配備ガイド

Sun™ ONE Directory Server

Version 5.2

817-4626-10
2003 年 6 月

目 次

本書について

Sun™ ONE Directory Server 5.2 は、業界標準の LDAP (Lightweight Directory Access Protocol) に基づく強力でスケーラブルな分散ディレクトリサーバーです。Sun ONE Directory Server ソフトウェアは、サービスをオンデマンドで構築、配備するために、Sun の標準に基づくソフトウェアビジョン、アーキテクチャ、プラットフォーム、専門知識を結集した Sun ONE (Sun Open Net Environment) の一部です。

Sun ONE Directory Server は、社内を結ぶイントラネット内、パートナー企業との通信で使用されるエクストラネット上、または顧客からのアクセスを獲得するために一般インターネット上で使用できる、集中型および分散型データリポジトリ構築のための基盤となります。

このマニュアルの目的

このマニュアルでは、ディレクトリの導入を計画するための基本事項について説明します。ここで提供する情報は、主にディレクトリの責任者、ソリューション設計者、および管理者を対象としています。

このマニュアルは、2 つの部分に分かれています。第 1 部では、スキーマの設計、ディレクトリツリー、トポロジー、レプリケーション、セキュリティ、監視など、ディレクトリの設計概念について説明します。第 2 部では、主な留意点や問題点の理解に役立つ Sun ONE Directory Server 5.2 の配備例と参照用レプリケーションアーキテクチャを示します。

前提条件

このマニュアルを参照する前に、オンラインリリースノートを読み、Sun ONE Directory Server のこのリリースの新機能と機能強化について最新の情報を確認することを強くお勧めします。リリースノートは、次の場所で取得できます。

```
http://docs.sun.com/doc/816-6703-10/
```

このマニュアルは、基本的なディレクトリサービスと LDAP の概念に習熟し、Sun ONE Directory Server 製品の基本情報を理解している方を対象に記述されています。これらの情報は、すべて『Sun ONE Directory Server Getting Started Guide』に記載されています。

表記上の規則

ここでは、このマニュアルで使用する表記規則について説明します。

`Monospaced font` (等倍フォント) : このフォントは、属性およびオブジェクトクラスの名前などを本文中で使用する場合など、リテラル文字列で使用します。また、URL、ファイル名、および例にも使用します。

Italic font (*イタリック体*) : このフォントは、強調、新出用語、および可変部分 (パス名など実際の値に置き換える必要がある文字列) で使用します。

大なり記号 (>) は、メニューまたはサブメニューの項目名を示すときに、区切り文字として使用されます。たとえば、「オブジェクト」>「新規」>「ユーザー」は、「オブジェクト」メニューの「新規」サブメニューから「ユーザー」という項目を選択することを意味します。

注	「注」、「注意」、および「ヒント」は、重要な条件または制限を強調するためのものです。必ずこれらの注意事項を読んでから、作業を続けるようにしてください。

デフォルトのパスとファイル名

Sun ONE Directory Server 製品のマニュアルでは、すべてのパスとファイル名の例は、次の 2 つの形式のいずれかで表記されます。

- *ServerRoot/...* : *ServerRoot* は、Sun ONE Directory Server 製品が格納されているの場所です。このパスには、Directory Server、Sun ONE Administration Server、およびコマンド行ツールの共有バイナリファイルが含まれます。

  実際の *ServerRoot* パスは、それぞれのプラットフォーム、インストール内容、および設定によって異なります。デフォルトパスは、製品プラットフォームとパッケージングによって異なります。表 1 を参照してください。

- *ServerRoot*/slapd-*serverID*/. . . : serverID は、インストール時または設定時に定義した Directory Server インスタンスの名前です。このパスには、各インスタンスに固有のデータベースと設定ファイルが格納されます。

注 このマニュアルに記載されているパスは UNIX 形式のスラッシュ (/) を使用しており、コマンドはファイル拡張子なしで記述されています。Windows バージョンの Sun ONE Directory Server を使用している場合は、バックスラッシュ形式に読み替えてください。一般に、Windows システムの実行可能ファイルには、UNIX 形式と同じ名前に `.exe` または `.bat` という拡張子が付けられています。

表 1 デフォルトの *ServerRoot* パス

製品のインストール	*ServerRoot* パス
Solaris 9[1]	`/var/mps/serverroot` : 設定後に、このディレクトリには次の場所へのリンクが含まれます。 • `/etc/ds/v5.2` (スタティックな設定ファイル) • `/usr/admserv/mps/admin` (Sun ONE Administration Server バイナリ) • `/usr/admserv/mps/console` (Server Console バイナリ) • `/usr/ds/v5.2` (Directory Server バイナリ)
Solaris およびその他の UNIX システムへの圧縮アーカイブインストール	`/var/Sun/mps`
Windows システムへの Zip インストール	`C:¥Program Files¥Sun¥MPS`

1. Solaris Operating Environment 環境で、インストールされている Sun ONE Directory Server ソフトウェアのバージョンがわからないときは、pkginfo コマンドを実行して SUNWdsvu などのキーパッケージが存在するかどうかを確認します。たとえば、次のようにします。pkginfo | grep SUNWdsvu

Directory Server インスタンスは、*ServerRoot*/slapd-*serverID*/ の下に格納されます。この *serverID* は、インスタンスの作成時に指定するサーバー識別子です。たとえば、Directory Server に dirserv という名前を指定した場合は、実際のパスは表 2 のようになります。Directory Server を別の場所に作成した場合は、それに合わせてパスを変更してください。

表 2　　dirserv インスタンスを例としたデフォルトの場所

製品のインストール	インスタンスの場所
Solaris 9	/var/mps/serverroot/slapd-dirserv
Solaris およびその他の UNIX システムへの圧縮アーカイブインストール	/usr/Sun/mps/slapd-dirserv
Windows システムへの Zip インストール	C:¥Program Files¥Sun¥MPS¥slapd-dirserv

Directory Server ツールのダウンロード

一部のプラットフォームには、Directory Server にアクセスするためのネイティブツールが用意されています。LDAP ディレクトリサービスのテストと管理のためのツールが必要なときは、Sun ONE Directory Server Resource Kit (DSRK) をダウンロードしてください。このソフトウェアは、次の場所にあります。

```
http://wwws.sun.com/software/download/
```

DSRK ツールのインストール方法と参照マニュアルについては、Sun ONE Directory Server Resource Kit Tools Reference を参照してください。

ディレクトリクライアントアプリケーションの開発用には、同じ場所から Sun ONE LDAP SDK for C および Sun ONE LDAP SDK for Java もダウンロードしてください。

さらに、JNDI (Java Naming and Directory Interface) テクノロジにより、Java アプリケーションから LDAP と DSML v2 を使って Directory Server にアクセスすることができます。JNDI に関する情報は、次の場所にあります。

```
http://java.sun.com/products/jndi/
```

JNDI Tutorial には、JNDI の使用方法に関する詳細な説明と例が含まれます。これは、次の場所にあります。

```
http://java.sun.com/products/jndi/tutorial/
```

参考文献

Sun ONE Directory Server 製品には次のマニュアルが用意されています。どのマニュアルにも HTML 形式と PDF 形式があります。

- 『Sun ONE Directory Server Getting Started Guide』: Directory Server 5.2 の主要機能について簡単に説明します。
- 『Sun ONE Directory Server 配備ガイド』: ディレクトリトポロジ、データ構造、セキュリティ、監視の計画方法について説明し、配備例を紹介します。
- 『Sun ONE Directory Server インストールおよびチューニングガイド』: Directory Server のインストールとアップグレードの手順、およびパフォーマンスの最適化について説明します。
- 『Sun ONE Directory Server 管理ガイド』: ディレクトリコンテンツの管理、および Directory Server のすべての機能の設定をコンソールとコマンド行から行う方法について説明します。
- 『Sun ONE Directory Server Reference Manual』: Directory Server の設定パラメータ、コマンド、ファイル、エラーメッセージ、スキーマの詳細情報を提供します。
- 『Sun ONE Directory Server Plug-In API Programming Guide』: Directory Server プラグインの開発方法を照会します。
- 『Sun ONE Directory Server Plug-In API Reference』: Directory Server プラグイン API のデータ構造と関数の詳細情報を提供します。
- 『Managing Servers with Sun ONE Console』: Sun ONE Administration Server および Java ベースのコンソールを使用してサーバーを管理する方法について説明します。
- 『Sun ONE Directory Server Resource Kit Tools Reference』: Sun ONE Directory Server Resource Kit のインストール方法と、多数の便利なツールの機能について説明します。

その他の有用な情報は、次の Web サイトから入手できます。

- オンラインの製品マニュアル : `http://docs.sun.com/coll/S1_DirectoryServer_52`
- Sun ソフトウェア : `http://wwws.sun.com/software/`
- Sun ONE サービス : `http://www.sun.com/service/sunps/sunone/`

- Sun サポートサービス : `http://www.sun.com/service/support/`
- 開発者向けの Sun ONE 情報 : `http://sunonedev.sun.com/`
- トレーニング : `http://suned.sun.com/`

注

Sun Microsystems Inc. は、このマニュアルに記載されているサードパーティ Web サイトの利用可能性について責任を負いません。Sun Microsystems Inc. は、このようなサイトまたはリソースで得られるあらゆる内容、広告、製品、およびその他素材を保証するものではなく、責任または義務を負いません。Sun Microsystems Inc. は、このようなサイトまたはリソースで得られるあらゆるコンテンツ、製品、またはサービスによって生じる、または生じたと主張される、または使用に関連して生じる、または信頼することによって生じる、いかなる損害または損失についても責任または義務を負いません。

第 1 部

Directory Server の設計

第 1 部では、ディレクトリ設計の概念を紹介し、データの設計とアクセス、スキーマの設計、ディレクトリツリー、トポロジ、レプリケーション、セキュリティ、監視など、設計プロセス全体について説明します。第 1 部の目的は、設計概念の詳細を十分に理解し、ディレクトリの配備を計画する上で注意すべき事項について理解することです。

第 1 章

Directory Server の設計と配備の概要

Sun ONE Directory Server は、イントラネット、ネットワーク、およびエクストラネットの情報を集中管理するディレクトリサービスを提供します。Directory Server は、既存のシステムと統合することができ、社員、顧客、サプライヤ、パートナーなどの情報を管理する中央リポジトリとして機能します。また、Directory Server を拡張して、ユーザーのプロファイルや環境設定、エクストラネットのユーザー認証などを管理することもできます。

LDAP とディレクトリの概念、および Sun ONE Directory Server の基本情報については、『Sun ONE Directory Server Getting Started Guide』を参照してください。この章では、ディレクトリの設計と配備のプロセスについてその概要を説明します。この章は次の節に分かれています。

- ディレクトリ設計の概要
- ディレクトリ配備の概要

ディレクトリ設計の概要

実際にディレクトリサービスを導入する前に設計計画を立てることは、ディレクトリを確実に成功させるためのもっとも重要な作業です。ディレクトリの設計段階では、環境、データソース、ユーザー、ディレクトリを使用するアプリケーションなど、ディレクトリに求められる要件に関するデータを収集します。これらのデータを活用することにより、要件を満たすディレクトリサービスを設計できます。

Sun ONE Directory Server が備える柔軟性により、Directory Server を導入した後も、予期しない状況や要件の変更に対応して設計を見直すことができます。ただし、優れた設計によって修正を避けるべきであることは言うまでもありません。

設計プロセスの概要

設計プロセスは 6 つの段階に分けられます。

- ディレクトリデータの計画とアクセス

  ディレクトリには、ユーザー名、電話番号、ユーザーが属するグループの情報などのデータが入ります。組織内のさまざまなデータソースを分析し、それらの相互関係を理解するには、第 2 章「ディレクトリデータの計画とアクセス」を参照してください。この章では、ディレクトリに保存するデータのタイプや、そのデータにどのようにアクセスするか、Directory Server のコンテンツの設計に必要なその他の作業について説明します。

- スキーマの設計

  Directory Server が 1 つ以上のディレクトリ対応アプリケーションをサポートするように設計します。これらのアプリケーションには、ディレクトリに格納するデータについて、形式などの要件があります。ディレクトリに格納するデータの特性は、ディレクトリスキーマで決定します。第 3 章「スキーマの設計」では、Sun ONE Directory Server に含まれる標準スキーマを紹介し、スキーマをカスタマイズする方法を説明します。スキーマの一貫性を保持するためのヒントも記載されています。

- ディレクトリツリーの設計

  ディレクトリに格納するデータを決めたら、そのデータを編成して、参照させる必要があります。ディレクトリツリーの目的は、この編成と参照です。第 4 章「ディレクトリツリーの設計」では、ディレクトリツリーについて説明します。データ階層の設計方法、およびエントリのグループ化を最適化するためのメカニズムと属性の管理について説明します。ディレクトリツリーの設計例も記載されています。

- ディレクトリトポロジの設計

  トポロジの設計では、ディレクトリツリーを複数の物理的な Directory Server 上に分割する方法や、これらのサーバー間での通信方法を決定します。第 5 章「ディレクトリトポロジの設計」では、トポロジ設計の基礎となる一般原則、複数のデータベースの使用方法、分散したデータをリンクするのに使用するメカニズム、Directory Server 自体で分散したデータを追跡する方法について説明します。

- レプリケーションの設計

  レプリケーションを使用すると、複数の Directory Server が同じディレクトリデータを保持するので、パフォーマンスが向上し、耐障害性が高まります。第 6 章「レプリケーションの設計」では、レプリケーションのしくみ、レプリケートできるデータの種類、一般的なレプリケーションの使用例について説明します。また、可用性の高いディレクトリサービスを構築するためのヒントを示します。

- 安全なディレクトリの設計

  ディレクトリ内のデータを保護する方法を計画し、ユーザーとアプリケーションのセキュリティ要件を満たすように、サービスを別の側面から検討することは重要です。第 7 章「安全なディレクトリの設計」では、一般的なセキュリティ上の危険、セキュリティ手法の概要、必要なセキュリティ要件を分析する際の手順について説明します。また、アクセス制御を設計し、ディレクトリデータの整合性を保持するためのヒントを示します。

- ディレクトリの監視

  ここまでは、できるだけ安全で、要件に即したディレクトリサービスを設計することに集中してきました。しかし、ディレクトリサービスを正しく監視できない限り、ディレクトリサービスの配備が成功したかどうかを正しく評価したり、毎日のディレクトリアクティビティを行うことができません。第 8 章「ディレクトリの監視」では、SNMP、Directory Server コンソール、ログファイル、および Directory Server に用意されているデータベース監視ツールとレプリケーション管理ツールを使ってディレクトリを監視する方法について説明します。

ディレクトリ配備の概要

ディレクトリサービスの設計が完了したら、配備フェーズに進みます。配備フェーズは、次の段階から構成されます。

- ディレクトリの試験
- 運用環境へのディレクトリの配備

ディレクトリの試験

配備フェーズの最初の段階では、サーバーのインスタンスを試験的にインストールし、サービスがユーザーの負荷を処理できるかどうかをテストします。サービスが適切でない場合は、設計を調整してパイロットテストを繰り返します。自信を持って全社的に導入できるような堅牢なサービスが完成するまで、設計を調整します。

試験的なディレクトリ作成と実装の概要については、『Understanding and Deploying LDAP Directory Services』(T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年) を参照してください。

運用環境へのディレクトリの配備

パイロットテストを実施して、サービスを調整したら、ディレクトリサービスを試験的な配備から実際の運用環境に配備する計画を立て、それを実行します。次のような内容の運用計画を作成します。

- 必要なリソースの見積もり

- サーバーをインストールする前に実行しておくべき作業
- 達成すべき事柄と作業日程
- 配備が成功したかどうかを測定するための一連の基準

ディレクトリの管理と保守については、『Sun ONE Directory Server 管理ガイド』を参照してください。

第 2 章

ディレクトリデータの計画とアクセス

ディレクトリに格納されるデータには、ユーザー名、電子メールアドレス、電話番号、ユーザーが所属するグループの情報が含まれ、それ以外の種類の情報が含まれる場合もあります。ディレクトリ内のデータのタイプによって、ディレクトリの構成方法、データへのアクセスが可能なユーザー、およびアクセスの要求方法と応答方法が決まります。Sun ONE Directory Server 5.2 では、LDAP または DSML 経由でディレクトリデータにアクセスできるので、アプリケーションがディレクトリデータと直接やり取りを行える可能性もあります。

この章では、ディレクトリデータの計画とアクセスに関連する問題点と手法について説明します。この章は、次の節から構成されます。

- ディレクトリデータの概要
- ディレクトリ要件の定義
- DSML による HTTP/SOAP 経由のディレクトリデータへのアクセス
- サイト調査の実施

ディレクトリデータの概要

データのタイプには、ディレクトリに適しているものとそうでないものがあります。ディレクトリに入れるのに最適なデータには、次のような特性があります。

- 読み取り回数が書き込み回数より多い

  ディレクトリは読み取り操作用に調整されているため、書き込み操作はサーバーのパフォーマンスを低下させます。

- 属性データの形式で表現できる (たとえば、`surname=jensen`)
- 多くのユーザーにとって重要である

  たとえば、社員の名前やプリンタが実際に置かれている場所は、多くのユーザーやアプリケーションにとって重要です。

- 複数の場所からアクセスされる

  たとえば、ある社員のアプリケーションの環境設定は、そのアプリケーションだけがこの情報にアクセスできればよいため、ディレクトリには適しません。ただし、このアプリケーションにディレクトリから環境設定を読み込む機能がある場合は、環境設定情報をディレクトリに入れておくと、別のサイトでそのアプリケーションを使用するときに自分の環境設定をディレクトリから読み込めるので便利です。

ディレクトリに格納できるデータ

次に、ディレクトリに格納できるデータの例を示します。

- 電話番号、住所、電子メールアドレスなどの連絡先情報
- 社員番号、役職、管理者 ID、業務に関する利害関係などの記述的な情報
- 電話番号、住所、管理者 ID、業務についての説明などの組織の連絡先情報
- プリンタの設置場所、プリンタの機種、プリンタの印刷速度などの機器に関する情報
- 会社の取引先、得意先、顧客などの連絡先情報と明細書情報
- 顧客名、期限、作業内容、価格情報などの契約情報
- ユーザーのソフトウェアの環境設定情報や設定情報
- Web サーバーへのポインタ、特定のファイルやアプリケーションのファイルシステムなどのリソースの場所

サーバー管理データとは別に、ディレクトリに次のような情報を格納することもできます。

- 契約や顧客アカウントに関する情報
- 給与データ
- 物理的なデバイス情報
- 自宅連絡先情報
- 企業のさまざまなサイトに関する職場連絡先情報

ディレクトリに含めないデータ

ディレクトリサーバーは、クライアントアプリケーションが読み取りや書き込み (頻繁ではない) を行う膨大な量のデータを管理するには適していますが、画像やほかのメディアなど、構造化されていない大きなオブジェクトを処理するようには設計されていません。このようなオブジェクトは、ファイルシステムで管理する必要があります。ただし、FTP、HTTP、またはその他の URL タイプを使用して、このようなタイプのアプリケーションへのポインタをディレクトリに格納することはできます。

ディレクトリは読み取り操作で効果を発揮するので、頻繁に更新させる情報はディレクトリに入れないようにします。ディレクトリで実行される書き込み操作の回数を減らすことにより、検索処理全体のパフォーマンスが向上します。

ディレクトリ要件の定義

ディレクトリデータを設計するときは、現在必要なデータだけでなく、将来含める可能性のあるデータも考慮に入れてください。設計の段階で将来ディレクトリに必要となる要件を考慮することが、ディレクトリデータを拡張したり分散させる際に影響します。

設計時には、次の点を考慮してください。

- 現時点でどのようなデータをディレクトリに入れるか。ディレクトリの配備により解決したい当面の問題は何か。使用するディレクトリ対応アプリケーションですぐに必要となるデータは何か
- 近い将来どのようなデータをディレクトリに入れるか。たとえば、使用している会計パッケージが現時点では LDAP に対応していないが、近い将来 LDAP または DSML に対応することがわかっている場合など。この場合、アプリケーションで使用するデータを判別しておき、その機能が利用可能になったときに、データをディレクトリに移行するようにする
- 将来ディレクトリに格納する可能性のあるデータがあるか。たとえば、ホスト環境の場合、新しい顧客が現在の顧客とは異なるデータを要求することも考えられる。新しい顧客が、JPEG 画像の格納にディレクトリを使用する可能性もある。これは予想される中でもっとも困難な場合であるが、考慮しておけば予想しなかった事態に対処できる可能性がある。少なくとも、このような方法で計画を行っていれば、ほかの方法では思いつかなかったデータソースを特定するのに役立つ

DSML による HTTP/SOAP 経由のディレクトリデータへのアクセス

ディレクトリデータへのアクセスに LDAP (Lightweight Directory Access Protocol) だけを使用する従来バージョンの Directory Server とは異なり、Sun ONE Directory Server 5.2 では、DSMLv2 (Directory Service Markup Language バージョン 2) を使用して HTTP/SOAP 経由でディレクトリデータにアクセスすることもできます。

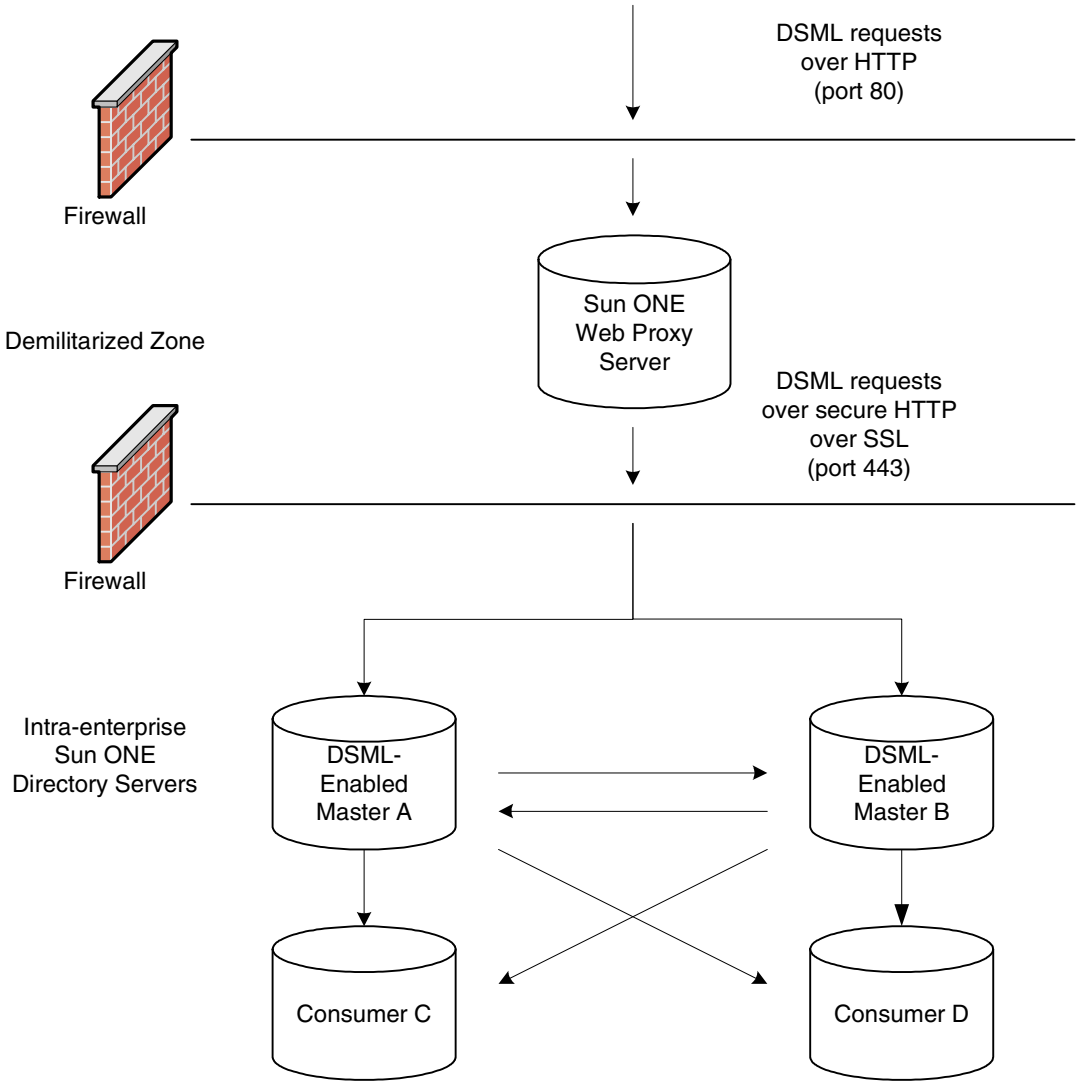
DSMLv2 はマークアップ言語、つまり、ディレクトリサービスによるデータ処理の構造と内容を XML (eXtensible Markup Language) ドキュメントとして記述するためのボキャブラリとスキーマです。DSMLv2 は、XML でディレクトリサービス情報を表現する方法を標準化します。この標準を使用することで、DSMLv2 を使用し、ディレクトリサービスのスケーラビリティ、レプリケーション、セキュリティ、管理の各機能を強化したアプリケーションを記述することができます。DSMLv2 はアクセスプロトコルではありませんが、ディレクトリに格納されているデータへの実際のアクセスする上で、DSMLv2 はアクセスプロトコルに依存します。Directory Server は、HTTP/1.1 (Hypertext Transfer Protocol) 経由での DSMLv2 の使用、および DSML コンテンツを転送するためのプログラミングプロトコルとして SOAP 1.1 (Simple Object Access Protocol) の使用をサポートしています。

HTTP/SOAP 経由で DSMLv2 を使用してディレクトリにアクセスできるようになったため、アプリケーションは LDAP アプリケーションである必要がなくなり、ディレクトリとの間でデータをやり取りするアプリケーションの範囲にも変化がもたらされています。次に、HTTP/SOAP 経由の新しい DSMLv2 によるアクセスの可能性について詳細に説明します。DSMLv2 を使用して HTTP/SOAP 経由でデータにアクセスしたり、データを検索する方法については、付録 A「DSMLv2 を使用した HTTP/SOAP 経由のデータへのアクセス」を参照してください。

HTTP/SOAP 経由の DSMLv2 の配備

DSML に対応した Directory Server と Sun ONE Web Proxy Server (非 LDAP クライアントがディレクトリデータを利用できます) を利用した配備の一例を 25 ページの図 2-1 に示します。

図 2-1　DSML 対応 Directory Server の配備例



非 LDAP クライアントアプリケーションからの DSML 形式の更新要求は、まず、HTTP ポート 80 でファイアウォールを通過し、非武装地帯に入ります。ここから、逆プロキシサーバーとして設定された Sun ONE Web Proxy Server は、セキュリティ保護された HTTP ポート 443 の使用を強制し、要求は第 2 のファイアウォールを越えてイントラネットドメインに入ります。要求は、DSML に対応していないコンシューマ C、D にレプリケートされる前に、2 つのマスターレプリカ Master A、B で処理されます。

この配備の主な目的は、非 LDAP アプリケーションがディレクトリ操作を行うことで、ディレクトリデータと通信できるようにすることです。クライアントからの要求が検索要求だけであれば、Directory Server に保持されているデータのコピーが読み取り専用であっても、読み書き両方に対応していても、どちらも検索要求を処理できるので問題ありません。しかし、非 LDAP クライアントが変更要求を送信する場合、DSML 対応 Directory Server は読み書きに対応したデータコピー (マスターレプリカ) を保持している必要があります。これはレプリケーションの特性で、変更要求を受け取るコンシューマ (読み取り専用のデータコピーを保持しています) は、デフォルトの設定では、クライアントからの変更要求を満足できる可能性のあるマスターの LDAP URL リストのリフェラルを返すためです。非 LDAP クライアントアプリケーションに対して HTTP 経由で LDAP URL を返しても意味がありませんし、クライアントとディレクトリの間のトラフィックで LDAP を使用しないという当初の目的を果たすことができません。このために読み書き両方に対応したコピーが必要なのです。25 ページの図 2-1 の配備では、クライアントからの変更要求を処理する DSML 対応 Directory Server のマスター A、B は読み書き両方に対応したデータコピーを保持し、DSML に対応していないコンシューマ C、D にデータがレプリケートされます。

前述のように、HTTP/SOAP 経由の DSML アクセスにより XML および Web サービスでもディレクトリデータを利用できるようになりますが、それに応じてセキュリティ上のリスクも高まります。Directory Server の DSML フロントエンドは、DSML HTTP ポスト処理だけを受け付け、SOAP/DSML 仕様に準拠していない要求を拒否することで、アクセスを制限した HTTP サーバーを構築しています。これにより、その他の種類の HTTP Web サーバーと比較してリスクの範囲は狭められています。配備に DSML 対応 Directory Server を含めるときは、それでもなお次のセキュリティ上の注意点に留意することをお勧めします。

- ファイアウォールを実装して、DSML 対応 Directory Server を保護します。
- ポート 443 で SSL によってセキュリティ保護された HTTP を使用するか、クライアントで SSL 経由の HTTP を使用したくない場合は、Web プロキシサーバーを実装します。

サイト調査の実施

サイト調査は、ディレクトリの内容を調べ、その特性を把握するための適切な手法です。ディレクトリアーキテクチャで重要なのはデータです。サイト調査には十分な時間をかけてください。サイト調査では次の作業を行います。ここではそれぞれについて簡単に説明し、詳細については後述します。

- ディレクトリを使用するアプリケーションを特定する

  導入するディレクトリ対応アプリケーションとそのデータ要件を決定します。

- アプリケーションがどのようにディレクトリにアクセスするかを特定する

  アプリケーションが LDAP または HTTP/SOAP 経由の DSML のどちらのモードを使用するかを決定します。

- データソースを特定する

  自社内を調査して、データの取得元 (Windows、NetWare ディレクトリ、PBX システム、人事部データベース、電子メールシステムなど) を確認します。

- ディレクトリに入れる必要があるデータの特性を把握する

  ディレクトリに入れる必要のあるオブジェクト (ユーザーまたはグループなど) と、ディレクトリ内で管理するオブジェクトの属性 (ユーザー名やパスワードなど) を決定します。

- 提供すべきサービスレベルを決定する

  クライアントアプリケーションにどの程度までディレクトリデータの利用を許可するかを決定し、それに応じてアーキテクチャを設計します。ディレクトリデータをどの程度まで利用可能にするかによって、データのレプリケート方法や、リモートサーバーに格納されているデータに接続するための連鎖ポリシーの設定が変わってきます。

  レプリケーションについては、113 ページの第 6 章「レプリケーションの設計」を参照してください。連鎖については、第 5 章「ディレクトリトポロジの設計」を参照してください。

- データマスターを特定する

  データマスターには、ディレクトリデータのプライマリソースが含まれます。このデータは、ロードバランスと回復の目的のために、ほかのサーバーにコピーされている場合があります。各データについて、そのデータのマスターを特定します。

- データ所有者を決定する

  各データについて、データを最新の状態に保つ責任のあるユーザーを決定します。

- データアクセスを決定する

ほかのソースからデータをインポートする場合は、一括インポートと増分更新の両方について計画を立てます。この手法の一環として、どのデータについてもマスターは一か所に配置し、そのデータを変更できるアプリケーションの数を制限します。また、データに書き込みを行えるユーザーの数も制限します。このグループのメンバー数が少ないほど、データの整合性が確保でき、管理に伴うオーバーヘッドを低減できます。

- サイト調査の結果を文書化する

  多くの組織がディレクトリによって影響を受けるため、影響のある各組織からの代表者によるディレクトリ導入チームを編成することをお勧めします。このチームでサイト調査を実施します。

  会社は一般に、人事、経理、製造 (1 つまたは複数)、営業 (1 つまたは複数)、開発 (1 つまたは複数) などの部署から形成されています。これらの各組織からの代表者をチームに加えると、調査を迅速に進めることができます。また、影響を受けるすべての組織が直接参加することで、部署ごとのローカルデータ保管から、ディレクトリによる集中化されたデータ管理へ移行しやすくなります。

- サイト調査の繰り返し

  企業の事務所が複数ある場合は、事務所ごとにサイト調査を繰り返す必要があります。各事務所でサイト調査チームを作り、結果を中央のサイト調査チーム (各地の代表者で構成) に送ります。

ディレクトリ対応のアプリケーションの特定

一般的に、ディレクトリにアクセスするアプリケーションと、それらのアプリケーションで必要となるデータが、ディレクトリの内容を決定する主な原因となります。ディレクトリを使用する一般的なアプリケーションには、次のアプリケーションが含まれます。

- White pages などのディレクトリブラウザアプリケーション。これらのアプリケーションは、通常は電子メールアドレス、電話番号、社員名などの情報にアクセスします。
- メッセージングアプリケーション、特に電子メールサーバー。すべての電子メールサーバーは、ディレクトリで利用できる電子メールアドレス、ユーザー名、およびルーティング情報を必要とします。サーバーの中には、ユーザーのメールボックスが格納されているディスク上の格納場所、休暇通知情報、およびプロトコル情報 (たとえば、IMAP や POP) など、さらに高度な情報を必要とするものもあります。
- ディレクトリ対応の人事管理アプリケーション。このアプリケーションは、政府指定の ID 番号、自宅の住所、自宅の電話番号、生年月日、給与、役職など、個人に関する詳細な情報を必要とします。

- セキュリティ、Web ポータル、個人化用アプリケーション。これらのアプリケーションは、プロファイル情報にアクセスします。

ディレクトリを使用するアプリケーションを調査するときは、各アプリケーションが使用する情報のタイプに注目します。次の表に、アプリケーションと各アプリケーションが使用する情報の例を示します。

表 2-1 アプリケーションが必要とするデータ

アプリケーション	データクラス	データ
電話帳	people	名前、電子メールアドレス、電話番号、ユーザー ID、パスワード、部署番号、マネージャ、メールの配信停止
Web サーバー	people、groups	ユーザー ID、パスワード、グループ名、グループ番号、グループの所有者
Calendar Server	people、meeting rooms	名前、ユーザー ID、広さ、会議室の名前
Web ポータル	people、groups	ユーザー名、ユーザー ID、パスワード、グループ名、グループ番号

アプリケーションおよび各アプリケーションが使用する情報を特定すると、いくつかのタイプのデータが複数のアプリケーションによって使用されていることがわかってきます。データの計画段階でこのような課題に取り組むことで、ディレクトリ内のデータが重複する問題を避けることができ、ディレクトリを利用するアプリケーションが必要とするデータの特定に役立ちます。

ディレクトリで管理するデータのタイプと、そのデータの管理を開始する時期についての最終的な決定は、次の要因に影響されます。

- さまざまな旧バージョンのアプリケーションで必要とされるデータと、そのアプリケーションを使用するユーザーの数
- 旧バージョンのアプリケーションが LDAP ディレクトリと通信できるかどうか

アプリケーションがディレクトリにアクセスする方法の特定

非 LDAP アプリケーションが、ディレクトリ操作を行ってディレクトリデータをやり取りしなければならない場合は、HTTP/SOAP 経由の DSML によるディレクトリへのアクセスを検討してください。ただし、クライアントアプリケーションが LDAP アプリケーションであれば、LDAP アクセスを選択します。選択するアクセスモードは、どのアプリケーションがディレクトリにアクセスするかによって異なります。

データソースの特定

ディレクトリに入れるすべてのデータを調べるには、現存のデータの格納場所について、次のような調査を実施する必要があります。調査では、次の作業を行います。

- 情報を提供する組織を特定する

  企業にとって重要な情報を管理している組織をすべて特定します。通常、この組織には、情報サービス部、人事部、および経理部が含まれます。

- 情報のソースであるツールとプロセスを特定する

  一般的な情報ソースには、ネットワーク用のオペレーティングシステム (Windows、Novell Netware、UNIX NIS)、電子メールシステム、セキュリティシステム、PBX (電話交換) システム、人事管理アプリケーションなどがあります。

- データの集中化が各データに与える影響を判定する

  集中データ管理では、新しいツールとプロセスが必要になることがあります。集中化によって、ある組織のスタッフを増員してほかの組織のスタッフを減らすことが必要になる場合は、問題が生じる可能性もあります。

調査中に、次の表のような雛形を用意して、企業内の情報ソースをすべて特定しておくことをお勧めします。

表 2-2 情報ソース

データソース	データクラス	データ
人事管理データベース	people	名前、住所、電話番号、部署番号、マネージャ
電子メールシステム	people、groups	名前、電子メールアドレス、ユーザー ID、パスワード、電子メールの環境設定
設備システム	facilities	建物の名前、フロアの名前、広さ、アクセスコード

ディレクトリデータの特徴づけ

ディレクトリに含めるために特定したすべてのデータは、次のような一般的な観点から特徴づけることができます。

- 形式
- サイズ
- さまざまなアプリケーションで使用される回数
- データ所有者
- ほかのディレクトリデータとの関係

ディレクトリに含める計画のある各データをよく調査して、ほかのデータと共通している特徴を明確にする必要があります。これにより、第 3 章「スキーマの設計」で詳しく説明しているスキーマの設計段階で、時間を節約することができます。

たとえば、ディレクトリデータの特徴を記載した次のような表を作成することができます。

表 2-3 ディレクトリデータの特徴

データ	形式	サイズ	所有者	関連するエントリ
社員の名前	テキストの文字列	128 文字	人事	ユーザーエントリ
ファックス番号	電話番号	14 桁の数字	設備	ユーザーエントリ
電子メールアドレス	テキスト	多くの文字	情報システム部	ユーザーエントリ

ディレクトリの可用性要件の特定

提供するサービスの可用性のレベルは、ディレクトリ対応のアプリケーションを利用するユーザーが必要とするサービスによって決まります。各アプリケーションで必要なサービスレベルを決定するには、まずそのアプリケーションがいつどのように使用されているかを確認します。

ディレクトリの利用が進むにつれて、通常の運用レベルからミッションクリティカルなレベルまで、さまざまなサービスレベルをサポートする必要が出てきます。ディレクトリの導入後にサービスレベルを上げることは困難なので、将来の要件にも対応できる設計を初期の段階から心がけるようにしてください。

たとえば、システム全体に及ぶような障害が発生する可能性を排除する場合は、同じデータに対して複数のマスターが存在する、マルチマスター設定を使用します。次に、データマスターの特定について詳しく説明します。

データマスターサーバーの特定

データマスターは、データのマスターソースになるサーバーです。データが複数の場所に存在する場合は、どのサーバーをデータマスターにするかを考慮します。たとえば、レプリケーションを使用する場合、あるいは LDAP 経由で通信できないアプリケーションを使用する場合は、データが複数のサイトに分散されていることがあります。あるデータが複数の場所に存在する場合は、マスターコピーを置くサーバーとこのマスターコピーから更新を受け取るサーバーを決定しておく必要があります。

レプリケーション時のデータマスターの作成

Sun ONE Directory Server では、複数のサーバーに情報のマスターソースを置くことができます。レプリケーションを使用する場合は、どのサーバーをデータのマスターソースにするかを決定します。Sun ONE Directory Server では、複製のサーバーが同じデータのマスターソースとなる可能性があるマルチマスター設定がサポートされています。レプリケーションとマルチマスターレプリケーションについては、113 ページの「レプリケーションの設計」を参照してください。

もっとも単純な構成では、すべてのデータのマスターソースを 2 つの Directory Server に置き、そのデータを 1 台または複数のコンシューマサーバーにレプリケートするようにします。2 つのマスターサーバーがあれば、1 つのサーバーが故障してオフラインになったときでも安全が保障されます。より複雑な構成では、データを複数のデータベースに格納し、データの更新または検索を行うアプリケーションの近くにあるサーバーが、エントリのマスターを作成するようにします。

複数アプリケーションにわたるデータマスターの作成

ディレクトリと間接的に通信するアプリケーションがある場合には、データのマスターソースについても考慮する必要があります。このような場合は、データの変更処理と、データ変更を実行する場所とを、できる限り単純に保ちます。単一のサイトでデータのマスターを作成することにした場合は、同じサイトでそこに含まれているほかのすべてのデータのマスターを作成します。このようにマスターの作成先を単一のサイトにしておくと、企業内でデータベースの同期がとれなくなった場合の障害追跡が簡単になります。

次に、データマスターの作成方法を示します。

- ディレクトリおよびそのディレクトリを使用しないすべてのアプリケーションの両方でデータマスターを作成する

マルチマスターを管理する場合は、ディレクトリやその他のアプリケーションとの間でデータをやり取りするためのカスタムスクリプトは必要ありません。ただし、一か所でデータが変更されると、ほかのすべてのサイトでもデータを変更する必要があります。ディレクトリおよびそのディレクトリを使用しないすべてのアプリケーションでマスターデータを管理すると、企業全体のデータが同期しなくなることがあります (このような状態をディレクトリが回避しようとする)。

- ディレクトリでデータマスターを作成し、Sun ONE Meta Directory を使用してほかのアプリケーションとデータの同期をとる

  さまざまなディレクトリ対応アプリケーションやデータベースアプリケーションを使用している場合は、ほかのアプリケーションと同期をとるデータマスターを管理する方法がもっとも合理的です。Sun ONE Meta Directory の詳細については、Sun ONE 製品の販売店にお問い合わせください。

  ディレクトリ以外のアプリケーションでデータマスターを作成してから、そのデータをディレクトリにインポートするスクリプト、プログラム、またはゲートウェイを作成する

  すでに使用しているアプリケーションの中にデータマスターを作成できるものが 1 つまたは 2 つあり、ディレクトリを検索の目的 (オンラインの企業電話帳など) にのみ使用する場合は、ディレクトリ対応以外のアプリケーションでデータマスターを作成する方法がもっとも合理的です。

