、allows に追加します。変更後の ipNetwork は次のようになります。

```
objectclass ipNetwork
    oid
         1.3.6.1.1.1.2.7
    requires
         objectClass,
         ipNetworkNumber
    allows
         cn,
         ipNetmaskNumber,
         manager,
         l,
```

(続く)

```
        description
```

▼ オブジェクトクラス定義を slapd.user_oc.conf に追加する

1. **NisKeyObject** オブジェクトクラスを追加します。

```
# NIS publickey objectclass
objectclass NisKeyObject
        oid 1.3.6.1.1.1.2.14
        superior top
        requires
                cn,
                nisPublickey,
                nisSecretkey
        allows
                uidNumber,
                description
```

2. **nisDomainObject** オブジェクトクラスを追加します。

```
# NIS domain objectclass
objectclass nisDomainObject
        oid 1.3.1.6.1.1.1.2.15
        superior top
        requires
                nisDomain
```

3. **SolarisNamingProfile** オブジェクトクラスを追加します。

```
# LDAP client profile objectclass
objectclass SolarisNamingProfile
        oid 1.3.6.1.4.1.42.2.27.5.2.7
        superior top
        requires
                cn,
                SolarisLDAPservers,
                SolarisSearchBaseDN
        allows
                SolarisBindDN,
                SolarisBindPassword,
                SolarisAuthMethod,
                SolarisTransportSecurity,
                SolarisCertificatePath,
                SolarisDataSearchDN,
                SolarisSearchScope,
                SolarisSearchTimelimit,
                SolarisPreferredServer,
                SolarisPreferredServerOnly,
                SolarisCacheTTL,
                SolarisSearchReferral
```

4. **mailGroup** オブジェクトクラスを追加します。

```
# mailGroup objectclass
objectclass mailGroup
        oid 2.16.840.1.113730.3.2.4
        superior top
        requires
                mail
        allows
                cn,
                mgrpRFC822MailMember
```

5. **nisMailAlias** オブジェクトクラスを追加します。

```
# nisMailAlias objectclass
objectClass nisMailAlias
        oid 1.3.6.1.4.1.42.2.27.1.2.5
        superior top
        requires
                cn
        allows
```

(続く)

```
                rfc822mailMember
```

6. **nisNetId** オブジェクトクラスを追加します。

```
# nisNetId objectclass
objectClass nisNetId
        oid 1.3.6.1.4.1.42.2.27.1.2.6
        superior top
        requires
                cn
        allows
                nisNetIdUser,
                nisNetIdGroup,
                nisNetIdHost
```

7. **SolarisAuditUser** オブジェクトクラスを追加します。

```
# User auditing objectclass
objectclass SolarisAuditUser
        oid 1.3.6.1.4.1.42.2.27.5.2.2
        superior top
        allows
                SolarisAuditAlways,
                SolarisAuditNever
```

8. **SolarisUserAttr** オブジェクトクラスを追加します。

```
# RBAC User attributes objectclass
objectclass SolarisUserAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.3
        superior top
        allows
                SolarisUserQualifier,
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisAttrKeyValue
```

9. **SolarisAuthAttr** オブジェクトクラスを追加します。

```
# RBAC Authorizations Objectclass
objectclass SolarisAuthAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.4
        superior top
        requires
                cn
        allows
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisAttrShortDesc,
                SolarisAttrLongDesc,
                SolarisAttrKeyValue
```

10. **SolarisProfAttr** オブジェクトクラスを追加します。

```
# RBAC Profile objectclass
objectClass SolarisProfAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.5
        superior top
        requires
                cn
        allows
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisAttrLongDesc,
                SolarisAttrKeyValue
```

11. **SolarisExecAttr** オブジェクトクラスを追加します。

```
# RBAC Execution objectlcass
objectClass SolarisExecAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.6
        superior top
        allows
                SolarisKernelSecurityPolicy,
                SolarisProfileType,
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisProfileID,
                SolarisAttrKeyValue
```

12. **nisKeyObject** オブジェクトクラスを追加します。

```
# Publickey objectclass
objectClass nisKeyObject
        oid 1.3.6.1.1.1.2.14
        superior top
        requires
                cn,
                nisPublicKey,
                nisSecretKey
        allows
                uidNumber,
                description
```

13. **SolarisProject** オブジェクトクラスを追加します。

```
# Project Accounting objectclass
objectclass SolarisProject
        oid 1.3.6.1.4.1.42.2.27.5.2.1
        superior top
        requires
                SolarisProjectID,
                SolarisProjectName
        allows
                memberUid,
                memberGid,
                description,
                SolarisProjectAttr
```

▼ slapd.user_at.conf ファイルに属性定義を追加する

1. **nisMapEntry** 属性を追加します。

```
# Sun nisMapEntry attributes
attribute nisDomain      1.3.6.1.1.1.1.30   cis
```

2. **LDAP** クライアントプロファイル属性を追加します。

```
# attributes for LDAP client profile
attribute SolarisLDAPServers     1.3.6.1.4.1.42.2.27.5.1.15   cis
attribute SolarisSearchBaseDN    1.3.6.1.4.1.42.2.27.5.1.16   dn single
attribute SolarisCacheTTL        1.3.6.1.4.1.42.2.27.5.1.17   cis  single
attribute SolarisBindDN          1.3.6.1.4.1.42.2.27.5.1.18   dn single
attribute SolarisBindPassword    1.3.6.1.4.1.42.2.27.5.1.19   ces  single
attribute SolarisAuthMethod      1.3.6.1.4.1.42.2.27.5.1.20   cis
attribute SolarisTransportSecurity  1.3.6.1.4.1.42.2.27.5.1.21   cis
attribute SolarisCertificatePath 1.3.6.1.4.1.42.2.27.5.1.22   ces single
attribute SolarisDataSearchDN    1.3.6.1.4.1.42.2.27.5.1.24   cis
attribute SolarisSearchScope     1.3.6.1.4.1.42.2.27.5.1.25   cis single
attribute SolarisSearchTimeLimit 1.3.6.1.4.1.42.2.27.5.1.26   int single
attribute SolarisPreferredServer 1.3.6.1.4.1.42.2.27.5.1.27   cis
attribute SolarisPreferredServerOnly 1.3.6.1.4.1.42.2.27.5.1.28 cis single
attribute SolarisSearchReferral  1.3.6.1.4.1.42.2.27.5.1.29   cis single
```

3. **mailGroup** 属性を追加します。

```
# Sun additional attributes to RFC2307 attributes (NIS)
attribute mgrpRFC822MailMember   2.16.840.1.113730.3.1.30     cis
attribute rfc822MailMember                                    ces
```