データのマスターコピーの管理方法は、個々の要件によって異なります。ただし、どのような管理方法を選択しても、簡単で一貫性のあるものにしてください。たとえば、複数のサイトでデータマスターを作成し、競合するアプリケーション間で自動的にデータを交換するようなことは避けてください。そのような方法では、「最新の変更が優先される」ことになり、管理に伴うオーバーヘッドが増大します。

たとえば、ある社員の自宅の電話番号を管理する場合を考えてみます。この情報は、LDAP ディレクトリと人事データベースの両方に格納されています。人事アプリケーションは LDAP 対応なので、LDAP ディレクトリと人事データベースの間でデータを転送する自動アプリケーションを作成することができます。ここで、その社員の電話番号への変更を LDAP ディレクトリと人事管理データの両方でマスタリングすると、最後に変更した電話番号のデータが、もう一方のデータベースの情報を上書きしてしまいます。これは、最後にデータを書き込んだアプリケーションが正しい情報を持っている場合には、適切な処理として許容できます。ところが、この情報が古い場合 (たとえば、人事管理データがバックアップから再読み込みされたものである場合など) は、LDAP ディレクトリ内の正しい電話番号が削除されてしまいます。

データ所有者の決定

「データ所有者」とは、データを最新の状態に維持する責任のある個人または組織のことです。データの設計時に、ディレクトリにデータを書き込めるユーザーを決めておきます。データ所有者を決めるには、一般に次の規則に従います。

- ディレクトリの内容を管理する少人数のマネージャグループを除くすべてのユーザーに対して、ディレクトリへの読み取り専用アクセスを許可する
- 各ユーザーが自分自身に関する重要な情報を管理できるようにする

  この情報には、パスワード、そのユーザーに関する情報と組織内での役割、自動車のナンバープレート番号、自宅やオフィスの電話番号などの連絡先の情報が含まれます。
- 人に関する重要な情報 (連絡先情報や役職など) を上司が書き込めるようにする
- 組織の管理者がその組織に関するエントリを作成および管理できるようにする

  この方法では、実質的に組織の管理者が、ディレクトリの内容の管理者にもなります。
- グループ内のユーザーに読み取りと書き込みのアクセス権限を与えるロールを作成する

  たとえば、人事、財務、経理などのロールを作成します。これらのロールごとに、給与情報や政府指定の ID 番号 (米国の場合は社会保障番号)、自宅の電話番号と住所など、そのグループが必要とするデータへの読み取りアクセス権、書き込みアクセス権、またはその両方を許可します。

  ロールとエントリのグループ化については、57 ページの「ディレクトリツリーの設計」を参照してください。

データに書き込みを許可するユーザーを決定するときに、複数のユーザーに対して同じデータへの書き込み権限が必要になることがあります。たとえば、社員のパスワードへの書き込み権限を、情報システムまたはディレクトリ管理グループに許可するとします。同時に、社員にも自分のパスワードへの書き込み権限を許可する場合があります。複数のユーザーに同じ情報への書き込み権限を与えなければならない場合がありますが、このような場合はこのグループを少人数に保ち、容易に識別できるようにします。グループを少人数に保つことにより、データの整合性を維持しやすくなります。

ディレクトリのアクセス制御の設定については、第 7 章の 157 ページの「安全なディレクトリの設計」を参照してください。

データアクセスの決定

データ所有者を決定したら、各データを読み取ることができるユーザーを決定します。たとえば、ある社員の自宅の電話番号をディレクトリに保存することにした場合を考えてみます。このデータは、その社員の上司や人事部など、多くの組織にとって有用です。また、その社員自身が確認のためにこの情報を読み込めるようにしたい場合もあります。しかし、自宅の連絡先情報は機密事項とも考えられます。したがって、この種のデータを企業全体で広く利用可能にするかどうかを決定する必要があります。

ディレクトリに格納する各情報について、次の事項を決定する必要があります。

- データを匿名で読み取れるようにするか

  LDAP プロトコルでは匿名アクセスがサポーされているので、オフィスの場所、電子メールアドレス、会社の電話番号などの一般情報を簡単に検索できます。ただし、匿名アクセスの場合は、ディレクトリへのアクセス権を持つユーザーであればだれでも一般情報にアクセスできてしまいます。そのため、匿名アクセスの使用はできるだけ避けてください。

- データを企業全体で広く読み取れるようにするか

  ディレクトリにログイン (あるいはバインド) しないかぎり、特定の情報を読み取れないようにアクセス制御を設定することができます。匿名アクセスとは異なり、この形式のアクセス制御では、ディレクトリの情報を閲覧できるユーザーを組織内のメンバーだけに限定できます。また、ログイン情報をディレクトリのアクセスログに取り込めるので、情報にアクセスしたユーザーの記録を残すことができます。

  アクセス制御については、183 ページの「アクセス制御の設計」を参照してください。

- データを読み取る必要があるユーザーグループまたはアプリケーションを特定できるようにするか

  一般に、データへの書き込み特権を持つユーザーには読み取り権限も必要です (パスワードへの書き込み権限は例外)。また、特定の組織やプロジェクトグループだけが必要とするデータが存在することがあります。これらのアクセス要件を特定しておくと、ディレクトリで必要となるグループ、ロール、およびアクセス制御を決める際に役立ちます。

  グループとロールについては、57 ページの第 4 章「ディレクトリツリーの設計」を参照してください。アクセス制御については、第 7 章の 157 ページの「安全なディレクトリの設計」を参照してください。

ディレクトリの各データに対してこれらの事項を決定する際には、ディレクトリのセキュリティポリシーを定義していることが前提となります。これらの決定は、サイトの性質と、サイトですでに利用可能なセキュリティのタイプによって異なります。たとえば、サイトにファイアウォールが使用されている場合や、インターネットに直接アクセスしていない場合は、インターネット上に直接ディレクトリを配置している場合に比べ、匿名アクセスをサポートしやすくなります。

多くの国では、データ保護に関する法律によって個人情報をどのように管理すべきかが規定されており、個人情報にアクセスする人を制限しています。たとえば、法律によって住所や電話番号への匿名アクセスが禁止されていたり、住所や電話番号を表すエントリ内の情報を閲覧、訂正する許可をユーザーに与えなければならない場合があります。必ず社内の法務部に問い合わせて、ディレクトリの導入が、業務拠点としている国々の該当する法律に違反していないことを確認してください。

セキュリティポリシーの作成と実装方法については、第 7 章の 157 ページの「安全なディレクトリの設計」で詳しく説明します。

サイト調査の記録

データの設計は複雑なため、サイト調査の結果は文書に記録しておきます。サイト調査の各段階で、データを把握するための簡単な表を作成することをお勧めしました。決定事項と未解決の問題を概説する簡単な表を作成することを検討してください。使い慣れたワードプロセッサでこの表を作成したり、表の内容を簡単に保存・検索できるようにスプレッドシートを使用することもできます。

36 ページの表 2-4 は、基本的なデータ追跡例を示しています。この表には、データ所有者と、サイト調査で特定した各データについてのデータアクセス権限が示されています。

表 2-4 サイト調査を記録するためのデータ追跡表の例

データ名	所有者	マスターサーバーアプリケーション	本人による読み取り / 書き込み	全員による読み取り	人事部 (HR) による書き込み	情報システム部 (IS) による書き込み
社員の名前	HR	People Soft	読み取り専用	可 (匿名)	可	可
ユーザーパスワード	IS	Directory US-1	読み取り / 書き込み	不可	不可	可
自宅の電話番号	HR	PeopleSoft	読み取り / 書き込み	不可	可	不可
社員の所属場所	IS	Directory US-1	読み取り専用	可 (ログインが必要)	不可	可
会社の電話番号	Facilities	電話スイッチ	読み取り専用	可 (匿名)	不可	不可

社員名を表す行には、次の項目が含まれています。

- 所有者

  人事部がこの情報を所有し、この情報の更新と変更の責任を負います。

- マスターサーバー / アプリケーション

  PeopleSoft というアプリケーションで社員名に関する情報を管理します。

- 本人による読み取り / 書き込み

  ユーザーは自分の名前の読み取りはできますが、書き込み (変更) はできません。

- 全員による読み取り

  ディレクトリへのアクセス権を持つすべてのユーザーは、匿名で社員名を読み取ることができます。

- 人事部 (HR) による書き込み

  人事グループのメンバーは、ディレクトリ内の社員名を変更、追加、および削除できます。

- 情報システム部 (IS) による書き込み

  情報サービスグループのメンバーは、ディレクトリ内の社員名を変更、追加、および削除できます。

サイト調査の繰り返し

サイト調査は数回実施した方が良い場合があります。特に、複数の都市や国にオフィスを持つ企業の場合は、これが当てはまります。情報に関する要件があまりにも複雑なため、中央のサイトで情報を一元的に管理するよりも、複数の組織がそれぞれの現地オフィスで情報を保管するようにしなければならない場合もあります。このような場合は、情報のマスターコピーを保管する各オフィスで、独自のサイト調査を実施するようにします。この過程の完了後、中央のチーム (各オフィスの代表者で構成される) に各調査結果が戻され、企業全体のデータスキーマモデルとディレクトリツリーの設計に使用します。

第 3 章

スキーマの設計

第 2 章で実施したサイト調査により、ディレクトリに格納しようと計画しているデータについての情報を収集できました。次に、このデータの表現方法を決める必要があります。ディレクトリスキーマは、ディレクトリに格納できるデータのタイプを表します。スキーマの設計時には、各データ要素を LDAP 属性に割り当て、関連する要素を集めて LDAP オブジェクトクラスに入れます。スキーマを適切に設計することで、ディレクトリに格納するデータ連鎖サフィックスの整合性を維持できます。

この章では、適切なスキーマを設計する方法について説明します。この章には次の節があります。

- Sun ONE Directory Server のスキーマ
- スキーマ設計の概要
- デフォルトスキーマへのデータの割り当て
- スキーマのカスタマイズ
- データの整合性の維持
- その他のスキーマ関連資料

Directory Server のオブジェクトクラスと属性、およびスキーマファイルとディレクトリ設定属性については、『Sun ONE Directory Server Reference Manual』を参照してください。サーバー間でのスキーマのレプリケーション方法については、153 ページの「スキーマのレプリケーション」を参照してください。

Sun ONE Directory Server のスキーマ

ディレクトリスキーマは、データ値のサイズ、範囲、および形式に制限を課すことにより、ディレクトリに格納されるデータの整合性を維持します。設計者は、ディレクトリに含まれるエントリの種類 (ユーザー、デバイス、組織など) と各エントリで使用できる属性を決めます。

Directory Server に事前に定義されているスキーマには、標準の RFC LDAP スキーマ、サーバーの機能をサポートするための追加アプリケーション固有のスキーマ、および Directory Server に固有のスキーマ拡張が含まれます。定義済みのスキーマは大半のディレクトリの要件を満たしますが、ユーザー独自の要件にも対応できるように、このスキーマを拡張して新しいオブジェクトクラスと属性を追加する必要がある場合もあります。スキーマの拡張方法については、44 ページの「スキーマのカスタマイズ」を参照してください。

Directory Server は、LDAP プロトコルバージョン 3 (LDAPv3) のスキーマ形式に準拠しています。このプロトコルでは、ディレクトリサーバーが LDAP 自体を通じてスキーマを公開することによって、ディレクトリクライアントアプリケーションがプログラムを使用してスキーマを検索し、検索したスキーマに基づいて自分の動作を調整できるようにする必要があります。Directory Server のグローバルスキーマセットは、`cn=schema` というエントリに含まれています。

Directory Server のスキーマは、RFC 2256 のコア LDAPv3 スキーマだけでなく、多数の一般的なスキーマもサポートしています。また、Directory Server はスキーマエントリ内で `X-ORIGIN` という非公開フィールドを使用します。このフィールドは、スキーマエントリの定義元を示します。たとえば、スキーマエントリが標準 LDAPv3 スキーマで定義されている場合、`X-ORIGIN` フィールドの値は RFC 2252 になります。スキーマエントリが、Directory Server 用に Sun ONE で定義されている場合は、`X-ORIGIN` フィールドに `Sun ONE Directory Server` という値が入ります。

たとえば、標準の person オブジェクトクラスはスキーマ内で次のように示されます。

```
objectclasses: ( 2.5.6.6 NAME 'person' DESC 'Standard Person
 Object Class' SUP top MUST (objectlass $ sn $ cn) MAY (description
 $ seealso $ telephoneNumber $ userPassword) X-ORIGIN 'RFC 2252' )
```

このスキーマエントリは、クラスのオブジェクト識別子 (OID) (`2.5.6.6`)、オブジェクトクラス名 (`person`)、クラスの説明 (`Standard Person Object Class`)、必須の属性 (`objectclass`、`sn`、および `cn`)、および許可された属性 (`description`、`seealso`、`telephoneNumber`、および `userPassword`) を示しています。

Sun ONE Directory Server のすべてのスキーマと同様に、オブジェクトクラスは直接 Directory Server で定義され、格納されています。これは、ディレクトリのスキーマを標準の LDAP 操作で問い合わせたり、変更したりできるということです。

スキーマ設計の概要

スキーマの設計時には、Directory Server によって格納されるエントリを表すのに使用するオブジェクトクラスと属性を選択および定義します。スキーマの設計は、次の手順で行います。

- できるかぎり多くの要件を満たす、定義済みのスキーマを選択する
- Directory Server の標準スキーマを拡張して、残りの要件を満たす新しい要素を定義する
- スキーマの保守を計画する

最善の方法は、Directory Server が提供する標準スキーマに定義されている既存のスキーマ要素を使用することです。標準スキーマの要素を選択すれば、ディレクトリを使用するアプリケーションとの互換性を保証できます。また、スキーマは LDAP 標準に基づいているので、多くのディレクトリユーザーによってレビューされ、承認されています。

デフォルトスキーマへのデータの割り当て

27 ページの「サイト調査の実施」で説明したように、サイト調査で識別したデータを既存のデフォルトディレクトリスキーマに割り当てる必要があります。この節では、既存のデフォルトスキーマを表示する方法と、適切な既存のスキーマ要素にデータを割り当てる方法について説明します。

既存のデフォルトスキーマとマッチしない要素が独自のスキーマ内にある場合は、カスタマイズしたオブジェクトクラスと属性を作成する必要があります。詳細は、44 ページの「スキーマのカスタマイズ」を参照してください。

デフォルトのディレクトリスキーマの表示

Sun ONE Directory Server 5.2 で提供されるスキーマは、次のディレクトリに保存される一連のファイルに記述されています。

ServerRoot/slapd-*serverID*/config/schema

このディレクトリには、Sun ONE 製品のすべての共通スキーマが入っています。LDAPv3 標準のユーザースキーマと組織スキーマは、00core.ldif ファイルに記述されています。旧バージョンのディレクトリで使用された設定スキーマは、50ns-directory.ldif ファイルに記述されています。

注　サーバーの稼動中は、このディレクトリ内のファイルを絶対に変更しないでください。

また、手動で加えた変更は、LDAP または Directory Server コンソールを使用して別の変更を加えるまでレプリケートされないことにも注意する必要があります。

データとスキーマ要素の対応付け

サイト調査で識別したデータを既存のディレクトリスキーマに割り当てる必要があります。この作業は、次の手順で行います。

- データを表すオブジェクトのタイプを識別する

  サイト調査に記載されているデータにもっとも適したオブジェクトを選択します。データで複数のオブジェクトを記述できる場合があります。必要に応じて別の要素をディレクトリスキーマに入れるかどうかを決める必要があります。たとえば、電話番号に社員の電話番号と会議室の電話番号を記述できます。これらの電話番号データを、ディレクトリスキーマで異なるオブジェクトとみなすかどうかは設計者が決めます。

- デフォルトスキーマから類似のオブジェクトクラスを選択する

  最善の方法は、groups、people、organizations などの共通オブジェクトクラスを使用することです。

- 対応するオブジェクトクラスから類似の属性を選択する

  対応するオブジェクトクラスから、サイト調査で識別したデータにもっとも適した属性を選択します。

- サイト調査で収集したデータの中で対応しないデータを識別する

  デフォルトのディレクトリスキーマで定義されているオブジェクトクラスと属性に対応しないデータがある場合は、スキーマをカスタマイズする必要があります。詳細は、44 ページの「スキーマのカスタマイズ」を参照してください。

次の表に、ディレクトリスキーマの要素を第 2 章のサイト調査で識別したデータに割り当てた例を示します。

表 3-1　デフォルトディレクトリスキーマに割り当てられたデータ

データ	所有者	オブジェクトクラス	属性
社員名	HR	person	cn (commonName)
ユーザーパスワード	IS	person	userPassword
自宅の電話番号	HR	inetOrgPerson	homePhone

表 3-1 デフォルトディレクトリスキーマに割り当てられたデータ (続き)

データ	所有者	オブジェクトクラス	属性
社員の所属場所	IS	inetOrgPerson	localityName
会社の電話番号	Facilities	person	telephoneNumber

表では、個人を社員名で表しています。デフォルトのディレクトリスキーマには、top オブジェクトクラスから継承する person オブジェクトクラスがあります。このオブジェクトクラスには複数の属性が許可されており、その中に個人の氏名を記述する cn (commonName) という属性があります。この属性は、社員名データを入れる目的にもっとも適しています。

ユーザーパスワードも person オブジェクトに関連付けることができます。person オブジェクトの許可された属性に userPassword が含まれています。

自宅の電話番号は、特定の個人のある側面を表すものです。しかし、person オブジェクトクラスに関連するリストには、該当する属性がありません。自宅の電話番号をより本質的に分析してみますと、これは、組織的な企業ネットワークにおいて、特定の個人を表すものだといえます。このためこのオブジェクトは、ディレクトリスキーマの inetOrgPerson オブジェクトクラスに対応します。inetOrgPerson オブジェクトクラスは organizationalPerson オブジェクトクラスから継承し、organizationalPerson オブジェクトクラスは person オブジェクトクラスから継承します。inetOrgPerson オブジェクトの許可された属性の中に、社員の自宅の電話番号を入れるのに適した homePhone 属性があります。

スキーマのカスタマイズ

標準スキーマの制限が多すぎて、ディレクトリの要件に対応できない場合は、標準スキーマを拡張できます。Directory Server コンソールは、スキーマ定義の管理に役立ちます。詳細については、『Sun ONE Directory Server 管理ガイド』の第 9 章「Extending the Directory Schema」を参照してください。

スキーマをカスタマイズするときは、次の規則に留意してください。

- できるかぎり既存のスキーマ要素を再利用する。既存のすべてのスキーマ要素のリストは、『Sun ONE Directory Server Reference Manual』の「Directory Server Schema」に記載されています。
- 各オブジェクトクラスに定義する必須の属性の数を最小限にする
- 同じ目的で複数のオブジェクトクラスまたは属性を定義しない
- できるかぎりスキーマを簡潔にする

注 スキーマをカスタマイズする場合は、標準スキーマの属性またはオブジェクトクラスの既存の定義の変更、削除、および置換は行わないでください。標準スキーマを修正すると、ほかのディレクトリや LDAP クライアントアプリケーションとの互換性に問題が生じます。

カスタムのオブジェクトクラスと属性は、次のファイル内に定義されます。

ServerRoot/slapd-*serverID*/config/schema/99user.ldif

次の項目ごとに、ディレクトリスキーマのカスタマイズについて詳しく説明します。

- スキーマの拡張が必要な場合
- オブジェクト識別子の取得と割り当て
- 属性とオブジェクトクラスの命名
- 新しいオブジェクトクラスの定義戦略
- 新しい属性の定義方法
- スキーマ要素の削除
- カスタムスキーマファイルの作成 - 最良の事例と落とし穴

スキーマの拡張が必要な場合

Directory Server が提供するオブジェクトクラスと属性は、ユーザーのほとんどの要件を満たしますが、既存のオブジェクトクラスによっては組織の特殊な情報を格納できない場合もあります。また、LDAP 対応アプリケーションの独自のデータ要件に適したオブジェクトクラスや属性をサポートできるように、スキーマを拡張しなければならない場合もあります。

オブジェクト識別子の取得と割り当て

各 LDAP オブジェクトクラスまたは属性には、一意の名前とオブジェクト識別子 (OID) が割り当てられている必要があります。スキーマを定義するときは、組織に固有の OID が必要です。1 つの OID ですべてのスキーマ要件に対応できます。別の階層レベルを追加するだけで、属性とオブジェクトクラスに新しい分岐を作成できます。OID の取得とスキーマでの割り当ては、次の手順で行います。

- IANA (Internet Assigned Numbers Authority) または国内の機関から組織の OID を取得する

  国によっては、企業にすでに OID が割り当てられてます。所属する組織がまだ OID を持っていない場合は、IANA から取得できます。詳細は、IANA の Web サイト `http://www.iana.org/cgi-bin/enterprise.pl` を参照してください。

- OID の割り当てを追跡できるように、OID レジストリを作成する

  OID レジストリは、ディレクトリスキーマで使用する OID と OID の説明を提供するリストで、作成者が保持します。このレジストリにより、OID が複数の目的に使用されないようにすることができます。次に、スキーマで OID レジストリを公開する必要があります。

- スキーマ要素を入れるために、OID ツリーに分岐を作成する

  OID 分岐またはディレクトリスキーマの下に少なくとも 2 つの分岐 (1 つは属性用の *OID.1*、もう 1 つはオブジェクトクラス用の *OID.2*) を作成します。独自のマッチングルールや制御を定義する場合は、必要に応じて *OID.3* などの新しい分岐を追加できます。

属性とオブジェクトクラスの命名

新しい属性やオブジェクトクラスに名前を付けるときは、できるかぎりわかりやすいものにします。わかりやすい名前にすると、Directory Server の管理者がスキーマを使いやすくなります。

作成するすべての要素に固有の接頭辞を付けて、作成したスキーマ要素と既存のスキーマ要素間での名前の衝突を防ぎます。たとえば、Example.com 社では、各カスタムスキーマ要素の前に `Example` という接頭辞を追加しています。また、ディレクトリ内の Example.com 社員を識別するために `ExamplePerson` という特別なオブジェクトクラスを追加しています。

新しいオブジェクトクラスの定義戦略

新しいオブジェクトクラスを作成するには、次の 2 つの方法があります。

- 属性を追加するオブジェクトクラス構造ごとに 1 つずつ、多数の新しいオブジェクトクラスを作成する
- ディレクトリ用に作成するすべての属性を含む 1 つのオブジェクトクラスを作成する。このオブジェクトクラスは AUXILIARY オブジェクトクラスとして定義して作成する

2 つの方法を併用するのがもっとも簡単です。

たとえば、サイトに `ExampleDepartmentNumber` と `ExampleEmergencyPhoneNumber` という属性を作成するとします。これらの属性にいくつかのサブセットを許可する複数のオブジェクトクラスを作成できます。`ExamplePerson` というオブジェクトクラスを作成し、そのオブジェクトクラスが `ExampleDepartmentNumber` と `ExampleEmergencyPhoneNumber` を許可するようにします。`ExamplePerson` の親は `inetOrgPerson` であるとします。`ExampleOrganization` というオブジェクトクラスを作成し、そのオブジェクトクラスが `ExampleDepartmentNumber` と `ExampleEmergencyPhoneNumber` を許可するようにします。`ExampleOrganization` の親は `organization` オブジェクトクラスであるとします。

新しいオブジェクトクラスは、LDAPv3 スキーマ形式では次のようになります。

```
objectclasses: ( 1.3.6.1.4.1.42.2.27.999.1.2.3 NAME 'ExamplePerson'
 DESC 'Example Person Object Class' SUP inetorgPerson STRUCTURAL MAY
 (ExampleDepartmentNumber  $ ExampleEmergencyPhoneNumber) )

objectclasses: ( 1.3.6.1.4.1.42.2.27.999.1.2.4 NAME
 'ExampleOrganization' DESC 'Example Organization Object Class' SUP
 organization STRUCTURAL MAY (ExampleDepartmentNumber $
 ExampleEmergencyPhoneNumber) )
```

このようにする代わりに、これらの属性のすべてを許可する 1 つのオブジェクトクラスを作成して、これらの属性を使用する任意のエントリでこのオブジェクトクラスを使用できるようにします。1 つのオブジェクトクラスは、次のようになります。

```
objectclasses: (1.3.6.1.4.1.42.2.27.999.1.2.5 NAME 'ExampleEntry'
DESC  'Example Auxiliary Object Class' SUP top  AUXILIARY MAY
 (ExampleDepartmentNumber $ ExampleEmergencyPhoneNumber) )
```

新しい ExampleEntry オブジェクトクラスには、構造上のオブジェクトクラスに関係なく任意のエントリで使用できることを示す AUXILIARY が付いています。

注　この例の新しいオブジェクトクラスの OID は、Sun ONE OID 接頭辞に基づいており、配備した製品内では使用できません。独自の新しいオブジェクトクラスを作成するには、独自の OID を取得する必要があります。詳細については、45 ページの「オブジェクト識別子の取得と割り当て」を参照してください。

目的に合った、新しいオブジェクトクラスを定義する方法を選択してください。新しいオブジェクトクラスを実装する方法を決めるときは、次の点に留意します。

- 複数の STRUCTURAL オブジェクトクラスを作成すると、作成および管理するスキーマ要素の数も増える

  一般に、要素の数が少なければ、管理の手間も少なくて済みます。しかし、スキーマに 2 ～ 3 つのオブジェクトクラスを追加する場合は、1 つのオブジェクトを使用する方が簡単です。

- 複数の STRUCTURAL オブジェクトクラスを作成する場合は、より厳密かつ注意深いデータ設計が必要となる

  データを厳密に設計するには、個々のデータを配置するオブジェクトクラス構造を考慮する必要があります。この手間を有用と思う人もいれば、面倒だと思う人もいます。

- 複数のタイプのオブジェクトクラス構造に入れたいデータがある場合は、1 つの AUXILIARY オブジェクトクラスを使用した方がデータ設計が簡単になる

  たとえば、preferredOS 属性を人のエントリとグループエントリの両方に設定するとします。このような場合は、1 つのオブジェクトクラスを作成して、そのクラスでこの属性が許可されるようにします。

- 意味のあるグループを構成する実際のオブジェクトとグループ要素に関連するオブジェクトクラスを設計する

- 新しいオブジェクトクラスに必須の属性を設定しない

  必須の属性を設定するとスキーマに柔軟性がなくなります。新しいオブジェクトクラスを作成する場合は、必須の属性より許可の属性にするようにします。

新しいオブジェクトクラスを定義したら、そのオブジェクトクラスの許可された属性と必須の属性、および継承するオブジェクトクラスを決める必要があります。

新しい属性の定義方法

ディレクトリのエントリに格納する必要がある情報の中に既存のオブジェクトクラスがサポートしていないものがある場合は、新しい属性とオブジェクトクラスを追加します。

できるかぎり、標準属性を使用するようにします。デフォルトのディレクトリスキーマにある属性を探し、それらを新しいオブジェクトクラスに関連付けて使用します。対応する属性がデフォルトのディレクトリスキーマにない場合は、新しい属性を作成します。

たとえば、`person`、`organizationalPerson`、または `inetOrgPerson` の各オブジェクトクラスがサポートしている以外の情報を、個人のエントリに格納したい場合があります。ディレクトリに生年月日を格納する場合、Sun ONE Directory Server の標準スキーマには対応する属性がありません。したがって、`dateOfBirth` という新しい属性を作成し、この属性を許可する新しい補助クラスを定義して、個人を表すエントリでこの属性を使用できるようにします。

スキーマ要素の削除

Directory Server に含まれているスキーマ要素は削除しないでください。未使用のスキーマ要素は、Directory Server の動作や管理において、オーバーヘッドになることはありません。ただし、標準 LDAP スキーマの一部を削除すると、将来提供される新しいバージョンの Directory Server や LDAP 対応アプリケーションとの互換性に問題が生じる可能性があります。

拡張したスキーマの新しい要素を使用しないときは、その不要要素を削除してもかまいません。スキーマ要素を削除する前に、ディレクトリ内のどのエントリもそれを使用しないことを確認してください。削除しようとしているスキーマ要素を使用するエントリがないことを確認する最も簡単な方法は、そのスキーマ要素を含むすべての要素を返す ldapsearch を実行することです。たとえば、`myObjectClass` というオブジェクトクラスを削除する前に、次の `ldapsearch` コマンドを実行します。

```
ldapsearch -h host -p port -s base "objectclass=myObjectClass"
```

これに該当するエントリが見つかった場合は、それを削除するか、スキーマから消去される部分を削除することができます。あるスキーマ定義を使用するエントリを削除する前にそのスキーマ定義を削除すると、その定義を使用するエントリを後から変更できなくなることがあります。不明な値をエントリから削除しない限り、変更されたエントリに関するスキーマ検査も失敗します。

カスタムスキーマファイルの作成 - 最良の事例と落とし穴

Directory Server に含まれている `99user.ldif` ファイルのほかに、独自のカスタムスキーマファイルを作成できます。ただし、カスタムスキーマファイルを作成するとき、特にレプリケーションが関連する場合は、次に示すいくつかの点に注意する必要があります。

- 新たに追加したスキーマ要素をオブジェクトクラスで使用するためには、事前にすべての属性が定義されている必要があります。属性とオブジェクトクラスは同じスキーマファイル内で定義できます。
- 作成する各カスタム属性またはオブジェクトクラスは、1 つのスキーマファイル内でだけ定義されている必要があります。これにより、サーバーが最新のスキーマを読み込むときに、前の定義を上書きするのを防ぐことができます (サーバーが最初に数字順、次にアルファベット順にスキーマを読み込むため)。
- 新しいスキーマ定義を手動で作成するときは、その定義を `99user.ldif` ファイルに追加する方法が最も適しています。

  LDAP を使用するスキーマ要素を更新すると、新しい要素が自動的に `99user.ldif` ファイルに書き込まれます。このため、カスタムスキーマファイルに加えたそれ以外のスキーマ定義の変更が上書きされる可能性があり、すべてのスキーマ定義を `99user.ldif` ファイルに追加する方法が推奨されます。この場合、スキーマ要素が重複する可能性と、スキーマの変更が後から上書きされる危険を回避できます。
- Directory Server がスキーマファイルを英数字順にロードする場合、つまり、数字が小さいものから先にロードする場合、カスタムスキーマファイルの名前を次のように指定する必要があります。

  `[00-99]`*`yourname`*`.ldif`

  この数字は、すでに定義されているどのディレクトリ標準スキーマよりも大きな値にする必要があります。

  標準スキーマファイルより小さな値をスキーマファイル名に付けた場合、スキーマのロード時にサーバーでエラーが発生し、さらに、カスタムスキーマ要素のロードが完了するまで、すべての標準属性とオブジェクトクラスがロードされなくなります。

- 作成するカスタムスキーマのファイル名は、数値やアルファベットの順序で "99user.ldif" を超えないよう、注意してください。なぜなら Directory Server は、内部スキーマの管理のために、(数値順、次にアルファベット順で) もっとも大きい名前を持つファイルを使用するからです。

  作成したスキーマファイルに `99zzz.ldif` という名前を付けると、次に LDAP または Directory Server コンソールを使用してスキーマを更新したときに、`'user defined'` (通常は `99user.ldif` ファイルに格納されています) の値として `X-ORIGIN` を持つすべての属性が `99zzz.ldif` に書き込まれます。その結果、重複した情報を持つ 2 つの LDIF ファイルが存在し、`99zzz.ldif` ファイル内のいくつかの情報が削除される可能性があります。

- 原則として、追加するカスタムスキーマ要素の識別には、次の 2 つの項目を使用します。

  - カスタムスキーマファイルの `X-ORIGIN` フィールドに指定されている `'user defined'`
  - および、`X-ORIGIN` フィールドの `'Example.com Corporation defined'` のように、カスタムスキーマ要素の内容を後から判断する上で役立つより説明的な情報。たとえば、`X-ORIGIN ('user defined' 'Example.com Corporation defined')` など

  スキーマ要素を手動で追加し、`X-ORIGIN` フィールドの `'user defined'` を使用しない場合、これらの要素は Directory Server コンソールの読み取り専用セクションに表示されるため、コンソールを使用して編集することができません。

  新しいスキーマの由来と関連するその他のスキーマを特定し、理解する上で役立つので、`'user defined'` を補完する説明的な識別子を使用することをお勧めします。サーバーによって自動的に追加される `'user defined'` という値以外に説明を加えない場合、後から LDAP または Directory Server コンソールを使用してカスタムスキーマ定義を追加しなければならなくなったときに、そのスキーマが何に関連しているかを確認することが困難になります。

- 変更は自動的にレプリケートされないので、作成したカスタムスキーマファイルをすべてのサーバーに伝達するか、すべてのサーバーで手動で変更を加えることが重要です。

  ディレクトリスキーマを変更すると、サーバーはスキーマがいつ変更されたのかを示すタイムスタンプを記録します。各レプリケーションセッションの最初に、サーバーはコンシューマのタイムスタンプとこのタイムスタンプを比較し、必要であればスキーマの変更をプッシュします。カスタムスキーマファイルについては、サーバーは `99user.ldif` ファイルに関連付けられている 1 つのタイムスタンプだけを維持します。つまり、カスタムスキーマファイルに加えた変更、また

は 99user.ldif ファイル以外のファイルに対する追加は、トポロジ内のその他のサーバーにはレプリケートされません。このため、トポロジ全体にすべてのスキーマ情報が行き渡るように、作成したカスタムスキーマファイルをすべてのサーバーに伝達するか、すべてのサーバーで変更を行う必要があります。

カスタムスキーマファイルに加えた変更は、次のいずれかの方法で伝達できます。

- schema_push.pl スクリプトを実行して変更をレプリケートする (各サーバーの再起動が必要)
- 作成したカスタムスキーマファイルをすべてのサーバーに手動でコピーする

新しいカスタムスキーマ定義をレプリケーションプロセスによってすべてのサーバーにレプリケートする方法を選択した場合は、1 つのマスターだけでスキーマを維持してください。スキーマ定義が存在しないコンシューマサーバーにスキーマ定義がレプリケートされると、それを定義したカスタムスキーマファイルではなく、99user.ldif ファイルに定義が格納されます。スキーマ要素をコンシューマの 99user.ldif ファイルに格納しても、1 つのマスターサーバーだけでスキーマを管理している限り問題はありません。

各サーバーにスキーマファイルをコピーする方法を選択した場合は、変更のたびにファイルをコピーする必要があります。変更のたびにコピーしない場合、変更がコンシューマ上の 99user.ldif ファイルにレプリケートおよび格納される可能性があります。変更が 99user.ldif ファイル内に格納されると管理がむずかしくなります。これは、一部の属性が、コンシューマ上の 2 つのスキーマファイル (サプライヤからコピーした元のカスタムスキーマファイルとレプリケート後の 99user.ldif ファイル) の両方に現れるためです。

- カスタムスキーマ要素がレプリケーショントポロジ内の別のサーバーにレプリケートされないようにするには、次の処理が必要です。
  - レプリケートしたくないスキーマ要素を別のファイルに定義する
  - その要素を X-ORIGIN フィールドの 'user defined' として識別しない
  - X-ORIGIN フィールドで 'user defined' というラベルを持つスキーマだけがレプリケートされるように、nsslapd-schema-repl-useronly 属性を on に設定する

注 バージョン 5.0 または 5.1 の Directory Server にレプリケートするときは、この nsslapd-schema-repl-useronly 属性を on に設定する必要があります。

スキーマのレプリケートについては、153 ページの「スキーマのレプリケーション」を参照してください。

データの整合性の維持

Directory Server 内のスキーマの整合性を維持すると、LDAP クライアントアプリケーションがディレクトリのエントリを検出しやすくなります。ディレクトリに格納している情報のタイプごとに、その情報をサポートするために必要なオブジェクトタイプと属性を選択し、常に同じものを使用する必要があります。整合性のないスキーマオブジェクトを使用すると、ディレクトリツリー内の情報を効率よく検出できなくなります。

次のようにすると、整合性のあるスキーマを維持できます。

- スキーマ検査を使用して、属性とオブジェクトクラスが必ずスキーマ規則にマッチしていることを確認する
- 整合性のあるデータ形式を選択して適用する

次に、スキーマ内で整合性を維持する方法について詳しく説明します。

スキーマ検査

スキーマ検査は、すべての新しいまたは変更したディレクトリエントリが、スキーマ規則にマッチしているかどうかを検査します。規則にマッチしていない場合、ディレクトリは変更要求を拒否します。

注	スキーマ検査では、適切な属性があるかどうかだけを検査します。属性値が当該属性について正しい構文で記述されているかどうかは検査しません。Directory Server 5.2 には `nsslapd-valuecheck` という属性があり、これを使用することで属性値の DN 構文だけを検索することができます。ただし、この属性はデフォルトではオフに設定されており、属性値の検査は行われません。

スキーマ検査はデフォルトで有効になっています。クライアントの更新を受け付けるサーバーでは、これをオフにすることはお勧めできません。スキーマ検査をオン、オフする方法については、『Sun ONE Directory Server 管理ガイド』の「Turning Schema Checking On and Off」を参照してください。

スキーマ検査を有効にする場合は、オブジェクトクラスで定義されている必須の属性と許可された属性に注意する必要があります。通常、オブジェクトクラス定義には少なくとも 1 つの必須の属性と 1 つ以上の省略可能な属性が含まれています。省略可能な属性とは、ディレクトリのエントリに追加できるが必須ではない属性のことです。エントリのオブジェクトクラス定義で必須でなく許可されてもいない属性をエントリに追加しようとすると、Directory Server はオブジェクトクラス違反メッセージを返します。

たとえば、エントリで organizationalPerson オブジェクトクラスを使用するように定義した場合は、commonName (cn) と surname (sn) がそのエントリの必須の属性になります。これらの属性にはエントリの作成時に値を指定する必要があります。さらに、オプションとしてエントリで使用できる属性の長いリストがあります。このリストには、telephoneNumber、Uid、streetAddress、userPassword などの説明属性が含まれています。

注 スキーマ検査の機能を設定するときは、次の点に留意してください。

- 一般に、スキーマ違反を回避するために、スキーマに定義されている各エントリのすべての必須属性をレプリケートします。ただし、部分レプリケーション機能を使用して一部の必須属性をフィルタリングする場合は、スキーマ検査を無効にする必要があります。
- 必須属性をフィルタリングすると ldif ファイルをロードできなくなるため、部分レプリケーションでスキーマ検査を有効にした場合、ldif ファイルからオフラインで初期化できなくなる可能性があります。
- スキーマ検査を無効にすると、パフォーマンスを向上できることがあります。
- 部分コンシューマレプリカでスキーマ検査を無効にした場合、その部分コンシューマレプリカが存在するサーバーインスタンス全体にスキーマが適用されなくなります。このため、同じサーバーインスタンスではサプライヤ (読み取り、書き込み) レプリカを設定しないようにする必要があります。
- 部分レプリケーションの設定ではサプライヤがスキーマをプッシュするため、部分コンシューマレプリカのスキーマは、マスターレプリカのスキーマのコピーとなり、適用される部分レプリケーション設定に対応しません。

整合性のあるデータ形式の選択

LDAP スキーマを使用して、必要なデータを任意の属性値に格納できます。ただし、LDAP クライアントアプリケーションとディレクトリユーザーに適切な形式を選択して、ディレクトリツリー内で整合性を維持するようにデータを格納することが重要です。

LDAP プロトコルと Sun ONE Directory Server を使用する場合は、RFC 2252 で規定されているデータ形式でデータを表す必要があります。

また、電話番号の正しい LDAP 形式は、次の ITU-T 勧告ドキュメントで定義されています。

- ITU-T 勧告 E.123

  国内および国際電話番号に関する注記

- ITU-T 勧告 E.163

  国際電話サービスの番号計画

たとえば、米国の電話番号は次のような形式になります。

```
+1 555 222 1717
```

`postalAddress` 属性には、区切り文字としてドル記号 ($) を使用する複数行文字列形式の属性値が必要です。適切に形式化されたディレクトリエントリは次のようになります。

```
postalAddress: 1206 Directory Drive$Pleasant View, MN$34200
```

レプリケートされたスキーマでの整合性の維持

レプリケート環境で整合性のあるスキーマを維持するには、次の点に留意します。

- コンシューマサーバーのスキーマを変更しない

  コンシューマサーバーのスキーマを変更すると、マスターサーバーのスキーマよりも新しいスキーマとなります。したがって、マスターがレプリケーションで更新をコンシューマに送信すると、コンシューマのスキーマが新しいデータをサポートできないため、多数のレプリケーションエラーが発生します。

- マルチマスターレプリケーション環境では、1 つのマスターサーバーだけでスキーマを変更する

  2 つのマスターサーバーのスキーマを変更すると、最後に更新されたマスターのスキーマがコンシューマに伝達されます。これは、コンシューマのスキーマと他方のマスターのスキーマとの間に整合性がないことを意味します。

注	Directory Server 5.2 では、RFC2307 に準拠するためにスキーマファイル 11rfc2307.ldif は変更されました。このファイルは 10rfc2307.ldif に対応します (5.1 zip インストールの場合)。5.2 サーバーと 5.1 サーバーの間のレプリケーションが有効な場合、レプリケーションが正しく機能するには、5.1 サーバー上の RFC2307 スキーマを修正する必要があります。5.2 インスタンスから 5.1 インスタンスに 11rfc2307.ldif ファイルをコピーし、10rfc2307.ldif ファイルを削除します。