4. **nisKeyObject** 属性を追加します。

```
# Sun nisKeyObject attributes
attribute nisPublickey    1.3.6.1.1.1.1.28    cis
attribute nisSecretkey    1.3.6.1.1.1.1.29    cis
```

5. **nisNetId** 属性を追加します。

```
# Sun nisNetId attributes
attribute nisNetIdUser    1.3.6.1.4.1.42.2.27.1.1.12    ces
attribute nisNetIdGroup   1.3.6.1.4.1.42.2.27.1.1.13    ces
attribute nisNetIdHost    1.3.6.1.4.1.42.2.27.1.1.14    ces
```

6. 監査属性を追加します。

```
# attributes for auditing
attribute SolarisAuditAlways   1.3.6.1.4.1.42.2.27.5.1.5   cis single
attribute SolarisAuditNever    1.3.6.1.4.1.42.2.27.5.1.6   cis single
```

7. **RBAC** 属性を追加します。

```
# attributes for RBAC
attribute SolarisAttrKeyValue   1.3.6.1.4.1.42.2.27.5.1.4   cis single
attribute SolarisAttrShortDesc  1.3.6.1.4.1.42.2.27.5.1.7   cis single
attribute SolarisAttrLongDesc   1.3.6.1.4.1.42.2.27.5.1.8   cis single
attribute SolarisKernelSecurityPolicy  1.3.6.1.4.1.42.2.27.5.1.9
   cis single
attribute SolarisProfileType    1.3.6.1.4.1.42.2.27.5.1.10  cis single
attribute SolarisProfileId      1.3.6.1.4.1.42.2.27.5.1.11  ces single
attribute SolarisUserQualifier  1.3.6.1.4.1.42.2.27.5.1.12  cis single
attribute SolarisAttrReserved1  1.3.6.1.4.1.42.2.27.5.1.13  cis single
attribute SolarisAttrReserved2  1.3.6.1.4.1.42.2.27.5.1.14  cis single
```

8. **nisKeyObject** 属性を追加します。

```
# attributes for nisKeyObject
attribute nisPublicKey    1.3.6.1.1.1.1.28    cis
attribute nisSecretKey    1.3.6.1.1.1.1.29    cis
```

9. プロジェクトアカウント属性を追加します。

```
# attributes for Project Accounting
attribute SolarisProjectID     1.3.6.1.4.1.42.2.27.5.1.1    int single
attribute SolarisProjectName   1.3.6.1.4.1.42.2.27.5.1.2    ces single
attribute SolarisProjectAttr   1.3.6.1.4.1.42.2.27.5.1.3    ces
```

```
attribute memberGid             1.3.6.1.4.1.42.2.27.5.1.30   ces
```

ディレクトリサーバーへのデータの読み込み

ディレクトリサーバーに Unix Crypt 形式でパスワードを格納していない場合、ディレクトリサーバーを構成してパスワードを格納します。Unix Crypt 形式のパスワードの設定方法についての詳細は、iPlanet のマニュアルを参照してください。

▼ ACI を設定する

1. ツリーのトップエントリの **ACI (**アクセス制御情報**)** を設定します。この **ACI** は所有者が自分のエントリを変更する権限を制御します。たとえば、デフォルトの **ACI** ではユーザーが自分のホームディレクトリを変更できますが、次の例のように変更した **ACI** では自分のホームディレクトリを変更できません。各自の環境に合わせて **ACI** を設定してください。

   次に示すようなツリーのトップエントリである「Allow self entry modification」ACI を、

```
aci=(targetattr = "*")(version 3.0; acl "Allow self entry modification";
allow (write)userdn = "ldap:///self";)
```

   次のように変更します。

```
aci=(targetattr!="cn || uid || uidNumber || gidNumber || homeDirectory
|| shadowLastChange || shadowMin || shadowMax || shadowWarning ||
shadowInactive || shadowExpire || shadowFlag || memberUid")
(version 3.0; acl "Allow self entry modification"; allow
(write) userdn = "ldap:///self"; )
```

注 - ユーザーが変更すべきでない属性 (uid など) の変更許可を与えないでください。それを許可すると、uid を 0 に設定することによって、ユーザーがスーパーユーザーになることができるようになってしまいます。

▼ ネームコンテナのエントリを追加する

ネームコンテナの一覧は、26ページの「ディレクトリ情報ツリー」を参照してください。

注 - 次のコンテナエントリは、28ページの「NIS ドメイン」で使用した nisDomain の例に基づいています。実際に使用するときは、これらのコンテナエントリを各自の環境に合わせて変更してください。

---

1. ドメインエントリを追加します。

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
associatedDomain: mkt.mainstore.com
objectClass: top
objectClass: domain
objectClass: domainRelatedObject
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
```

2. ネームコンテナのエントリを追加します。

```
dn: ou=people,dc=mkt,dc=mainstore,dc=com
ou: people
objectClass: top
objectClass: organizationalUnit

dn: ou=Group,dc=mkt,dc=mainstore,dc=com
ou: Group
objectClass: top
objectClass: organizationalUnit

dn: ou=rpc,dc=mkt,dc=mainstore,dc=com
ou: rpc
objectClass: top
objectClass: organizationalUnit

dn: ou=protocols,dc=mkt,dc=mainstore,dc=com
ou: protocols
objectClass: top
objectClass: organizationalUnit

dn: ou=networks,dc=mkt,dc=mainstore,dc=com
ou: networks
objectClass: top
objectClass: organizationalUnit

dn: ou=netgroup,dc=mkt,dc=mainstore,dc=com
ou: netgroup
objectClass: top
objectClass: organizationalUnit

dn: ou=aliases,dc=mkt,dc=mainstore,dc=com
ou: aliases
objectClass: top
objectClass: organizationalUnit

dn: ou=Hosts,dc=mkt,dc=mainstore,dc=com
ou: Hosts
objectClass: top
objectClass: organizationalUnit

dn: ou=services,dc=mkt,dc=mainstore,dc=com
ou: services
objectClass: top
objectClass: organizationalUnit

dn: ou=Ethers,dc=mkt,dc=mainstore,dc=com
ou: Ethers
objectClass: top
objectClass: organizationalUnit

dn: ou=profile,dc=mkt,dc=mainstore,dc=com
ou: profile
objectClass: top
objectClass: organizationalUnit
```

(続く)

```
dn: nismapname=auto_home,dc=mkt,dc=mainstore,dc=com
nismapname: auto_home
objectClass: top
objectClass: nisMap

dn: nismapname=auto_direct,dc=mkt,dc=mainstore,dc=com
nismapname: auto_direct
objectClass: top
objectClass: nisMap

dn: nismapname=auto_master,dc=mkt,dc=mainstore,dc=com
nismapname: auto_master
objectClass: top
objectClass: nisMap

dn: nismapname=auto_shared,dc=mkt,dc=mainstore,dc=com
nismapname: auto_shared
objectClass: top
objectClass: nisMap
```

▼ パフォーマンスと上限に関するパラメータを設定する

これらのパラメータの値は、読み込むデータ量、データの使用パターン、使用するハードウェアなどによってサーバーごとに異なります。

1. キャッシュの最大エントリ数、キャッシュの最大サイズ (バイト) を設定し、上限値を確認します。

   システムで使用できるメモリーおよびディスク容量に合わせて、キャッシュパラメータを変更してください。

2. サイズおよび時間の上限パラメータを、環境に合わせて設定します。

   `sizelimit` および `timelimit` を `-1` に指定すると、取り得る上限値として設定されます。各自のシステムに合わせて値を選択してください。

▼ プロキシエージェントにパスワードの読み取り権を与える

注 - 次のプロキシエージェント ACI (アクセス制御情報) は、28ページの「NIS ドメイン」で使用した nisDomain の例に基づいています。実際に使用するときは、これらのプロキシエージェント ACI を各自の環境に合わせて変更してください。

---

1. すべてのクライアントで認証に pam_unix を使用する場合は、ldapmodify を使用して検索起点識別名に読み取り **ACI** を設定することによって、プロキシエージェントにパスワードの読み取り権を与えます。

```
#ldapmodify -D "cn=Directory Manager" -w nssecret -f aci.ldif
```

aci.ldif の内容は次のとおりです。

```
dn: dc=mkt,dc=mainstore,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=mkt,dc=mainstore,dc=com")
 (targetattr="userPassword")(version 3.0; acl "password read";
 allow (compare,read,search) userdn = "ldap:///cn=proxyagent,
 ou=profile,dc=mkt,dc=mainstore,dc=com"; )
```

2. ldapsearch を使用して新しい **ACI** 設定を確認します。

ldapsearch は変更された ACI を表示します。

```
#ldapsearch -L -h <servername> -b "dc=mkt,dc=mainstore,dc=com" \
  -s base "objectclass=*"
```

ldapsearch によって返される変更後の ACI は、次のようになります。

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
associateddomain: mkt.mainstore.com
objectclass: top
objectclass: domain
objectclass: domainRelatedObject
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
aci: (target="ldap:///dc=mkt,dc=mainstore,dc=com")
 (targetattr="userPassword")(version 3.0; acl "password read";
 allow (compare,read,search) userdn = "ldap:///cn=proxyagent,
 ou=profile,dc=mkt,dc=mainstore,dc=com"; )
```

pam_ldap 認証はサーバー側で実行されるため、プロキシエージェントにパスワード属性の読み取り権を与える必要はありません。pam_ldap についての詳細は、33ページの「PAM」を参照してください。

▼ NIS データを LDIF 書式に変換する

NIS(YP) 環境から LDAP 環境に移行するには、dsimport を使用して NIS データを LDIF 書式に変換します。dsimport は NIS 拡張の一部で、iPlanet Advantage Software CD vol.1 に収録されています。マニュアルは、次の Web サイトで参照できます。http://docs.iplanet.com/docs/manuals/directory.html

1. **NIS** パスワードを **LDIF** 書式に変換します。

```
# cat passwd.nis | dsimport -n -m nis.mapping -t passwd \
 -M SIMPLE -D "" -w "" >passwd.ldif
```

passwd.ldif ファイルを LDAP サーバーに読み込みます。

2. **NIS** グループデータを **LDIF** 書式に変換します。

```
# cat group.nis | dsimport -n -m nis.mapping -t group \
-M SIMPLE -D "" -w "" > group.ldif
```

group.ldif を LDAP サーバーに読み込みます。

3. 上記の手順を繰り返して、すべてのネームコンテナファイルを変換します。

4. **ns-slapd** の ldif2db コマンドまたは ldapadd コマンドを使用して、**LDIF** 書式ファイルをディレクトリデータベースにインポートします。

   ns-slapd ldif2db コマンドについての詳細は、『*Directory Server Administrator's Guide*』の「Managing Directory Server Databases」を参照してください。ldapadd については、ldapadd(1) のマニュアルページを参照してください。

注 - ファイルデータを LDIF 書式に変換するために、dsimport は、エントリの格納方法を定義するマップファイルを変更する必要があります。

▼ インデックスを作成して検索パフォーマンスを向上させる

注 - インデックスの作成方法については、『*iPlanet Directory Server Administrator's Guide*』の「Managing Indexes」を参照してください。

---

1. 次の **Solaris** クライアント属性にインデックスを付けます。

```
membernisnetgroup    pres,eq,sub
nisnetgrouptriple    pres,eq,sub
memberuid            pres,eq
macAddress           pres,eq
uid                  pres,eq
uidNumber            pres,eq
```

```
gidNumber            pres,eq
ipHostNumber         pres,eq
ipNetworkNumber      pres,eq
ipProtocolNumber     pres,eq
oncRpcNumber         pres,eq
ipServiceProtocol    pres,eq
ipServicePort        pres,eq
nisDomain            pres,eq
nisMapName           pres,eq
mail                 pres,eq
```

2. ldapsearch を使用して、**VLV** 制御タイプが **1.2.840.113556.1.4.473**、**VLV** 制御値が **2.16.840.1.113730.3.4.9** の **OID** によって特定される仮想リスト表示 **(VLV)** を、ディレクトリがサポートしているかどうかを確認します。

```
# ldapsearch -b "" -s base objectclass=\*
```

ldapsearch は次の検索結果を返します。

```
objectclass=top
namingcontexts=dc=sun,dc=com
namingcontexts=o=NetscapeRoot
subschemasubentry=cn=schema
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=2.16.840.1.113730.3.4.4
supportedcontrol=2.16.840.1.113730.3.4.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=2.16.840.1.113730.3.4.9
supportedcontrol=2.16.840.1.113730.3.4.12
supportedsaslmechanisms=EXTERNAL
supportedldapversion=2
supportedldapversion=3
dataversion=atitrain2.east.sun.com:389 020000605172910
netscapemdsuffix=cn=ldap://:389,dc=atitrain2,dc=east,dc=sun,dc=com
```

3. 次の **VLV** 属性にインデックスを付けます。

```
getpwent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getspent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getgrent:      vlvFilter: (objectclass=posixGroup),       vlvScope: 1
gethostent:    vlvFilter: (objectclass=ipHost),           vlvScope: 1
getnetent:     vlvFilter: (objectclass=ipNetwork),        vlvScope: 1
getprotoent:   vlvFilter: (objectclass=ipProtocol),       vlvScope: 1
getrpcent:     vlvFilter: (objectclass=oncRpc),           vlvScope: 1
getaliasent:   vlvFilter: (objectclass=rfc822MailGroup),  vlvScope: 1
getserviceent: vlvFilter: (objectclass=ipService),        vlvScope: 1
```