スキーマレプリケーションについては、153 ページの「スキーマのレプリケーション」を参照してください。

その他のスキーマ関連資料

標準 LDAPv3 スキーマについては、次のドキュメントを参照してください。

- Internet Engineering Task Force (IETF) http://www.ietf.org
- 『Understanding and Deploying LDAP Directory Services』 (T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年)
- RFC 2252: LDAPv3 Attribute Syntax Definitions http://www.ietf.org/rfc/rfc2252.txt
- RFC 2256: Summary of the X.500 User Schema for Use with LDAPv3 http://www.ietf.org/rfc/rfc2256.txt
- RFC 2251:Lightweight Directory Access Protocol (v3) http://www.ietf.org/rfc/rfc2251.txt

第 4 章

ディレクトリツリーの設計

ディレクトリ情報ツリー (DIT) を使用することで、ディレクトリに格納されているデータを参照することができます。格納した情報のタイプ、企業の地理的な特性、ディレクトリで使用するアプリケーション、および使用するレプリケーションのタイプによって、ディレクトリツリーの設計方法が決まります。この章では、ディレクトリツリーの設計手順について概要を説明します。説明する内容は次のとおりです。

- ディレクトリツリーの概要
- ディレクトリツリーの設計
- ディレクトリエントリのグループ化と属性の管理
- ディレクトリツリーの設計例
- その他のディレクトリツリー関連資料

ディレクトリツリーの概要

ディレクトリツリーを使用すると、クライアントアプリケーションからディレクトリデータに名前を付けたり、参照したりできるようになります。ディレクトリツリーは、ディレクトリデータの分散、レプリケート、アクセス制御の方法など、その他の設計判断と緊密に連携します。最初に適切なディレクトリツリーを設計することは、配備後に不適切なディレクトリツリーを設計しなおすより時間的にずっと効率的です。

適切に設計されたディレクトリツリーでは、次のことが可能になります。

- ディレクトリデータの管理を簡単にする
- レプリケーションポリシーとアクセス制御の作成における柔軟性
- ディレクトリを使用するアプリケーションをサポートする
- ユーザーが簡単にディレクトリを操作する

ディレクトリツリーの構造は、階層型の LDAP モデルに従います。ディレクトリツリーでは、たとえば、グループ、ユーザー、あるいは場所ごとにデータを編成できます。また、ディレクトリツリーによって複数のサーバー間でどのようにデータを分散して配置するかが決まります。たとえば、各データベースでは、サフィックスレベルでデータを分割する必要があります。適切なディレクトリツリー構造がなければ、複数のサーバー間で希望どおりにデータを分散することができない場合があります。

これらの留意点のほかに、選択するディレクトリツリー構造の種類によって、設定できるレプリケーションの可能性が制限されることに注意してください。ディレクトリツリーのパーティションは、レプリケーションが正しく機能するように定義する必要があります。ディレクトリツリーの一部だけをレプリケートするのであれば、ディレクトリツリーの設計時にそれを設計に反映させる必要があります。同様に、分岐点にアクセス制御を適用する場合も、ディレクトリツリーの設計時にそれを考慮する必要があります。

ディレクトリツリーの設計

この節では、ディレクトリツリーの設計段階で決定する事柄について説明します。ディレクトリツリーの設計では、データを格納するサフィックスの選択、データエントリ間の階層関係の決定、ディレクトリツリー階層内のエントリのネーミングを行います。次に、ディレクトリツリー設計の各段階について詳しく説明します。

- サフィックスの選択
- ディレクトリツリー構造の作成
- エントリのネーミング

サフィックスの選択

サフィックスは、ツリーのルートにあるエントリの名前です。サフィックスの下にディレクトリデータが格納されます。ディレクトリには複数のサフィックスを持たせることができます。一般的なルートポイントを持たない情報のディレクトリツリーが複数ある場合、複数のサフィックスを使用することもできます。

デフォルトでは、Sun ONE Directory Server を標準的な設定で配備すると、データの格納用に 1 つのサフィックスが、設定情報やディレクトリのスキーマなど、ディレクトリの内部処理に必要なデータ用に複数のサフィックスが使用されます。これらの標準ディレクトリサフィックスについては、『Sun ONE Directory Server 管理ガイド』の第 3 章「Creating Your Directory Tree」を参照してください。

サフィックスの命名規則

ディレクトリ内のすべてのエントリは、サフィックスと呼ばれる共通のベースエントリの下に格納する必要があります。ディレクトリには複数のサフィックスを持たせることができます。ディレクトリサフィックスに名前を付けるときは、次の推奨事項に注意してください。

- グローバルに一意の名前にする
- 変更しない、あるいはまれにしか変更しないようにする
- そのサフィックスの下にあるエントリが画面上で読みやすいように短い名前にする
- ユーザーが容易に入力および記憶できるものにする

1 つの企業環境では、企業の DNS 名またはインターネットドメイン名に合わせてディレクトリのサフィックスを選択します。たとえば、企業が `Example.com` というドメイン名を所有している場合は、次のようなディレクトリサフィックスを使用します。

```
dc=example,dc=com
```

`dc` (`domainComponent`) 属性は、ドメイン名をコンポーネントに分割してサフィックスを表します。

通常、サフィックスの名前を付けるときには、任意の属性を使用できます。ただし、ホスティングサービス事業者環境では、サフィックスに使用する属性は次のものに限定することをお勧めします。

- `c` (`countryName`)

  ISO の定義に準拠した、国名を表す 2 桁のコードを含めます。

- `l` (`localityName`)

  エントリが存在する、あるいはエントリに関連付けられた国や都市などの地域を示します。

- `st` (`stateOrProvinceName`)

  エントリがある州または県を示します。

- `o` (`organizationName`)

  エントリが属する組織の名前を示します。

これらの属性がサフィックスに含まれていると、顧客のアプリケーションとの相互運用性が高まります。たとえば、ホスティングサービス事業者がこれらの属性を使用して、クライアントの Example.com に、次のようなルートサフィックスを作成する場合を考えてみます。

```
o=Example.com,st=Washington,c=US
```

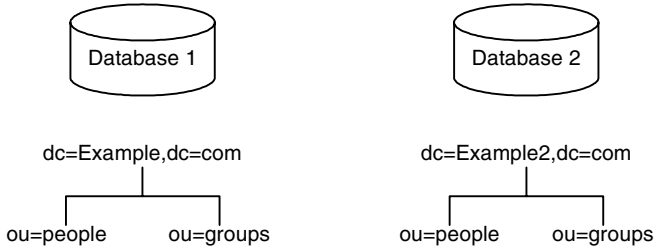
これらの属性については、62 ページの表 4-1 を参照してください。

組織名のあとに国名コードを使用する指定方法は、X.500 のサフィックスの命名規則に従っています。

複数のサフィックスの命名

ディレクトリで使用する各サフィックスが、それぞれ固有のディレクトリツリーを示します。複数のディレクトリツリーを作成し、各ディレクトリツリーを Directory Server によって提供される別々のデータベースに格納することができます。たとえば、60 ページの図 4-1 に示すように Example.com と Example2.com 用に別々のサフィックスを作成し、それぞれを異なるデータベースに格納できます。

図 4-1 2 つの異なるデータベースに格納される 2 つのサフィックス



データベースは、リソースの制限に応じて 1 つのサーバーまたは複数のサーバーに格納できます。

ディレクトリツリー構造の作成

ツリーの構造をフラットなものにするか階層型にするかを決定する必要があります。一般には、ディレクトリツリーはできるだけフラットにします。ただし、あとで複数のデータベース間にデータを分散したり、レプリケートできるようにしたり、アクセス制御を設定したりする場合は、ある程度階層化することも必要です。

ツリー構造を決定するには、次の手順と検討事項に従います。

- ディレクトリの分岐点の作成
- 分岐点の特定
- レプリケーションに関する検討事項
- アクセス制御に関する検討事項

ディレクトリの分岐点の作成

階層を設計するときは問題の生じる名前の変更を避けるようにします。ネームスペースがフラットなほど、名前を変更する確率は低くなります。名前を変更する確率は、名前が変更される可能性があるコンポーネントが、ネームスペース内に多く含まれているほど高くなります。ディレクトリツリーの階層が深いほどネームスペース内のコンポーネントは多くなり、名前を変更する確率が高くなります。

次に、ディレクトリツリーの階層設計に適用されるガイドラインを示します。

- 企業組織内でもっとも大きい部門区分のみを表すようにツリーを分岐させる

  このような分岐点は、部門 (企業情報サービス、カスタマサポート、販売サービス、専門サービスなど) だけを表します。ディレクトリツリーを分岐させる部門は、安定したものにします。組織変更が頻繁に行われる場合は、この種の分岐は使用しないようにします。

- 分岐点には組織の実際の名前ではなく、機能を表す名前または一般的な名前を使用する

  組織名は変更されることが考えられます。会社が部門の名前を変更するたびにディレクトリツリーを変更する必要に迫られるのでは困ります。代わりに、組織の機能を表す一般的な名前を使用します (たとえば、「`Widget` 研究開発」ではなく「エンジニアリング」を使用します)。

- 似たような機能を持つ組織が複数ある場合は、部門の構成に基づいて分岐点を作成するのではなく、その機能を表す分岐点を 1 つ作成する

  たとえば、特定の製品ラインを担当する複数のマーケティング部門がある場合でも、1 つのマーケティング サブツリーのみを作成します。すべてのマーケティングエントリは、そのツリーに所属させます。

次に、会社とホスト環境に固有のガイドラインを示します。

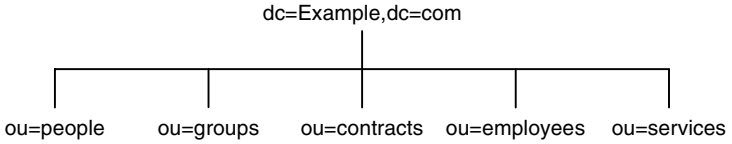
企業環境での分岐点の作成

変更される可能性の低い情報に基づいてディレクトリ構造を決定すれば、名前の変更を避けられます。たとえば、組織ではなくツリー内のオブジェクトのタイプに基づいて構造を定義します。次に、ツリー構造の定義に使用するオブジェクトの例を示します。

- `ou=people`
- `ou=groups`
- `ou=contracts`
- `ou=employees`
- `ou=services`

62 ページの図 4-2 は、これらのオブジェクトを使用した Example.com 社のディレクトリツリーの構造例を示しています。

図 4-2 5 つの分岐点を持つディレクトリ情報ツリーの例



よく使用されている従来型の属性だけを使用することをお勧めします (62 ページの表 4-1 を参照)。従来からある属性を使用すると、サードパーティの LDAP クライアントアプリケーションとの互換性が保たれる可能性が高くなります。また、従来からある属性は、デフォルトのディレクトリスキーマで認識可能なので、分岐 DN のエントリを作成しやすくなります。

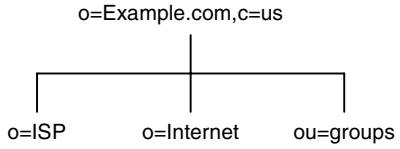
表 4-1 従来からある DN 分岐点の属性

属性名	定義
c	国名
o	組織名。通常、この属性は、企業の部門、教育機関での学部 (人文学部、理学部など)、子会社、企業内の主要部門など、大きな部門を表すために使用する。また、59 ページの「サフィックスの命名規則」で説明したように、ドメイン名を表す場合もこの属性を使用する必要がある
ou	組織の構成単位。通常この属性は、組織よりも小さな組織内の部門を表すために使用する。組織単位は、一般にすぐ上の組織に属する
st	州あるいは県の名前
l	地域 (都市、地方、オフィス、施設名など)
dc	59 ページの「サフィックスの命名規則」で説明されているドメインのコンポーネント

ホスト環境での分岐点の作成

ホスト環境では、`organization(o)` オブジェクトクラスの 2 つのエントリと `organizationalUnit(ou)` オブジェクトクラスの 1 つのエントリをルートサフィックスの下に含むツリーを作成します。63 ページの図 4-3 は、ISP である Example.com のディレクトリ分岐の例を示しています。

図 4-3 ISP Example.com のディレクトリ情報ツリー



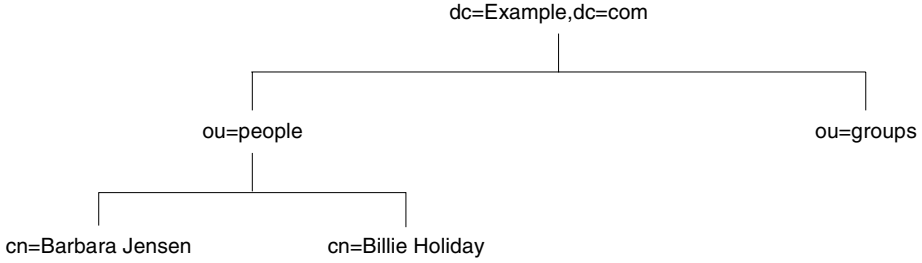
分岐点の特定

ディレクトリツリーをどのように分岐させるかを決めるには、その分岐点を特定するために使用する属性を決定することが必要になります。DN は、属性とデータのペアで構成される一意の文字列です。たとえば、Example.com 社の社員 Barbara Jensen 用のエントリの DN は次のようになります。

```
cn=Barbara Jensen,ou=people,dc=example,dc=com
```

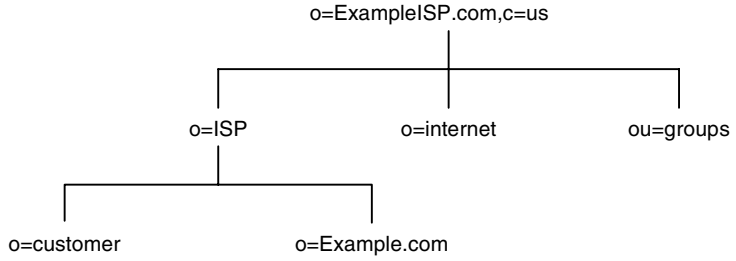
各属性とデータのペアは、ディレクトリツリーの分岐点を表します。63 ページの図 4-4 は、Example.com という企業のディレクトリツリーの例を示しています。

図 4-4 Example.com 社のディレクトリ情報ツリー



64 ページの図 4-5 は、インターネットホストを持つ ExampleHost.com 社のディレクトリツリーを示しています。

図 4-5 ExampleHost.com 社のインターネットホストディレクトリ情報ツリー



サフィックスのエントリ `o=ExampleHost.com,c=US` の下で、ツリーは 3 つに分岐しています。ISP の分岐には、顧客データと Example.com の社内情報が含まれています。internet の分岐は、ドメインのツリーです。groups の分岐には、管理グループに関する情報が含まれています。

分岐点の属性を選択するときは、一貫性を持たせることが重要です。ディレクトリツリー全体で識別名 (DN) の形式が統一されていないと、一部の LDAP アプリケーションで混乱が生じる可能性があります。つまり、ディレクトリツリーのある部分で `l (localityName)` が `o (organizationName)` の下位にある場合、ディレクトリのほかの部分でも `l` が `o` の下位にあることを確認してください。

注 よくある間違いに、識別名で使用されている属性に基づいてディレクトリを検索してしまうことがあります。識別名はディレクトリエントリをほかと識別するだけのもので、これを検索対象にすることはできません。ただし、エントリ自体に格納された属性とデータのペアに基づいてエントリを検索することは可能です。

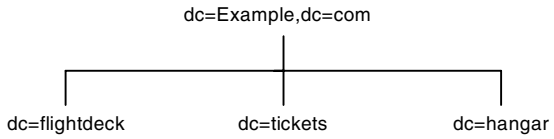
レプリケーションに関する検討事項

ディレクトリツリーの設計時に、レプリケートするエントリを検討します。エントリセットをレプリケートする場合は、サブツリーの頂点で識別名 (DN) を指定し、その下にあるエントリをすべてレプリケートするのが自然な方法です。また、このサブツリーは、ディレクトリデータの一部を含むディレクトリパーティションである、データベースに対応します。

たとえば、企業環境では自社内のネットワーク名に対応させてディレクトリツリーを編成できます。ネットワーク名が変更されることはほとんどないので、ディレクトリツリー構造は安定したものになります。また、レプリケーションを使用して別のディレクトリサーバーを連動させる場合は、ネットワーク名を使用してディレクトリツリーの最上位の分岐点を作成する方法が有効です。

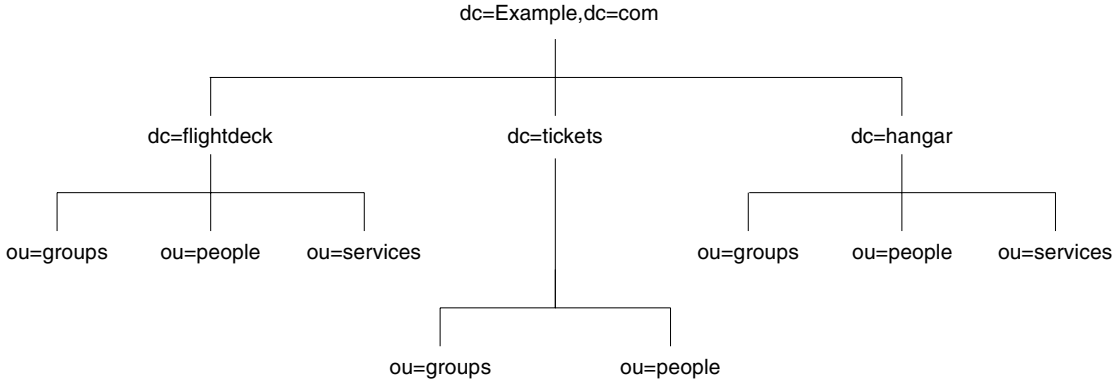
たとえば、Example.com 社に `flightdeck.Example.com`、`tickets.Example.com`、`hanger.Example.com` という 3 つのプライマリネットワークがあるとします。65 ページの図 4-6 は、このディレクトリツリーの最初の分岐を示しています。

図 4-6 Example.com 社 DIT の 3 つの主要ネットワーク



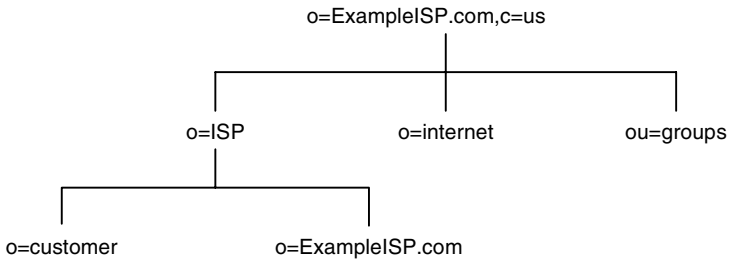
ツリーの最初の構造を作成した後、ネットワークをさらに 65 ページの図 4-7 のように分岐させています。

図 4-7 Example.com 社 DIT の 3 つの主要ネットワークの詳細



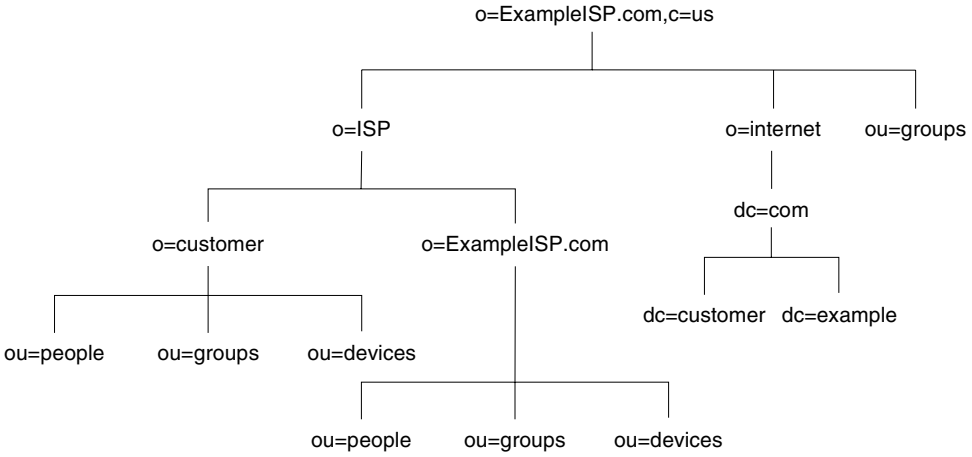
もう一つ例を示します。66 ページの図 4-8 は、ISP 企業である ExampleISP.com のディレクトリ分岐を示しています。

図 4-8　ExampleISP.com 社のディレクトリ情報ツリー



ディレクトリツリーの最初の構造を作成した後、ネットワークをさらに 66 ページの図 4-9 のように分岐させています。

図 4-9　ExampleISP.com 社の詳細な DIT



どちらの企業も、あまり変更されることのない情報に基づいてデータ階層を設計しています。

アクセス制御に関する検討事項

ディレクトリツリーを階層化すると、特定のタイプのアクセス制御を使用できるようになります。レプリケーションの場合と同様に、類似したエントリをグループ化すると、それらのエントリを 1 つの分岐点から簡単に管理できます。

また、ディレクトリツリー階層で管理を分散させることができます。たとえば、営業部の管理者に営業のエントリへのアクセス権を与え、マーケティング部の管理者にマーケティングのエントリへのアクセス権を与える場合は、ディレクトリツリーの設計を通じてアクセス権を付与することができます。

ディレクトリツリーではなくディレクトリの内容に基づいてアクセス制御を設定することができます。ACI フィルタを適用するターゲットメカニズムを使用すると、ディレクトリエントリが特定の属性値を含むすべてのエントリへのアクセスを持つように規定した、1 つのアクセス制御規則を定義できます。たとえば、`ou=Sales` という属性を含むすべてのエントリへのアクセス権を営業部の管理者に与える ACI フィルタを設定できます。

ただし、ACI フィルタは管理が簡単ではありません。ディレクトリツリー階層で組織に対応した分岐を作成する、ACI フィルタを適用する、あるいは両者を組み合わせるなどして、ディレクトリにもっとも適したアクセス制御方法を決める必要があります。ディレクトリで ACI を管理できるように、Sun ONE Directory Server 5.2 には getEffectiveRights という機能が用意されています。この機能は、指定したユーザーがディレクトリのエントリと属性に対して持つアクセス制御権を要求、取得します。getEffectiveRights 機能を使用することで、ユーザーの管理、アクセス制御ポリシーの検証、デバッグを簡単に実行できます。詳細については、第 7 章「安全なディレクトリの設計」で説明します。

エントリのネーミング

ディレクトリツリー階層を設計したあとは、ツリー構造内のエントリに名前を付けるときに使用する属性を決定する必要があります。一般に、名前は 1 つ以上の属性値を選び、相対識別名 (RDN) を形成して作成します。RDN とは、一番左の DN 属性値のことです。名前を付けるエントリのタイプによって使用する属性が変わります。

エントリの名前は、次の規則に従って付ける必要があります。

- 変更される可能性の低い属性を選んで名前を付ける
- 名前はディレクトリ全体で一意でなければならない。名前を一意にすると、DN によってディレクトリ内の複数のエントリが参照されることがなくなる

エントリを作成するときは、エントリ内で RDN を定義します。エントリ内で少なくとも RDN を定義しておけば、簡単にエントリを検出できます。これは、検索が実際の DN を対象にして実行されるのではなく、エントリ自体に格納されている属性値を対象に実行されるからです。

属性名には意味があるので、属性が表すエントリのタイプに合った属性名を使用するようにします。たとえば、組織を表すために `l` (`locality`) を使用したり、組織の構成単位を表すために `c` (`country`) を使用したりしないでください。

エントリに名前を付けるときの手法について、次の項目ごとに説明します。

- 人のエントリのネーミング
- 組織エントリのネーミング
- その他のエントリのネーミング

人のエントリのネーミング

人のエントリの名前 (DN) は一意である必要があります。従来、識別名ではその人のエントリに名前を付けるときに、commonName または cn 属性を使用しています。つまり、Babs Jensen という名前のユーザーのエントリには、次のような識別名が付けられます。

```
cn=Babs Jensen,dc=example,dc=com
```

このエントリと関連付けられているユーザーを識別するのは簡単ですが、同じ名前のユーザーが組織に 2 人いる場合は一意にならない場合があります。この場合、DN の名前の衝突として知られている、複数のエントリが同じ識別名を持つ問題が生じます。

共通名の衝突は、一意の識別子を共通名に追加することで避けられます。たとえば、次のようにします。

```
cn=Babs Jensen+employeeNumber=23,dc=example,dc=com
```

ただし、大きなディレクトリの場合は、扱いにくい共通名となり、管理が難しくなります。

より良い方法は、cn 以外の属性で人のエントリを特定することです。次のいずれかの属性を使用することを検討してください。

- uid

  Uid (userID) 属性を使用して、個人に固有な値を指定します。たとえば、ユーザーのログイン ID や社員番号などが使用できます。ホスト環境の加入者は、uid 属性で識別する必要があります。

- mail

  mail 属性を使用して、個人の電子メールアドレスの値を追加します。この方法でも、重複した属性値を含む扱いにくい DN になる場合があるので (たとえば mail=bjensen@Example.com, dc=example,dc=com)、Uid 属性で使用可能な一意の値がなかった場合にのみ、この方法を使用します。たとえば、会社が社員番号やユーザー ID を臨時社員や契約社員に割り当てない場合は、Uid 属性の代わりに mail 属性を使用します。

- employeeNumber

  inetOrgPerson オブジェクトクラスの社員には、employeeNumber などの会社側が割り当てた属性値を使用することを検討します。

人のエントリの RDN の属性とデータのペアにどのような属性値を使用する場合でも、必ず一意で永続的な値を使用します。

ホストされる環境での人のエントリに関する検討事項

ユーザーがサービスの加入者の場合、そのエントリは inetUser オブジェクトクラスにし、Uid 属性を含める必要があります。属性は顧客のサブツリーで一意である必要があります。

ユーザーがホスティングサービス事業者に属している場合は、nsManagedPerson オブジェクトクラスの inetOrgPerson として表します。

DIT 内への人のエントリの配置

次に、ディレクトリツリーに人のエントリを配置する場合のガイドラインを示します。

- 企業内のユーザーのエントリは、組織のエントリの下にあるディレクトリツリーに配置する必要がある
- ホスティングサービス事業者の加入者は、ホストされる組織の ou=people 分岐の下に配置する必要がある

組織エントリのネーミング

組織エントリ名は、ほかのエントリと同様に一意である必要があります。ほかの属性値と組織の法的な名前を組み合わせて使用すると、名前は確実に一意になります。たとえば、次のように組織エントリに名前を付けます。

```
o=Example.com+st=Washington,o=ISP,c=US
```

登録商標を使用することもできますが、一意である保証はありません。

ホスト環境では、組織エントリに次の属性を含ませる必要があります。

- o (organizationName)
- top、organization、および nsManagedDomain の値を持つ objectClass

その他のエントリのネーミング

地域、州、国、デバイス、サーバー、ネットワーク情報、その他のタイプのデータなど、ディレクトリには多くのものを表すエントリが含まれています。

これらのタイプのエントリには、可能な場合は RDN 内で commonName (cn) 属性を使用してください。たとえば、グループエントリに名前を付ける場合は、次のように名前を付けます。

```
cn=allAdministrators,dc=example,dc=com
```

ただし、commonName 属性がサポートされていないオブジェクトクラスを持つエントリに名前を付けなければならないこともあります。この場合は、エントリのオブジェクトクラスでサポートされている属性を使用します。

エントリの DN で使用される属性とエントリ内で実際に使用されている属性が対応している必要はありません。ただし、指定する属性を DN で見えるようにすると、ディレクトリツリーを簡単に管理できます。

ディレクトリエントリのグループ化と属性の管理

ディレクトリツリーは、エントリの情報を階層構造で構成します。階層もグループ化メカニズムの 1 つですが、分散しているエントリの関連付け、頻繁に変更される組織、または多数のエントリで繰り返されるデータには適していません。Directory Server には、グループ化メカニズムとしてグループとロールも提供しています。こちらのほうがエントリ間の関係を柔軟に定義できます。

これらのグループ化メカニズムのほかに、Directory Server ではサービスクラス (CoS) というメカニズムを利用できます。これは、アプリケーションの介入なしでエントリ間で属性を共有できるように、属性を管理するメカニズムです。ロールメカニズムと同様に、エントリが検出されると、CoS がそのエントリに対して仮想属性を生成します。ただし、CoS はメンバーを定義するのではなく、一貫性の保持と容量の節約のために、関連するエントリがデータを共有できるようにします。

次に、エントリのグループ化と属性管理のメカニズム、およびそれぞれの利点と制限について説明します。

- スタティックグループとダイナミックグループ
- 管理されているロール、フィルタを適用したロール、入れ子のロール
- ロールの列挙とロールメンバーシップの列挙
- ロールの範囲
- ロールの制限事項
- グループとロールのどちらを使用するかの決定
- サービスクラス (CoS) による属性の管理
- CoS について
- CoS 定義エントリと CoS テンプレートエントリ
- CoS の優先順位
- ポインタ CoS、間接 CoS、クラシック CoS
- CoS の制限事項

スタティックグループとダイナミックグループ

グループとは、そのメンバーであるほかのエントリを特定するエントリです。グループのメンバーの範囲は、ディレクトリ全体です。グループ定義エントリの場所は関係しません。グループ定義エントリは、特に変更を頻繁に行う企業では、グループ化メカニズムに柔軟性を与えてくれます。グループ名がわかっている場合は、そのすべてのメンバーエントリを簡単に検索できます。次に、スタティックグループとダイナミックグループの特徴について説明します。どのような場合にどちらのグループを使用するべきかを理解してください。

- スタティックグループは、そのメンバーエントリの名前を明示的に指定します。スタティックグループを定義するエントリは、`groupOfNames` または `groupOfUniqueNames` オブジェクトクラスを使用し、各メンバーの DN をそれぞれ `member` または `uniqueMember` の属性値として含みます。`member` 属性には、サーバーがグループのメンバーシップを確立するときにチェックする DN が含まれ、`uniqueMember` 属性には、DN とオプションとして一意の識別子が続く構文が含まれます。メンバーシップのチェックは、この構文に対して行われます。ただし現時点では、Directory Server はネイティブアクセス制御処理の `groupOfNames` (`member` 属性) だけをサポートしています。`uniqueMember` 属性の構文については、『RFC 2256: A Summary of the X.500(96) User Schema for use with LDAPv3』(http://www.ietf.org/rfc/rfc2256.txt) を参照してください。
- スタティックグループは、ディレクトリ管理者のグループのようにメンバーの少ないグループに適しており、数千のメンバーが含まれるグループには適していません。パフォーマンスが著しく低下するため、メンバー数が 20,000 を超えるスタティックグループを作成することは避けてください。これ以上のサイズのグループには、ダイナミックグループまたはロールを使用することをお勧めします。メンバー数が 20,000 を超えるグループを定義するためにスタティックグループを使用する必要がある場合は、1 つの大きなスタティックグループを使用するのではなく、グループのグループ化を使います。
- ダイナミックグループはフィルタを指定し、そのフィルタと一致するすべてのエントリがグループのメンバーとなります。このようなグループは、フィルタが評価されるたびにメンバーが定義されるため、ダイナミックグループと呼ばれます。ダイナミックグループの定義エントリは、`groupOfUniqueNames` および `groupOfURLs` オブジェクトクラスに属します。グループメンバーシップは、`memberURL` 属性の LDAP URL 値として表現される 1 つまたは複数のフィルタ、または `uniqueMember` 属性の値として表現される 1 つまたは複数の DN としてリストされます。

注 別のグループの DN をダイナミックグループの `uniqueMember` 属性に挿入することで、グループ内に別のグループを入れ子にすることができます。

どちらのグループも、ディレクトリ内のどこにいるメンバーでも識別できますが、グループ定義自体は、ou=Groups などの適切な名前のノードに格納することをお勧めします。たとえば、このように格納することで、バインドの証明情報がグループのメンバーである場合に、アクセスを許可または制限するアクセス制御命令 (ACI) を定義するときに、検索が簡単になります。

管理されているロール、フィルタを適用したロール、入れ子のロール

ロールはエントリをグループ化する新しいメカニズムで、エントリがメンバーとなっているすべてのロールを自動的に特定します。各ロールはメンバー (そのロールを所有するエントリ) を持ちます。グループと同じようにロールのメンバーを明示的またはダイナミックに指定できます。ディレクトリ内のエントリを取得するときに、そのエントリがメンバーとして属するすべてのロール定義の DN を含む nsRole 属性がロールメカニズムによって自動的に生成されるため、そのエントリがどのロールに属するかを直ちに知ることができます。これにより、グループ化メカニズムの主な欠点が解消されます。

ディレクトリがすべてのロールのメンバーを自動的に算出するため、ロールメカニズムはクライアントから非常に簡単に使用できます。ロールに属しているすべてのエントリに nsRole 仮想属性が指定されます。この属性の値は、そのエントリがメンバーになっているすべてのロールの DN です。nsRole が仮想属性であるのは、実行中にサーバーによって生成され、実際にはディレクトリに格納されないためです。つまり、ロールを使って処理を実行すると、グループを使う場合よりもサーバー側でより多くのリソースが消費されます。これは、クライアントアプリケーションのためにサーバーがその処理を実行するためです。ただし、ロールのメンバーの検査方法は一貫しており、サーバー側で透過的に実行されます。

Sun ONE Directory Server は、次の 3 種類のロールをサポートしています。

- 管理されているロール : 明示的にメンバーエントリにロールを割り当てる
- フィルタを適用したロール : 指定した LDAP フィルタと一致するエントリを割り当てる。このため、ロールは各エントリに含まれている属性によって異なる
- 入れ子のロール : ほかのロールを含むロールを作成できる

管理されているロール

管理されているロールは、メンバーがロール定義エントリではなく各メンバーエントリ内で定義されていることを除いて、スタティックグループと似ています。管理されているロールを使用すると、管理者は、対象となるエントリに nsRoleDN 属性を追加することにより、特定のロールを割り当てることができます。この属性の値は、ロール定義エントリの DN です。スタティックロールの定義エントリは、対象の範囲だけを定義します。そのロールのメンバーは指定範囲内で、nsRoleDN 属性がロール定義エントリの DN を指定しているエントリです。

フィルタを適用したロール

フィルタを適用したロールはダイナミックグループと似ており、ロールのメンバーを決定するフィルタを定義します。nsRoleFilter 属性の値は、フィルタを適用したロールを定義します。サーバーが、フィルタ文字列と一致する、フィルタを適用したロールの適用範囲内のエントリを返す場合、そのエントリにはロールを識別する nsRole 属性が常に含まれています。

入れ子のロール

入れ子のロールはほかのロールの定義エントリをリストし、それらのロールのすべてのメンバーを組み合わせます。つまり、あるエントリが入れ子のロールにリストされているロールのメンバーである場合、そのエントリは入れ子のロールのメンバーでもあることになります。入れ子のロールを使用して、別のロールを含むロールを作成できます。

ロールの列挙とロールメンバーシップの列挙

ロールの列挙

nsRole 属性はほかの属性と同様に読み取られます。クライアントはこの属性を使用して、任意のエントリが属しているすべてのロールを列挙することができます。ロールメカニズムで使用されるのは、nsRole 属性だけで、この属性はすべての変更操作から保護されています。ただし、読み取りは可能なので、ロールメンバーシップへの読み取りアクセスがセキュリティ要件で禁じられている場合は、アクセス制御を定義して読み取りから保護してください。

ロールメンバーシップの列挙

Directory Server の従来のリリースでは実行できなかった仮想属性に対する検索が可能になりました。これにより、nsRole 属性を検索し、ロールのメンバーを列挙することができます。ただし、検索操作でインデックスが付けられていない属性は、パフォーマンスに大きな影響を与える可能性があるため、注意が必要です。等価フィル

タに基づく検索の多くにはインデックスが付けられるため効率的ですが、たとえば否定検索にはインデックスが付けられず、パフォーマンスの低下を招くことになります。nsRoleDN 属性にはデフォルトでインデックスが付けられるので、管理されているロールに対する検索は比較的効率的です。ただし、フィルタを適用したロールと入れ子のロールでは、インデックスが付けられた属性と付けられていない属性の両方がフィルタに含まれている可能性があるため、インデックスが付けられていない検索を行わないように、少なくとも 1 つのインデックス付き属性がフィルタに含まれていることを確認する必要があります。

ロールの範囲

Directory Server の従来のリリースでは、ロールの範囲はロール定義のサブツリーに限定されていました。異なるサブツリーの間でロールを共有するには、そのサブツリーがツリーのルートにある必要があったため、複数のサブツリーにまたがるロールは限定され、管理も複雑でした。Example.com という大企業のエンジニアリング部門に勤務するエンジニアが、販売部門によって制御されるアプリケーションにアクセスしなければならない場合を考えてください。ロールが販売部門の管理者のために作成され、販売サブツリー内のこのアプリケーションへのアクセスを制御している場合、エンジニアリングサブツリーには属しているが、販売サブツリーには属さないこのエンジニアは、ロールの範囲がサブツリーに限定されているため、そのアプリケーションにアクセスすることはできません。グループに同じ範囲制限が適用された場合 (現在でも適用されます)、それを解決するには、そのアプリケーションにアクセスできる販売部門のユーザーのグループにエンジニアを追加する必要がありました。しかし、この解決策はグループベースであり、結果としてロールメカニズムの利点は失われます。

Sun ONE Directory Server 5.2 では、ロールの範囲をロール定義エントリのサブツリーを超えて拡張するための新しい属性を使用できます。これは、nsRoleScopeDN という 1 つの値を属性で、既存のロールに追加する範囲の DN を含みます。

注　範囲拡張のための新しい nsRoleScopeDN 属性を追加できるのは、入れ子のロールだけです。

販売アプリケーションにアクセスする必要のある、前述の Example.com のエンジニアの例で、ロールの範囲をどのように拡張できるかについて説明します。Example.com のディレクトリツリーには、エンジニアリングサブツリー用の o=eng,dc=example,dc=com と、販売サブツリー用の o=sales, dc=example,dc=com という 2 つのサブツリーがあると仮定します。販売アプリケーションへのアクセスを管理する SalesAppManagedRole という販売サブツリーロールの範囲を拡張し、エンジニアリングサブツリーの 1 人のユーザーを含めるには、次の手順を実行します。

1. エンジニアリングサブツリーに、たとえば EngineerManagedRole のようにエンジニアユーザーのロールを作成します。(前述の例では管理されているロールを作成しているが、フィルタを適用したロールでも、入れ子のロールでも構わない。)
2. 販売サブツリーに、たとえば SalesAppPlusEngNestedRole のような入れ子のロールを作成し、新たに作成した EngineerManagedRole ロールと、当初からの SalesAppManagedRole ロールを格納します。
3. 新しい SalesAppPlusEngNestedRole 入れ子ロールに nsRoleScopeDN 属性を追加します。属性値には、追加するエンジニアリングサブツリーの範囲の DN を指定します。この例では、o=eng,dc=example,dc=com を指定します。

新しい SalesAppPlusEngNestedRole 入れ子ロールは、次のようになります。

```
dn:cn=SalesAppPlusEngNestedRole,dc=example,dc=com
objectclass: LDAPsubentry
objectclass:nsRoleDefinition
objectclass:nsComplexRoleDefinition
objectclass:nsNestedRoleDefinition
nsRoleDN:cn=SalesAppManagedRole,o=sales,dc=example,dc=com
nsRoleDN:cn=EngineerManagedRole,o=eng,dc=example,dc=com
nsRoleScopeDN:o=eng,dc=example,dc=com
```

アクセス制御の点から見ると、販売アプリケーションにアクセスするエンジニアリングユーザーには、SalesAppPlusEngNestedRole ロールおよび実際の販売アプリケーションにアクセスするための適切なアクセス権を与えることが重要です。同様に、レプリケーションの面から考えると、ロールの範囲全体をレプリケートする必要があります。これは、拡張された範囲をレプリケートしない場合に問題が発生するためです。

注 範囲の拡張は入れ子のロールに限定されているため、これまで 1 つのドメインのロールを管理していた管理者は、その他のドメインにすでに存在するロールを使用する権限だけを持つことになり、その他のドメインで任意のユーザーの任意のロールを作成することはできなくなります。

ロールの制限事項

ディレクトリサービスをサポートするロールを作成する場合は、次の制限事項を考慮する必要があります。

- ロールと連鎖

連鎖機能を使用してディレクトリツリーを複数のサーバーに分散している場合は、ロールを定義するエントリをそれらのロールを所有するエントリと同じサーバーに配置する必要があります。連鎖を介して、サーバー A が別のサーバー B からエントリを受け取る場合は、それらのエントリにはサーバー B で定義されたロールが含まれますが、サーバー A で定義されたロールは割り当てられません。

- フィルタを適用したロールは CoS によって生成された属性を使用できない

フィルタを適用したロールでは、CoS 仮想属性の値に基づくフィルタ文字列を使用できません。詳細は、79 ページの「CoS について」を参照してください。ただし、CoS 定義の指示子属性は、ロール定義によって生成された `nsRole` 属性を参照できます。ロールベースの属性の作成については、『Sun ONE Directory Server 管理ガイド』の第 5 章に記載されている「Creating Role-Based Attributes」を参照してください。