ツリー内の ou のうち、多数のオブジェクトを含むかアクセス頻度の高いものに、これらのインデックスを作成します。

4. パスワードエントリ **(getpwent)** については、ディレクトリに次のエントリを追加します。

```
dn: cn=getpwent,cn=config,cn=ldbm
objectclass: top
objectclass: vlvSearch
cn: getpwent
vlvBase: ou=people,dc=eng,dc=sun,dc=com
vlvScope: 1
vlvFilter: (objectclass=posixAccount)
aci: (target="ldap:///cn=getpwent,cn=config,cn=ldbm")(targetattr="*")
 (version 3.0; acl "Config";allow(read,search,compare)userdn="ldap:///
anyone";)

dn: cn=getpwent,cn=getpwent,cn=config,cn=ldbm
cn: getpwent
vlvSort: cn uid
objectclass: top
objectclass: vlvIndex
```

5. **getpwent** の **VLV** インデックスを作成します。

```
# cd /usr/netscape/server4/slapd*
# ./vlvindex getpwent
OK# ./vlvindex getgrent
OK# ./vlvindex gethostent
OK# ./vlvindex getspent
OK#
# ./vlvindex
[05/Jun/2000:15:34:31 -0400] - ldbm2index: Unknown VLV Index named ''
```

(続く)

```
[05/Jun/
2000:15:34:31 -0400] - ldbm2index: Known VLV Indexes are: 'getgrent',
'gethostent', 'getnetent', 'getpwent', 'getspent',
```

6. 残りの **VLV** 属性について、手順 **4** と **5** を繰り返します。

▼ すべてのユーザーに VLV 要求制御の読み取り権/検索権/比較権を与える

1. `ldapsearch` を使用して、**VLV** 制御 **ACI** を表示します。

```
#ldapsearch -D "cn=Directory Manager" -w nssecret -b cn=features, \
cn=config objectclass=\*
```

検索結果は次のとおりです。

```
cn=features,cn=config
objectclass=top
cn=features

cn=options,cn=features,cn=config
objectclass=top
cn=options

oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectclass=top
objectclass=directoryServerFeature
oid=2.16.840.1.113730.3.4.9
cn=VLV Request Control
aci=(targetattr != "aci")(version 3.0; acl "VLV Request \
 Control"; allow( read,
```

(続く)

```
search, compare ) userdn = "ldap:///all";)
```

2. ldapmodify を使用して、すべてのユーザーに、**VLV** 機能の読み取り/検索/比較の権限を与えます。これにより、**anonymous (**匿名**)** による検索で **VLV** 制御を使用しても失敗しなくなります。

```
#ldapmodify -D "cn=Directory Manager" -w nssecret -f vlvcntrl.ldif
```

vlvcntrl.ldif の内容は次のとおりです。

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
changetype: modify
replace: aci
aci: (targetattr !="aci")(version 3.0; acl "VLV Request Control";
 allow (compare,read,search) userdn = "ldap:///anyone"; )
```

3. ldapsearch を使用して、変更された **VLV** 制御 **ACI** を表示します。

```
#ldapsearch -L -b "cn=features,cn=config" -s one \
oid=2.16.840.1.113730.3.4.9
```

ldapsearch が返す ACI は次のようになります。

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectclass: top
objectclass: directoryServerFeature
oid: 2.16.840.1.113730.3.4.9
cn: VLV Request Control
aci: (targetattr !="aci")(version 3.0; acl "VLV Request Control";
 allow (compare,read,search) userdn = "ldap:///anyone"; )
```

▼ LDAP サーバーに proxyagent エントリを追加する

注 - この手順は、proxyagent エントリを使用する場合だけ必要になります。

---

1. **proxyagent** エントリを **LDAP** サーバーに追加します。

```
#ldapadd -D "cn=Directory Manager" -w nssecret -f proxyagent.ldif
```

`proxyagent.ldif` の内容は次のとおりです。

```
dn: cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com
cn: proxyagent
sn: proxyagent
objectclass: top
objectclass: person
userpassword: proxy_agent_password
```

注 - ou は、ou=profile または ou=person に設定できます。

▼ クライアントプロファイルを生成する

1. クライアントプロファイルを生成し、それを **LDAP** サーバーに追加します。

```
ldap_gen_profile -P profile -b baseDN -D bindDN \
  -w bindDNpasswd ldapServer_IP_address(es)[:port#]
```

`bindDN` はプロキシエージェントのバインド DN です。障害発生時に他の LDAP サーバーで処理を継続することを可能にする場合は、2 つ以上の LDAP サーバー

の IP アドレスを指定します。上のコマンドの実行結果を、profile.ldif などの名前のファイルに格納します。

典型的なコマンド行を示します。

```
ldap_gen_profile -P myProfile -b "dc=mkt,dc=mainstore,dc=com" \
-D "cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com" \
-w proxy_agent_pswd -a simple 100.100.100.100 > profile.ldif
```

2. このクライアントプロファイルを **LDAP** サーバーに追加して、クライアントがダウンロードできるようにします。

第 4 章

クライアントの設定

この章では、LDAP ネームサービスを使用するための Solaris クライアントの設定方法について説明します。

概要

Solaris クライアントを LDAP クライアントにした場合、NIS/NIS+ または NFS を使用した Solaris クライアントと同様に動作します。このクライアントは強固な検索を実行します。つまり、`getXXbyYY()` 呼び出しは、応答が返ってくるまで待ちます。通常 NIS(YP) では、ローカルのサブネット上にサーバーを置きますが、これはブロードキャストを使用するように設定されているためです。Solaris 2.0 から、NIS (YP) サーバーがあまり使用されていないにしても、ローカルサブネットの外にあるサーバーを使用できるようになりました。NIS+ はローカルサーバーを使用しないように設定するのが普通です。LDAP は、非ローカルサーバーを使用することが多いという点で、NIS+ に似ています。

したがって、クライアントマシンを動作させるには、ルーターが不可欠になります。

クライアントは、少なくとも 1 つの LDAP サーバーに常にアクセスできるようにしておく必要があります。それには、ネットワークの適度な信頼性を確保するか (ケーブルを切断したり、ルーターの電源を落としたりしない限り、ほとんどのネットワークには適度な信頼性があります)、サーバーをローカルサブネット上に置くようにします。ただし、サーバーをローカルサブネット上に置き、イーサネットスイッチ経由で接続したとしても、ケーブル切断や電源断にはやはり対処できません。

クライアントを常に機能させるには、複数のサーバーを配備し、それらのサーバーを最新の状態に (同じデータが保持されるように) し、クライアントがすべてのサーバーにアクセスできるようにするのが最善の方法です。サーバー優先機能を使用して自分のクライアントを特定のサーバー群に強制的にバインドしている場合は、それらのサーバーが同じ基準を満たしている必要があります。

完全指定ドメイン名 (FQDN)

LDAP クライアントと NIS/NIS+ クライアントの大きな違いは、LDAP クライアントは常に FQDN (完全指定ドメイン名) を返す点で、これは DNS に似ています。たとえば、ドメイン名が `engineering.example.net` のとき、ホスト名 `server` を `getipnodebyname()` で検索したとします (現行の LDAP は IPv4 上でのみ動作しますが、IPv6 への移行に備えて、この API を使用します)。`gethostbyname()` と `getipnodebyname()` はどちらも、`server.engineering.example.net` のように FQDN を返します。また、`server-#` のようなインタフェース固有の別名を使用した場合も、FQDN の長いリストが返されます。

ホスト名を使用してファイルシステムを共有したり、他にこのような検査を行うアプリケーションを使用している場合は、この鍵となる違いを理解し、対処する必要があります。特に、ローカルホストは非 FQDN、(DNS で解決された) リモートホストは FQDN と考えている場合は注意が必要です。DNS と異なるドメイン名を使用して LDAP を設定すると、同じホストでも検索元によって FQDN が異なることがあります。

ldap_cachemgr デーモン

`ldap_cachemgr(1M)` は、LDAP クライアント上で動作するデーモンです。このデーモンは、構成ファイル内の情報をサーバーからの情報で更新します。

ldap_cachemgr が実行されていないと、構成は更新されません。

ldap_cachemgr は、更新機能の他に、更新照会中の不正な構文を検出する堅牢な解析機能を備えています。

NIS/NIS+ から LDAP への移行

NIS/NIS+ クライアントであったマシンを SunOS 5.8 (Solaris 8) にアップグレードして、LDAP クライアントにするには、ldapclient(1M) を実行します。

ldapclient を実行するには、プロファイル名と最低 1 つのサーバーの IP アドレスが必要です。次の例では、プロファイル名が myprofile、LDAP サーバーが IP アドレス 100.100.100.100 上でデフォルトの LDAP ポート番号 389 を使用して動作しているものとします。

▼ LDAP クライアントを作成する

1. スーパユーザーになります。

2. ldapclient(1M) を実行します。

```
# ldapclient -P myprofile 100.100.100.100
```

   ldapclient は構成ファイルを作成し、このクライアントが LDAP を使用してネームサービス検索を行うように /etc/nsswitch.conf ファイルを変更します。

3. クライアントをリブートします。

ログインすると、ldap で認証されます。

ldaplist コマンド

ldaplist は、LDAP サーバーの名前情報をリストするユーティリティです。詳細は、ldaplist(1) のマニュアルページを参照してください。

▼ LDAP サーバーの名前情報をリストする

1. **起点識別名 (baseDN) のコンテナをリストします。**

```
# ldaplist hosts myhost
dn: cn=myhost+ipHostNumber=100.100.100.100,ou=Hosts,
dc=mkt,dc=mainstore,dc=com
```

引数を省略すると、ldaplist は、現在の検索 baseDN 内のすべてのコンテナを返します。

付録 **A**

スキーマ

Solaris LDAP ネームサービスクライアントをサポートするには、Solaris 固有のスキーマと IETF 定義のスキーマがいくつか必要です。

この付録の内容は次のとおりです。

IETF スキーマ

LDAP は IETF 定義の 2 つのスキーマを必要とします。改訂版 RFC 2307 NIS スキーマと LDAP メールグループインターネットドラフトの 2 つです。

RFC 2307 Network Information Service スキーマ

LDAP サーバーは改訂版 RFC 2307 をサポートするように構成する必要があります。

nisSchema OID は 1.3.6.1.1 です。RFC 2307 の属性は、次のとおりです。

```
( nisSchema.1.0 NAME 'uidNumber'
DESC 'An integer uniquely identifying a user in an
  administrative domain'
EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )
```

(続く)

```
( nisSchema.1.1 NAME 'gidNumber'
DESC 'An integer uniquely identifying a group in an
  administrative domain'
EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.2 NAME 'gecos'
DESC 'The GECOS field; the common name'
EQUALITY caseIgnoreIA5Match
SUBSTRINGS caseIgnoreIA5SubstringsMatch
SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.3 NAME 'homeDirectory'
DESC 'The absolute path to the home directory'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.4 NAME 'loginShell'
DESC 'The path to the login shell'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.5 NAME 'shadowLastChange'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.6 NAME 'shadowMin'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.7 NAME 'shadowMax'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.8 NAME 'shadowWarning'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.9 NAME 'shadowInactive'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.10 NAME 'shadowExpire'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.11 NAME 'shadowFlag'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.12 NAME 'memberUid'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
```

(続く)

```
SYNTAX 'IA5String' )

( nisSchema.1.13 NAME 'memberNisNetgroup'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String' )

( nisSchema.1.14 NAME 'nisNetgroupTriple'
DESC 'Netgroup triple'
SYNTAX 'nisNetgroupTripleSyntax' )

( nisSchema.1.15 NAME 'ipServicePort'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.16 NAME 'ipServiceProtocol'
SUP name )

( nisSchema.1.17 NAME 'ipProtocolNumber'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.18 NAME 'oncRpcNumber'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.19 NAME 'ipHostNumber'
DESC 'IP address as a dotted decimal, eg. 192.168.1.1
      omitting leading zeros'
SUP name )

( nisSchema.1.20 NAME 'ipNetworkNumber'
DESC 'IP network as a dotted decimal, eg. 192.168,
      omitting leading zeros'
SUP name SINGLE-VALUE )

( nisSchema.1.21 NAME 'ipNetmaskNumber'
DESC 'IP netmask as a dotted decimal, eg. 255.255.255.0,
       omitting leading zeros'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}' SINGLE-VALUE )

( nisSchema.1.22 NAME 'macAddress'
DESC 'MAC address in maximal, colon separated hex
      notation, eg. 00:00:92:90:ee:e2'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}' )

( nisSchema.1.23 NAME 'bootParameter'
DESC 'rpc.bootparamd parameter'
SYNTAX 'bootParameterSyntax' )

( nisSchema.1.24 NAME 'bootFile'
```