- ロールの範囲拡張

ロールの範囲は、同一サーバーインスタンス上の異なるサブツリーに限定されます。ロールの範囲を別のサーバーにまで拡張すると、予期せぬ動作が生じることがあります。

グループとロールのどちらを使用するかの決定

グループとロールのメカニズムは機能の一部が重複しているため、あいまいさを招く可能性があります。エントリをグループ化するどちらの方法にも、利点と欠点があります。一般に、より新しく設計されたロールメカニズムのほうが、頻繁に必要となる機能を効率的に提供できるように設計されています。しかし、グループ化メカニズムは、サーバーの複雑さに影響を及ぼし、クライアントによるメンバー情報の処理方法を決めるため、グループ化メカニズムは慎重に計画する必要があります。どのメカニズムが適しているかを判断するには、メンバーシップクエリーと、実行する必要のある管理操作の種類を理解する必要があります。次の 2 つの項目では、設計時の選択に役立てられるように、両方のメカニズムの長所と短所について説明します。管理者がグループ化ポリシーを後から一貫して維持できるように、選択内容を文書化しておくことを強くお勧めします。

この節には、次の項目があります。

- グループメカニズムの利点
- ロールメカニズムの利点

グループメカニズムの利点

- メンバーシップが 20,000 以下の場合にメンバーを列挙するには、スタティックグループが適している

  特定セットのメンバーを列挙するだけであれば、スタティックグループを使用するほうが負担が少なくなります。ただし、20,000 を超えるメンバーを含むスタティックグループはパフォーマンスを低下させるため、メンバー数が 20,000 以下であることが前提となります。member 属性を取得してスタティックグループのメンバーを列挙することは、同じロールを共有するすべてのエントリを調べるより簡単であるため、メンバーの列挙操作により適しています。

- メンバーの割り当てと削除などの管理操作には、スタティックグループが適している

  グループにユーザーを追加する上で、特別なアクセス権を追加する必要がないため、メンバーの割り当てと削除を行う場合の最適なグループ化メカニズムは、スタティックグループとなります。

  グループエントリを作成する権限を持つということは、グループにメンバーを割り当てる権限を持つということになります。これに対して、管理されているロールとフィルタが適用されたロールでは、管理者はユーザーエントリに nsroledn 属性を書き込む権限も持つ必要があります。このアクセス権の制限は、入れ子のロールにも間接的に適用されます。これは、入れ子のロールを作成するということは、すでに定義されているその他のロールをまとめる権限が必要であることを意味するためです。

- フィルタベースの ACI で使用するには、ダイナミックグループが適している

  ACI でのバインドルールの指定など、フィルタに基づいてすべてのメンバーを検索するだけの場合は、ダイナミックグループを使用します。フィルタを適用したロールはダイナミックグループと似ていますが、nsRole 仮想属性を生成するロールメカニズムを開始します。クライアントが nsRole 値を必要としない場合は、ダイナミックグループを選択することで、この計算のオーバーヘッドを回避できます。

- 既存のセットに対してセットの追加や削除を行うときは、グループが適している

  既存のセットに対してセットの追加や削除を行う場合は、入れ子の制限がないため、グループメカニズムが最も適しています。ロールメカニズムでは、他のロールを受け入れられるのは入れ子のロールに限られています。

- グループ化範囲の自由度が配備で重要になる場合は、グループが適している

  グループのメンバー範囲は、グループ定義エントリの場所に関係なくディレクトリ全体であるため、範囲の面ではグループには柔軟性があります。ロールでも、特定のサブツリーを超えて範囲を拡張することができますが、入れ子のロールに範囲拡張属性 nsRoleScopeDN を追加する必要があるため、範囲拡張には制限があります。

ロールメカニズムの利点

- 指定セットのメンバーを列挙し、指定エントリのすべてのメンバーを検索する場合は、ロールが適している

  指定セットのメンバーを列挙し、指定エントリのすべてのメンバーを検索する場合は、ロールメカニズムが最も適しています。ロールはこの情報をユーザーエントリにプッシュします。そこでは情報をキャッシュできるので、以後のメンバーシップのテストをより効率的に行えます。サーバーがすべての計算を実行し、クライアントは `nsRole` 属性の値を読み取るだけです。さらに、この属性にすべてのタイプのロールが含まれ、クライアントはすべてのロールを均等に処理できます。一般に、グループよりもロールの方が、両方の処理をより効率よく実行でき、クライアント側での処理が簡単になります。

- CoS、パスワードポリシー、アカウントの無効化、ACI など、Directory Server の既存のグループ化メカニズムを統合する場合は、ロールが適している

  サーバー上のセットのメンバーシップを「自然に」使用する、つまり、メンバーシップに関する計算をサーバーが自動的に行う利点を活用する場合は、ロールメカニズムが最適です。これはロールが設計された目的でもあります。ロールは、リソース指向の ACI で、CoS のベースとして、より複雑な検索フィルタ、パスワードポリシー、アカウントの無効化などの一部として使用することができます。グループを使用してこのような統合を行うことはできません。

サービスクラス (CoS) による属性の管理

ディレクトリエントリのグループ化と属性の管理に関する節の冒頭でも説明したように、サービスクラス (CoS) メカニズムを使用することで、アプリケーションによる介入なしでエントリ間で属性を共有することができます。CoS は、ロールメカニズムと同じように、エントリの取得時にエントリの仮想属性を生成します。CoS は、メンバーシップを定義しません。ロールメカニズムのようにエントリをグループ化するのではなく、一貫性の保持と容量の節約のために、関連するエントリがデータを共有できるようにします。ここでは、CoS メカニズムの詳細について説明します。説明する内容は次のとおりです。

- CoS について
- CoS 定義エントリとテンプレートエントリ
- CoS の優先順位
- ポインタ CoS、間接 CoS、クラシック CoS
- CoS の制限事項

CoS について

facsimileTelephoneNumber 属性の値が等しい何千ものエントリがディレクトリに格納されているとします。従来のやり方で FAX 番号を変更するには、各エントリを個別に更新する必要がありました。これは管理者にとって大きな負担であり、時間がかかるというだけでなく、全てのエントリが更新されないという危険がありました。CoS を使用すると、FAX 番号は 1 か所に保存され、Directory Server は、エントリが返されるときに、関係のあるすべてのエントリに対して facsimileTelephoneNumber 属性を自動的に生成します。

クライアントアプリケーションでは、生成された CoS 属性はほかの属性と同じように検出されます。ただし、ディレクトリ管理者が管理するのは 1 つの FAX 番号値だけです。さらに、実際にディレクトリに格納される値が少ないため、データベースが使用するデータ領域が少なくて済みます。また、CoS メカニズムを使用すると、生成された値をエントリで上書きすることも、同じ属性に対して複数の値を生成することも可能です。

注 CoS 仮想属性にはインデックスが付けられないため、LDAP 検索フィルタで参照した場合、パフォーマンスに影響が生じる可能性があります。

生成される CoS 属性には、複数のテンプレートにより複数の値が設定されている場合があります。指示子が複数のテンプレートエントリを指定する場合もあれば、同じ属性に複数の CoS 定義がある場合もあります。また、選択したすべてのテンプレートから 1 つの値だけが生成されるように、テンプレートの優先順位を指定することもできます。詳細については、『Sun ONE Directory Server 管理ガイド』の第 5 章に記載されている「Defining Class of Service (CoS)」を参照してください。

ロールとクラシック CoS を組み合わせて使用すると、ロールベースの属性を指定できます。エントリは関連付けられたサービスクラステンプレートを持つ特定のロールを所有しているため、これらの属性がエントリ上に現れます。たとえば、ロールに基づいた属性を使用して、ロール単位で検索制限を設定できます。

CoS の機能は再帰的に使用できます。つまり、Directory Server では、CoS で生成されたほかの属性によって指定される CoS から属性を生成できます。CoS スキーマを複雑にすると、クライアントアプリケーションから情報へのアクセスが単純化され、繰り返し使用する属性を簡単に管理できるようになりますが、同時に管理が複雑になり、サーバーのパフォーマンスが低下します。極端に複雑な CoS スキーマは避けてください。たとえば、多くの間接 CoS スキーマは、クラシック CoS またはポインタ CoS として再定義することができます。

最後に、必要以上に CoS 定義を変更しないようにすることが非常に重要です。サーバーは CoS 情報をキャッシュに保存するため、CoS 定義への変更がすぐには反映されません。キャッシュを利用することで、生成された属性エントリに高速にアクセスできるようになりますが、CoS 情報が変更されると、サーバーはキャッシュを再構築しなくてはなりません。これは、多少時間のかかるタスク (通常は数秒) です。キャッシュの再構築中は、読み取り操作は新たに変更された情報ではなく、古いキャッシュに残されされている情報に対して行われます。このため、CoS 定義を頻繁に変更した場合、古くなったデータにアクセスする可能性が高くなります。

CoS 定義エントリと CoS テンプレートエントリ

CoS メカニズムは、CoS 定義エントリと CoS テンプレートエントリという 2 種類の支援エントリに依存します。ここでは、2 つのエントリの詳細について説明します。説明する内容は次のとおりです。

- CoS 定義エントリ
- CoS テンプレートエントリ

CoS 定義エントリ

CoS 定義エントリは、使用中の CoS のタイプおよび生成される CoS 属性の名前を特定します。このエントリは、ロール定義エントリと同様に、`LDAPsubentry` オブジェクトクラスから継承されます。CoS 偏差の範囲は、定義エントリの格納場所によって決定されます。この範囲は、CoS 定義エントリの親の下にあるサブツリー全体となります。定義エントリの親の分岐内のすべてのエントリを CoS 定義のターゲットエントリと呼びます。同じ CoS 属性に複数の定義が存在することもあります。この場合は、複数の値が含まれます。

CoS 定義エントリは、`cosSuperDefinition` オブジェクトクラスのインスタンスです。CoS 定義エントリは、CoS のタイプを指定する、次のオブジェクトクラスのいずれかから継承されます。

- `cosPointerDefinition`
- `cosIndirectDefinition`
- `cosClassicDefinition`

CoS 定義エントリには、必要に応じて、仮想 CoS 属性、テンプレート DN、およびターゲットエントリの指示子属性を指定できるように、CoS のそれぞれのタイプに固有の属性が含まれています。デフォルトでは、CoS メカニズムは、CoS 属性と同じ名前を持つ既存の属性の値を上書きしません。ただし、CoS 定義エントリの構文を使用することで、この動作を制御できます。

注 スキーマ検査が有効になっている場合は、その CoS 属性を設定できるすべてのターゲットエントリにこの属性が生成されます。スキーマ検査が無効になっている場合は、すべてのターゲットエントリに CoS 属性が生成されます。

CoS テンプレートエントリ

CoS テンプレートエントリには、CoS 属性について生成される値が含まれます。CoS の適用範囲内のすべてのエントリで、ここに定義された値が使用されます。それぞれが異なる値を持つ複数のテンプレートが存在することもあります。この場合、生成される属性は複数の値を持ちます。CoS メカニズムは、定義エントリとターゲットエントリの内容に基づいていずれかの値を選択します。

CoS テンプレートエントリは、`cosTemplate` オブジェクトクラスのインスタンスです。CoS テンプレートエントリには、CoS メカニズムによって生成された 1 つ以上の値が含まれます。特定の CoS 用のテンプレートエントリは、その CoS 定義と同じレベルのディレクトリツリー内に格納されます。

注 管理を容易にするため、定義エントリとテンプレートエントリはできるだけ同じ場所に格納してください。また、それが提供する機能を説明するような名前を付けることをお勧めします。たとえば、定義エントリ DN に `"cn=classicCosGenerateEmployeeType,ou=People,dc=example,dc=com"` などの名前を付けると、`"cn=ClassicCos1,ou=People,dc=example,dc=com"` よりもわかりやすくなります。CoS の各タイプに関連するオブジェクトクラスと属性については、『Sun ONE Directory Server 管理ガイド』の第 5 章に記載されている「Defining Class of Service (CoS)」を参照してください。

CoS の優先順位

属性値を得る上で互いに競合する CoS スキームが作成される場合があります。たとえば、CoS 定義エントリに複数の値を持つ cosSpecifier があると仮定します。このような場合、どのテンプレートが属性値を提供するかを決定するために、各テンプレートエントリにテンプレート優先順位を指定することができます。テンプレート優先順位の設定には、cosPriority 属性を使用します。この属性は、特定のテンプレートのグローバルな優先順位を数字で表します。優先順位 0 は、優先順位が最も高いことを示します。

cosPriority 属性を持たないテンプレートは、優先順位が最も低いとみなされます。2 つ以上のテンプレートが属性値を提供する状況で、その優先順位が同じまたは設定されていない場合は、任意の値が選択されます。

次に、CoS テンプレートエントリの例を示します。

```
dn: cn=exampleUS,cn=data
objectclass:top
objectclass: cosTemplate
postalCode: 44438
cosPriority: 0
```

このテンプレートエントリには、postalCode 属性の値が含まれます。優先順位にはゼロが指定されているため、別の postalCode 値を持つ競合テンプレートに優先して適用されます。

ポインタ CoS、間接 CoS、クラシック CoS

テンプレートの選択方法が異なり、それによって生成される値が異なる 3 つのタイプの CoS があります。次に、これらの 3 種類の CoS について詳しく説明します。

- ポインタ CoS
- 間接 CoS
- クラシック CoS

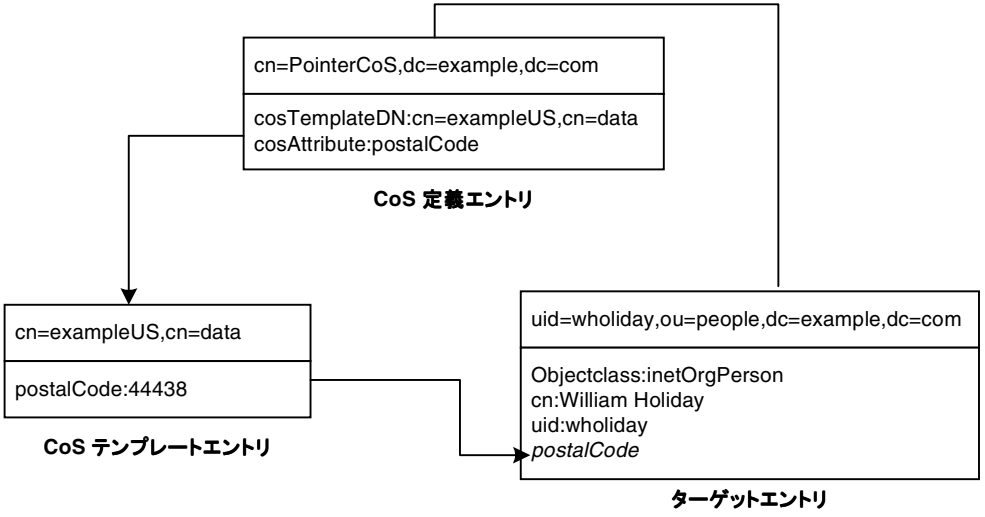
ポインタ CoS

ポインタ CoS はもっとも単純です。ポインタ CoS 定義エントリによって、cosTemplate オブジェクトクラスの特定のテンプレートエントリの DN が決定されます。すべてのターゲットエントリに、このテンプレートで定義されているものと同じ CoS 属性値が設定されます。

ポインタ CoS の例

この例では、dc=example,dc=com の下に格納されるすべてのエントリに共通の郵便番号を定義する CoS を示します。83 ページの図 4-10 は、この例の 3 つのエントリ (CoS 定義エントリ、CoS テンプレートエントリ、ターゲットエントリ) を示しています。

図 4-10　ポインタ CoS の定義とテンプレートの例



テンプレートエントリは、CoS 定義エントリ内の DN (cn=exampleUS,cn=data) によって特定されます。エントリ dc=example,dc=com で postalCode 属性が照会されるたびに、Directory Server は、テンプレートエントリ cn=exampleUS,cn=data 内の使用可能な値を返します。したがって、郵便コードは、エントリ uid=wholiday,ou=people,dc=example,dc=com と一緒に表示されますが、このエントリには格納されません。エントリ内に実際に存在する属性の代わりに、CoS によって生成されるいくつかの属性が数千または数百万のエントリで共有される例を考えると、CoS を利用することで節約できる格納のための容量とパフォーマンスの向上を理解することができます。

間接 CoS

間接 CoS を使用すると、ディレクトリ内の任意のエントリをテンプレートにして、CoS 値を指定することができます。間接 CoS 定義エントリは、間接指示子と呼ばれる属性を識別します。ターゲットエントリに含まれるこの属性の値によって、そのエントリで使用されるテンプレートが決定されます。ターゲットエントリ内の間接指示子属性には、DN が含まれています。間接 CoS を使用することで、各ターゲットエントリで異なるテンプレートを使用できるため、CoS 属性に異なる値を指定することができます。

たとえば、`departmentNumber` 属性を生成する間接 CoS では、指示子として社員の上司を使用できます。ターゲットエントリを検索する場合、CoS メカニズムはテンプレートとして `manager` 属性の DN 値を使用します。そして、上司の部門番号と同じ値を使用して、社員の `departmentNumber` 属性を生成します。

警告

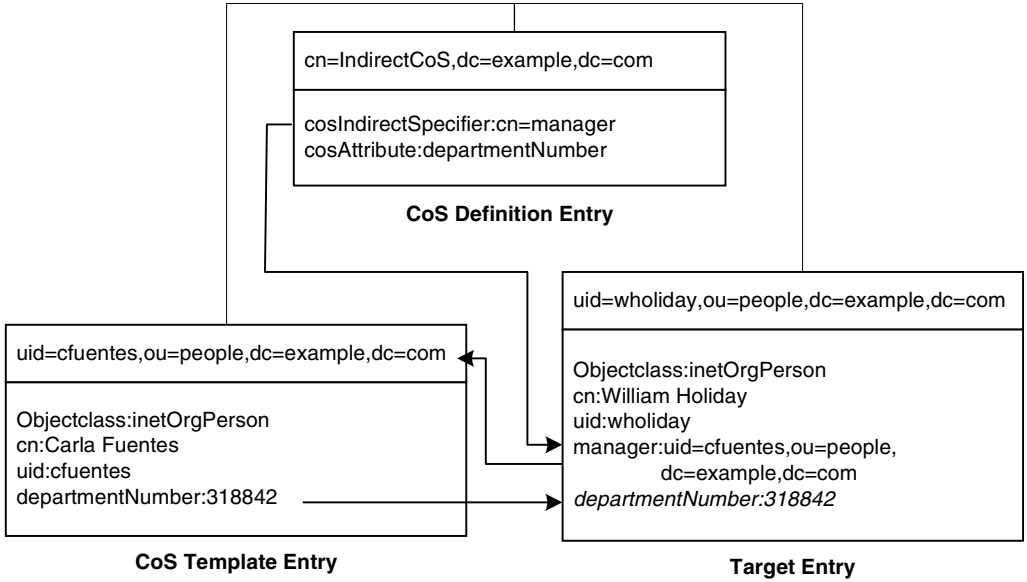
間接 CoS は使いすぎないようにしてください。テンプレートは、ディレクトリツリー内の任意の場所にある任意のエントリである可能性があるため、間接 CoS へのアクセス制御の実装は、非常に複雑になります。間接 CoS はリソースを多用するため、パフォーマンスが重要な配備でも使いすぎを避けるべきです。

多くの場合、クラシック CoS を使用してターゲットエントリの場所を限定するか、比較的柔軟性の低いポインタ CoS メカニズムを使用して、間接 CoS を使用した場合と同様の結果を得ることができます。

間接 CoS の例

次の間接 CoS の例では、ターゲットエントリの `manager` 属性を使用してテンプレートエントリを特定しています。CoS メカニズムでは、この方法で、すべての従業員に対して上司と同じ `departmentNumber` 属性を生成することにより、常に最新の状態を維持できます。85 ページの図 4-11 は、この例の 3 つのエントリを示しています。

図 4-11　間接 CoS の定義とテンプレートの例



間接 CoS の定義エントリは、指示子属性の名前を指定します。この例では、`manager` 属性です。William Holiday のエントリは、この CoS のターゲット エントリの 1 つであり、その `manager` 属性には、`cn=Carla Fuentes,ou=people,dc=example,dc=com` の DN が含まれます。したがって、Carla Fuentes のエントリは、`departmentNumber` 属性値 `318842` を提供するテンプレートです。

クラシック CoS

クラシック Co s は、ポインタ CoS と間接 Co s の動作を組み合わせたものです。クラシック CoS の定義エントリは、テンプレートのベース DN と指示子属性を特定します。次のように、ターゲットエントリの指示子属性の値は、テンプレートエントリの DN の構築に使用されます。

`cn=`*`specifierValue,baseDN`*

CoS 値を含むテンプレートは、ターゲットエントリの指示子属性の RDN (relative domain name) 値とテンプレートのベース DN を組み合わせることで決定されます。

クラシック CoS テンプレートは、任意の間接 CoS テンプレートに関連するパフォーマンスの問題を回避するための、`cosTemplate` オブジェクトクラスのエントリです。

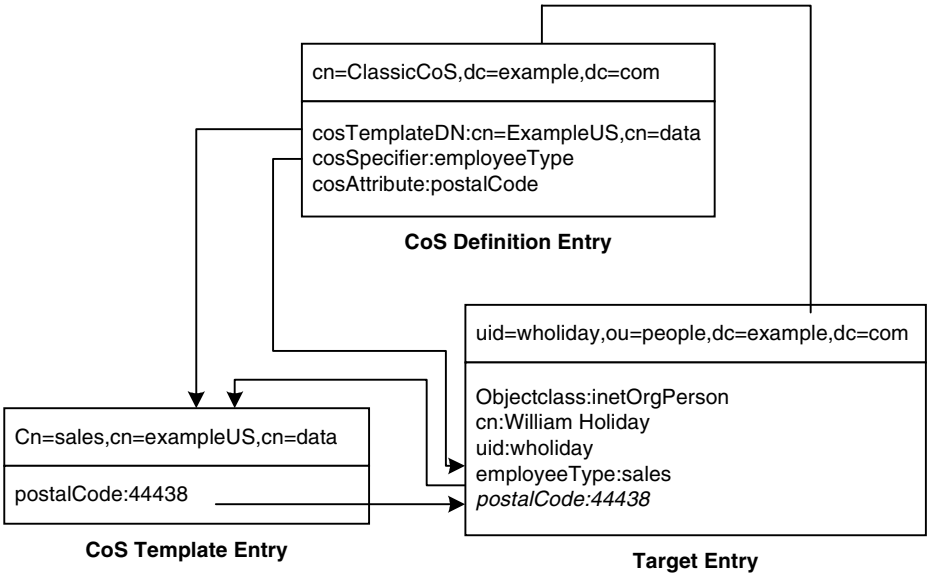
クラシック CoS の例

クラシック CoS メカニズムでは、定義エントリで指定されたベース DN とターゲットエントリの指示子属性からテンプレートの DN が決定されます。指示子属性の値は、テンプレート DN の cn 値として使用されます。したがって、クラシック CoS のテンプレート DN は、次のような構造になります。

cn=*specifierValue,baseDN*

86 ページの図 4-12 の例は、郵便番号属性の値を生成するクラシック CoS の定義を示しています。

図 4-12 クラシック CoS の定義とテンプレートの例



この例では、CoS 定義エントリの cosSpecifier 属性が、employeeType 属性を指定します。この属性とテンプレート DN を組み合わせると、テンプレートエントリを cn=sales,cn=exampleUS,cn=data として識別できます。このテンプレートエントリは、postalCode 属性の値をターゲットエントリに与えます。

CoS の制限事項

CoS 機能は、複雑なメカニズムであり、パフォーマンスおよびセキュリティ上の理由から次の制限事項が適用されます。

- サブツリーの制限

`cn=config` サブツリーと `cn=schema` サブツリーには、CoS 定義を作成できません。

- 属性が CoS 生成属性として宣言されているサフィックスの検索は、インデックスが付けられていない検索となる

属性が CoS 生成属性として宣言されているサフィックスの検索は、インデックスが付けられていない検索となります。これは、パフォーマンスに重大な影響を生じる可能性があります。同じ属性が CoS 属性として宣言されていないサフィックスでは、検索にインデックスが付けられます。

- 属性タイプの制限

次の属性タイプは、同じ名前の実際の属性と動作が異なるため、CoS メカニズムでは生成できません。

  - `userPassword` : CoS で生成されたパスワード値は、ディレクトリサーバーへのバインドに使用できない
  - `aci` : Directory Server では、CoS によって定義された仮想 ACI 値の内容に基づいてアクセス制御を適用しない
  - `objectclass` : Directory Server では、CoS によって定義された仮想オブジェクトクラスの値を検査するスキーマが実行されない
  - `nsRoleDN` : CoS によって生成された `nsRoleDN` 値は、サーバーによるロールの生成に使用されない

- すべてのテンプレートをローカルに配置する必要がある

CoS 定義またはターゲットエントリの指示子に指定されているテンプレートエントリの DN は、ディレクトリ内のローカルエントリを参照する必要があります。テンプレートとそこに含まれる値は、ディレクトリ連鎖またはリフェラルからは取得できません。

- CoS 仮想値と実際の値と組み合わせることはできない

CoS 属性の値では、エントリの実際の値とテンプレートの仮想値を組み合わせることはできません。CoS により実際の属性値が上書きされると、実際の値はすべてテンプレートの値に置き換えられます。ただし、『Sun ONE Directory Server 管理ガイド』の「CoS Limitations」で説明しているように、CoS メカニズムでは、複数の CoS 定義の仮想値を組み合わせることができます。

- フィルタが適用されたロールは CoS によって生成される属性を使用できない

フィルタを適用したロールでは、CoS 仮想属性の値に基づくフィルタ文字列を使用できません。ただし、CoS 定義の指示子属性は、ロール定義によって生成された `nsRole` 属性を参照できます。詳細については、『Sun ONE Directory Server 管理ガイド』の「Creating Role-Based Attributes」を参照してください。

- アクセス制御命令 (ACI)

格納されている通常の属性へのアクセスと同様に、CoS によって生成された属性へのアクセスが制御されます。ただし、CoS によって生成された属性値に依存するアクセス制御規則は、87 ページの「属性タイプの制限」で説明されている条件に従います。

- CoS キャッシュの応答時間

CoS キャッシュは、パフォーマンスを向上させるためにすべての CoS データをメモリに保持する Directory Server の内部構造です。このキャッシュは、仮想属性の算出時に使用される CoS データの取得用に最適化されており、CoS 定義エントリおよびテンプレートエントリの更新中でも使用できます。したがって、定義エントリおよびテンプレートエントリを追加または変更すると、変更内容が反映されるまでわずかに時間がかかる場合があります。この遅延時間は、CoS 定義の数と複雑さ、および現在のサーバーの負荷によって異なりますが、通常は、数秒間かかります。複雑な CoS 設定を行うときは、この遅延時間に留意してください。

ディレクトリツリーの設計例

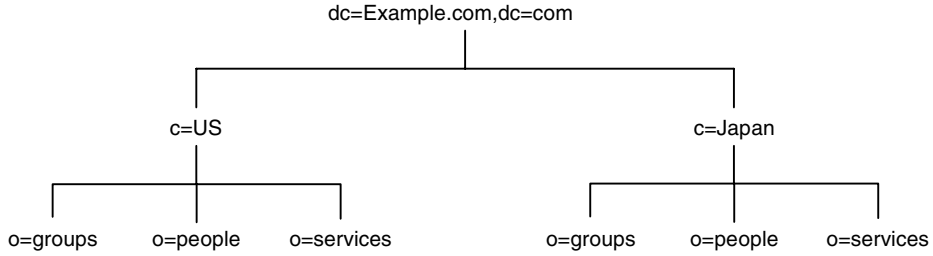
次に、フラットな階層をサポートするディレクトリツリーの設計例と、より複雑な階層を含む設計例を示します。

国際企業向けのディレクトリツリー

国際企業の要件に対応するには、ディレクトリツリーのルートをインターネットのドメイン名に設定し、そのルートポイントのすぐ下で、企業が事業を展開している国ごとにツリーを分岐させます。59 ページの「サフィックスの命名規則」では、国名指示子をディレクトリツリーのルートとして設定しないように勧めています。これは、企業が国際的な組織である場合には、特に当てはまります。

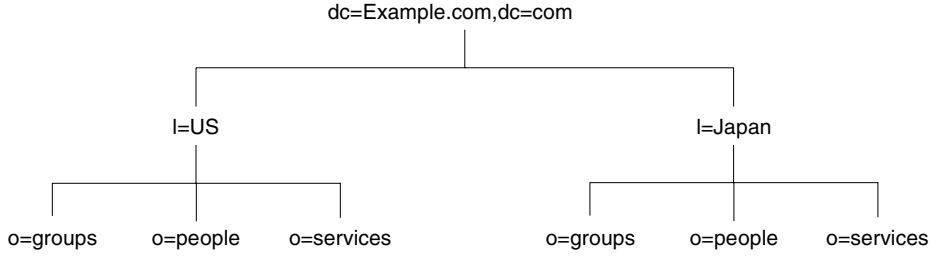
LDAP は DN 内の属性の順序については何の規制も設けていないため、89 ページの図 4-13 のように `c` (`countryName`) 属性を使用して各国の分岐を表すことができます。

図 4-13　c (countryName) 属性を使用してディレクトリツリーで国を表現した例



ただし、この方法ではスタイルに統一感がないと感じる管理者もいるので、代わりに `l` (`locality`) 属性を使用して、各国を次のように表すこともできます。

図 4-14　l (locality) 属性を使用してディレクトリツリーで国を表現した例



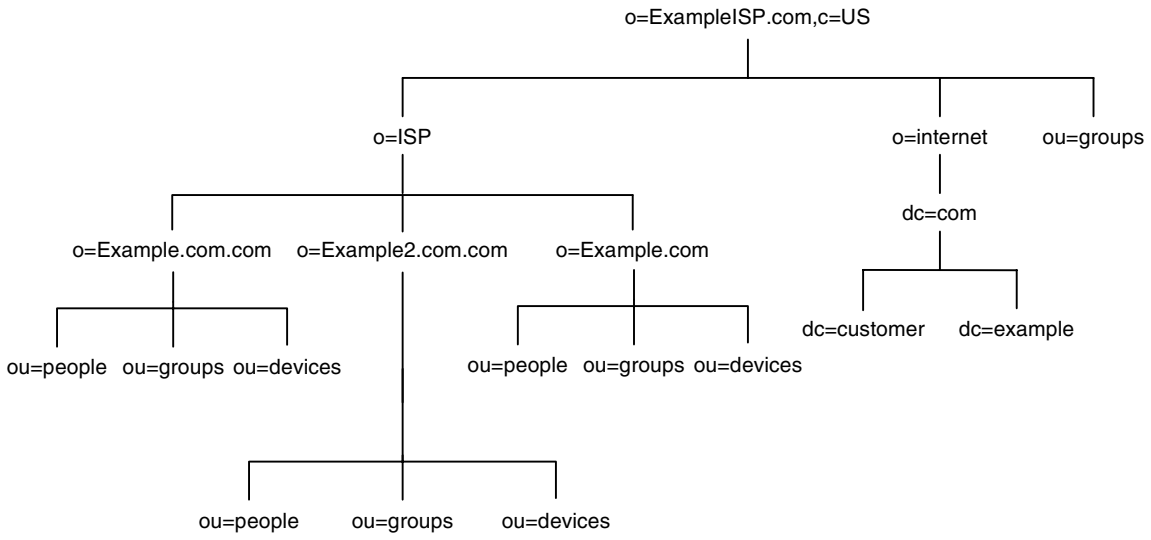
ISP 向けのディレクトリツリー

IPS (インターネットサービスプロバイダ) は、ディレクトリを通じて複数の企業をサポートする可能性があります。ISP の場合は、各顧客を個別の企業とみなし、ディレクトリツリーを設計します。セキュリティ上の理由から、それぞれが一意のサフィックスを持つ個別のディレクトリツリーをアカウントごとに提供し、各ディレクトリツリーに適した個別のセキュリティポリシーを設定します。

各顧客に別々のデータベースを割り当て、別々のサーバーにこれらのデータベースを格納することもできます。各ディレクトリツリーを専用のデータベース内に置けば、ほかの顧客に影響を与えずに各ディレクトリツリーにあるデータをバックアップしたり復元したりできます。

また、データを分散して配置することにより、ディスクの競合に起因するパフォーマンス上の問題を減らし、ディスク障害によって影響を受けるアカウントの数を減らすこともできます。次の 90 ページの図 4-15 は、ISP 企業である Example.com のディレクトリツリーを示しています。

図 4-15　Example.com という ISP のディレクトリツリー



その他のディレクトリツリー関連資料

ディレクトリツリーの設計については、次のリンクを参照してください。

- RFC 2247:Using Domains in LDAP/X.500 Distinguished Names (LDAP/X.500 識別名でドメインを使用する方法)
  http://www.ietf.org/rfc/rfc2247.txt
- RFC 2253:LDAPv3, UTF-8 String Representation of Distinguished Names (LDAPv3、識別名を UTF-8 文字列で表す方法)
  http://www.ietf.org/rfc/rfc2253.txt

第 5 章

ディレクトリトポロジの設計

第 4 章「ディレクトリツリーの設計」では、ディレクトリでのエントリの格納方法について説明しました。Directory Server は膨大な数のエントリを格納できるので、エントリを複数のサーバーに分散して配置しなければならない場合があります。ディレクトリのトポロジは、ディレクトリツリーをどのように複数の物理的な Directory Server に分割するか、およびこれらのサーバーをどのように相互にリンクするかを示します。

この章では、ディレクトリのトポロジの計画について説明します。この章で説明する内容は次のとおりです。

- トポロジの概要
- データの分散
- リフェラルと連鎖について

トポロジの概要

第 4 章「ディレクトリツリーの設計」で設計したディレクトリツリーが複数の物理的な Directory Server に分散されている、分散型のディレクトリを構成するように、Sun ONE Directory Server の配置を設計できます。ディレクトリを複数のサーバーに分割する手法は、次のような機能強化を実現するのに役立ちます。

- ディレクトリを使用するアプリケーションに最大限のパフォーマンスを提供する
- ディレクトリの可用性を高める
- ディレクトリの管理を向上させる

データベースは、レプリケーション、バックアップ、データの復元などの作業に使用する基本単位です。

ディレクトリを複数のサーバーに分割すると、各サーバーはディレクトリツリーの一部分だけを処理します。分散されたディレクトリの動作は、DNS ネームスペースの各部分を特定の DNS サーバーに割り当てるドメイン名サービス (DNS) に似ています。DNS と同様に、ディレクトリのネームスペースを複数のサーバーに分散し、クライアント側からは単一のディレクトリツリーを持つ 1 つのディレクトリとして管理させることができます。

Sun ONE Directory Server には、異なるデータベースに格納されているディレクトリデータをリンクするために、リフェラルと連鎖のメカニズムも用意されています。

この章では、データベース、リフェラル、連鎖について、およびデータベースのパフォーマンスを向上するためにインデックスを設計する方法について説明します。

データの分散

データを分散すると、社内の各サーバー上にディレクトリのエントリを物理的に格納することなく、ディレクトリを複数のサーバーにわたって拡張できます。これらのサーバーは、パフォーマンス要件を満たすために複数のマシンに格納することができます。そのため、分散型ディレクトリは、1 つのサーバーで保持できる数よりはるかに多くのエントリを保持できます。

また、分散されていることがユーザーからは見えないようにディレクトリを設定することもできます。ユーザーとアプリケーションからは、ディレクトリへのクエリーに応答する単一のディレクトリがあるようにしか見えません。

次に、データ分散のしくみについて詳しく説明します。

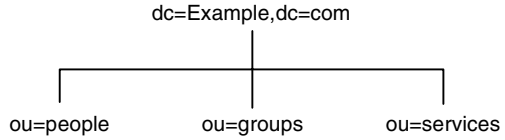
- 複数のデータベースの使用
- サフィックスについて

複数のデータベースの使用

Sun ONE Directory Server はデータを LDBM データベースに格納します。LDBM データベースはディスクベースの高パフォーマンスデータベースです。各データベースは、割り当てられたすべてのデータを含む大きなファイルのセットから構成されます。

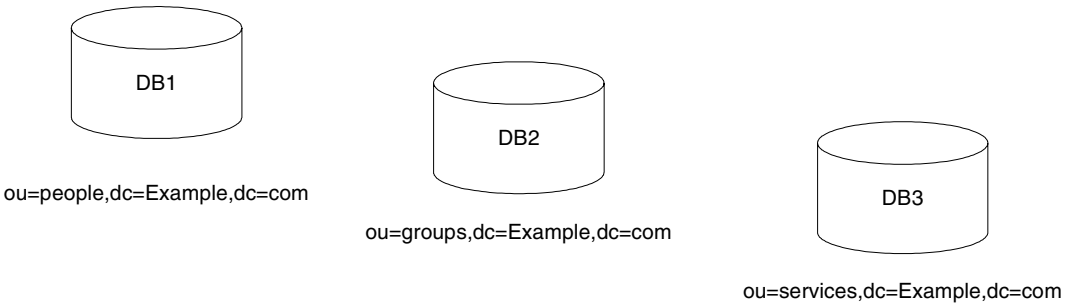
ディレクトリツリーの異なる部分を別々のデータベースに格納できます。たとえば、図 5-1 のようなディレクトリツリーがあるとします。

図 5-1　3 つのサブサフィックスを持つディレクトリツリー



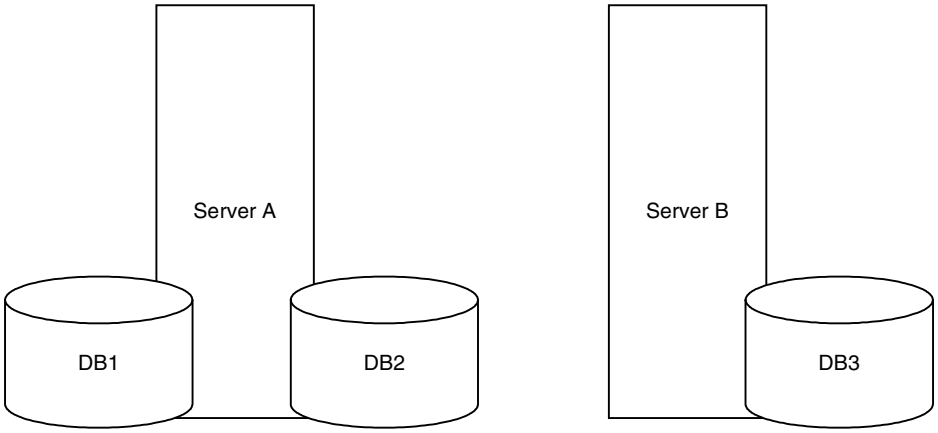
ツリーにある 3 つのサブサフィックスのデータを、図 5-2 のように 3 つの異なるデータベースに格納できます。

図 5-2　3 つの異なるデータベースに格納された 3 つのサブサフィックス



ディレクトリツリーを多数のデータベースに分割すると、それらのデータベースを複数のサーバーに分散できます。通常は、パフォーマンスを向上するために複数のマシンにインストールされた複数のデータベースに分散します。たとえば、ディレクトリツリーの 3 つのサブサフィックスを含めるために作成した 3 つのデータベースを、93 ページの図 5-1 のように 2 つのサーバーに格納できます。

図 5-3　2 つのサーバー A、B に格納された Example.com の 3 つのデータベース



サーバー A にはデータベース DB1 と DB2、サーバー B には DB3 が含まれます。データベースを複数のサーバーに分散すると、各サーバーが処理しなければならない作業量を削減できます。このようにして、1 つのサーバーで保持できる数よりもはるかに多いエントリに対応するようにディレクトリを拡張できます。

さらに、Sun ONE Directory Server ではデータベースをダイナミックに追加できるので、ディレクトリにデータベースが必要になったときに、ディレクトリ全体をオフラインにしなくても新しいデータベースを追加できます。

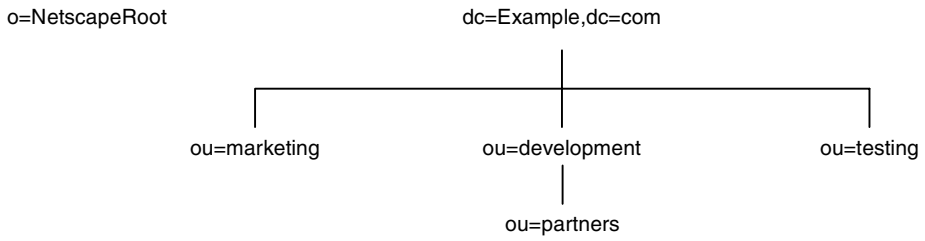
サフィックスについて

各データベースにはディレクトリサーバーのサフィックスにあるデータが格納されます。サフィックスとサブサフィックスの両方を作成して、ディレクトリツリーの内容を編成できます。サフィックスはツリーのルートにあるエントリです。このサフィックスは、ディレクトリツリーのルートか、またはディレクトリサーバー用に設計したもっと大きなツリーの一部を形成します。

サブサフィックスは、サフィックスの下にある分岐です。サフィックスとサブサフィックスのデータはデータベースに格納されます。

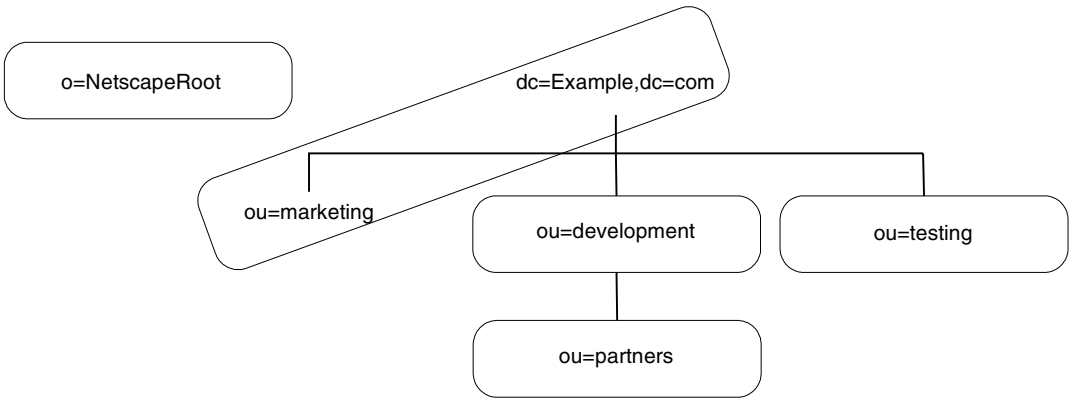
たとえば、ディレクトリデータの分散を表すためにサブサフィックスを作成するとします。Example.com 社のディレクトリツリーは、95 ページの図 5-4 のようになります。

図 5-4　Example.com 社のディレクトリツリー



Example.com 社が、ディレクトリツリーを 95 ページの図 5-5 のように 5 つの異なるデータベースに分割するとします。

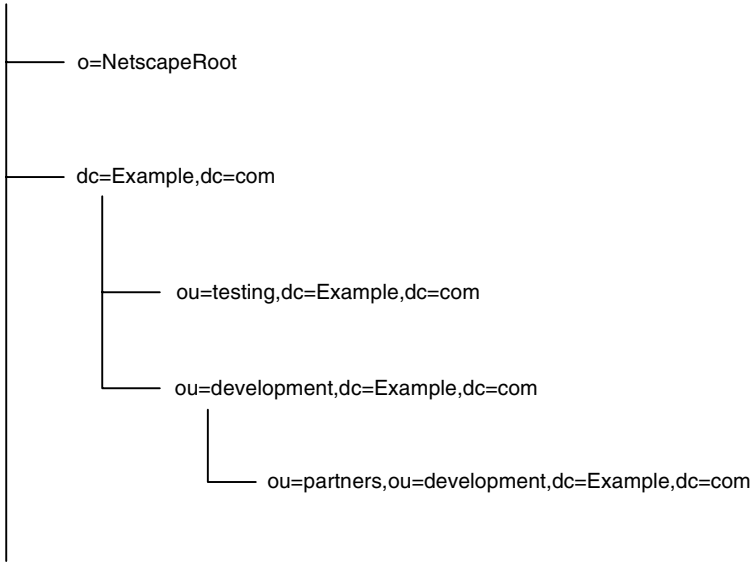
図 5-5　5 つのデータベースに分割された Example.com 社のディレクトリツリー



`o=NetscapeRoot` と `dc=Example,dc=com` はどちらもサフィックスです。それ以外の `ou=testing,dc=Example,dc=com`、`ou=development,dc=Example,dc=com`、`ou=partners,ou=development,dc=Example,dc=com` の各サフィックスは、どれも `dc=Example,dc=com` サフィックスのサブサフィックスです。サフィックス `dc=Example,dc=com` には、元のディレクトリツリーの分岐である `ou=marketing` のデータが含まれます。

この分割により、サフィックスとサブサフィックスは 96 ページの図 5-6 のようにエントリを格納します。

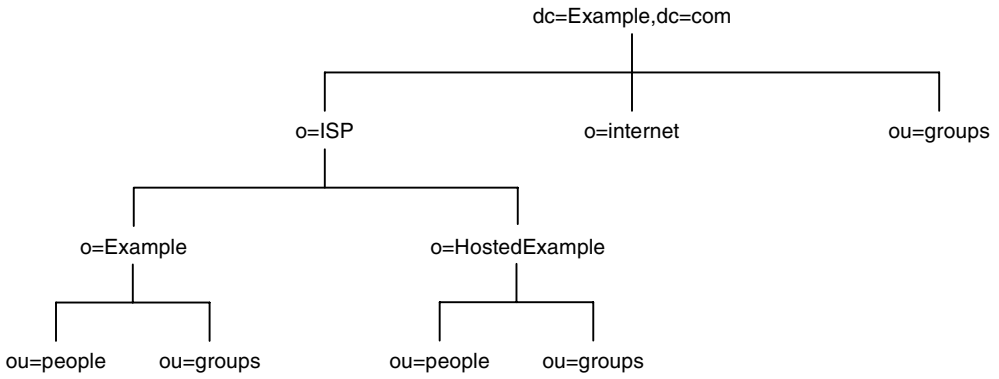
図 5-6　Example.com 社のサフィックスと関連エントリ



ディレクトリには複数のサフィックスを持たせることができます。たとえば、Example.com という ISP が複数の Web サイトをホスティングし、1 つが独自の Web サイトである Example.com、もう 1 つが HostedExample2.com という別の Web サイトであると仮定します。ISP は、すべてを格納する 1 つのサフィックスを作成するか、同社がホスティングする ISP 部分と Example.com 社の社内データのために 2 つの異なるサフィックスを作成するかを選択できます。

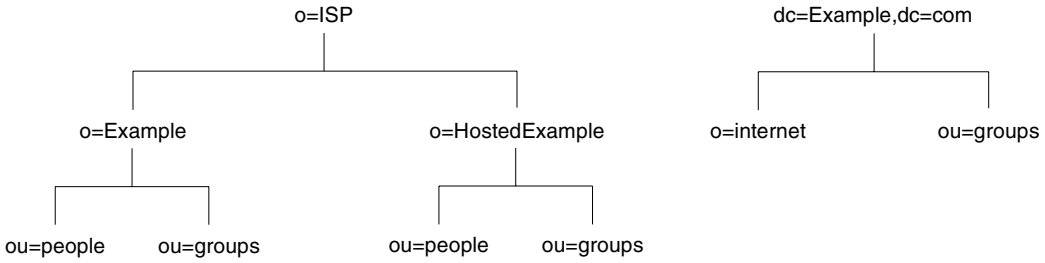
すべてのデータを 1 つのサフィックスに格納するソリューションでは、ディレクトリ情報ツリーは 97 ページの図 5-7 のようになります。

図 5-7　1 つのサフィックスを持つ ISP、Example.com 社のディレクトリツリー



ISP が dc=Example,dc=com という命名コンテキストに対応するサフィックスと、同社の o=ISP 部分に対応するサフィックスの 2 つのサフィックスを作成する 2 番目のソリューションでは、ディレクトリ情報ツリーは 97 ページの図 5-8 のようになります。

図 5-8　Example.com という ISP のディレクトリツリー



dc=Example,dc=com エントリと o=ISP エントリはサフィックスを表します。ホスティングされる各 ISP のエントリ (o=Example と o=HostedExample) は o=ISP サフィックスのサブサフィックスで、ou=people と ou=groups はホスティングされる各 ISP のサブサフィックスとして分岐します。

リフェラルと連鎖について

データを複数のデータベースに分散したあとは、分散したデータ間の関係を定義する必要があります。これは、別々のデータベースに保持されているディレクトリ情報へのポインタを使用して行います。Sun ONE Directory Server には、分散されたデータを単一のディレクトリツリーとしてリンクするために、リフェラルと連鎖というメカニズムが用意されています。

- リフェラル

  サーバーは、要求を完了するために別のサーバーに接続する必要があることを示す情報をクライアントアプリケーションに返します。

- 連鎖

  サーバーはクライアントアプリケーションの代わりに別のサーバーに接続し、操作が終了すると、結合した結果をクライアントアプリケーションに返します。

次に、2 つのメカニズムをより詳細に比較します。

リフェラルの使い方

リフェラルはサーバーが返す情報であり、操作要求を完了するために接続する必要があるサーバーをクライアントアプリケーションに指示します。Directory Server は、次の 3 種類のリフェラルをサポートしています。

- デフォルトリフェラル

  クライアントアプリケーションが提示した DN に対応するサフィックスがサーバーにない場合、ディレクトリはデフォルトリフェラルを返します。デフォルトリフェラルは、`nsslapd-referral` 属性を使用してサーバーレベルで設定します。

- サフィックスリフェラル

  サフィックスリフェラルはデフォルトリフェラルとは異なり、データベースレベルで格納されるリフェラルです。サフィックスリフェラルは、サフィックスの設定情報として設定されます。サフィックスリフェラルは、サフィックスごと、つまり、一般的にはデータベースごとに設定できます。

- スマートリフェラル

  スマートリフェラルは、ディレクトリ自体のエントリに格納されています。このリフェラルは、そのスマートリフェラルが格納されているエントリの DN と一致する DN を持つサブツリーに関する情報を保有している Directory Server をポイントします。

すべてのリフェラルは、LDAP URL (Uniform Resource Locator) の形式で返されます。次に、LDAP リフェラルの構造と、Directory Server がサポートする 3 つのタイプのリフェラルについて説明します。

LDAP リフェラルの構造

LDAP リフェラルには LDAP URL 形式の情報が含まれます。LDAP URL には、次の情報が含まれます。

- 接続先サーバーのホスト名
- サーバーのポート番号
- 検索操作の場合は ベース DN、追加、削除、および変更操作の場合はターゲット DN

たとえば、クライアントアプリケーションが `Jensen` という姓を持つエントリを `dc=Example,dc=com` 内で検索するとします。リフェラルは、次の LDAP URL をクライアントアプリケーションに返します。

```
ldap://europe.Example.com:389/ou=people,l=europe,dc=Example,dc=com
```

リフェラルは、LDAP ポート `389` のホスト `europe.Example.com` に接続し、`ou=people,l=europe,dc=Example,dc=com` にルート設定された検索要求を送信するように、クライアントアプリケーションに指示します。

リフェラルがどのように処理されるかは、使用している LDAP クライアントアプリケーションによって決まります。一部のクライアントアプリケーションは、参照先サーバーに対して自動的に操作を再実行します。別のクライアントアプリケーションは、ユーザーにリフェラル情報を返します。コマンド行ユーティリティなど、Sun ONE Directory Server が提供するほとんどの LDAP クライアントアプリケーションは、自動的にリフェラルを実行します。参照先サーバーへのアクセスには、最初のサーバー要求で指定したバインド証明情報が使用されます。

ほとんどのクライアントアプリケーションは、リフェラルの制限数、あるいはホップ数だけリフェラルを実行します。実行するリフェラル数を制限すると、クライアントアプリケーションがディレクトリ検索要求を完了しようとして費やす時間を短縮でき、また循環リフェラルパターンが原因で発生する処理停止を防ぐのにも役に立ちます。

デフォルトリフェラル

ディレクトリサーバーは、要求されたディレクトリオブジェクトの DN とローカルサーバーでサポートされているディレクトリサフィックスを比較して、デフォルトリフェラルを返すかどうかを決めます。DN とサポートされているサフィックスが一致しない場合はデフォルトリフェラルを返します。

たとえば、ディレクトリクライアントが次のディレクトリエントリを要求したとします。

```
uid=bjensen,ou=people,dc=Example,dc=com
```

ところが、サーバーは dc=europe,dc=Example,dc=com サフィックスの下に格納されているエントリしか管理していません。このような場合、ディレクトリは、dc=Example,dc=com サフィックスに格納されているエントリを取得するために接続する必要があるサーバーを示すリフェラルをクライアントに返します。デフォルトリフェラルを受け取ったクライアントは適切なサーバーに接続して、元の要求を再送信します。

デフォルトリフェラルは、ディレクトリの分散に関する情報をより多く保持しているディレクトリサーバーをポイントするように設定します。サーバーのデフォルトリフェラルは、dse.ldif 設定ファイルに格納されている nsslapd-referral 属性で設定します。

デフォルトリフェラルの設定については、『Sun ONE Directory Server 管理ガイド』の「Setting Default Referrals」を参照してください。

サフィックスリフェラル

サフィックスリフェラルを使用することで、リフェラルをサフィックスレベルで設定できます。この機能によりリフェラルの詳細度が向上し、1 つの設定属性を指定して、更新が実際に行われる場所を制御できるようになりました。ディレクトリインストール時の各データベースのデフォルトリフェラルは、dse.ldif 設定ファイル内のマッピングツリーエントリに含まれる nsslapd-referral 属性によっても設定されます。

米国内に 2 つの主要サイトがあり、1 つはニューヨーク、もう 1 つはロサンジェルスに基盤を置いていると仮定します。クライアントアプリケーションは、ニューヨークのサイトを次のように参照します。

```
uid=bjensen,ou=people,dc=US,dc=Example,dc=com
```

サフィックスリフェラルを dc=NewYork,dc=US,dc=Example,dc=com と設定することで、dc=NewYork サブツリーを含むサフィックスによって要求が処理されるようになります。

サフィックスは、4 つの異なる状態で稼動するように設定できます。このうち、2 つはサフィックスリフェラルに関するものです。サフィックスリフェラルが必要ない場合は、すべての操作をバックエンドデータベースが処理する nsslapd-referral: backend 状態、または処理を担当するデータベースは存在せず、クライアントアプリケーションからの要求に応じてエラーを返す nsslapd-referral: disabled 状態を選択できます。

しかし、サフィックスリフェラルを設定する場合は、2 つの別の状態を選択できます。nsslapd-referral: referral 状態を選択すると、このサフィックスに対するすべての要求でリフェラルが返されます。nsslapd-referral: referral on update 状態を選択した場合は、すべての操作はデータベースによって行われ、更新要求に対してはリフェラルが返されます。2 番目の referral on update 状態は、レプリケー

ションが設定された場合に、コンシューマが更新要求を処理しないように内部的に使用されます。ただし、ロードバランスやパフォーマンス上の理由から特定のデータベースに対する読み取り操作のアクセスを制限する必要がある場合は、サフィックスリフェラルのこの設定で解決できることがあります。

サフィックスリフェラルの設定については、『Sun ONE Directory Server 管理ガイド』の「Creating Suffix Referrals」を参照してください。

スマートリフェラル

Directory Server では、スマートリフェラルを使用するようにディレクトリを設定することもできます。スマートリフェラルを使用すると、ディレクトリのエントリまたはディレクトリツリーを特定の LDAP URL に関連付けることができます。ディレクトリのエントリを特定の LDAP URL に関連付けると、次のいずれかに要求を転送できます。

- 異なるサーバー上の同じネームスペース
- ローカルサーバー上の異なるネームスペース
- 同じサーバー上の異なるネームスペース

デフォルトリフェラルとは異なり、スマートリフェラルはディレクトリ自体に格納されます。スマートリフェラルの設定と管理については、『Sun ONE Directory Server 管理ガイド』の「Creating Smart Referrals」を参照してください。

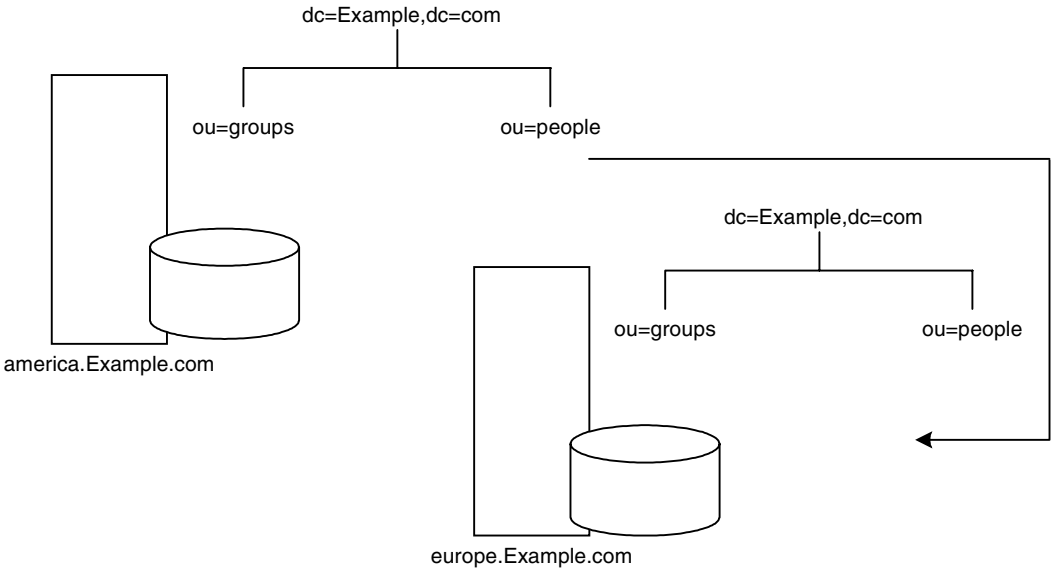
たとえば、Example.com 社の米国支社のディレクトリに、`ou=people,dc=Example,dc=com` というディレクトリ分岐点があるとします。

`ou=people` エントリ自体にスマートリフェラルを指定することで、この分岐への要求をすべて Example.com 社のヨーロッパ支社の `ou=people` 分岐に転送できます。このスマートリフェラルは、次のようになります。

```
ldap://europe.Example.com:389/ou=people,dc=Example,dc=com
```

米国支店のディレクトリにある people 分岐への要求はすべて、ヨーロッパのディレクトリに転送されます。102 ページの図 5-9 は、このスマートリフェラルが、米国のディレクトリに対する要求をヨーロッパのディレクトリに転送する様子を示しています。

図 5-9　米国のディレクトリからヨーロッパのディレクトリへのスマートリフェラル

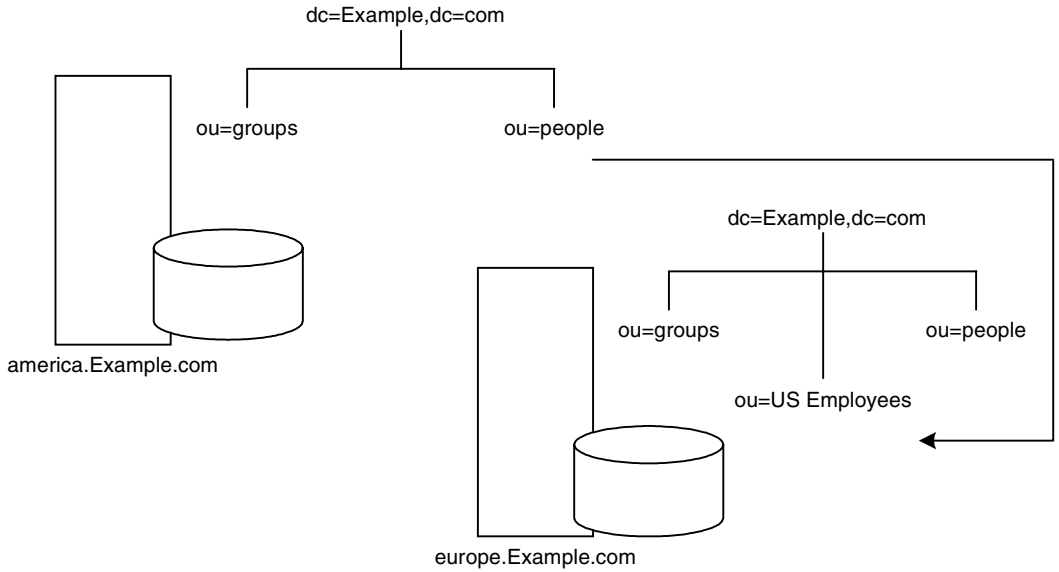


同じメカニズムを使用して、異なるネームスペースを使っている別のサーバーにクエリーを転送できます。たとえば、Example.com 社のイタリア支社の従業員がヨーロッパのディレクトリに米国の Example.com 社員の電話番号を要求したとします。ディレクトリは次のリフェラルを返します。

```
ldap://europe.Example.com:389/ou=US employees,dc=Example,dc=com
```

103 ページの図 5-10 は、ヨーロッパ支社のイタリア人従業員が米国支社の米国人従業員の電話番号を要求する場合のスマートリフェラルを示しています。

図 5-10　電話番号要求のスマートリフェラル

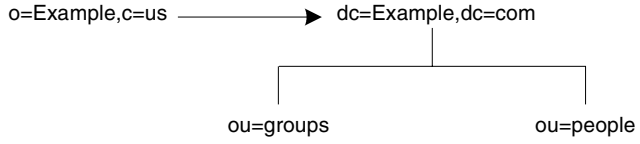


同じサーバー上に複数のサフィックスを設定している場合は、あるネームスペースから同じマシン上の別のネームスペースにクエリーを転送できます。ローカルマシン上の `o=Example,c=us` に対するすべてのクエリーを `dc=Example,dc=com` に転送する場合は、`o=Example,c=us` エントリに次のスマートリフェラルを設定します。

```
ldap:///dc=Example,dc=com
```

図 5-11 を参照してください。

図 5-11　スマートリフェラルのトラフィック



1 つのネームスペースからのクエリーを同一マシン上の別のネームスペースに転送するため、通常は URL の 2 番目のスラッシュの後に指定される `host:port` の情報ペアを指定する必要はありません。URL にこの情報ペアが指定されていないため、同じ Directory Server をポイントする URL には 3 つのスラッシュが含まれます。

注	リフェラルを最大限に活用できるように、リフェラルが設定されている場所の下を検索のベースにしないでください。

LDAP URL の詳細については、『Sun ONE Directory Server Reference Manual』の「LDAP URL」を参照してください。Sun ONE Directory Server のエントリにスマート URL を含める方法については、『Sun ONE Directory Server 管理ガイド』の「Setting Referrals」を参照してください。

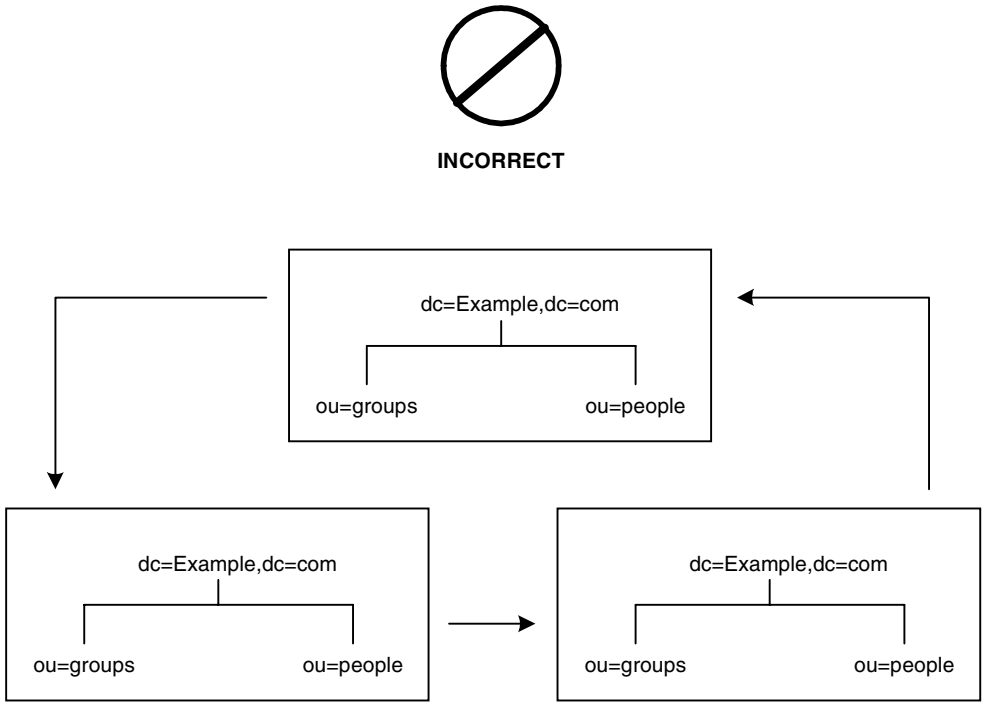
スマートリフェラルを設計する際のヒント

スマートリフェラルを使用する前に、次の点を考慮してください。

- 設計を単純にする

  複雑に交錯したリフェラルを使用してディレクトリを運用すると、管理が難しくなります。また、スマートリフェラルを使いすぎると、循環リフェラルパターンを引き起こしてしまう場合もあります。例えば、リフェラルがある LDAP URL をポイントし、その LDAP URL が別の LDAP URL をポイントするといったことが、連鎖のどこかで最後に元のサーバーに戻ってしまうまで続くという状況です。図 5-12 は、循環リフェラルのパターンを示しています。

図 5-12　スマートリフェラルの使いすぎによる循環リフェラルのパターン



- 主要な分岐点で転送する

  ディレクトリツリーのサフィックスレベルで転送を処理するように、リフェラルの使用を制限します。スマートリフェラルを使用すると、最下位のエントリ (分岐ではありません) への検索要求を別のサーバーや DN に転送できます。そのため、スマートリフェラルをエイリアスメカニズムとして使用することがよくあり、その結果、ディレクトリ構造の安全保護を複雑で困難なものにしています。リフェラルの使用をディレクトリツリーのサフィックスまたは主要な分岐点に限定することで、管理するリフェラル数が減少し、ディレクトリの管理に伴う負荷を低減できます。

- セキュリティへの影響を考慮する

  アクセス制御はリフェラルの境界を越えると機能しません。要求を発信したサーバーがエントリへのアクセスを許可している場合でも、スマートリフェラルがクライアントの要求を別のサーバーに転送したときは、クライアントアプリケーションがアクセスを許可されないこともあります。

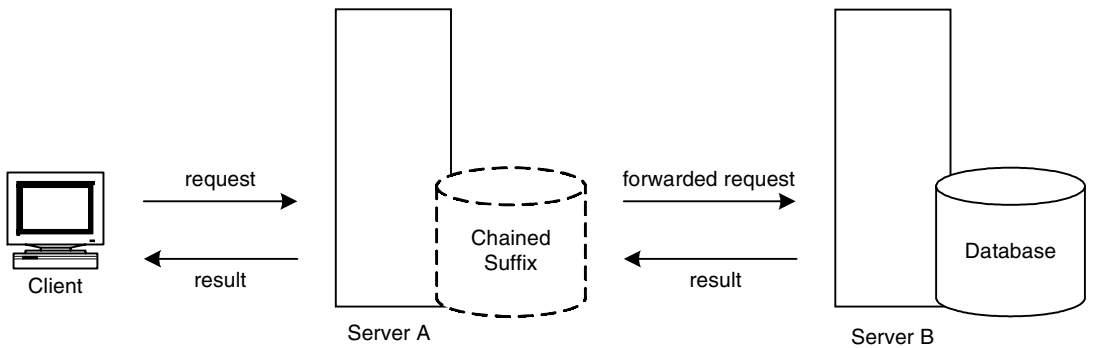
  また、クライアントが認証されるためには、クライアントは転送先のサーバー上でクライアント証明情報を使用できなければなりません。

連鎖の使用方法

連鎖は要求を別のサーバーに中継する手法の 1 つです。この手法は連鎖サフィックスを介して実行されます。「データの分散」で説明したように、連鎖サフィックスにはデータは含まれていません。連鎖サフィックスは、クライアントアプリケーションの要求をデータがあるリモートサーバーに転送します。

連鎖時に、サーバーは格納されていないデータに対する要求をクライアントアプリケーションから受け取ります。サーバーは連鎖サフィックスを使用して、クライアントアプリケーションの代わりに他のサーバーにアクセスし、結果をクライアントアプリケーションに返します。106 ページの図 5-13 は、この処理を示しています。

図 5-13　連鎖による処理



各連鎖サフィックスは、データを保持しているリモートサーバーに関連付けられています。障害が発生したときに連鎖サフィックスが使用する、データのレプリカが入った代替リモートサーバーを設定することもできます。連鎖サフィックスの設定については、『Sun ONE Directory Server 管理ガイド』の「Creating Chained Suffixes」を参照してください。

連鎖サフィックスには次の機能があります。

- リモートデータへの透過的なアクセス

  連鎖サフィックスがクライアントからの要求を処理するので、データの分散はクライアントからは見えません。

- 動的な管理

システム全体をクライアントアプリケーションが使用できる状態にしたままで、ディレクトリを部分的にシステムに追加したり、システムから削除したりできます。連鎖サフィックスは、エントリがディレクトリに再分散されるまで、リフェラルを一時的にアプリケーションに返すことができます。サフィックスを使用してこの機能を実装することもできます。サフィックスはクライアントアプリケーションをデータベースに転送するのではなく、リフェラルを返します。

- アクセス制御

  連鎖サフィックスは、クライアントアプリケーションに代わって該当する認証情報をリモートサーバーに提示します。アクセス制御を評価する必要がない場合は、リモートサーバーに対するユーザー代行機能を無効にすることができます。アクセス制御と連鎖サフィックスの関係については、『Sun ONE Directory Server 管理ガイド』の「Access Control Through Chained Suffixes」を参照してください。

リフェラルと連鎖の選択

分割されたディレクトリ部分をリンクする場合、どちらの手法にも利点と欠点があります。ディレクトリの要件に応じて、どちらか一方、あるいは両方を組み合わせて使用します。

リフェラルと連鎖の大きな違いは、分散された情報の検索方法を認識するための情報が置かれている場所です。連鎖システムでは、この情報がサーバー内に実装されます。リフェラルを使用するシステムでは、この情報はクライアントアプリケーション内に実装されます。

連鎖では、クライアント側の処理は簡単になりますが、その分サーバー側の処理が複雑になります。連鎖対象のサーバーは、リモートサーバーと協調して結果をディレクトリクライアントに送信する必要があります。

リフェラルでは、クライアントがリフェラルを検索して検索結果を照合する必要があります。ただし、連鎖よりもリフェラルを使用した方がクライアントアプリケーションを柔軟に作成でき、開発者は分散型ディレクトリの進行状況をより適切な形でユーザーにフィードバックできます。

次に、リフェラルと連鎖の違いについて詳しく説明します。

使用方法の違い

リフェラルをサポートしないクライアントアプリケーションもあります。連鎖を使用すると、クライアントアプリケーションは 1 つのサーバーと通信するだけで、多数のサーバーに格納されているデータにアクセスすることができます。企業のネットワークがプロキシを使用している場合は、リフェラルが機能しないことがあります。たとえば、クライアントアプリケーションがファイアウォール内の 1 つのサーバーとだけ通信する権限を持っているとします。クライアントアプリケーションが別のサーバーに転送された場合、クライアントはそのサーバーに正常に接続できません。

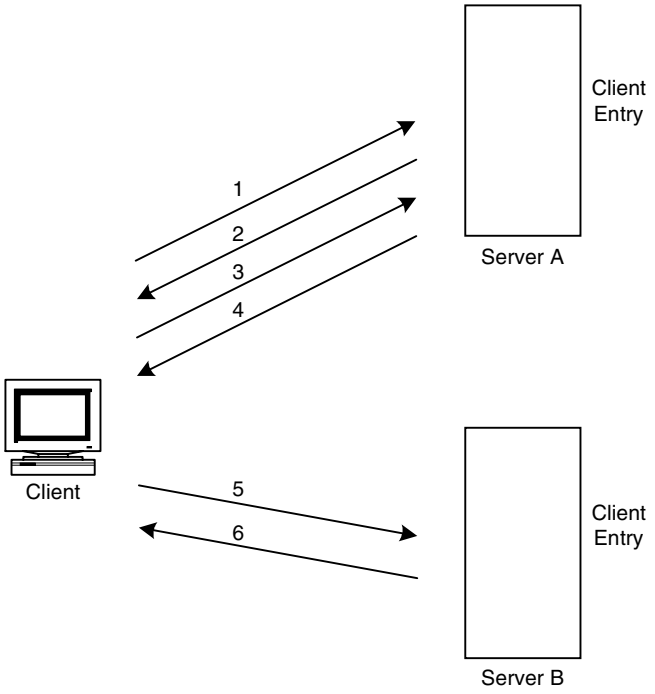
また、リフェラルでは、クライアントを認証する必要があります。つまり、クライアントの転送先のサーバーにクライアントの証明情報が格納されている必要があります。連鎖では、クライアント認証は 1 回だけで済みます。要求の連鎖先のサーバーでクライアントを再び認証する必要はありません。

アクセス制御の評価

連鎖は、リフェラルとは異なる方法でアクセス制御を評価します。リフェラルでは、バインド DN エントリがすべての転送先サーバー上に存在しなければなりません。連鎖では、クライアントエントリがすべての転送先サーバー上に存在している必要はありません。

たとえば、クライアントが検索要求をサーバー A に送信する場合を考えてみます。108 ページの図 5-14 は、リフェラルを使用した場合の動作を示しています。

図 5-14 リフェラルによってサーバー B に転送される、クライアントアプリケーションからサーバー A に対する検索要求



上の図では、クライアントアプリケーションは次の手順を実行します。

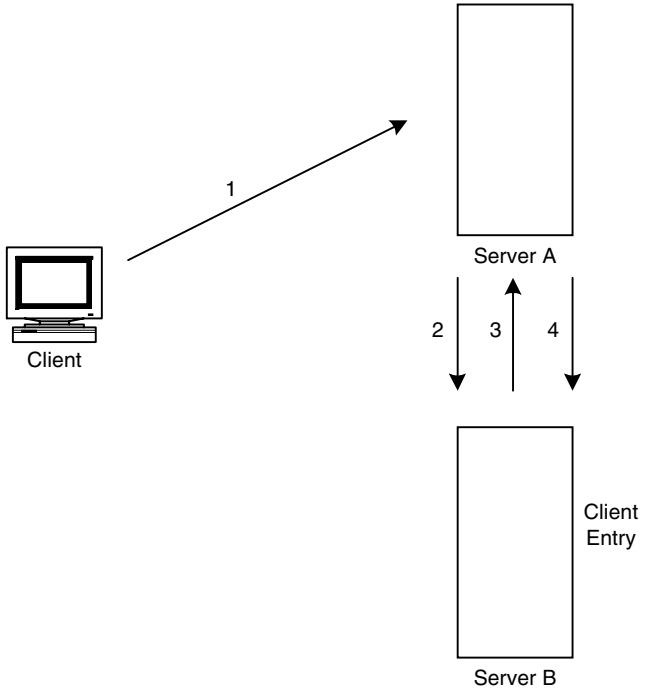
1. クライアントアプリケーションは、まずサーバー A にバインドします。

2. サーバー A は、ユーザー名とパスワードを提供するクライアントのエントリを保持しているので、バインド受け入れメッセージを返します。リフェラルが機能するためには、サーバー A にクライアントエントリが存在する必要があります。
3. クライアントアプリケーションがサーバー A に操作要求を送信します。
4. しかし、サーバー A には要求された情報が格納されていません。サーバー A は、代わりにリフェラルをクライアントアプリケーションに返し、サーバー B へのアクセスを指示します。
5. クライアントアプリケーションが、バインド要求をサーバー B に送信します。サーバー がバインドに成功するには、サーバー B にクライアントアプリケーションのエントリが存在する必要があります。
6. バインドに成功したので、クライアントアプリケーションは再び検索操作をサーバー B に送信できます。

この方法では、サーバー A からレプリケートされたクライアントエントリのコピーがサーバー B に必要です。

連鎖では、この問題は発生しません。110 ページの図 5-15 は、連鎖を使用した場合の検索要求の動作を示しています。

図 5-15 連鎖を使用した検索要求

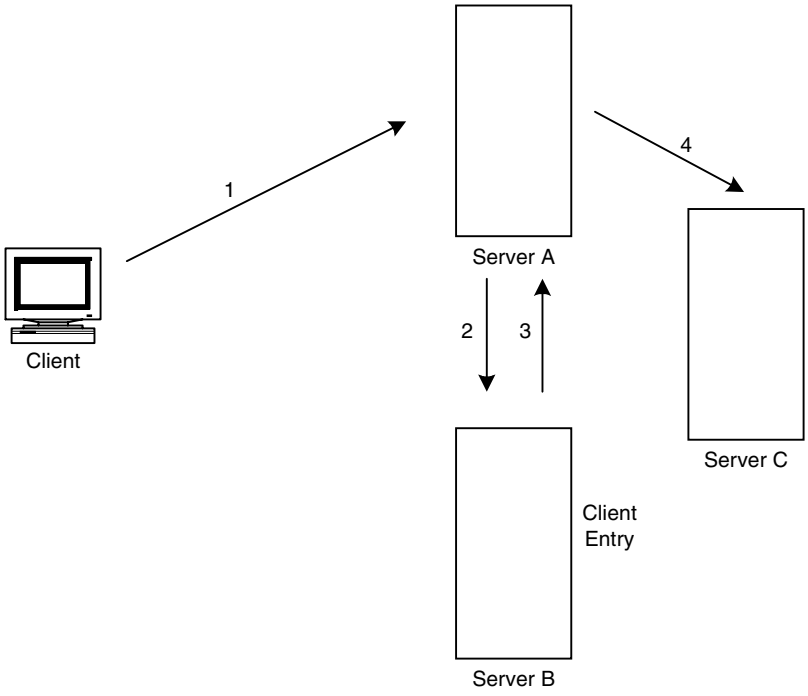


上の図では、次の手順が実行されます。

1. クライアントアプリケーションがサーバー A にバインドし、サーバー A はユーザー名とパスワードが正しいことを確認しようとします。
2. サーバー A にはクライアントアプリケーションに対応するエントリが存在しません。その代わりに、サーバー A にはクライアントの実際のエントリがあるサーバー B への連鎖サフィックスがあります。サーバー A はバインド要求をサーバー B に送信します。
3. サーバー B はサーバー A にバインド受け入れメッセージを返信します。
4. サーバー A は連鎖サフィックスを使用してクライアントアプリケーションからの要求を処理します。連鎖サフィックスはサーバー B にあるリモートデータストアに接続して検索操作を処理します。

連鎖システムでは、クライアントアプリケーションに対応するエントリが、クライアントが要求するデータと同じサーバー上にある必要はありません。111 ページの図 5-16 は、クライアントからの検索要求を処理するために、2 つの連鎖サフィックスを使用する方法を示しています。

図 5-16　クライアントからの検索要求を処理するために 2 つの連鎖サフィックスを使用する連鎖



上の図では、次の手順が実行されます。

1. クライアントアプリケーションがサーバー A にバインドし、サーバー A はユーザー名とパスワードが正しいことを確認しようとします。
2. サーバー A にはクライアントアプリケーションに対応するエントリが存在しません。その代わりに、サーバー A にはクライアントの実際のエントリがあるサーバー B への連鎖サフィックスがあります。サーバー A はバインド要求をサーバー B に送信します。
3. サーバー B はサーバー A にバインド受け入れメッセージを返信します。
4. サーバー A は別の連鎖サフィックスを使用してクライアントアプリケーションからの要求を処理します。連鎖サフィックスはサーバー C にあるリモートデータストアに接続して検索操作を処理します。

ただし、連鎖サフィックスは、次のアクセス制御をサポートしません。

- ユーザーエントリが別のサーバー上にある場合、そのユーザーエントリの内容にアクセスする必要がある制御はサポートされません。これには、グループ、フィルタ、およびロールに基づくアクセス制御が含まれます。

- クライントの IP アドレスまたは DNS ドメインに基づく制御は拒否される場合があります。これは、連鎖サフィックスがクライアントとしてリモートサーバーに接続するためです。リモートデータベースに IP に基づくアクセス制御が含まれている場合、リモートデータベースは、元のクライアントのドメインではなく、連鎖サフィックスのドメインを使用してアクセス制御を評価します。

第 6 章

レプリケーションの設計

ディレクトリの内容をレプリケートすると、ディレクトリの可用性とパフォーマンスが向上します。第 4 章と第 5 章では、ディレクトリツリーとディレクトリトポロジの設計について検討しました。この章では、データの物理的および地理的な場所に焦点を当て、特に、レプリケーションを使用していつでもどこでも必要なときにデータを使用できるようにする方法について考察します。

またこの章では、レプリケーションの使用方法についても説明し、個々のディレクトリ環境に合ったレプリケーション方法を設計するための指針も示します。この章は、次の節で構成されています。

- レプリケーションについて
- 一般的なレプリケーションの例
- レプリケーション戦略の定義
- レプリケーションとほかのディレクトリ機能との併用
- レプリケーションの監視

レプリケーションについて

レプリケーションとは、1 つの Directory Server から別の Directory Server にディレクトリのデータを自動的にコピーするメカニズムのことです。レプリケーションを行うと、それ自身のデータベースに格納しているディレクトリツリーやサブツリーをサーバー間でコピーできます。ただし、設定や監視の情報サブツリーはコピーされません。

レプリケーションを行うと、可用性の高いディレクトリサービスを提供でき、データを地理的に分散することができます。具体的には、レプリケーションには次のような利点があります。

- 耐障害性とフェイルオーバー

ディレクトリツリーを複数のサーバーにレプリケーションすると、ハードウェア、ソフトウェア、またはネットワークに障害が発生し、ディレクトリクライアントアプリケーションが特定の Directory Server にアクセスできない場合でも、ディレクトリを利用可能にできます。読み取りや書き込み操作を実行するために、クライアントを別のディレクトリに転送することができます。書き込みフェイルオーバーに対応するには、レプリケーション環境にデータのマスターコピーが複数必要であることに注意してください。

- ロードバランス

  ディレクトリツリーを複数のサーバーにレプリケートすることで、各マシン上のアクセス負荷を軽減し、サーバーの応答時間を改善できます。

- パフォーマンスの向上と応答時間の短縮

  ディレクトリのエントリをユーザーに近い場所にレプリケートすることで、ディレクトリの応答時間を大幅に改善できます。

- ローカルでのデータの管理

  レプリケーションを行うと、ローカルにデータを所有して管理でき、同時に全社レベルでそのデータをほかの Directory Server と共有できます。

ディレクトリ情報のレプリケーション戦略を決定する前に、レプリケーションの動作を理解しておく必要があります。以下に、次の事項について説明します。

- レプリケーションの概念
- データの整合性

レプリケーションの概念

レプリケーションの配備を計画するときは、常に次の基本事項を決定することから始めます。

- レプリケートする情報
- 情報のマスターコピーを保持するサーバー (複数も可)
- 情報の読み取り専用コピーを保持するサーバー (複数も可)
- コンシューマレプリカを保持するマシンがクライアントアプリケーションから変更要求を受け取ったときに、その要求を転送する転送先サーバー