(続く)

```
DESC 'Boot image name'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' )

( nisSchema.1.26 NAME 'nisMapName'
SUP name )

( nisSchema.1.27 NAME 'nisMapEntry'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String{1024}' SINGLE-VALUE )

( nisSchema.1.28 NAME 'nisPublicKey'
DESC 'NIS public key'
SYNTAX 'nisPublicKeySyntax' )

( nisSchema.1.29 NAME 'nisSecretKey'
DESC 'NIS secret key'
SYNTAX 'nisSecretKeySyntax' )

( nisSchema.1.30 NAME 'nisDomain'
DESC 'NIS domain'
SYNTAX 'IA5String' )
```

nisSchema OID は 1.3.6.1.1.1 です。RFC 2307 定義のオブジェクトクラスは、次のとおりです。

```
( nisSchema.2.0 NAME 'posixAccount' SUP top AUXILIARY
  DESC 'Abstraction of an account with POSIX attributes'
  MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
  MAY ( userPassword $ loginShell $ gecos $ description ) )

( nisSchema.2.1 NAME 'shadowAccount' SUP top AUXILIARY
  DESC 'Additional attributes for shadow passwords'
  MUST uid
  MAY ( userPassword $ shadowLastChange $ shadowMin
        shadowMax $ shadowWarning $ shadowInactive $
        shadowExpire $ shadowFlag $ description ) )

( nisSchema.2.2 NAME 'posixGroup' SUP top STRUCTURAL
  DESC 'Abstraction of a group of accounts'
  MUST ( cn $ gidNumber )
  MAY ( userPassword $ memberUid $ description ) )

( nisSchema.2.3 NAME 'ipService' SUP top STRUCTURAL
  DESC 'Abstraction an Internet Protocol service.
        Maps an IP port and protocol (such as tcp or udp)
        to one or more names; the distinguished value of
        the cn attribute denotes the service's canonical
```

(続く)

```
        name'
  MUST ( cn $ ipServicePort $ ipServiceProtocol )
  MAY ( description ) )

( nisSchema.2.4 NAME 'ipProtocol' SUP top STRUCTURAL
  DESC 'Abstraction of an IP protocol. Maps a protocol number
        to one or more names. The distinguished value of the cn
        attribute denotes the protocol's canonical name'
  MUST ( cn $ ipProtocolNumber )
  MAY  description )

( nisSchema.2.5 NAME 'oncRpc' SUP top STRUCTURAL
  DESC 'Abstraction of an Open Network Computing (ONC)
        [RFC1057] Remote Procedure Call (RPC) binding.
        This class maps an ONC RPC number to a name.
        The distinguished value of the cn attribute denotes
        the RPC service's canonical name'
  MUST ( cn $ oncRpcNumber $ description )
  MAY  description )

( nisSchema.2.6 NAME 'ipHost' SUP top AUXILIARY
  DESC 'Abstraction of a host, an IP device. The distinguished
        value of the cn attribute denotes the host's canonical
        name. Device SHOULD be used as a structural class'
  MUST ( cn $ ipHostNumber )
  MAY ( l $ description $ manager $ userPassword ) )

( nisSchema.2.7 NAME 'ipNetwork' SUP top STRUCTURAL
  DESC 'Abstraction of a network. The distinguished value of
        the cn attribute denotes the network's canonical name'
  MUST ipNetworkNumber
  MAY ( cn $ ipNetmaskNumber $ l $ description $ manager ) )

( nisSchema.2.8 NAME 'nisNetgroup' SUP top STRUCTURAL
  DESC 'Abstraction of a netgroup. May refer to other netgroups'
  MUST cn
  MAY ( nisNetgroupTriple $ memberNisNetgroup $ description ) )

( nisSchema.2.9 NAME 'nisMap' SUP top STRUCTURAL
  DESC 'A generic abstraction of a NIS map'
  MUST nisMapName
  MAY description )

( nisSchema.2.10 NAME 'nisObject' SUP top STRUCTURAL
  DESC 'An entry in a NIS map'
  MUST ( cn $ nisMapEntry $ nisMapName )
  MAY description )

( nisSchema.2.11 NAME 'ieee802Device' SUP top AUXILIARY
  DESC 'A device with a MAC address; device SHOULD be
        used as a structural class'
  MAY macAddress )
```

(続く)

```
( nisSchema.2.12 NAME 'bootableDevice' SUP top AUXILIARY
  DESC 'A device with boot parameters; device SHOULD be
  used as a structural class'
  MAY ( bootFile $ bootParameter ) )

( nisSchema.2.14 NAME 'nisKeyObject' SUP top AUXILIARY
  DESC 'An object with a public and secret key'
  MUST ( cn $ nisPublicKey $ nisSecretKey )
  MAY ( uidNumber $ description ) )

( nisSchema.2.15 NAME 'nisDomainObject' SUP top AUXILIARY
  DESC 'Associates a NIS domain with a naming context'
  MUST nisDomain )
```

メール別名スキーマ

LDAP サーバーは、メール別名情報をサポートするように構成する必要があります。メール別名情報は、LDAP メールグループインターネットドラフト (`draft-steinback-ldap-mailgroups`) で定義されたスキーマを使用します。新しいスキーマが使用可能になるまで、Solaris LDAP クライアントは、このメール別名情報のスキーマを使用し続けます。

注 - インターネットドラフトとは、最長 6 ヶ月間有効な草稿文書で、他の文書によっていつでも更新、置換、廃止される可能性があります。

インターネットドラフトに定義された LDAP メールグループスキーマには、多数の属性とオブジェクトクラスが含まれています。このうち、Solaris クライアントが使用するのは、2 つの属性と 1 つのオブジェクトクラスだけです。次にそれらを示します。

メール別名の属性は、次のとおりです。

```
( 0.9.2342.19200300.100.1.3
  NAME 'mail'
  DESC 'RFC822 email address for this person'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(256)'
  SINGLE-VALUE )

( 2.16.840.1.113730.3.1.30
  NAME 'mgrpRFC822MailMember'
  DESC 'RFC822 mail address of email only member of group'
  EQUALITY CaseIgnoreIA5Match
```

(続く)

```
  SYNTAX 'IA5String(256)' )
```

メール別名のオブジェクトクラスは、次のとおりです。

```
( 2.16.840.1.113730.3.2.4
  NAME 'mailGroup'
  SUP top
  STRUCTURAL
  MUST mail
  MAY ( cn $ mailAlternateAddress $ mailHost $ mailRequireAuth $
   mgrpAddHeader $ mgrpAllowedBroadcaster $ mgrpAllowedDomain $
   mgrpApprovePassword $ mgrpBroadcasterModeration $ mgrpDeliverTo $
   mgrpErrorsTo $ mgrpModerator $ mgrpMsgMaxSize $
   mgrpMsgRejectAction $ mgrpMsgRejectText $ mgrpNoMatchAddrs $
   mgrpRemoveHeader $ mgrpRFC822MailMember )
)
```