これらの事項は、Directory Server がこれらの概念をどのように処理するかを理解していないと効率的に決定できません。たとえば、レプリケーションする情報を決める場合は、Directory Server が処理できる最小のレプリケーション単位を知っておく必要があります。

レプリケーションのプロセスと、Directory Server の配備で利用できる可能性について理解を深めるために、次に Directory Server で使用されるレプリケーションの概念について説明します。ここでは、全体的な決定事項について検討するための安定した枠組みを示します。

レプリカ

レプリケーションに関与するデータベースを、レプリカと呼びます。レプリカには、いくつかの種類があります。

- マスターレプリカ (読み書き可能レプリカ) : ディレクトリデータのマスターコピーを格納する読み書き可能データベース。マスターレプリカは、ディレクトリクライアントからの更新要求を処理できる
- コンシューマレプリカ : マスターレプリカに保持される情報のコピーを格納する読み取り専用データベース。コンシューマレプリカは、ディレクトリクライアントからの検索要求を処理できるが、更新要求はマスターレプリカを照会する
- ハブレプリカ : コンシューマレプリカと同じく、読み取り専用データベース。ただしハブレプリカは、1 つまたは複数のコンシューマのサプライヤとして動作する Directory Server に格納される

Directory Server は、複数のレプリカを管理するように設定できます。各レプリカには、レプリケーションにおいて異なる役割を持たせることができます。たとえば、マスターレプリカに `dc=engineering,dc=example,dc=com` サフィックス、コンシューマレプリカに `dc=sales,dc=example,dc=com` サフィックスを格納する 1 つの Directory Server を持つことができます。

レプリケーションの単位

Directory Server では、レプリケーションの最小単位はデータベースです。レプリケーションメカニズムでは、サフィックスとデータベースが 1 対 1 で対応している必要があります。つまり、カスタム分散ロジックを使用している 2 つ以上のデータベースにまたがって分散されているサフィックス (またはネームスペース) はレプリケーションできません。レプリケーション単位の概念は、コンシューマとサプライヤの両方に適用されます。つまり、1 つのデータベースだけを保持するコンシューマに 2 つのデータベースをレプリケートすることはできません。また、その反対も同様です。

レプリカ ID

マスターレプリカは一意のレプリカ識別子 (ID) を必要とし、すべてのコンシューマは同じレプリカ ID を持ちます。マスターのレプリカ ID は 1 ～ 65534 の間の任意の 16 ビット整数で、すべてのコンシューマレプリカのレプリカ ID は 65535 です。レプリカ ID は、どのレプリカで変更が発生したかを識別し、これによってレプリケーションが正しく行われます。このため、レプリカ ID はレプリケーションメカニズムで中心的な役割を果たします。

注 サーバーが複数のレプリカをホスティングする場合、1 つのレプリケートされた命名コンテキストまたはサフィックスを持つ複数のマスターの間でレプリカ ID が一意であれば、それぞれのレプリカが同じレプリカ ID を持つことができます。

サプライヤとコンシューマ

別のサーバーにレプリケートされるサーバーをサプライヤと呼びます。別のサーバーによって更新されるサーバーをコンシューマと呼びます。

サーバーは、場合によってはサプライヤとコンシューマの両方の機能を持ちます。これは、次の場合に当てはまります。

- Directory Server がマスターレプリカとコンシューマレプリカの組み合わせを管理する場合
- Directory Server がハブレプリカを含む場合。つまり、サプライヤから更新を受け取り、変更内容をコンシューマにレプリケートする場合。詳細については、132 ページの「カスケード型レプリケーション」を参照
- マルチマスターレプリケーションで、一方の Directory Server が他方の Directory Server のサプライヤおよびコンシューマとして動作する 2 つの Directory Server がマスターレプリカを保持する場合。詳細については、125 ページの「マルチマスターレプリケーション」を参照

サーバーがコンシューマとしてだけ機能する場合、このサーバーにはコンシューマレプリカだけが保持されます。このようなサーバーを専用コンシューマと呼びます。

Directory Server では、レプリケーションは常にサプライヤサーバーから開始されます。コンシューマサーバーから開始されることはありません。サプライヤがコンシューマにデータをプッシュすることから、これをサプライヤ主導レプリケーションと呼びます。

Directory Server の初期のバージョンでは、サプライヤからのデータの送信に関して、コンシューマ側に設定するコンシューマ主導のレプリケーションも許されていました。Directory Server 5.0 移行のリリースでは、コンシューマが更新の送信をサプライヤに要求する手順に切り替えられました。

マスターレプリカでは、サーバーは次のように機能する必要があります。

- ディレクトリクライアントからの更新要求 (追加、削除、変更) に応答する
- レプリカの履歴情報と更新履歴ログを保持する
- コンシューマへのレプリケーションを開始する

  マスターレプリカを保持するサーバーは、管理するマスターレプリカに対して行われる変更を常に記録する必要があります。これにより、すべての変更がコンシューマにレプリケーションされます。

ハブレプリカでは、サーバーは次のように機能する必要があります。

- 読み取り要求に応答する
- マスターレプリカを保持するサーバーに更新要求を送信する
- レプリカの履歴情報を維持する
- コンシューマへのレプリケーションを開始する

カスケード型レプリケーションについては、132 ページの「カスケード型レプリケーション」を参照してください。

コンシューマレプリカでは、サーバーは次のように機能する必要があります。

- 読み取り要求に応答する
- レプリカの履歴情報を維持する
- マスターレプリカを保持するサーバーに更新要求を送信する

  コンシューマは、エントリの追加、削除、変更の要求を受け取ると、その要求はマスターレプリカを保持するサーバーにクライアント経由で送信されます。つまり、レプリケーションの流れの中で、サーバーがサプライヤとして機能します。サプライヤは要求を実行し、変更をレプリケートします。

  セキュリティまたはパフォーマンス上の理由から、リフェラルではなく、エラーを返すようにコンシューマレプリカまたはハブレプリカを設定することもできます。詳細は、128 ページの注を参照してください。

レプリカのオンライン昇格とオンライン降格

Sun ONE Directory Server 5.2 には、レプリカのオンライン昇格とオンライン降格の機能が用意されています。オンライン昇格またはオンライン降格が完了すると、サーバーは更新の受け入れを直ちに開始または停止します。コンシューマレプリカをマスターレプリカに昇格させるには、まず、ハブレプリカに昇格させ、それからマスターレプリカに昇格させます。オンライン降格にもこれと同じ段階的なアプローチが適用されます。

オンライン昇格および降格によって柔軟性が向上するだけでなく、フェイルオーバー機能も向上します。ロードバランスとフェイルオーバーのために 2 つのハブが設定された双方向のマルチマスター環境を例に考えてみます。いずれかのマスターがオフラインになった場合は、いずれかのハブを昇格させるだけで読み取り、書き込みの最適な可用性を維持できます。マスターレプリカがオンラインに復帰したときは、ハブレプリカに降格させるだけで元の状態を回復できます。

注 ハブをコンシューマに降格する前に、そのハブが他のサーバーと同期していることを確認する必要があります。コンシューマレプリカは更新履歴ログを持たないため、このレプリカは変更を伝達できなくなります。ハブの同期を確認するには、レプリケーション監視ツール `insync` を使用します。このツールについては、155 ページの「レプリケーションの監視」を参照してください。

更新履歴ログ

サプライヤとして機能するすべてのサーバー (マスターレプリカ、ハブレプリカ) は、更新履歴ログを維持します。更新履歴ログは、マスターレプリカに対して行われた変更を記述した記録です。サプライヤとして機能するサーバーは、この変更をコンシューマ上で再現します。

エントリの変更、名称変更、追加、または削除が行われると、実行された LDAP 操作を記述する変更レコードが更新履歴ログに記録されます。

Directory Server の従来のバージョンでは、LDAP から更新履歴ログにアクセスできました。しかし現在では、更新履歴ログはサーバーが内部的に使用するだけで、専用データベースに格納され、LDAP 経由でアクセスできなくなりました。使用しているアプリケーションで更新履歴ログを読み取る必要がある場合は、レトロログのプラグインを使用して、下位互換性を保つことができます。レトロログのプラグインについては、『Sun ONE Directory Server 管理ガイド』の「Using the Retro Change Log Plug-In」を参照してください。

注 更新履歴ログからエントリが削除されると、その更新はレプリケートされなくなります。更新履歴ログのサイズは慎重に設定してください。必要となるディスク容量は変更の種類によって異なるため、予定されているトラフィックの種類を検討し、更新履歴ログ用に十分なディスク容量を確保する必要があります。

レプリケーションの識別情報

2 つのサーバーの間でレプリケーションが行われると、サプライヤとして機能するサーバーがコンシューマとして機能するサーバーにバインドしてレプリケーション更新を送信するときに、コンシューマがサプライヤを認証します。この認証処理を実行するには、サプライヤがコンシューマにバインドする際に使用するエントリがコンシューマサーバーに格納されている必要があります。このエントリをレプリケーションマネージャエントリと呼びます。レプリケーション時に Directory Server コンソールが DN またはバインド DN を参照する場合、レプリケーションマネージャエントリのバインド DN が参照されます。

レプリケーションマネージャエントリあるいはその役割を果たすために作成したエントリは、次のような条件を満たしている必要があります。

- コンシューマとして機能するレプリカが少なくとも 1 つすべてのサーバーに存在する (マルチマスター環境における専用コンシューマ、ハブ、マスターもこれに該当する)
- セキュリティ上の理由および初期化の問題から、このエントリをレプリケートされたデータの一部にしないこと

注 このエントリは、コンシューマサーバーに定義されたすべてのアクセス制御規則を迂回する、特別なユーザープロファイルとなります。ただし、この特別ユーザープロファイルはレプリケーションだけで有効です。

2 つのサーバー間でレプリケーションを設定する場合は、両方のサーバーでレプリケーションマネージャエントリを識別する必要があります。

- コンシューマとして機能するサーバーでは、レプリケーショントポロジにコンシューマレプリカ、ハブレプリカ、またはマスターレプリカ (マルチマスターレプリケーションの場合) を設定するときに、レプリケーション更新の実行が承認されたエントリとしてこれを指定する必要があります。
- サプライヤとして機能するサーバー、つまりすべてのマスターレプリカとハブレプリカでは、レプリケーションアグリーメントを設定するときに、このエントリのバインド DN をレプリケーションアグリーメントに指定する必要があります。

注 Directory Server コンソールでは、レプリケーションマネージャエントリはデフォルトで作成されます。ただし必要に応じて、Directory Server コンソールを使用してこのエントリを独自に作成することもできます。

SSL とレプリケーションを使用している場合、認証を行うには次の 2 つの方法があります。

- SSL サーバー認証を使用する場合は、認証するサーバーのレプリケーションマネージャエントリと、認証のためのパスワードが必要です。
- SSL クライアント認証を使用する場合は、認証するサーバーにエントリが含まれ、証明書が保持されている必要があります。このエントリは、レプリケーションマネージャエントリとマッピングさせる、またはマッピングさせないことができます。

レプリケーションアグリーメント

Directory Server では、レプリケーションアグリーメントを使用してレプリケーションを定義します。レプリケーションアグリーメントは、1 つのサプライヤと 1 つのコンシューマの間のレプリケーションを定義します。アグリーメントは、サプライヤ上に設定されます。レプリケーションが機能するには、レプリケーションアグリーメントが有効化されている必要があります。レプリケーションアグリーメントには、次の内容が定義されます。

- レプリケートするデータベース
- データがレプリケートされるコンシューマサーバー
- 部分レプリケーションを設定する場合は、レプリケート対象からデータを外す、または含めるための属性セットへのポインタ
- レプリケーションを実行できる時間帯
- サプライヤがコンシューマへのバインドに使用するバインド DN と証明情報 (レプリケーションマネージャエントリ、詳細は、118 ページの「レプリケーションの識別情報」を参照)
- 接続をセキュリティ保護する方法 (SSL、クライアント認証)
- 1 つの要求にグループ化できる変更の数と、コンシューマからの受信通知が必要となるまでに送信できる要求の数を設定するグループサイズとウィンドウサイズ
- レプリケーションアグリーメントの状態情報
- Solaris および Linux システムでは、レプリケーション時に使用する圧縮のレベル情報

注 Sun ONE Directory Server 5.2 では、既存のレプリケーションアグリーメントを無効にするか、有効にするかを選択できます。これは、特定のレプリケーションアグリーメントの使用が一時的に必要ないが、将来の必要性を考えて保持しておきたい場合に便利です。

コンシューマの初期化 (完全更新)

コンシューマの初期化は、サプライヤとして機能するサーバーからコンシューマとして機能するサーバーにすべてのデータを物理的にコピーする完全更新プロセスです。レプリケーションアグリーメントを作成したら、アグリーメントに指定されているコンシューマを初期化する必要があります。サプライヤが更新操作をコンシューマ上で再現 (レプリケート) できるのは、コンシューマの初期化が完了した後だけです。通常の動作環境では、それ以後のコンシューマの初期化は必要ありません。ただし、何らかの理由でサプライヤ上のデータをバックアップから復元した場合は、そのサプライヤに合わせて一部のコンシューマを初期化し直さなければならない可能性があります。たとえば、復元されたサプライヤが、トポロジ内でそのコンシューマの唯一のサ

プライヤである場合、このコンシューマを初期化し直す必要があります。コンシューマは、オンラインとオフライン (手動) の両方で初期化できます。コンシューマの初期化手順については、『Sun ONE Directory Server 管理ガイド』の「Initializing Replicas」を参照してください。Directory Server 5.2 には、高度なバイナリコピー機能も用意されています。この機能を利用することで、1 つのサーバーのバイナリバックアップファイルからマスターレプリカまたはコンシューマレプリカをクローン化し、同じディレクトリ内用を別のサーバー上で復元することができます。特定の制限が適用されるため、この機能が実用的で時間の節約となるのは、大規模なデータベースファイルのレプリカを対象とする場合だけです。バイナリバックアップ手順とこの機能の詳しい制限については、259 ページの「バイナリバックアップ (db2bak)」を参照してください。

マルチマスターレプリケーショントポロジでは、バックアップファイルまたは ldif ファイルからオンラインまたはオフラインで再初期化された読み書き可能レプリカは、デフォルトではクライアントからの更新要求を拒否します。これは、Directory Server の従来のバージョンとは対照的です。レプリカはデフォルトで永続的に読み取り専用モードになり、更新操作の要求についてはトポロジ内の他のサプライヤを参照します。この場合、管理者は次の 2 つの方法で更新の受け入れをレプリカに再設定することができます。

- Directory Server コンソールを使用するか、`ds5BeginReplicaAcceptUpdates` 属性に `start` を設定することで、読み書き可能モードを手動で有効化します。これにより、管理者はレプリケーション監視ツール `insync` を使用して、レプリカがトポロジ内のその他のサプライヤに完全に合流したことを確認できます。更新操作の受け入れを開始する前に管理者がレプリカの同期を確認できるので、この手順が推奨されます。
- レプリカ固有の `ds5referralDelayAfterInit` 属性に指定した遅延時間後に読み書き可能モードへの自動変更が行われるようにレプリカを設定します。この手順では、レプリカが完全に他のマスターレプリカと同期する前に更新操作が受け入れられ、予期せぬエラーが発生する危険があります。

これらの手順については、『Sun ONE Directory Server 管理ガイド』の「Initializing Replicas」を参照してください。レプリケーション設定属性については、『Sun ONE Directory Server Reference Manual』の「Core Server Configuration Attributes」に記載されているレプリケーション属性リストを参照してください。

差分更新

差分更新は、コンシューマの初期化または完全更新後にサプライヤからコンシューマに更新をレプリケートするプロセスです。Directory Server の従来のリリースとは異なり、Sun ONE Directory Server 5.2 では、それぞれの更新が異なるレプリカ ID に由来する場合は、複数のサプライヤが同時にコンシューマを差分更新することができます。複数のサプライヤ (ただし異なるレプリカ ID を持つ) が同時に差分更新できるようになったことで、差分更新プロセスのパフォーマンスが向上します。

データの整合性

整合性とは、レプリケートされたデータベースの内容が、任意の時点でどの程度一致しているかを示します。2 つのサーバー間でレプリケーションを設定する場合、設定の一部として更新のスケジュールが含まれます。Directory Server では、いつコンシューマを更新すべきかを判断し、レプリケーションを開始するのは、常にサプライヤとして機能するサーバーです。レプリケーションを行えるのは、コンシューマの初期化が完了してからだけです。

Directory Server には、レプリカの同期を常に維持するオプションと、特定の時間または曜日に更新をスケジュールするオプションが用意されています。レプリカの同期を常に維持する利点は、データの整合性がより確実に保証されるという点です。ただしその場合は、頻繁な更新操作によってネットワークトラフィックが増大します。このソリューションは、次の場合に最適です。

- サーバー間で信頼性が高く高速な接続を利用できる場合
- ディレクトリのサービスを受けるクライアント要求が主に検索、読み取り、および比較の操作であり、追加と変更の操作は比較的少ない場合

データの整合性が低くてもかまわない場合は、必要に応じて更新の頻度を選択して、ネットワークトラフィックへの影響を軽減することができます。このソリューションは、次の場合に最適です。

- ネットワーク接続の信頼性が低いか、あるいは断続的なネットワーク接続を使用している場合 (ダイヤルアップ接続を使用してレプリカの同期をとっている場合など)
- ディレクトリがサービスするクライアント要求が主に追加と変更の操作である場合
- 通信コストを削減する必要がある場合

マルチマスターレプリケーションの場合は、一般に、各マスターに格納されているデータ間に違いがある可能性があるため、各マスター上のレプリカは緩やかな整合性を保っている状態といえます。これは、常にレプリカの同期を維持するように選択している場合にも当てはまります。理由は次のとおりです。

- マスター間のレプリケーションの更新操作の伝達に遅延があるため
- 追加または変更の操作を実行したマスターは、2 番目のマスターがその更新操作を検証するのを待たずに「操作は正常に完了しました」というメッセージをクライアントに返すため

一般的なレプリケーションの例

レプリケーション要件に適したレプリケーション戦略を構築するために、レプリケーションの更新情報をサーバー間でやり取りする方法と、更新情報を伝達するときのサーバー間の相互動作の方法を決める必要があります。次の 5 つの基本的な例について説明します。

- シングルマスターレプリケーション
- マルチマスターレプリケーション
- カスケード型レプリケーション
- 部分レプリケーション
- 混合環境

次に、上記の例について説明し、環境に最も適した方法を決定するための指針を示します。これらの基本的な例を組み合わせて、要件に最も合ったレプリケーショントポロジを構築することもできます。

注 どのレプリケーションモデルを選択した場合でも、スキーマのレプリケーションを考慮するようにしてください。詳細は、153 ページの「スキーマのレプリケーション」を参照してください。

シングルマスターレプリケーション

レプリケーションの最も基本的な設定では、サプライヤとして機能するサーバーが 1 つまたは複数のコンシューマサーバーにマスターレプリカを直接コピーします。この設定では、マスターレプリカに対するすべてのディレクトリ変更はサプライヤに格納され、コンシューマには読み取り専用のデータコピーが維持されます。

サプライヤは、マスターレプリカに対するすべての更新を記録した更新履歴ログを管理します。サプライヤには、レプリケーションアグリーメントも格納されます。

サプライヤがレプリケーションの更新を送信するためにコンシューマにバインドしたときに、コンシューマがサプライヤを認証できるように、コンシューマにはレプリケーションマネージャエントリに対応したエントリが格納されます。

サプライヤサーバーはすべての変更をコンシューマレプリカに伝達する必要があります。124 ページの図 6-1 に、この設定の簡単な例を示しています。

図 6-1　シングルマスターレプリケーション

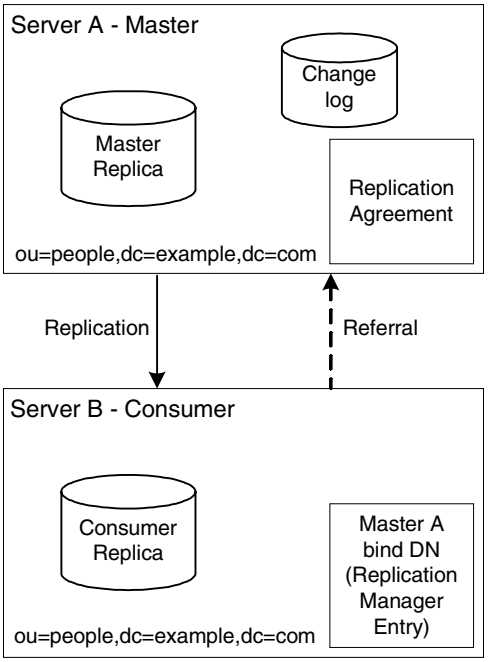


図 6-1 の例では、`ou=people,dc=example,dc=com` サフィックスが、クライアントから多数の検索要求と更新要求を受け取ります。したがって、負荷を分散するために、サーバー A 上にマスターのあるこのサフィックスは、サーバー B 上にあるコンシューマレプリカにレプリケートされます。

サーバー B は、クライアントからの検索要求を処理できますが、ディレクトリエントリの変更要求は処理できません。サーバー B は、クライアントにサーバー A に対するリフェラルを返すことによって、クライアントから受け取った変更要求を処理します。

注　レプリケーションでは、コンシューマとして機能するサーバーにサプライヤとして機能するサーバーのリフェラル情報が格納されますが、変更要求はクライアントからサプライヤに転送されません。クライアントは、コンシューマによって返されたリフェラルを実行する必要があります。

図 6-1 では、1 つのサーバーだけがコンシューマとして機能していますが、サプライヤは複数のコンシューマにレプリケートすることができます。1 つのサプライヤが管理できるコンシューマの総数は、ネットワークの速度と 1 日当たりのエントリの変更総数によって異なりますが、1 つのサプライヤサーバーで複数のコンシューマサーバーを管理できると想定して特に問題ありません。

マルチマスターレプリケーション

マルチマスターレプリケーション環境では、同じ情報を持つマスターレプリカが複数のサーバー上に存在します。ここでは、マルチマスターレプリケーションについて次の内容を説明します。

- マルチマスターレプリケーションの基本概念
- マルチマスターレプリケーションの機能
- 完全に接続された 4 方向のマルチマスタートポロジ
- 広域ネットワーク (WAN) を介したマルチマスターレプリケーション

マルチマスターレプリケーションの基本概念

同じ情報を持つマスターレプリカが複数のサーバー上に存在するマルチマスター構成では、複数の異なる場所でデータを同時に更新することができます。つまり、レプリケーショントポロジに関与するマスターレプリカの更新履歴ログを各サーバーが管理しています。各サーバー上で行われた変更は、そのトポロジの別のサーバーにレプリケートされます。つまり、各サーバーはサプライヤとコンシューマの両方の役割を果たします。マルチマスター設定には、次の利点があります。

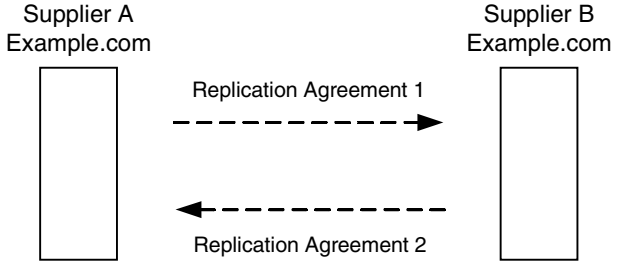
- 1 つのサプライヤにアクセスできなくなった場合でも、自動的に書き込み処理のフェイルオーバーが実行される
- 地域分散型環境のローカルサプライヤで更新処理を実行できる

両方のサーバーでほとんど同時に同じデータが変更された場合は、頂点手順が適用され、最も新しい変更が優先されます。ただし、競合する一部の変更が LDAP モデルを破損した場合は、そのエントリは競合エントリとしてマークされます。これらの「競合エントリ」を解決するには、管理者がこれらのエントリの処理を決定し、手動で更新する必要があります。

注 マルチマスターレプリケーション環境で属性の一意性が重要となる配備では、属性値一意性検査プラグインを使用して、名前の競合を減らすことを強くお勧めします。属性値一意性検査プラグインについては、123 ページの「一般的なレプリケーションの例」を参照してください。

独立した 2 つのサーバーが同じデータのマスターコピーを保持することはできますが、1 つのレプリケーションアグリーメントにおいては、1 つのサプライヤと 1 つのコンシューマだけしか存在できません。このため、同じデータの管理責任を共有する 2 つのサプライヤ間にマルチマスター環境を構築するには、2 つのレプリケーションアグリーメントを作成する必要があります。124 ページの図 6-1 は、この設定を示しています。

図 6-2　マルチマスターレプリケーションの設定 (2 つのマスター)



サプライヤ A とサプライヤ B は、それぞれが同じデータを持つマスターレプリカを保持し、マルチマスター設定のレプリケーションの流れを制御する 2 つのレプリケーションアグリーメントが存在します。

Directory Server 5.2 は、マルチマスターレプリケーショントポロジで最大 4 つのマスターをサポートします。コンシューマとハブの数は、理論的には無制限です。ただし、1 つのサプライヤがレプリケートできるコンシューマの数は、サプライヤサーバーの性能によって異なります。

マルチマスターレプリケーションの機能

レプリケーション要件やパフォーマンス要件に配備を簡単に適応させられるように、Sun ONE Directory Server 5.2 はより効率的で柔軟なプロトコルを提供します。Sun ONE Directory Server 5.2 では、次の処理が可能です。

- レプリカ ID に基づいて更新をレプリケートします。レプリカ ID ベースの更新により、複数のサプライヤがコンシューマを同時に更新できるようになるため、パフォーマンスが向上します (それぞれの更新が異なるレプリカ ID に由来する必要があります)。
- 指定したコンシューマのレプリケーションアグリーメントを有効化または無効化することができます。これにより、配備でのレプリケーション設定の柔軟性が向上します。トポロジを後日変更することを前提に特定のトポロジを設定し、後から簡単に変更することができます。

完全に接続された 4 方向のマルチマスタートポロジ

127 ページの図 6-3 は、完全に接続された 4 方向のマスタートポロジを示しています。4 方向のマスターフェイルオーバー設定により、完全に接続されたこのトポロジは高い可用性を提供し、データの整合性を保証します。これは、読み書き対応フェイルオーバー機能の面では最も安全ですが、パフォーマンスの面ではこのフェイルオーバー機能によってオーバーヘッドが生じることに注意が必要です。完全に接続された

マルチマスター構成の配備が必要かどうかは、高可用性の要件によって異なります。高可用性が比較的重要でない場合、またはパフォーマンス上の理由からレプリケーショントラフィックを削減する場合は、読み書き可能フェイルオーバーの面でより「軽量」の配備が適しています。

図 6-3　完全に接続された 4 方向のマルチマスターレプリケーション構成

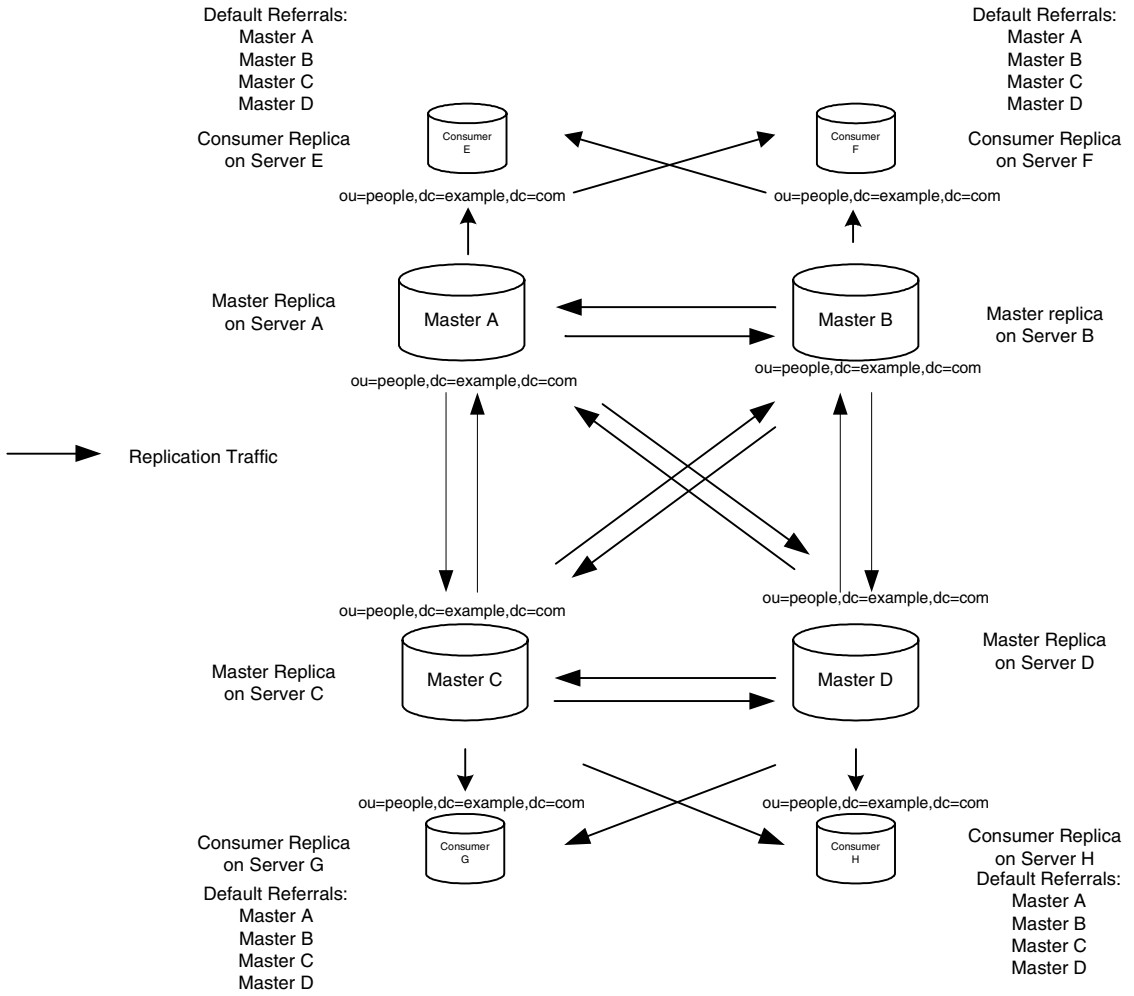


図 6-3 では、4 つのマスターで `ou=people,dc=example,dc=com` サフィックスが維持され、変更要求に常に対応できるようにしています。各マスターは、自身の更新履歴ログを管理します。いずれかのマスターがクライアントからの変更要求を処理すると、そのマスターは操作を更新履歴ログに記録します。次に、その他のサーバーにレプリ

ケーション更新を送信し、そこからその他のコンシューマに伝達されます。このためマスターは、マスター間のレプリケーションアグリーメントだけでなく、コンシューマとのレプリケーションアグリーメントも保持する必要があります。また各マスターには、レプリケーション更新を送信するために他のマスターにバインドできるように、他のマスターの認証に使用するレプリケーションマネージャエントリも格納されます。

図 6-3 では、各コンシューマはレプリケーションマネージャエントリに対応する 2 つのエントリを保持しています。これにより、コンシューマはレプリケーション要求を送信する際にマスターにバインドするときに、マスターを認証できます。各コンシューマが 1 つのレプリケーションマネージャエントリだけを保持し、すべてのマスターが同じレプリケーションエントリを認証に使用するように設定することもできます。

デフォルトでは、コンシューマはトポロジ内のすべてのマスターに対するリフェラルを保持します。クライアントから変更要求を受け取ると、コンシューマはマスターへのリフェラルをクライアントに返します。

注

レプリケーション環境では、コンシューマはクライアントからの変更要求をサプライヤとして機能するサーバーに転送しません。変更要求を受け取ったコンシューマは、クライアントからの変更要求を正しく実行できる可能性のあるマスターの URL リストを返します。

Sun ONE Directory Server 5.2 では、サーバーによって自動的に設定されるリフェラルを独自のリフェラルに書き換えることで、これらのリフェラルを制御できます。

リフェラルを制御できることで、配備のセキュリティとパフォーマンスを次のように最適化できます。

- リフェラルがセキュリティ保護されたポートだけをポイントするようにする
- ロードバランスのために Sun ONE Directory Proxy Server をポイントできる
- WAN によって隔てられた複数サーバーへの配備に限定し、ローカルサーバーにリダイレクトすることができる
- 4 方向のマルチマスタートポロジのサブセットだけにリフェラルを限定することができる

リフェラルの設定については、『Sun ONE Directory Server 管理ガイド』の「Setting Referrals」を参照してください。

レプリケーションの要素を理解いただくために、完全に接続された 4 方向のマルチマスターレプリケーションの設定と配備について説明します。図 6-4 は、マスター A で設定する必要があるレプリケーションアグリーメント、変更履歴ログ、レプリケーションマネージャを示しています。図 6-5 は、コンシューマ E で設定する必要がある同じ要素を示しています。

図 6-4 完全に接続された 4 方向のマルチマスターレプリケーションのマスター A のレプリケーション設定

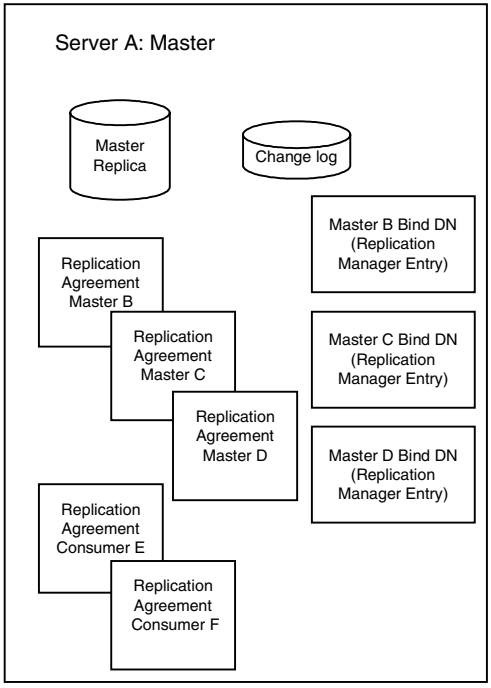


図 6-4 に示されるように、マスター A は、マスターレプリカ、更新履歴ログ、マスター B、C、D 用のレプリケーションマネージャエントリまたはバインド DN (4 つすべてのマスターでは、必ずしも同じレプリケーションマネージャエントリを使用しない場合) を必要とします。更新履歴ログとレプリケーションマネージャエントリのほかに、マスター A は、3 つの他のマスター B、C、D、およびコンシューマ E、F とのレプリケーションアグリーメントも必要とします。

図 6-5 完全に接続された 4 方向のマルチマスターレプリケーションのコンシューマサーバー E のレプリケーション設定

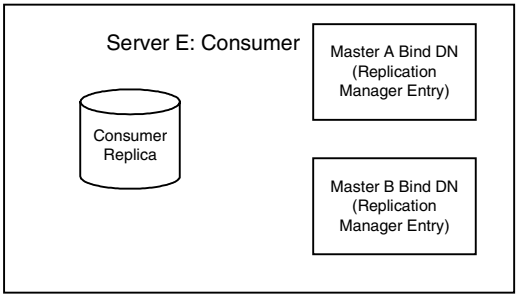


図 6-5 は、コンシューマ E のレプリケーション設定の詳細を示しています。コンシューマ E は、コンシューマレプリカ、およびレプリケーション更新の送信時にマスター A、B とのバインドで認証に使用されるレプリケーションマネージャエントリを必要とします。

広域ネットワーク (WAN) を介したマルチマスターレプリケーション

広域ネットワーク (WAN) を介したマルチマスターレプリケーション (MMR) は、Sun ONE Directory Server 5.2 の新機能です。この機能を利用することで、地理的に離れた国際的な配備や複数のデータセンター配備にまたがって MMR を構成できます。従来の Directory Server で、MMR を完全にサポートするためには、高速で待ち時間の少ない、100M バイト / 秒以上の転送スピードのネットワークにマスター Directory Server を接続することが必要でした。この制限のため、WAN を介した MMR は、実現できませんでした。しかし、この制限がなくなります。現在、Sun ONE Directory Server では、WAN を介した MMR をサポートしています。地理的に離れていることは、マルチマスターレプリケーションの障壁とはならなくなりました。この新機能の柔軟性は、大規模な配備に大きく貢献します。

注 プロトコルが異なるため、WAN を介したマルチマスターレプリケーションは、Directory Server の従来のリリースとの互換性はありません。このため、WAN を介したマルチマスターアプリケーションの設定では、WAN によって分散されるすべての Directory Server インスタンスは、*5.2* インスタンスである必要があります。

WAN を介した MMR (MMR over WAN) の配備を実現するために、Sun ONE Directory Server 5.2 の新しいレプリケーションプロトコルは、非同期を完全にサポートし、ウィンドウとグループ化のメカニズムを提供します。次に、これらのメカニズムについて詳しく説明します。

注 MMR over WAN はプロトコルの改善によって実現しましたが、この改善は、ローカルエリアネットワーク (LAN) を介した配備にも同様に有効です。

グループ化とウィンドウのメカニズム

レプリケーションの流れを最適化するために、Directory Server では変更を個別に送信する代わりにグループ化して送信することができます。また、サプライヤが処理継続のためのコンシューマからの受信通知を待たずにコンシューマに送信できる要求の数を指定することもできます。1 つの更新要求にグループ化できる変更の数を指定するときは `ds5ReplicaTransportGroupSize` 属性を使用し、コンシューマからの受信通知が必要になる前に送信できる sendUpdate 要求の数を指定するときは `ds5ReplicaTransportWindowSize` 属性を使用します。デフォルトのグループサイズは 1 で、デフォルトのウィンドウサイズは 10 です。つまり、別の値を指定しない場合、デフォルトのレプリケーションは要求のグループ化を行いませんが、コンシューマからの受信通知が必要になるまでに 10 の sendUpdate 要求を送信できます。

警告 グループ化とウィンドウのメカニズムはエントリのサイズに基づくため、十分なエントリサイズがない場合はレプリケーションのパフォーマンスを最適化することは難しくなります。エントリのサイズが比較的均一である場合は、グループ化とウィンドウのメカニズムを使用して、差分更新と完全更新を最適化することができます。また、MMR over WAN レプリケーションのトラフィックフローのパフォーマンスは、使用する WAN 接続の待ち時間と帯域幅によって異なる点にも注意が必要です。

MMR over WAN を設定するときは、これらの要因を慎重に分析してください。

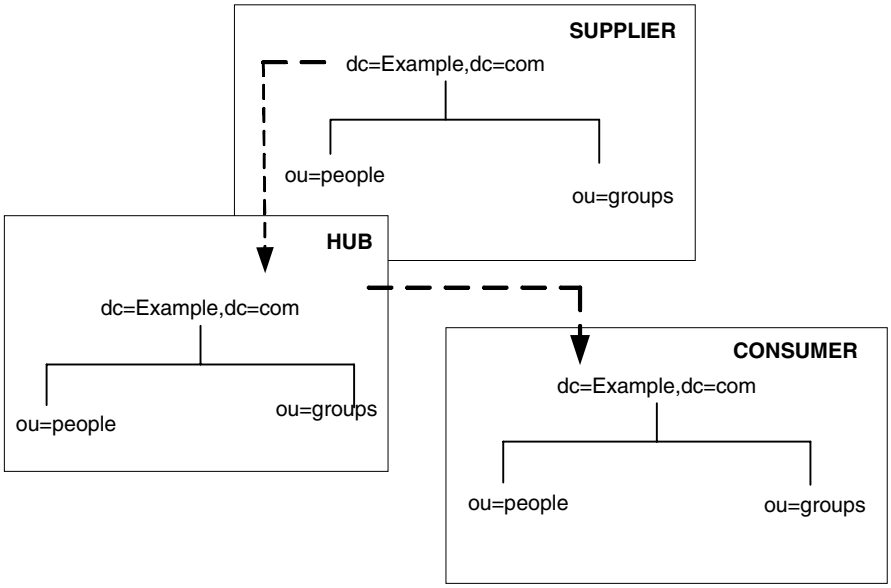
カスケード型レプリケーション

カスケード型レプリケーションでは、ハブとして機能するサーバーがサプライヤとして機能するサーバーから更新を受け取り、その更新をコンシューマ上で再現します。ハブはコンシューマとサプライヤの、両者の機能を合わせ持っています。つまり、通常のコンシューマと同様にデータの読み取り専用コピーを保持すると同時に、通常のサプライヤと同様に更新履歴ログも保持しています。

ハブは、元のマスターから受け取ったマスターデータのコピーを渡し、ディレクトリクライアントからの更新要求についてマスターを参照します。

図 6-6 は、このカスケード型レプリケーションの例を示しています。

図 6-6 カスケード型レプリケーションの例



カスケード型レプリケーションは、次の場合に便利です。

- 非常に大きなトラフィック負荷を均等にする必要がある場合：たとえば、マスターはすべての更新トラフィックを処理する必要があるため、コンシューマへのすべてのレプリケーショントラフィックをサポートするには、非常に大きな負荷がかかります。多数のコンシューマに対するレプリケーション更新を実行できるハブにレプリケーショントラフィックを任せることで、負荷を軽減できます。
- 地域分散型環境でローカルハブサプライヤを使用することで、接続費用を削減したい場合

- ディレクトリサービスのパフォーマンスを向上させる場合 : 読み取り操作を行うすべてのクライアントアプリケーションをコンシューマへ、更新操作を行うすべてのクライアントアプリケーションをマスターへ導くことができる場合は、ハブのすべてのインデックス (システムインデックスを除く) を削除できます。これにより、マスターとして機能するサーバーとハブとして機能するサーバーの間のレプリケーション速度が飛躍的に向上します。

同じ例を別の視点から図にすると、図 6-7 のようになります。この図は、レプリケーションアグリーメント、履歴変更ログ、デフォルトリテラルの面からサーバーがどのように設定されるかを示しています。

図 6-7 カスケード型レプリケーションのサーバー設定

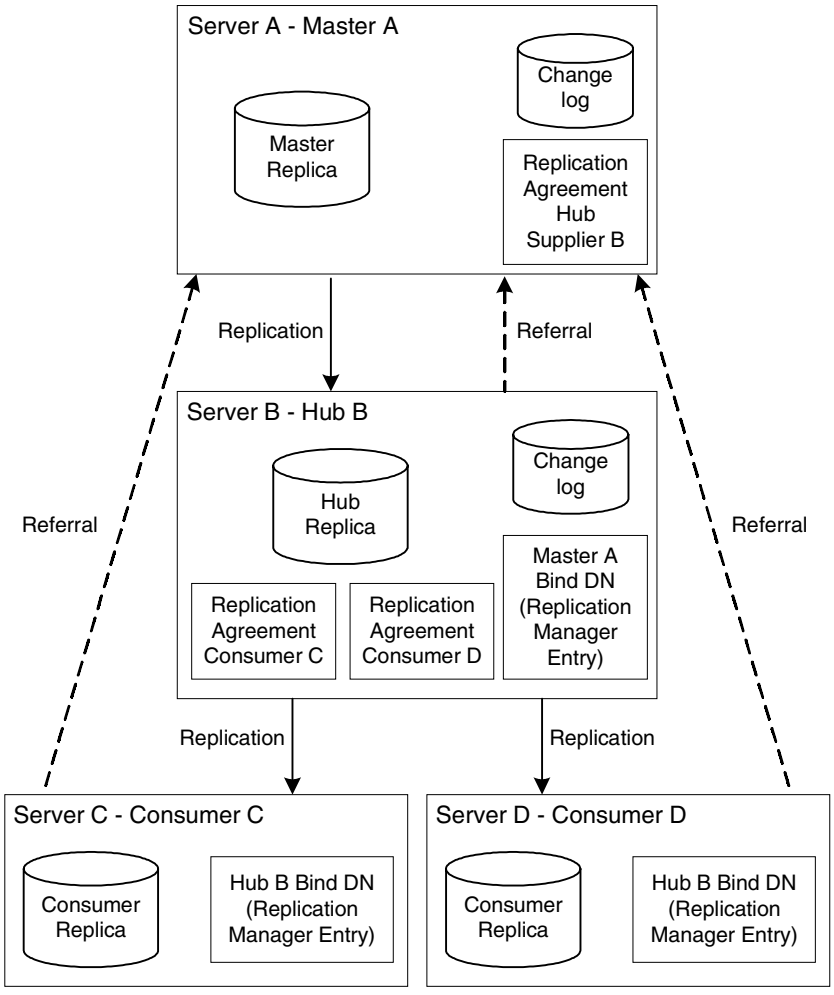


図 6-7 の例では、マスタとして機能するサーバーとハブとして機能するサーバーの間でハブを共有することで、レプリケーション更新の負荷をバランスしています。

マスターとハブは、どちらも更新履歴ログを維持します。ただし、クライアントからの変更要求を直接処理できるのは、マスターだけです。ハブにはマスター A のレプリケーションマネージャエントリが含まれるので、マスター A はハブにバインドし、レプリケーション更新を送信できます。コンシューマ C、D にはハブ B のレプリケーションマネージャエントリが含まれ、コンシューマへの更新の送信時の認証にはこれらのエントリが使用されます。

コンシューマとハブは、クライアントからの検索要求を処理できますが、変更要求の場合は、マスターへのリフェラルをクライアントに送信します。図 6-7 は、コンシューマ C、D がマスター A へのリフェラルを保持していることを示しています。これは、ハブとコンシューマの間のレプリケーションアグリーメントを作成するときに自動的に作成されるリフェラルです。ただしすでに指摘したように、パフォーマンスまたはセキュリティ上の理由で必要となる場合は、これらのリフェラルを書き換えることもできます。詳細は、128 ページの注を参照してください。

注 マルチマスターレプリケーションとカスケード型レプリケーションを組み合わせることができます。たとえば、135 ページの図 6-8 のマルチマスターの例では、サーバー C とサーバー D を、ハブとして設定することにより、これらのハブから任意の数のコンシューマにレプリケーションを実行できます。

混合環境

上で説明した例を自由に組み合わせて、要件に最も合った環境を構築できます。たとえば、マルチマスター設定とカスケード型設定を組み合わせて、図 6-8 に示すようなトポロジを構築できます。

図 6-8　マルチマスターレプリケーションとカスケード型レプリケーションの組み合わせ

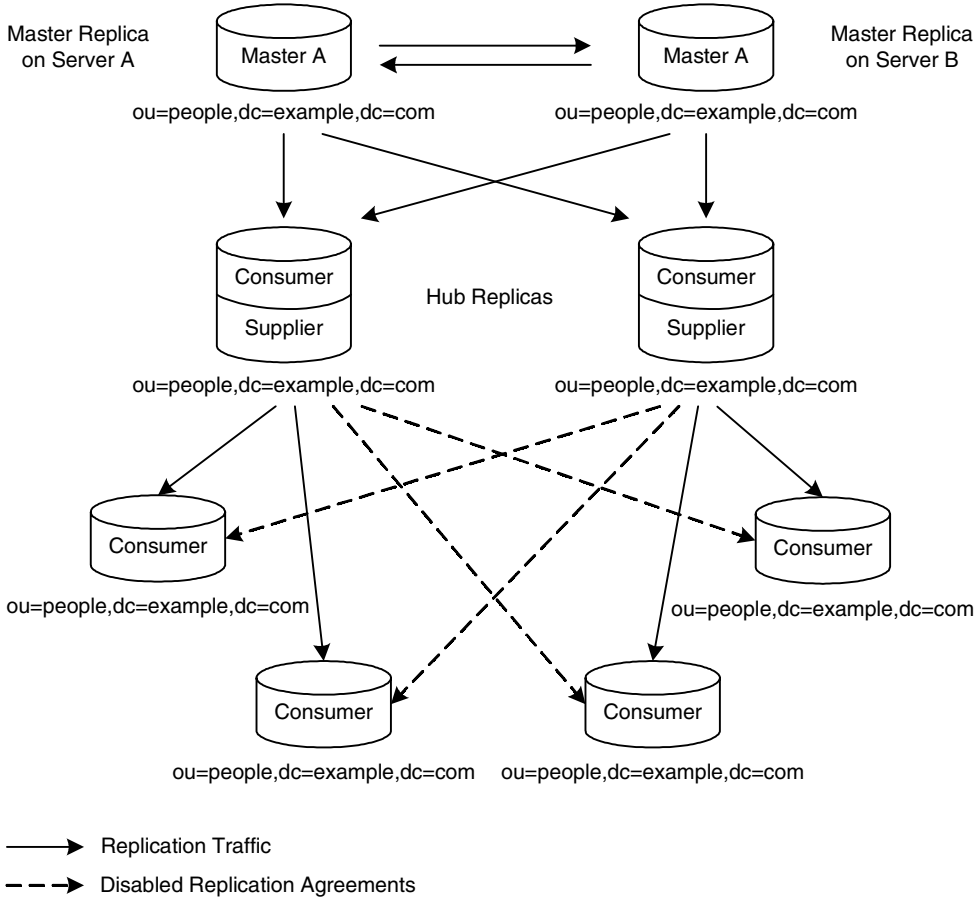


図 6-8 の例では、2 つのマスターと 2 つのハブが 4 つのコンシューマにデータをレプリケートしています。マスターとハブの間でハブを共有することで、レプリケーション更新の負荷をバランスします。このような設定は、管理するレプリケーション更新の負荷が非常に大きい場合に効果的です。

図 6-7 の例では、ハブ、マスター A、B が更新履歴ログを維持します。ただし、クライアントからの変更要求を直接処理できるのは、マスターだけです。クライアントからの変更要求をハブまたはコンシューマが受信すると、要求を処理するためにマスターへのリフェラルがクライアントに送信されます。図 6-8 にはリフェラルは示されていませんが、4 つのコンシューマと 2 つのマスターの間、および各ハブとマスターの間にリフェラルが存在します。これらのリフェラルは、トポロジの定義時に自動的に作成されます。

図 6-8 の例に示される点線は、無効化されたレプリケーションアグリーメントを示しています。これらのレプリケーションアグリーメントを有効化しない場合、いずれかのハブがオフラインになった場合にこのトポロジにはシングルポイント障害が発生します。レプリケーションアグリーメントを有効化して完全な読み書き対応フェイルオーバーを提供するかどうかは、そのトポロジの高可用性要件によって異なります。ただし、アグリーメントを有効化しない場合に、シングルポイント障害の危険が生じる可能性があることには注意が必要です。

部分レプリケーション

Directory Server の前回のリリースでは、レプリケーションの最小単位はデータベースで、特定のデータベース内の情報のサブセットだけをレプリケートすることはできませんでした。レプリケーションの最小単位がデータベースであることに変わりはありませんが、Sun ONE Directory Server 5.2 では、部分レプリケーションという新機能を提供することで、レプリケーションの細分化要件に対応しています。ここで説明する内容は次のとおりです。

- 部分レプリケーションとは
- 部分レプリケーションの設定

部分レプリケーションとは

部分レプリケーションでは、指定したデータベースのすべてのエントリの属性の中から、そのサブセットをサプライヤからコンシューマにレプリケートできます。部分レプリケーションが便利な機能であることを理解するために、2 つの簡単な例を紹介します。

- イントラネットサーバーとエクストラネットサーバーの間で同期が必要で、セキュリティ上の理由から一部のコンテンツをフィルタリングしなければならない場合に、部分レプリケーションがフィルタリング機能を提供します。
- レプリケーションによるリソースの消費を削減しなければならない場合、部分レプリケーションによってレプリケートする内容を選択できます。すべての場所で特定の属性を利用できるようにすることだけが必要な配備では、すべての属性をレプリケートする代わりに、部分レプリケーション機能を使用して必要な属性だけをレプリケートできます。たとえば、電子メールアドレスと電話番号の属性だけをレプリケートし、それ以外の属性をレプリケートしない場合、その他の属性が頻繁に変更されるようであれば、その結果として生じるトラフィックの負荷は非常に大きくなります。部分レプリケーションを使用することで、必要な属性をフィルタリングし、トラフィックを最小限に抑えることができます。このフィルタリング機能は、Directory Server が WAN によって分散しているレプリケーション環境で特に効果的です。

警告 Sun ONE Directory Server 5.2 の部分レプリケーション機能は、Directory Server の従来のバージョンとの逆互換性を持ちません。部分レプリケーションを使用する場合は、Directory Server のすべてのインスタンスが 5.2 インスタンスである必要があります。

部分レプリケーションの設定

部分レプリケーションを設定するには、レプリケートの対象から除外する、または対象に含める属性を選択します。これは、コンソールから簡単に設定できます。ただし、部分レプリケーションの設定を後から変更するには、変更を加える前に、アプリケーションアグリーメントを無効化する必要があります。変更が完了したら、レプリケーションアグリーメントを有効化し、新しい設定が適用されるようにコンシューマを初期化します。

警告 部分レプリケーションを設定するときは、次の 2 点に注意してください。

- 部分レプリケーションを設定するときは、レプリケートされるサーバーが読み取り専用レプリカであることが重要です。
- 排他的な設定アプローチを強くお勧めします。ACI、CoS、ロールなど、一部の機能の複雑さ、および特定の属性に対するこれらの機能の依存性を考えると、対象から除外する属性のリストを管理するほうが、対象に含める属性のリストを管理するより安全であり、人的エラーの可能性が少なくなります。

一般には、スキーマ違反を回避するために、スキーマに定義されている各エントリのすべての必須属性をレプリケートします。ただし、部分レプリケーション機能を使用して一部の必須属性をフィルタリングする場合は、スキーマ検査を無効にする必要があります。必須属性をフィルタリングすると ldif ファイルをロードできなくなるため、部分レプリケーションでスキーマ検査を有効にした場合、ldif ファイルからオフラインで初期化できなくなる可能性があります。スキーマ検査を無効にすると、パフォーマンスを向上できることがあります。また、部分コンシューマレプリカでスキーマ検査を無効にした場合、その部分コンシューマレプリカが存在するサーバーインスタンス全体にスキーマが適用されなくなります。このため、同じサーバーインスタンスでは、別の情報ディレクトリツリーのサプライヤ (読み書き可能) レプリカを設定しないようにする必要があります。

部分レプリケーションの設定ではサプライヤがスキーマをプッシュするため、部分コンシューマレプリカのスキーマは、マスターレプリカのスキーマのコピーとなり、適用される部分レプリケーション設定に対応しないことにも注意してください。

レプリケーション戦略の定義

使用するレプリケーション手法は、提供するサービスによって異なります。

- 高可用性を最優先する場合は、1 つのサイトに複数のディレクトリサーバーを配備したデータセンターを構築する必要があります。読み取りフェイルオーバーを提供するにはシングルマスターレプリケーションを、書き込みフェイルオーバーを提供するにはマルチマスターレプリケーションを使用できます。高可用性を実現するためのレプリケーションの設定方法については、142 ページの「高可用性を実現するためのレプリケーションの使用」を参照してください。
- 障害回復を最優先する場合は、2 つの地理的に異なる場所に独立したデータセンターを作成し、WAN を使用して分散させます。各データセンターは、フェイルオーバーのために 2 つのマスターをホスティングします。データセンターを二重化することで、1 つの場所で災害が発生した場合でも、もう一方は保護されます。地理的に離れた場所にまたがって書き込みフェイルオーバーの高い可用性を維持するには、WAN を介した 4 方向のマルチマスターレプリケーションを使用します。
- ローカルでのデータの可用性を最優先する場合は、レプリケーションを使用して、世界中の事務所に配置されているディレクトリサーバーにデータを物理的に分散する必要があります。本社などの単一の場所にすべての情報のマスターコピーを保管するか、あるいはそれぞれのローカルサイトが自身のサイトに関連する DIT 部分を管理するようにするかを選択できます。レプリケーションの設定タイプについては、143 ページの「ローカルでのデータの可用性を高めるためのレプリケーションの使用」を参照してください。
- どの場合でも、ディレクトリサーバーがサービスする要求の負荷をバランスして、ネットワークへの過剰負荷を防ぐ必要があります。ディレクトリサーバーとネットワークへの負荷をバランスする戦略については、144 ページの「ロードバランスのためのレプリケーションの使用」を参照してください。

レプリケーション手法を決定するには、ネットワーク、ユーザー、アプリケーション、および提供するディレクトリサービスがユーザーやアプリケーションによってどのように使われるかを調査することから始めます。この調査のガイドラインについては、「レプリケーション調査」を参照してください。

レプリケーション手法を決定したら、ディレクトリの配備を開始できます。ディレクトリサービスを段階的に配備していくことをお勧めします。ディレクトリを実際の環境に配備する段階に入ると、ディレクトリを全体的に配備した際の負荷をより正確に把握できるようになります。実際に運用されているディレクトリに基づいた負荷分析を行うことができない場合は、ディレクトリの使用状況をよく把握して、ディレクトリを修正するために準備してください。

次に、レプリケーション方法の決定に影響する要因について詳しく説明します。

- レプリケーションの逆互換性
- レプリケーション調査

- レプリケーションリソースの要件
- 高可用性を実現するためのレプリケーションの使用
- ローカルでのデータの可用性を高めるためのレプリケーションの使用
- ロードバランスのためのレプリケーションの使用
- 小規模サイト向けのレプリケーション方法の例
- 大規模サイト向けのレプリケーション方法の例

レプリケーションの逆互換性

最初に決定しなければならない事項の 1 つは、レプリケーションの設定に使用する Directory Server のバージョンです。レプリケーション設定が正しく機能させることができるように、139 ページの表 6-1 の情報を参照してください。これは、Directory Server の異なるバージョンの間で考えられるマスターとコンシューマの組み合わせ、および関連する制限を示しています。

表 6-1 Directory Server バージョン 4.x、5.0/5.1、5.2 の間のレプリケーションの逆互換性

	4.x コンシューマ	5.0/5.1 コンシューマ	5.0/5.1 マスター	5.2 コンシューマ	5.2 マスター	5.0/5.1/5.2 ハブサプライヤ
4.x マスター	可	可	可	可	可	不可
5.0/5.1 マスター	不可	可	可	可	可	可
5.2 マスター	不可	可	可	可	可	可

注

逆互換性については、次の 3 つの点に注意する必要があります。

- 4.0 マスターから 5.x マスターにレプリケートする設定で、5.x マスター上で旧バージョンのレプリケーションを有効にする場合、5.x マスターは、トポロジ内のその他の 5.x マスターからクライアント更新とレプリケーション更新を受信できなくなります。この場合、4.x マスターからのレプリケーション更新だけが受信可能です。ただし、旧バージョンのレプリケーションを無効にすると、5.x マスターのマスターレプリケーションとしての動作は完全に復元されます。
- 5.2 サーバーから 5.0/5.1 サーバーにレプリケートする場合、5.2 の新しい機能および機能拡張を使用しないようにする必要があります。使用した場合、5.0/5.1 サーバーで予期せぬ動作が発生する可能性があります。
- 5.2 スキーマ拡張によって 5.1 サーバーが混乱することがないように、`nsslapd-schema-replicate-useronly` 属性は on に設定する必要があります。

レプリケーション調査

レプリケーション手法を決定するときは、調査を通じて次のような情報を収集する必要があります。

- 異なる建物間やリモートサイト間を接続するネットワークの品質と使用可能な帯域幅
- ユーザーの物理的な位置、各サイトのユーザー数、ユーザーのアクティビティ

  たとえば、人事データベースや財務情報を管理するサイトは、ディレクトリを単に電話帳として使用する技術スタッフを含むサイトより、ディレクトリに大きな負荷をかけます。
- ディレクトリにアクセスするアプリケーションの数と、書き込み操作に対する読み取り、検索、および比較の操作の比率

  たとえば、メッセージングサーバーがディレクトリを使用する場合は、処理するメールメッセージごとにディレクトリが実行する操作数を調べる必要があります。ディレクトリを使用するその他の認証アプリケーションには、META Directory アプリケーションなどがあります。アプリケーションごとに、ディレクトリで実行される操作のタイプと頻度を調べる必要があります。
- ディレクトリに格納するエントリの数とサイズ

次に、これらの問題について検討し、レプリケーショントポロジを配備する上で考慮すべき重要な問題について説明します。

レプリケーションリソースの要件

レプリケーションを使用する場合は、リソースがさらに必要になります。レプリケーション手法を決定するときは、次のリソース要件を検討してください。

- ディスク使用量

  サプライヤでは、各更新操作後に更新履歴ログが書き込まれます。レプリケートされた複数のデータベースを保持するサプライヤでは、更新履歴ログはより頻繁に使用されるため、ディスク使用量はさらに増えます。

警告 ボトルネックを回避するため、コンシューマのマシンの規模は、少なくともサプライヤと同等である必要があります。

- サーバースレッド

  各レプリケーションアグリーメントは、2 つの追加スレッドを作成します。レプリケーションアグリーメントのスレッドは、操作スレッドとは分けられます。このため、いくつかのレプリケーションアグリーメントが存在する場合、クライアントアプリケーションで使用できるスレッドの数が少なくなり、クライアントアプリケーションに対するサーバーのパフォーマンスに影響が生じる可能性があります。

- ファイルディスクリプタ

  サーバーで使用できるファイルディスクリプタの数が、更新履歴ログ (1 ファイルディスクリプタが必要) と各レプリケーションアグリーメント (契約ごとに 1 ファイルディスクリプタが必要) によって減少します。

高可用性を実現するためのレプリケーションの使用

レプリケーションを使用して、単一のサーバーの障害によってディレクトリが使用できなくなることを防止します。最低限、ローカルのディレクトリツリーを少なくとも 1 つのバックアップサーバーにレプリケートしておきます。

ディレクトリの設計者の中には、データの信頼性を最大限保証するために物理的な場所ごとに 3 回はレプリケートしておく必要があると主張する人もいます。しかし、耐障害性を目的としたレプリケーションの使用頻度はユーザーの決定事項ですが、その決定はディレクトリで使用するハードウェアとネットワークの品質を考慮したものでなければなりません。信頼性の低いハードウェアでは、より多くのバックアップサーバーが必要になります。

注 通常のデータバックアップポリシー代わりにレプリケーションを使用することはできません。ディレクトリデータのバックアップについては、『Sun ONE Directory Server 管理ガイド』の「Backing Up Data」および 259 ページの「バックアップ方法の選択」を参照してください。

すべてのディレクトリクライアントに書き込みフェイルオーバーを保証する必要がある場合は、マルチマスターレプリケーションの手法を使用する必要があります。マルチマスターレプリケーションのフローで利用できるグループ化とウィンドウのメカニズムによって、レプリケーションのパフォーマンスが最適化されるようにレプリケーションアグリーメントを設定することができます。ただし、読み取りフェイルオーバーで充分な可用性が達成できる場合は、シングルマスターレプリケーションを使用できます。

LDAP クライアントアプリケーションは通常、1 つの LDAP サーバーだけを検索するように設定できます。つまり、カスタムクライアントアプリケーションを異なる DNS ホスト名にある LDAP サーバーを循環するように作成していないかぎり、LDAP クライアントアプリケーションが Directory Server の単一の DNS ホスト名を検索するように設定するだけで済みます。したがって、バックアップ用の Directory Server にフェイルオーバーを提供するには、DNS ラウンドロビンまたはネットワークソートのどちらかを使用する必要があります。DNS ラウンドロビンとネットワークソートの設定方法と使用方法については、DNS のマニュアルを参照してください。

地理的に離れた場所にまたがって書き込みフェイルオーバーの高い可用性を維持するには、WAN を介した 4 方向のマルチマスターレプリケーションを使用します。1 つの場所で 2 つのマスターを設定し、別の場所にも 2 つのマスターを設定します。これらのマスターを WAN 経由で完全に接続 することで、1 つのマスターがオフラインになった場合の安全対策とします。LAN 経由のマルチマスターレプリケーションと同様に、グループ化とウィンドウのメカニズムを利用して、レプリケーションのパフォーマンスを最適化することができます。

Sun ONE Directory Proxy Server 製品を代わりに使用することもできます。Sun ONE Directory Proxy Server については、`http://jp.sun.com/software/` を参照してください。

ローカルでのデータの可用性を高めるためのレプリケーションの使用

レプリケーションを使用して、ローカルでもデータを利用できるようにする必要があるかどうかはネットワークの品質とそのサイトで何を行うかによって決まります。また、ディレクトリに格納するデータの特性と、データが一時的に使用できなくなった場合の会社への影響について慎重に考慮する必要があります。データの重要性が高ければ、それだけ品質の低いネットワーク接続によって引き起こされる業務停滞は、許容できないものになります。

次の理由により、ローカルでもデータを利用するためにレプリケーションを使用する必要があります。

- データのローカルマスターコピーが必要である

  これは、特定の国の従業員だけにとって重要なディレクトリ情報を管理する必要がある、大規模な国際企業にとって重要な戦略です。また、データのローカルマスターコピーを保有することは、経営方針としてデータを部門レベルまたは組織レベルで管理するようにしている企業にとっても重要です。

- 信頼性が低いネットワーク接続、または断続的に利用可能なネットワーク接続を使用している

  国際ネットワークでよく見られるように、信頼性の低い WAN 使用している場合はネットワーク接続が断続的になることがあります。

- ネットワークに過度な負担が定期的にかかり、ディレクトリのパフォーマンスが著しく低下する

  たとえば、旧式のネットワークを使用している企業では、通常の営業時間帯にこのような状態が発生します。

- マスターレプリカでのネットワーク負荷と作業負荷を軽減する

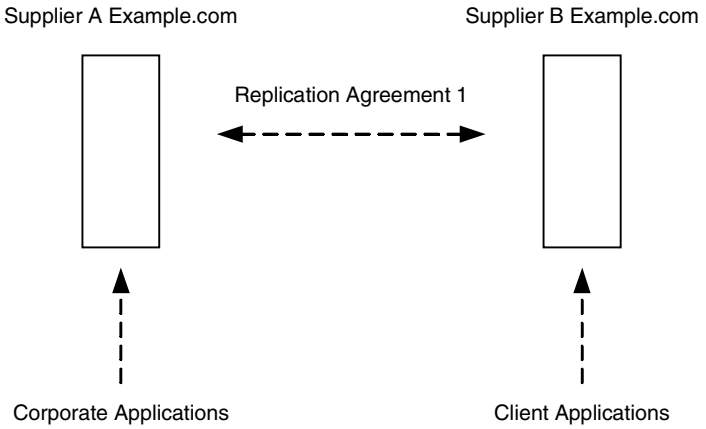
ネットワークの信頼性と可用性に問題がない場合でも、それに関係なくネットワーク使用量の削減が求められることがあります。

ロードバランスのためのレプリケーションの使用

レプリケーションを使用すると、次のような方法で Directory Server にかかる負荷をバランスすることができます。

- ユーザーの検索アクティビティを複数のサーバーに分散する
- 書き込みはマスターレプリカを保持するサーバーだけに限定し、その他のサーバーは読み取り専用に設定する
- メールサーバーのように、特定のタスクに専用サーバーを割り当てる

図 6-9 ロードバランスのためのマルチマスターレプリケーションの使用



ディレクトリデータをレプリケートする重要な理由の 1 つとして、ネットワーク負荷のバランスが挙げられます。可能であれば、比較的高速で信頼性の高いネットワーク接続を介してアクセス可能なサーバーにデータを移動します。最も重要な点は、サーバーとディレクトリユーザーとの間のネットワーク接続の通信速度と信頼性です。

通常、ディレクトリエントリの平均サイズは約 1K バイトです。したがって、ディレクトリの検索ごとにネットワークの負荷が約 1K バイト増加します。ディレクトリユーザーが 1 日当たり約 10 回検索を実行すると、ネットワークの負荷はユーザー 1 人につき 1 日当たり約 10,000 バイト増加します。低速な、負荷が大きい、あるいは信頼性が低い WAN を使用している場合は、ディレクトリツリーをローカルサーバーにレプリケートする必要がある場合もあります。

データをローカルに利用できるという利点が、レプリケーションを使用したことによるネットワーク負荷の増加という不利益を上回るかどうか慎重に検討してください。たとえば、ディレクトリツリー全体をリモートサイトにレプリケーションする場合は、ユーザーのディレクトリの検索によるトラフィックに比べ、はるかに大きな負荷をネットワークに課すことになります。このことは、ディレクトリツリーが頻繁に変更されるのに対して、リモートサイトの少数のユーザーが、1 日当たり数回のディレクトリ検索しか実行しない場合は、特に当てはまります。

たとえば、ディレクトリツリーに平均して 1,000,000 件を超えるエントリがあり、毎日 10 % 前後が変更される場合を考えてみます。ディレクトリエントリのサイズが平均して 1K バイトとすると、ネットワークの負荷が 1 日当たり 100M バイト増えることになります。しかし、リモートサイトの従業員がたとえば 100 人と少なく、1 日当たり平均して 10 回のディレクトリ検索を実行している場合は、ディレクトリアクセスによるネットワークの負荷は 1 日当たり 1M バイトにすぎません。

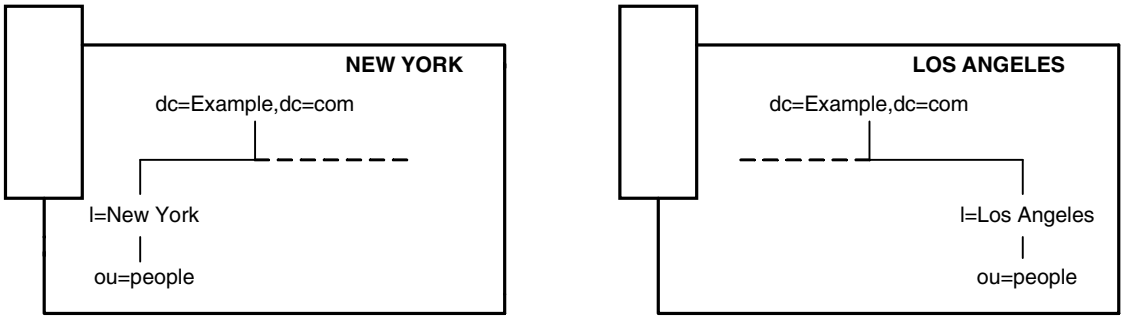
レプリケーションによる負荷と通常のディレクトリの使用による負荷の違いから、ネットワークのロードバランスを目的とするレプリケーションは好ましくないという結論に達する場合もあります。反対に、ネットワークにかかる負荷を考慮しても、ディレクトリデータをローカルで利用できる利点の方が勝っていると判断する場合もあります。

ネットワークに過度な負荷をかけずにデータをローカルサイトで使用できるようにするには、スケジュールされたレプリケーションを使用します。データの整合性とレプリケーションスケジュールについては、122 ページの「データの整合性」を参照してください。

ネットワークのロードバランスの例

2 つの都市に事務所を持つ企業を考えてみます。各事務所には、図 6-10 のような方法で管理する特定のサブツリーがあります。

図 6-10　ニューヨークとロサンゼルスでそれぞれ管理されるサブツリー



各事務所には高速のネットワークがありますが、2 都市間の通信にはダイヤルアップ接続を使用しています。このような場合は、以下のようにネットワークの負荷をバランスします。

- 事務所ごとに、ローカルで管理するデータのマスターサーバーとなるサーバーを 1 つ選択する

  ローカルで管理するデータを、選択したサーバーからリモートオフィスのマスターにレプリケートします。それぞれの地域でマスターコピーを保持することで、ユーザーはダイヤルアップ接続経由で更新と検索を行う必要がなくなり、パフォーマンスを最適化できます。

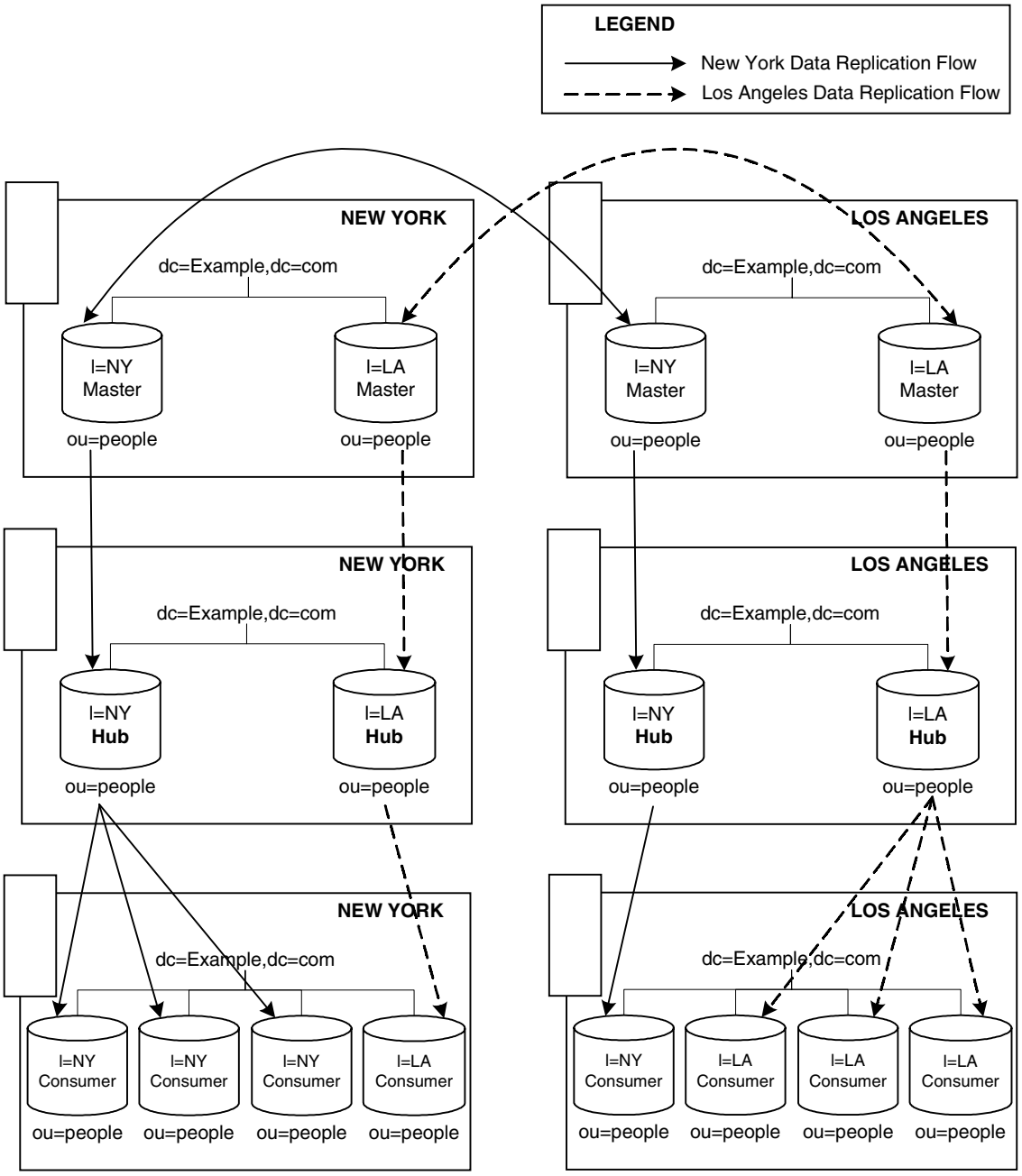
- ディレクトリデータの可用性を保証するために、リモートオフィスからのデータも含め、各マスター上のディレクトリツリーを少なくとも 1 つのローカル Directory Server にレプリケーションする

- それぞれの地域でカスケード型レプリケーションを設定し、ローカルデータに対する検索に特化したコンシューマの数を増やすことで、ロードバランスをさらに進める

  ニューヨーク事務所では、ロサンジェルスに関する検索よりもニューヨークに関する検索のほうが多く行われるため、この例では、ニューヨーク事務所に 3 つのニューヨークデータコンシューマと、1 つのロサンジェルスデータコンシューマが設定されています。同様に、ロサンジェルス事務所には 3 つのロサンジェルスデータコンシューマと 1 つのニューヨークデータコンシューマがあります。

147 ページの図 6-11 は、このネットワークロードバランスの設定を示しています。

図 6-11　マルチマスターレプリケーションとカスケード型レプリケーションによるロードバランス



パフォーマンス向上のためのロードバランスの例

ディレクトリで 15,000,000 のエントリを格納して 10,000,000 のユーザーをサポートする必要があり、各ユーザーが 1 日当たり 10 件のディレクトリ検索を実行する場合を考えてみます。また、1 日当たり 250,000,000 通のメールメッセージを処理し、メールメッセージごとに 5 件のディレクトリ検索を実行するメッセージングサーバーを使用しているとします。この場合、メール処理に 1 日当たり 1,250,000,000 件のディレクトリ検索が実行されると予測できます。ディレクトリとメッセージングシステムの合計トラフィックでは 1 日当たり 1,350,000,000 件のディレクトリ検索が実行されることになります。

1 日の就業時間は 8 時間、10,000,000 人のディレクトリユーザーが 4 つの時間帯に分かれているとすると、4 つの時間帯にわたる就業時間 (または、ピーク時の利用率) は 12 時間となります。したがって、1 日 12 時間で 1,350,000,000 件のディレクトリ検索をサポートする必要があります。これは毎秒 31,250 (1,350,000,000 / (60 × 60 × 12)) 件の検索をサポートするのと同じです。つまり、次のようになります。

10,000,000 人のユーザー	1 ユーザーにつき 10 件の検索 =	100,000,000 件の読み取り / 日
250,000,000 通のメッセージ	1 メッセージにつき 5 件の検索 =	1,250,000,000 件の読み取り / 日
	合計読み取り / 日 =	1,350,000,000
12 時間は 43,200 秒	合計読み取り / 秒 =	31,250

ここで、毎秒 5,000 件の読み取りをサポートできる CPU と RAM の組み合わせを Directory Server で使用しているとします。簡単な割り算で、この負荷をサポートするには 6 ～ 7 つの Directory Server が必要であることがわかります。ただし、10,000,000 人のディレクトリユーザーを有する企業の場合は、ローカルでデータを利用することも考慮するとさらに Directory Server を追加する必要があります。

注 Directory Server 5.2 では、適切なハードウェアと設定により、1 台のサーバーで、1 秒あたり 5,000 回を超える読み取り処理の継続が可能になります。

これらの計算から、次のような方法でレプリケートします。

- 1 つの都市に 2 つの Directory Servers をマルチマスター設定で配置し、すべての書き込みトラフィックを処理する

この設定は、すべてのディレクトリデータを 1 か所で管理することを想定しています。

- 上記のマスターを使用して 1 つまたは複数のハブにレプリケートする

  ディレクトリがサービスする読み取り、検索、および比較の要求はコンシューマで処理されるので、マスターは書き込み要求の処理に専念できます。ハブの定義については、132 ページの「カスケード型レプリケーション」を参照してください。

- ハブを使用して、会社全体のローカルサイトにレプリケートする

  ローカルサイトにレプリケートすることにより、サーバーや WAN の作業負荷をバランスし、ディレクトリデータの可用性を高めることができます。国内の 4 つのサイトにレプリケーションする場合を考えてみます。この場合、各ハブに 4 つのコンシューマが存在することになります。

- 各サイトで、少なくとも 1 回はレプリケートして可用性を高める。また、最低でも読み取り操作を実行できることを確認する

  DNS ソートを使用して、ローカルユーザーがディレクトリ検索に使用できるローカルの Directory Server を必ず見つけられるようにします。

小規模サイト向けのレプリケーション方法の例

会社全体が 1 つの建物内にある場合を考えてみます。この建物には、毎秒 100M バイトの高速で使いやすいネットワークが装備されています。ネットワークは非常に安定しており、サーバーのハードウェアと OS プラットフォームの信頼性も十分に高いものとします。また、1 つのサーバーの処理能力でサイトの負荷を容易に処理できるものとします。

このような場合、保守やハードウェアのアップグレードのためにプライマリサーバーが停止されたときでも、ディレクトリデータが利用できるようにしておくために、少なくとも 1 回はレプリケートしておきます。また、Directory Server の 1 つが使用できなくなったときに LDAP 接続のパフォーマンスを上げるために、DNS ラウンドロビンを設定します。代替方法として、Sun ONE Directory Proxy Server などの LDAP プロキシを使用することもできます。Sun ONE Directory Proxy Server については、`http://www.sun.com/software` を参照してください。

大規模サイト向けのレプリケーション方法の例

会社が 2 つの建物に分かれている場合を考えてみます。各建物には、毎秒 100M バイトの高速で使いやすいネットワークが装備されています。ネットワークは非常に安定しており、サーバーのハードウェアと OS プラットフォームの信頼性も十分に高いものとします。また、1 つのサーバーの処理能力で、各建物内のサーバーにかかる負荷を容易に処理できるものとします。

建物間は低速接続 (ISDN) で、通常の営業時間中に大きな負荷がかかります。

レプリケーション方法は次のようになります。

- いずれかの建物で、ディレクトリデータのマスターコピーを格納する 1 つのサーバーを選択する

  このサーバーは、ディレクトリデータのマスターコピーの管理に責任を持つ人が最も多くいる建物内に設置するようにします。これを建物 A とします。

- ディレクトリデータが常に利用できるようにするために、建物 A で少なくとも 1 回はレプリケートする

  書き込みフェイルオーバーを保証する必要がある場合は、マルチマスターレプリケーション設定を使用します。

- もう一方の建物 (建物 B) 内に 2 つのレプリカを作成する
- データのマスターコピーとレプリケートされたコピーの間で厳密な整合性が必要ない場合は、ピーク時間外にレプリケーションが行われるようにスケジュールする

大規模な国際企業のレプリケーション戦略

企業の 2 つの主要サイトがフランスと米国にあり、WAN によって分割されていると仮定します。WAN 経由によるレプリケーションが必要なだけでなく、パートナー企業にすべてのデータを開示しないために、一部のデータをフィルタリングしなければなりません。通常業務時間内は、使用する接続はとても混み合っています。

レプリケーション方法は次のようになります。

- ディレクトリデータのマスターコピーを、両方の地域のそれぞれのサーバーで保持する
- フランスサイト内と米国サイト内の書き込みフェイルオーバーのために、それぞれの地域の第 2 のマスターにデータをレプリケートする
- フランスと米国の間に完全に接続された 4 方向のマルチマスターレプリケーショントポロジを配備し、企業の配備全体で完全な高可用性と書き込みフェイルオーバーを実現する