Solaris スキーマ

Solaris オペレーティング環境に必要なスキーマは次の 4 つです。

拡張ユーザーアカウントスキーマ
役割によるアクセス制御スキーマ
Solaris クライアントネーミングプロファイルスキーマ
プロジェクトスキーマ

拡張ユーザーアカウントスキーマ

ユーザーと役割に関する拡張属性のシステムごとの設定は、/etc/user_attr に置かれます。詳細は、user_attr(4) のマニュアルページを参照してください。

拡張ユーザーアカウントの属性は、次のとおりです。

```
( 1.3.6.1.4.1.42.2.27.5.1.1 NAME 'SolarisProjectID'
  DESC 'Unique ID for a Solaris Project entry'
  EQUALITY integerMatch
  SYNTAX INTEGER SINGLE )

( 1.3.6.1.4.1.42.2.27.5.1.2 NAME 'SolarisProjectName'
  DESC 'Name of a Solaris Project entry'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String SINGLE )

( 1.3.6.1.4.1.42.2.27.5.1.3 NAME 'SolarisProjectAttr'
  DESC 'Attributes of a Solaris Project entry'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String )

( 1.3.6.1.4.1.42.2.27.5.1.30 NAME 'memberGid'
  DESC 'Posix Group Name'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' )
```

拡張ユーザーアカウントのオブジェクトクラスは、次のとおりです。

```
( 1.3.6.1.4.1.42.2.27.5.2.1 NAME 'SolarisProject'
  SUP top STRUCTURAL
  MUST ( SolarisProjectID $ SolarisProjectName )
  MAY ( memberUid $ memberGid $ description $ SolarisProjectAttr ) )
```

役割によるアクセス制御スキーマ

ユーザーと役割に関する拡張属性のシステムごとの設定は、/etc/user_attr に置かれます。詳細は、user_attr(4) のマニュアルページを参照してください。

役割によるアクセス制御の属性は、次のとおりです。

```
( 1.3.6.1.4.1.42.2.27.5.1.4 NAME 'SolarisAttrKeyValue'
  DESC 'Semi-colon separated key=value pairs of attributes'
  EQUALITY caseIgnoreIA5Match
  SUBSTRINGS caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.7 NAME 'SolarisAttrShortDesc'
  DESC 'Short description about an entry, used by GUIs'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.8 NAME 'SolarisAttrLongDesc'
  DESC 'Detail description about an entry'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )
```

(続く)

```
( 1.3.6.1.4.1.42.2.27.5.1.9 NAME 'SolarisKernelSecurityPolicy'
  DESC 'Solaris  kernel security policy'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.10 NAME 'SolarisProfileType'
  DESC 'Type of object defined in profile'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.11 NAME 'SolarisProfileId'
  DESC 'Identifier of object defined in profile'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.12 NAME 'SolarisUserQualifier'
  DESC 'Per-user login attributes'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.13 NAME 'SolarisReserved1'
  DESC 'Reserved for future use'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.14 NAME 'SolarisReserved2'
  DESC 'Reserved for future use'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )
```

役割によるアクセス制御のオブジェクトクラスは、次のとおりです。

```
( 1.3.6.1.4.1.42.2.27.5.2.3 NAME 'SolarisUserAttr' SUP top AUXILIARY
  DESC 'User attributes'
  MAY ( SolarisUserQualifier $ SolarisAttrReserved1 $ \
        SolarisAttrReserved2 $ SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.4 NAME 'SolarisAuthAttr' SUP top STRUCTURAL
  DESC 'Authorizations data'
  MUST cn
  MAY ( SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
        SolarisAttrShortDesc $ SolarisAttrLongDesc $ \
        SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.5 NAME 'SolarisProfAttr' SUP top STRUCTURAL
  DESC 'Profiles data'
  MUST cn
  MAY ( SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
        SolarisAttrLongDesc $ SolarisAttrKeyValue ) )
```

(続く)

```
( 1.3.6.1.4.1.42.2.27.5.2.6 NAME 'SolarisExecAttr' SUP top AUXILIARY
  DESC 'Profiles execution attributes'
  MAY ( SolarisKernelSecurityPolicy $ SolarisProfileType $ \
        SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
        SolarisProfileId $ SolarisAttrKeyValue ) )
```

Solaris クライアントネーミングプロファイルスキーマ

ユーザー、役割、プロファイルに関する拡張属性のシステムごとの設定は、/etc/user_attr に置かれます。詳細は、user_attr(4) のマニュアルページを参照してください。

実行プロファイル名、説明、実行プロファイルのその他の属性のシステムごとの設定は、/etc/security/prof_attr に置かれます。詳細は、prof_attr(4) のマニュアルページを参照してください。

Solaris クライアントネーミングプロファイルの属性は、次のとおりです。

```
( 1.3.6.1.4.1.42.2.27.5.1.15 NAME 'SolarisLDAPServers'
  DESC 'LDAP Server address eg. 76.234.3.1:389'
  EQUALITY caseIgnoreIA5Match
  SYNTAX SolarisLDAPServerSyntax)

( 1.3.6.1.4.1.42.2.27.5.1.16
  NAME 'SolarisSearchBaseDN'
  DESC 'Search Base Distinguished Name'
  EQUALITY distinguishedNameMatch
  SYNTAX DN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.17
  NAME 'SolarisCacheTTL'
  DESC 'TTL value for the Domain information eg. 1w, 2d, 3h, 10m, or 5s'
  EQUALITY caseIgnoreMatch
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.18
  NAME 'SolarisBindDN'
  DESC 'DN to be used to bind to the directory as proxy'
  EQUALITY distinguishedNameMatch
  SYNTAX DN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.19
  NAME 'SolarisBindPassword'
  DESC 'Password for bindDN to authenticate to the directory'
  EQUALITY caseExactIA5Match
  SYNTAX OctetString SINGLE-VALUE)
```

(続く)

```
( 1.3.6.1.4.1.42.2.27.5.1.20
  NAME 'SolarisAuthMethod'
  DESC 'Authentication method to be used eg. "NS_LDAP_AUTH_NONE",
       "NS_LDAP_AUTH_SIMPLE" or "NS_LDAP_AUTH_SASL_CRAM_MD5"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String)

( 1.3.6.1.4.1.42.2.27.5.1.21
  NAME 'SolarisTransportSecurity'
  DESC 'Transport Level Security method to be used eg.
        "NS_LDAP_SEC_NONE" or "NS_LDAP_SEC_SASL_TLS"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.22
  NAME 'SolarisCertificatePath'
  DESC 'Path to certificate file/device'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.23
  NAME 'SolarisCertificatePassword'
  DESC 'Password or PIN that grants access to certificate.'
  EQUALITY caseExactIA5Match
  SYNTAX OctetString SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.24
  NAME 'SolarisDataSearchDN'
  DESC 'Search DN for data lookup in "<database>:(DN0),(DN1),..." format'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String)

( 1.3.6.1.4.1.42.2.27.5.1.25
  NAME 'SolarisSearchScope'
  DESC 'Scope to be used for search operations eg.
        "NS_LDAP_SCOPE_BASE", "NS_LDAP_SCOPE_ONELEVEL" or
       "NS_LDAP_SCOPE_SUBTREE"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.26
  NAME 'SolarisSearchTimeLimit'
  DESC 'Time Limit in seconds for search operations'
  EQUALITY integerMatch
  SYNTAX INTEGER SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.27
  NAME 'SolarisPreferredServer'
  DESC 'Preferred LDAP Server address or network number'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IAString)

( 1.3.6.1.4.1.42.2.27.5.1.28
```

(続く)

```
  NAME 'SolarisPreferredServerOnly'
  DESC 'Boolean flag for use of preferredServer or not'
  EQUALITY booleanMatch
  SYNTAX BOOLEAN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.29
  NAME 'SolarisSearchReferral'
  DESC 'referral chasing option eg.
       "NS_LDAP_NOREF" or "NS_LDAP_FOLLOWREF"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)
```

Solaris クライアントネーミングプロファイルのオブジェクトクラスは、次のとおりです。

```
( 1.3.6.1.4.1.42.2.27.5.2.7 NAME 'SolarisNamingProfile'
  SUP top STRUCTURAL
  DESC 'Solaris LDAP Naming client profile objectClass'
  MUST ( cn $ SolarisLDAPServers $ SolarisSearchBaseDN )
  MAY ( SolarisBindDN $ SolarisBindPassword $ SolarisAuthMethod $
        SolarisTransportSecurity $ SolarisCertificatePath $
        SolarisCertificatePassword $ SolarisDataSearchDN $
        SolarisSearchScope $ SolarisSearchTimeLimit $
        SolarisPreferredServer $ SolarisPreferredServerOnly $
        SolarisCacheTTL $ SolarisSearchReferral )
 )
```

付録 **B**

構成に関する問題の解決

この付録では、LDAP の構成で発生する問題とその解決方法を示します。

構成で発生する問題とその解決法

LDAP の構成で発生する問題とそれらの解決方法について説明します。

未解決のホスト名

Solaris LDAP クライアントのバックエンドは、ホスト検索時に、`gethostbyname(3N)` や `getipnodebyname(3N)` によって返されるホスト名と同様に、完全指定ホスト名を返すように設計されています。ディレクトリに格納されている名前が完全指定されている場合 (つまり 1 つ以上のドットが含まれている場合)、クライアントはその名前をそのまま返します。たとえば、格納されている名前が `hostB.eng` であれば、返される名前も `hostB.eng` です。

LDAP ディレクトリに格納されている名前が完全指定されていない場合 (ドットが 1 つも含まれない場合)、クライアントのバックエンドは、その名前にドメイン部分を追加します。たとえば、格納されている名前が `hostA` であれば、返される名前は `hostA.domainname` となります。

LDAP ドメイン内のシステムにリモートからアクセスできない

DNS ドメイン名が LDAP ドメイン名と異なる場合は、nsswitch.conf ファイルを変更してください。ホストエントリ行に dns を指定するか、ldap の前に dns を追加します。

sendmail がリモートユーザーとのメールの送受信に失敗する

メールドメイン (通常は DNS ドメイン) が LDAP ドメインと異なると、メールの転送に失敗することがあります。sendmail(1M) は、メールドメインを、gethostname(3N) が返すホスト名のドメイン部分から取り出します。つまり、返されるアドレスは LDAP ドメインになります。メール/DNS ドメインは LDAP ドメインとは異なるため、外部ユーザーはこのようなメールに返信できません。この問題を解決するには、nsswitch.conf ファイル内の当該ホストエントリを dns に変更するか、ldap の前に dns を追加します。

ログインできない

LDAP クライアントは、ログイン時のユーザー認証に PAM(3) モジュールを使用します。UNIX 標準の PAM モジュールでは、パスワードをサーバーから読み込み、クライアント側で検査します。これは、次のいずれかの理由で失敗する場合があります。

1. ldap が、/etc/nsswitch.conf ファイル内に情報源として存在しない。
2. サーバー上のパスワードをプロキシエージェントから読むことができないようになっている。プロキシエージェントが比較のためにパスワードをクライアントに返すので、少なくともプロキシエージェントはパスワードを読めなければならない。
3. プロキシエージェントの設定が間違っていると、認証を実行できない。
4. 当該エントリに shadowAccount オブジェクトクラスが定義されていない。

検索が遅い

LDAP データベースは、インデックスを使用することによって検索パフォーマンスが向上します。インデックスが正しく作成されていない場合、大幅にパフォーマンスが低下することがあります。このマニュアルの第 2 章に、インデックス付けが必

要な一連の属性をまとめてあります。また、独自のインデックスを追加して、パフォーマンスの向上を図ることができます。

ldapclient がサーバーにバインドできない

-P プロファイルオプションを使用しているときに、ldapclient がクライアントの初期化に失敗したと考えられます。次のいずれかが原因である可能性があります。

1. ldap_cachemgr が動作していることを確認する。ps -ef |grep ldap とすれば、確認できます。
2. ldapclient -l を実行して、LDAP クライアントのキャッシュファイルの内容を検査する。

注 - 構成ファイルや資格ファイルは必ずしも ASCII 形式とは限らないので、これらを直接表示しないでください。

---

3. 指定されたクライアントドメインのエントリポイントを表す nisDomain 属性が DIT (ディレクトリ情報ツリー) 内に設定されていない。
4. サーバー上で仮想リスト表示のインデックス付けが正しく行われていない。
5. サーバー上でアクセス制御情報が正しく設定されていないため、anonymous (匿名) ユーザーが LDAP データベースを検索できない。
6. ldapclient コマンドに渡されたサーバーアドレスが間違っている。ldapsearch(1) を使用してサーバーのアドレスを確認します。
7. ldapclient コマンドに渡されたプロファイル名が間違っている。ldapsearch(1) を使用して、DIT 内のプロファイル名を確認します。
8. クライアントのネットワークインタフェースに対して snoop(1M) を実行して、外向きのトラフィックを検査し、どのサーバーとやり取りしているかを確認します。

索引