- 各地域で必要なだけのコンシューマを配備し、検索によるマスターへの負荷をできるだけ軽減する
- 両地域のマスターとコンシューマの間に部分レプリケーションを設定し、パートナー企業にアクセスさせたくないデータをフィルタリングする
- 帯域幅の性能を最適化できるように、混雑していない時間帯に実行されるようにレプリケーションをスケジュールする

レプリケーションとほかのディレクトリ機能との併用

レプリケーションは Directory Server のほかの機能と連携して、高度なレプリケーション機能を提供します。次に、最適なレプリケーションの設計に役立つ、機能の併用について説明します。

レプリケーションとアクセス制御

ディレクトリはエントリの属性として ACI を格納しています。つまり、ACI はほかのディレクトリ内容と一緒にレプリケートされます。Directory Server は ACI をローカルに評価するため、これは重要です。

ディレクトリのアクセス制御の設計方法については、第 7 章に記載されている 157 ページの「安全なディレクトリの設計」を参照してください。

レプリケーションと Directory Server のプラグイン

レプリケーションは、Directory Server に付属するほとんどのプラグインと一緒に使用できます。次に、いくつかの例外と制限について説明します。

- レプリケーションと旧バージョン形式の更新履歴ログプラグイン
- レプリケーションと参照整合性プラグイン
- レプリケーションと操作前、操作後プラグイン

レプリケーションと旧バージョン形式の更新履歴ログプラグイン

旧バージョン形式の更新履歴ログプラグインは、Directory Server リリース 4.x との逆互換性のためにサポートされていますが、マルチマスターレプリケーション環境で機能するようには設計されていません。旧バージョン形式の更新履歴ログプラグインは、ローカルサーバーに現れる順に変更を行い、これらの変更がシステムに適用された順では変更を行いません。旧バージョン形式の更新履歴ログプラグインを 2 つのサーバーに設定した場合、変更が同じ順序、および同じ変更番号でログに記録されるとは限りません。レプリケーションプロセスでは変更の順序が重要であるため、マルチマスターレプリケーションで旧バージョン形式の更新履歴ログプラグインを使用した場合、どのような変更がログに記録されたのかを確認することはできますが、実際に変更された内容を適用する上では信頼することができません。

レプリケーションと参照整合性プラグイン

参照整合性プラグインがすべてのマスターレプリカで有効になっている場合は、このプラグインとマルチマスターレプリケーションを一緒に使用できます。

注　デフォルトでは、参照整合性プラグインは無効化されているので、Directory Server コンソールまたはコマンド行から有効化する必要があります。

整合性チェックにはメモリと CPU が多大に消費されるので、変更要求を送信するサーバーで参照整合性プラグインを有効化する前に、パフォーマンスリソース、時間、整合性の必要性を分析してください。

レプリケーションと操作前、操作後プラグイン

操作前プラグインと操作後プラグインをレプリケーションで使用する場合は、レプリケーションがこれらの操作前プラグインと操作後プラグインを検出できる必要があります。これらのレプリケーション操作に対して変更を加えるかどうかを決定することができますが、操作がレプリケートされた操作である場合は、変更によって予期せぬ動作が生じる可能性があるので注意してください。操作前プラグインと操作後プラグインについては、『Sun ONE Directory Server Plug-In API Programming Guide』の「Extending Client Request Handling」を参照してください。

レプリケーションと連鎖サフィックス

連鎖を使用してエントリを分散する場合は、連鎖サフィックスを保持するサーバーが、実際のデータを含むリモートサーバー (ファームサーバー) をポイントします。このような環境では、連鎖サフィックス自体を複製することはできません。ただし、実際のデータを格納しているデータベースのレプリケートは可能です。

注 マルチプレクサ上ではなく、ファームサーバー上のレプリケーションアグリーメントを設定する必要があります。

レプリケーションを連鎖サフィックスのバックアップに使用しないでください。連鎖サフィックスは手動でバックアップする必要があります。連鎖とエントリの分散については、第 5 章を参照してください。

スキーマのレプリケーション

レプリケーション環境で Directory Server を使用する場合は、レプリケーションに含まれるすべてのディレクトリサーバーについてスキーマの一貫性を維持する必要があります。サーバー間でスキーマの一貫性が維持されない場合は、レプリケーションで多くのエラーが発生する可能性があります。

スキーマの一貫性を確保するために、マルチマスターレプリケーション環境の場合でも、1 つのマスターサーバーでスキーマの変更を行うことをお勧めします。

スキーマのレプリケーションは自動的に行われます。サプライヤとコンシューマ間でレプリケーションが設定されている場合は、デフォルトでスキーマがレプリケートされます。

注 Directory Server 5.2 には、`nsslapd-schema-repl-useronly` という新しい属性が用意されています。この属性は、ユーザー定義のスキーマ、つまり、LDAP 経由で追加したスキーマ、または `X-ORIGIN` フィールドに `'user defined'` という値を指定することでファイルとして追加したスキーマだけをレプリケートするように設定できます。これにより、転送されるデータの量を削減し、スキーマのレプリケーションを高速にすることができます。

Directory Server でスキーマのレプリケーションに使用されるロジックはどのレプリケーションでも同じで、次のように説明できます。

1. データをコンシューマにプッシュする前に、サプライヤは自身のスキーマが、コンシューマ上で保持されているバージョンのスキーマと同期しているかどうかを検査します。
2. サプライヤとコンシューマ両方のスキーマエントリが同じであれば、レプリケーションが実行されます。
3. サプライヤ上のスキーマがコンシューマに格納されているものよりも新しい場合、サプライヤはデータのレプリケーションを実行する前に自身のスキーマをコンシューマにレプリケートします。

   Directory Server 5.2 には、ユーザー定義のスキーマだけをレプリケートするための属性があります。ユーザー定義のスキーマは、LDAP 経由で追加したスキーマ、または `X-ORIGIN` フィールドに `'user defined'` という値を指定することでファイルとして追加したスキーマです。これにより、転送されるデータの容量を削減し、スキーマのレプリケーションプロセスを高速にすることができます。

注 Directory Server の従来のバージョンとは異なり、スキーマに含まれる ACI はレプリケートされるようになりました。

マルチマスターセットの 2 つのマスターサーバーでスキーマに変更を加えた場合、最後に変更されたマスターの変更内容が優先され、そのスキーマがコンシューマに伝達されます。つまり、後から加えた変更が、もう一方のマスターに加えた変更と異なる場合は、その内容が失われる危険があります。変更の喪失を回避するために、常に 1 つのマスターだけでスキーマの変更を行うようにしてください。

注 コンシューマ上のスキーマは絶対に更新しないでください。サプライヤは起こりうる不整合を解決できないので、レプリケーションは失敗します。コンシューマ上のスキーマを更新し、その結果としてサプライヤ上のスキーマのバージョンがコンシューマ上のスキーマのバージョンより古くなった場合は、コンシューマを検索したとき、またはサプライヤ上で更新操作を試みたときにエラーが発生します。

スキーマは、マルチマスターレプリケーショントポロジの単一のマスターで保持する必要があります。標準の `99user.ldif` ファイルを使用している場合、これらの変更はすべてのコンシューマにレプリケートされます。カスタムスキーマファイルを使用している場合は、マスター上で変更を行ったら、必ずこれらのファイルをすべてのサーバーにコピーしてください。ファイルをコピーした後に、サーバーを再起動する必要があります。詳細は、49 ページの「カスタムスキーマファイルの作成 - 最良の事例と落とし穴」を参照してください。

カスタムスキーマファイルへの変更は、スキーマが LDAP または Directory Server コンソールを使用して更新されている場合にだけレプリケートされます。これらのカスタムスキーマファイルは、すべてのサーバー上で情報を同じスキーマファイルに保持するために、各サーバーにコピーする必要があります。詳細については、49 ページの「カスタムスキーマファイルの作成 - 最良の事例と落とし穴」を参照してください。

スキーマの設計については、第 3 章「スキーマの設計」を参照してください。

レプリケーションと複数パスワードポリシー

複数パスワードポリシーを使用する環境では、レプリケートされたエントリに適用するポリシーの定義を含む LDAP サブエントリをレプリケートする必要があります。レプリケートしない場合、デフォルトのパスワードポリシーが適用されますが、デフォルト以外のパスワードポリシーが設定されているエントリに対しては機能しません。これらのエントリを 5.0/5.1 サーバーにレプリケートした場合、レプリケーションは正常に機能しますが、Directory Server 5.2 に固有の複数のパスワードポリシーが設定されていたとしても、5.0/5.1 サーバーではパスワードポリシーは強制されません。

レプリケーションの監視

Sun ONE Directory Server 5.2 には、サーバー間のレプリケーションを監視するためのレプリケーション監視ツールが用意されています。レプリケーションアクティビティを監視できることは、レプリケーションに関する問題を特定したり、トラブルシューティングを行う上で役立ちます。Directory Server のすべてのレプリケーション監視ツールは、LDAP が有効な場合にだけ使用できます。次の 3 種類のレプリケーション監視ツールがあります。

- insync
- entrycmp
- repldisc

これらのレプリケーション監視ツールについては、『Sun ONE Directory Server Reference Manual』の「Replication Monitoring Tools」を参照してください。特定のレプリケーション属性による監視については、『Sun ONE Directory Server Reference Manual』の「Core Server Configuration Attributes」の章を参照してください。

注 これらのツールが LDAP クライアントを構成するため、サーバーへの認証と、`cn=config` に対する読み取りアクセス権を持つバインド DN が必要となることに注意してください。

insync

insync ツールは、マスターレプリカと 1 つまたは複数のコンシューマレプリカの間の同期状態を示します。競合の可能性を管理する場合は、同期の度合いに注意する必要があります。

entrycmp

entrycmp ツールを使用することで、複数のサーバー上の同一エントリを比較することができます。マスターレプリカからエントリが取得され、エントリの nsuniqueid を使用して指定コンシューマから同じエントリを取得します。エントリのすべての属性と値が比較され、すべてが同じであれば、エントリは同一であると見なされます。

注 insync または entrycmp を実行するマシンが、ファイアウォール、VPN、またはその他のネットワーク設定 (たとえば、トポロジ内のハブなど) によって照会先のホストに接続できない場合、insync ツールと entrycmp ツールを使用することは困難になります。

---

repldisc

repldisc ツールを使用することで、レプリケーショントポロジを推測することができます。トポロジの推測は、1 つのサーバーから始まり、トポロジ内のすべての既知のサーバーをグラフ化することで行われます。次に、repldisc ツールはトポロジを示す隣接マトリクスを出力します。このレプリケーショントポロジ推測ツールは、配備したグローバルトポロジを思い出すことが難しい、大規模で複雑な配備で便利です。

注 レプリケーション監視ツールを使用するときは、次の 2 つの点に注意してください。

- まず、ホストを識別するときは、すべてを記号名で指定するか、またはすべてを IP アドレスで指定する。2 つを組み合わせた場合、問題が生じる可能性が高くなる
- 第 2 に SSL が有効な場合、ツールを実行するマシンには、トポロジ内の他のサーバーが使用するすべての証明書のコピーが保持されている必要がある

第 7 章

安全なディレクトリの設計

Directory Server のデータを保護する方法は、これまで説明してきたすべての設計領域に影響します。セキュリティの設計は、ディレクトリに格納されているデータを保護し、ユーザーとアプリケーションのセキュリティおよび機密性の要件を満たすものでなければなりません。

この章では、セキュリティ要件の分析方法と、その要件を満たすディレクトリの設計方法について説明します。この章は、次の節で構成されています。

- セキュリティに対する脅威について
- セキュリティ要件の分析
- セキュリティ手法の概要
- 適切な認証方法の選択
- アカウントの無効化による認証の防止
- パスワードポリシーの設計
- アクセス制御の設計
- SSL による接続のセキュリティ保護
- 属性の暗号化
- エントリの安全なグループ化
- 設定情報のセキュリティ保護
- その他のセキュリティ関連資料

セキュリティに対する脅威について

ディレクトリのセキュリティに対する脅威となるものは数多く存在します。一般的な脅威についての理解を深めておくと、全体的なセキュリティ設計を行うときに役立ちます。ディレクトリのセキュリティに対する代表的な脅威は、次のカテゴリに分類できます。

- 不正なアクセス
- 不正な改ざん
- サービス拒否

次に、ディレクトリのセキュリティポリシーを設計するときに役立つように、一般的なセキュリティの脅威について説明します。

不正なアクセス

不正なアクセスからディレクトリを保護することは簡単なようにみえますが、実際にはこの問題はかなり複雑です。ディレクトリ情報の配信経路には、権限のないクライアントがデータにアクセスできる箇所がいくつもあります。

たとえば、権限のないクライアントが別のクライアントの証明情報を利用してデータにアクセスする場合があります。権限のないクライアントが、正当なクライアントと Directory Server の間でやり取りされる通信情報を傍受する場合も考えられます。

権限のないアクセスは社内から発生することも、また、会社がエクストラネットやインターネットに接続している場合は、外部から発生することもあります。

ここに挙げた例は、権限のないクライアントからディレクトリデータにアクセスする例の一部にすぎません。

Sun ONE Directory Server に備わっている認証方法、パスワードポリシー、およびアクセス制御のメカニズムは、不正アクセスの防止に効果があります。これらの問題に関しては、164 ページの「適切な認証方法の選択」、170 ページの「パスワードポリシーの設計」、および 183 ページの「アクセス制御の設計」を参照してください。

不正な改ざん

侵入者がディレクトリへアクセスしたり、Directory Server とクライアントアプリケーションの間の通信を傍受した場合は、ディレクトリデータが変更 (あるいは改ざん) される潜在的な危険があります。クライアントがデータを信頼できなかったり、ディレクトリ自体がクライアントから受信する変更や照会を信頼できない場合、ディレクトリは役に立たなくなります。

たとえば、ディレクトリが改ざんを検出できない場合、侵入者がクライアントからサーバーへの要求を変更し (または転送せずに)、サーバーからクライアントへの応答を変更することができます。SSL や、それと同様の技術を利用して、接続の両端で情報に署名することで、この問題は解決できます。Sun ONE Directory Server での SSL の使用については、202 ページの「SSL による接続のセキュリティ保護」を参照してください。

サービス拒否

サービス拒否攻撃とは、侵入者がディレクトリによるクライアントへのサービスの提供を妨害することです。たとえば、侵入者はひたすらシステムのリソースを消費して、ほかのユーザーがリソースを使用するのを妨害します。

Sun ONE Directory Server では、特定のバインド DN に割り当てるリソースに制限を設定することで、サービスの拒否攻撃を防ぎます。ユーザーのバインド DN に基づくリソース制限の設定については、『Sun ONE Directory Server 管理ガイド』の「Setting Resource Limits Based on the Bind DN」を参照してください。

セキュリティ要件の分析

自社のセキュリティ要件を特定するには、環境とユーザーを分析する必要があります。第 2 章「ディレクトリデータの計画とアクセス」でサイト調査を実施した際に、ディレクトリ内の各データについてどのユーザーが読み取りまたは書き込みができるかの基本的な決定は終わっているはずです。この情報が、ここではセキュリティの設計の基盤となります。

また、セキュリティの実装方法は、ディレクトリをどのように使用して業務をサポートするかによって異なります。イントラネットを供給するディレクトリでは、エクストラネットをサポートするディレクトリやインターネット上に公開されている電子商取引アプリケーションと同等のセキュリティレベルが要求されることはありません。

ディレクトリがイントラネットだけにサービスを提供する場合は、次の事項を考慮する必要があります。

- 業務の遂行に必要な情報へのアクセスをユーザーとアプリケーションに与える
- 社員と業務に関する機密データを通常のアクセスから保護する
- 情報の完全性を保証する

ディレクトリがエクストラネットにサービスを提供したり、インターネットを介して電子商取引アプリケーションをサポートしたりする場合は、上記の点に加えて、次の事項を考慮する必要があります。

- 顧客とパートナー企業に機密性を保証する
- 情報の完全性を保証する

次に、セキュリティ要件の分析について、以下の項目ごとに説明します。

- アクセス権限の決定
- データの機密性と完全性の保証
- 定期的な監査の実行
- セキュリティ要件の分析例

アクセス権限の決定

データ解析を実行するときは、ユーザー、グループ、取引先、顧客、およびアプリケーションがアクセスする必要のあるデータを特定します。

次の 2 通りの方法でアクセス権を与えます。

- すべてのカテゴリのユーザーに、自己管理を行うまたは管理を委託する権限を与え、同時に機密性の高いデータを保護する

  この開かれた方法を選ぶ場合は、どのデータが業務上の機密事項あるいは重要事項に該当するのかを十分検討する必要があります。

- 各カテゴリのユーザーに業務に必要な最小限のアクセスだけを与える

  この方法は制限が多いので、この方法を選ぶ場合は、各カテゴリの内部ユーザーが必要とする情報、また場合によっては外部のユーザーが必要とする情報について、ある程度の時間をかけて検討する必要があります。

どちらの方法でアクセス権限を与える場合でも、組織のユーザーが属するカテゴリとそのカテゴリに与えるアクセス権限を一覧にした、簡単な表を作成してください。また、ディレクトリに保持する機密データ、および各データを保護するために使用した手法を一覧にした表も必要に応じて作成してください。

ユーザーの識別方法については、164 ページの「適切な認証方法の選択」を参照してください。ディレクトリ情報へのアクセスを制限する方法については、183 ページの「アクセス制御の設計」を参照してください。

データの機密性と完全性の保証

エクストラネットを介して取引先との情報交換を行う場合、あるいはインターネット上で顧客が使用する電子商取引アプリケーションをサポートするためにディレクトリを使用する場合は、交換するデータの機密性と完全性を保証する必要があります。

これには、次のような方法があります。

- データを暗号化する
- 証明書を使用してデータに署名する
- データ転送を暗号化する

Sun ONE Directory Server で使用可能な暗号化方法については、174 ページの「パスワード保存スキーマ」および 204 ページの「属性の暗号化」を参照してください。データの署名については、202 ページの「SSL による接続のセキュリティ保護」を参照してください。

定期的な監査の実行

さらにセキュリティを高めるために、セキュリティポリシー全体の効率を検証する定期的な監査を実行する必要があります。定期的な監査では、ログファイルと SNMP エージェントによって記録された情報を検査します。ディレクトリの監視については、第 8 章「ディレクトリの監視」を参照してください。

セキュリティ要件の分析例

ここでは、架空の ISP である Example.com 社が自社のセキュリティ要件をどのように解析したかを例に示しながら説明します。

Example.com 社の業務は、Web ホスティングとインターネットへのアクセスを提供することです。Example．com 社では業務の一部として、クライアント企業のディレクトリをホストしています。また、多数の個人加入者にインターネットへのアクセスを提供しています。

このため、Example.com 社はディレクトリ内に次の 3 つの主要な情報カテゴリを持ちます。

- Example.com 社の社内情報
- 法人顧客に属する情報
- 個人加入者に関係する情報

Example．com 社は、次のようなアクセス制御を必要とします。

- Example2 社と Example3 社のディレクトリ管理者に自社のディレクトリ情報へのアクセスを許可する
- Example2 社と Example3 社のディレクトリ情報について、それぞれの企業のアクセス制御ポリシーを実装する
- Example.com 社のサーバーを使用して自宅からインターネットにアクセスするすべての個人クライアントについて、標準のアクセス制御ポリシーを実装する
- Example.com 社のディレクトリへの外部からのアクセスをすべて拒否する
- Example.com 社の加入者用ディレクトリに対する読み取り権限をすべてのユーザーに与える

セキュリティ手法の概要

Sun ONE Directory Server では、個々の要件に合ったセキュリティポリシーを設計するためのさまざまな手法が用意されています。セキュリティポリシーは、権限のないユーザーが機密情報を変更したり取り出したりできないような強固なものであると同時に、簡単に情報の管理ができるものでなければなりません。複雑なセキュリティポリシーを作成すると、許可したユーザーが情報にアクセスできなかったり、あるいはアクセスを許可していないユーザーがディレクトリ情報を変更したり取り出したりする問題につながります。

Sun ONE Directory Server では、次のセキュリティ手法を使用できます。

- 認証

  一方が他方の識別情報を検証する方法です。たとえば、LDAP のバインド操作時に、クライアントは Directory Server にパスワードを呈示します。

- パスワードポリシー

  たとえば、有効期限、長さ、構文など、パスワードの有効性を証明するための条件を定義します。

- 暗号化

  情報の機密性を保護します。データを暗号化すると、データは受信者だけが理解できるような方法で符号化されます。

- アクセス制御

  さまざまなディレクトリユーザーに与えるアクセス権限を調整し、必要な証明情報またはバインド属性を指定する方法を提供します。

- アカウントの無効化

  ユーザーアカウント、アカウントのグループ、またはドメイン全体を無効にして、すべての認証の試行に対して、自動的に拒否するようにします。

- SSL (Secure Sockets Layer)

  情報の完全性を保持します。送信する情報に暗号化とメッセージダイジェストを適用した場合、受信者は、その情報が転送中に改ざんされていないことを確認できます。

- 監査

  ディレクトリのセキュリティが危険にさらされていないかを確認できます。たとえば、ディレクトリで保持されるログファイルを監査できます。

セキュリティを管理するこれらのツールは、セキュリティの設計と組み合わせて使用できます。セキュリティの設計をサポートするために、レプリケーションやデータの分散など、ほかのディレクトリの機能を使用できます。

適切な認証方法の選択

セキュリティポリシーに関して決定が必要な項目に、Directory Server に対するユーザーのアクセス方法があります。匿名アクセスを許可するかどうか、または Directory Server を使用するすべてのユーザーにディレクトリへのバインドを要求するかどうかを決定します。

Sun ONE Directory Server は、次の認証方法をサポートしています。

- 匿名アクセス
- 簡易パスワード
- プロキシ承認
- セキュリティ保護された接続での簡易パスワード
- 証明書に基づくクライアント認証
- SASL ベースのクライアント認証

Directory Server は、相手がユーザーか LDAP 対応アプリケーションかにかかわらず、すべてのユーザーに対して同じ認証メカニズムを使用します。

クライアント単位またはクライアントのグループ単位での認証の防止方法については、169 ページの「アカウントの無効化による認証の防止」を参照してください。

匿名アクセス

匿名アクセスは、ディレクトリにアクセスするもっとも簡単な形式です。匿名アクセスを使用すると、認証とは関係なくだれでもディレクトリデータを利用できます。

しかし、匿名アクセスでは、だれがどのような検索を実行しているのかを追跡することはできず、検索を実行しているユーザーがいるということしかわかりません。匿名アクセスを許可すると、ディレクトリに接続するユーザーはだれでもデータにアクセスできます。

したがって、特定のユーザーまたはユーザーのグループによるディレクトリデータへの読み取りを禁止しようとしても、そのデータへの匿名アクセスを許可していれば、ユーザーは匿名でディレクトリにバインドすることでデータへのアクセスが可能になります。

匿名アクセスの特権は制限できます。通常、ディレクトリ管理者は、匿名アクセスに対して読み取り、検索、および比較の特権だけを許可します (書き込み、追加、削除、本人による書き込みは許可しません)。また、アクセスは、ユーザー名、電話番号、電子メールアドレスなど、一般的な情報を含む属性のサブセットに限定されるのが普通です。匿名アクセスは、政府指定の ID (米国の社会保障番号など)、自宅の電話番号と住所、給与情報など機密データには絶対に許可しないでください。

ユーザーパスワード属性が含まれていないエントリでユーザーがバインドを試行した場合、Directory Server は次のどちらかを実行します。

- ユーザーがパスワードの入力を試みない場合、匿名アクセスを許可する
- ユーザーがパスワードに、何らかの文字列の入力を試みた場合、アクセスを拒否する

たとえば、次の ldapsearch コマンドを考えてみます。

```
% ldapsearch -h ds.example.com -D "cn=joe,dc=Example,dc=com"
-w secretpwd -b "dc=Example,dc=com cn=joe" objectclass=*
```

この場合、Directory Server によって読み取りについての匿名アクセスが許可されますが、`ldapsearch` コマンドで与えたパスワードと一致するパスワードが自分のエントリに含まれていないため、Joe は自分のエントリにアクセスできません。

簡易パスワード

匿名アクセスを設定しない場合、ディレクトリの内容にアクセスするには Directory Server への認証を行う必要があります。簡易パスワード認証では、クライアントは再使用可能な簡単なパスワードを送信して、サーバーへの認証を行います。

たとえば、識別名と証明情報を送信するバインド操作によって、クライアントは Directory Server への認証を行います。サーバーはクライアント DN に対応したディレクトリ内のエントリを検出し、クライアントから送信されたパスワードとエントリ内に格納されている値が一致するかどうかを確認します。一致した場合、サーバーはクライアントを認証します。一致しない場合、認証操作は失敗し、クライアントにエラーメッセージが返されます。

多くの場合、バインド DN はユーザーのエントリに対応しています。ただし、ユーザーのエントリとしてではなく管理者のエントリとしてバインドする方が便利だと考えるディレクトリ管理者もいます。Directory Server では、バインドに使用するエントリは、`userPassword` 属性を使用できるオブジェクトクラスのエントリである必要があります。これにより、ディレクトリがバインド DN とパスワードを認識することができます。

ユーザーが DN の長い文字列を記憶できない場合もあるため、多くの LDAP クライアントはバインド DN をユーザーに表示しません。クライアントがバインド DN をユーザーに表示しない場合、クライアントは次のようなバインドアルゴリズムを使用します。

1. ユーザーはユーザー ID などの一意の識別子を入力する (たとえば、`bjensen`)
2. LDAP クライアントアプリケーションはこの識別子でディレクトリを検索し、関連付けられている識別名を返す
   (`uid=bjensen,ou=people,dc=Example,dc=com` など)

3. LDAP クライアントアプリケーションは、検出した識別名とユーザーが入力したパスワードを使用してディレクトリにバインドする

注	簡易パスワード認証の欠点は、パスワードがクリアテキストでネットワークに送信されることです。悪意を持ったユーザーが盗聴している場合、承認されたユーザーになりすます可能性があり、Directory Server のセキュリティを危険にさらすことになります。

簡易パスワード認証ではユーザーを簡単に認証できますが、組織のイントラネットだけに使用を限定した方がよいでしょう。この認証では、エクストラネットを介した取引先との転送やインターネット上での顧客との転送に求められるレベルのセキュリティは提供されません。

プロキシ承認

プロキシ承認は特殊な形式の認証です。ユーザーは自分の ID を使用して Directory Server にバインドしますが、プロキシ承認によって別のユーザーの権限が与えられます。

たとえば、プロキシ承認を使用すると、ディレクトリ管理者は一般ユーザーとして Directory Server へのアクセスを要求できます。ディレクトリ管理者は自分の証明情報を使用してディレクトリにバインドしますが、アクセス制御の評価のために、一般ユーザーの権限が与えられます。このような仮のユーザーはプロキシユーザーと呼ばれ、そのユーザーの DN は プロキシ DN と呼ばれます。

プロキシ要求を実行できる Directory Server を設定するには、次の手順を実行します。

- 管理者には、ほかのユーザーとしてのプロキシ権限を与える
- 一般ユーザーには、アクセス制御ポリシーで定義されている通常のアクセス権限を与える

注	Directory Manager 以外のディレクトリのすべてのユーザーにプロキシ権限を与えることができます。プロキシ権限により、すべての DN (Directory Manager DN を除く) をプロキシ DN として指定する権限が与えられるので、プロキシ権限を与える場合には十分な注意が必要です。

プロキシのメカニズムは非常に強力です。プロキシの主な利点の 1 つとして、Directory Server に要求を送信している複数のユーザーにサービスを提供するために、LDAP アプリケーションが 1 つのバインドで 1 つのスレッドを使用できるようにする点が挙げられます。ユーザーごとにバインドして認証する代わりに、クライアントアプリケーションはプロキシ DN を使用して Directory Server にバインドします。

プロキシ承認については、『Sun ONE Directory Server 管理ガイド』の第 6 章「Managing Access Control」を参照してください。

セキュリティ保護された接続での簡易パスワード

セキュリティ保護された接続では、暗号化によって第三者がデータを読めないようにした上で、Directory Server とクライアントの間でデータを送信します。クライアントは、次のいずれかの方法でセキュリティ保護された接続を確立できます。

- SSL (Secure Socket Layer) を使用してセキュリティ保護されたポートにバインドする
- 匿名アクセスでセキュリティ保護されていないポートにバインドし、Start TLS 制御を送信して TLS (Transport Layer Security) を使用する

どちらの場合も、サーバーにはセキュリティ証明書が必要で、この証明書を信頼するようにクライアントを設定する必要があります。サーバーは、証明書をクライアントに送信することで、公開鍵暗号化方式を使用してサーバー認証を行います。その結果、クライアントは目的のサーバーに接続していること、およびサーバーが改ざんされていないことを認識します。

これ以後、クライアントとサーバーは、この接続を通じて伝送されるすべてのデータを機密保護のために暗号化します。クライアントは、暗号化された接続でバインド DN とパスワードを送信してユーザー認証を受けます。それ以後のすべての操作は、そのユーザーの ID、またはバインド DN に別のユーザー ID へのプロキシ権限が含まれる場合はプロキシ ID による操作として実行されます。操作の結果がクライアントに返されるときは、すべてが暗号化されます。

SSL については、202 ページの「SSL による接続のセキュリティ保護」を参照してください。証明書の設定と SSL の有効化については、『Sun ONE Directory Server 管理ガイド』の第 11 章「Implementing Security」を参照してください。

証明書に基づくクライアント認証

SSL または TLS を使用する暗号化された接続を確立するときに、サーバーがクライアント認証を要求するように設定することもできます。クライアントは、ユーザー ID の確認のためにサーバーに証明書を送信する必要があります。バインド DN の決定には、ユーザーの DN ではなく、証明情報が使用されます。クライアント認証は、ユーザーの成りすましを防ぐ保護で、最も安全な接続です。

クライアントが送信できる証明情報の 1 つにユーザー証明書があります。証明書ベースの認証を行うには、証明書のマッピングを行うようにディレクトリを設定し、すべてのユーザーは各自の証明書をそれぞれのエントリに格納しておく必要があります。クライアントからユーザー証明書を受け取ると、サーバーは証明書の内容に基づいてマッピングを行い、ディレクトリからユーザーエントリを検索します。このエントリには、そのユーザーの証明書とまったく同じコピーが含まれている必要があり、これによってユーザー ID の有効性が識別されます。すべての操作はこのエントリの DN をバインド DN として行われ、すべての結果は SSL または TLS 接続によって暗号化されます。

証明書のマッピングについては、『Managing Servers with Sun ONE Console』の第 10 章「Using Client Authentication」を参照してください。また、『Sun ONE Directory Server 管理ガイド』の第 11 章に記載されている「Configuring Certificate-Based Authentication in Clients」も参照してください。

SASL ベースのクライアント認証

SSL または TLS 接続でクライアントを認証する方法としては、SASL (Simple Authentication and Security Layer) によるクライアント ID の確認もあります。Directory Server は、SASL の汎用セキュリティインタフェースを通じて次のメカニズムをサポートします。

- DIGEST-MD5 : このメカニズムは、クライアントから送信されたハッシュ値とユーザーパスワードのハッシュを比較することでクライアントを認証します。ただし、このメカニズムはユーザーパスワードを読み取る必要があり、DIGEST-MD5 による認証を希望するすべてのユーザーは、ディレクトリ内に {CLEAR} パスワード (クリアテキスト形式のパスワード) を持つ必要があります。
- GSSAPI : Solaris 操作環境で利用できる GSSAPI (General Security Services API) では、Directory Server は Kerberos V5 セキュリティシステムとの対話によってユーザー ID の有効性を確認します。クライアントアプリケーションは Kerberos システムに証明情報を提示し、このシステムがユーザーの ID を Directory Server に返します。

どちらの SASL メカニズムを使用する場合も、ID のマッピングを行うようにサーバーを設定する必要があります。SASL 証明情報はプリンシパルと呼ばれ、各メカニズムは特定のマッピングを使用してプリンシパルの内容からバインド DN を決定します。プリンシパルが 1 つのユーザーエントリにマッピングされ、SASL メカニズムがそのユーザーの ID を検証すると、ユーザーの DN がその接続のバインド DN となります。

詳細は、『Sun ONE Directory Server 管理ガイド』の第 11 章に記載されている「SASL Authentication Through DIGEST-MD5」および「SASL Authentication Through GSSAPI (Solaris Only)」を参照してください。

アカウントの無効化による認証の防止

ユーザーアカウントまたはアカウントのセットを一時的に無効にできます。アカウントが無効になると、ユーザーは Directory Server にバインドできないため、このユーザーの認証操作は失敗します。

アカウントの無効化は、`nsAccountLock` オペレーショナル属性を使用して実装されます。エントリに `true` の値を持つ `nsAccountLock` 属性が含まれている場合、サーバーはバインドを拒否します。

ユーザーとロールの無効化にも、同じ手法を使用します。ただし、ロールの無効化は、そのロールのメンバー全員を無効にしますが、ロールのエントリ自体は無効にはしません。ロールについては、72 ページの「管理されているロール、フィルタを適用したロール、入れ子のロール」を参照してください。

パスワードポリシーの設計

パスワードポリシーは、システム内でパスワードがどのように管理されるかを規定した規則の集合です。Directory Server でパスワードポリシーが定義する内容は次のとおりです。

- パスワード変更ポリシー
- 最小パスワード長
- パスワードの最大有効日数
- パスワードの有効期限ポリシーと、それに関連する警告手順
- パスワード構文検査ポリシー
- パスワード保存スキーマ
- パスワード履歴手順
- パスワード失敗記録手順
- アカウントロックアウト手順

Directory Server の前回のリリースと比較して、Sun ONE Directory Server 5.2 のパスワードポリシー機能は柔軟性を増しています。ディレクトリ全体に適用される 1 つのグローバルポリシーではなく、複数のパスワードポリシーを設定できます。複数のパスワードポリシーを設定し、それを特定のユーザーに適用するか、CoS およびロール機能を使用してユーザーセット全体に適用することができます。特定のユーザーまたはロールに合わせてパスワードポリシーを設定できるので、複雑なセキュリティ要件にも正確に対応でき、パスワードポリシーによるセキュリティ対策の実装時に Directory Server のユーザーと管理者に一層の柔軟性が与えられます。

ここではまず、パスワードポリシーの基本的な機能について説明します。次に、パスワードポリシーの各種設定方法と、複数のパスワードポリシーの適用を制御する優先順位について説明します。最後に、アカウントロックアウトポリシー、およびレプリケートされた環境用のパスワードポリシーの設計について説明します。ユーザーに適したパスワードポリシーを作成するために必要な、ユーザーが自由に使用できる属性については、『Sun ONE Directory Server Reference Manual』の「Password Policy Attributes」および「Account Lockout Attributes」を参照してください。この節は、次の項目で構成されています。

- パスワードポリシーの機能
- パスワードポリシーの設定
- アカウントのロックアウトポリシーの設計
- レプリケーション環境でのパスワードポリシーの設計

パスワードポリシーの機能

ここでは、パスワードポリシーの主な機能について説明します。説明する内容は次のとおりです。

- ユーザー定義のパスワード
- 最初のログインまたはリセット後のパスワードの変更
- パスワードの有効期限
- 期限切れの警告
- パスワードの構文検査
- パスワード長
- パスワードの最小有効日数
- パスワードの履歴
- パスワード保存スキーマ

ユーザー定義のパスワード

パスワードポリシーを設定して、ユーザーが自分のパスワードを変更することを許可または禁止することができます。適切なパスワードを用いることは、パスワードポリシーを強固なものにする上で重要です。適切なパスワードとは、安易な単語、つまり辞書に載っているような単語、ペットや子供の名前、誕生日、ユーザー ID、あるいは、簡単に見破られる可能性があるユーザーに関するその他の情報 (ディレクトリ自体に格納されている情報も含む) を使用していないものです。

また、パスワードには、文字、数字、記号などの組み合わせを含めるようにしてください。しかし、ユーザーは単に覚えやすいパスワードを使用する傾向があります。そのため、「適切な」パスワードの条件を満たすパスワードを事前に設定し、ユーザーによるパスワードの変更を許可しない企業もあります。

ただし、ユーザーにパスワードを割り当てる作業は、管理者にとってかなりの時間がかかる作業です。また、自分にとって意味があり覚えやすいパスワードをユーザー自身が選択するのではなく、管理者がパスワードを提供すると、ユーザーはそのパスワードをどこかに書き留めてしまい、だれかが見つけてしまう危険があります。デフォルトでは、ユーザー定義のパスワードは許可されています。

最初のログインまたはリセット後のパスワードの変更

Directory Server のパスワードポリシーでは、初回のログイン後または管理者がパスワードをリセットしたあとに、ユーザーがパスワードを変更する必要があるかどうかを決めることができます。

多くの場合、管理者が設定した初回のパスワードは、ユーザーのイニシャル、ユーザー ID、会社名など、ある種の表記規則に従って設定されます。一度表記規則が発見されると、通常ハッカーがシステムに侵入するために最初に入力を試みる候補となります。そのため、初回のログイン後または管理者によるリセット後に、パスワードの変更をユーザーに義務付けるとよいでしょう。このオプションをパスワードポリシーに設定すると、ユーザーが定義したパスワードが無効になっている場合でも、ユーザーはパスワードを変更するように要求されます。詳細は、171 ページの「ユーザー定義のパスワード」を参照してください。

パスワードの変更をユーザーに許可しないようにした場合、管理者は簡単に推測できる表記規則に従ったパスワードを割り当てるのではなく、簡単に見破られないパスワードを設定します。

デフォルトでは、ログインまたはリセット後にユーザーがパスワードを変更する必要はありません。

パスワードの有効期限

パスワードポリシーは、ユーザーが同じパスワードを無期限に使用できるように設定することができます。また、一定期間が過ぎると、パスワードが期限切れになるように設定することもできます。一般に、パスワードの有効期限が長いほど見破られやすくなります。その一方で、パスワードが頻繁に期限切れになると、ユーザーはパスワードを憶えることが困難になり、パスワードを書き留めるようになってしまいます。一般的なポリシーでは、パスワードを 30 から 90 日で期限切れにします。

Directory Server では、パスワードの有効期限を無効にしても、パスワードの有効期限の設定は残されます。つまり、パスワードの有効期限のオプションを有効に戻した場合、パスワードの有効期限は、最後にこの機能を無効にしたときに設定していた期間になります。たとえば、パスワードが 90 日で期限切れになるように設定して、次にパスワードの有効期限を無効にするように設定したとします。パスワード期限をもう一度有効にするように設定を戻すと、この機能を無効にする前には有効期限を 90 日に設定していたので、デフォルトのパスワードの有効期限は 90 日になります。

デフォルトでは、ユーザーパスワードは期限切れになりません。

期限切れの警告

一定の期間が過ぎると、ユーザーパスワードが期限切れになるようにパスワードポリシーを設定した場合は、パスワードが期限切れになる前にユーザーに警告を送信します。パスワードが期限切れになる 1 ～ 24,855 日前に、ユーザーに警告が送信されるようにポリシーを設定できます。ユーザーがサーバーにバインドすると、Directory Server によって警告が表示されます。パスワードの有効期限をオンに設定した場合、ユーザーのクライアントアプリケーションがこの機能をサポートしていれば、デフォルトではユーザーパスワードが期限切れになる 1 日前に (LDAP メッセージを介して) 警告がユーザーに送信されます。

パスワードの構文検査

パスワードポリシーには、パスワード文字列の構文ガイドラインがあります。パスワードの構文検査メカニズムによって、パスワード文字列がパスワードポリシーで定義したパスワード構文のガイドラインに従っているかどうかが検査されます。デフォルトでは、パスワードの構文検査はオフに設定されています。

パスワード長

Directory Server では、ユーザーパスワードの最低文字数を指定できます。一般に、パスワードが短いほど、不正な手段による解読が簡単になります。2 ～ 512 文字のパスワードを要求できます。パスワードに適した長さは、8 文字です。これは不正な手段で解読することが難しく、またユーザーが記録しておかなくても覚えられる長さです。

デフォルトの最小パスワード長は 6 文字です。最小パスワード長の検査は、パスワードの構文検査が有効な場合にだけ行われます。

パスワードの最小有効日数

指定期間中はユーザーがパスワードを変更できないように、Directory Server を設定できます。この機能と `passwordInHistory` 属性を組み合わせると、ユーザーが古いパスワードを再使用するのを防止できます。

たとえば、パスワードの最短有効日数 (`passwordMinAge`) 属性を 2 日に設定すると、1 つのセッションの間にパスワードを繰り返し変更して古いパスワードを履歴からいったんなくし、その後古いパスワードを再使用するという方法を防止できます。0 ～ 24,855 日の間の任意の数字を指定できます。ゼロ (0) の値は、ユーザーがすぐにパスワードを変更できることを示します。

パスワードの履歴

`passwordInHistory` 属性に値を設定できることで、最大で 24 の使用パスワードを格納するように Directory Server を設定できます。この値を `0` に設定すると、パスワード履歴機能は無効化されます。この場合、過去に使用されたパスワードは格納されず、ユーザーはパスワードを再使用できます。

一方、`passwordInHistory` 属性に `1` ～ `24` の値を指定した場合は、ディレクトリは指定数の古いパスワードを `passwordHistory` 属性に格納します。ユーザーが Directory Server に格納されているパスワードを再使用しようとすると、ディレクトリはそのパスワードを拒否します。この機能を使用した場合、覚えやすい数個のパスワードをユーザーが再使用することはできなくなります。

デフォルトでは、Dirctory Server は過去に使用されたパスワードを格納しません。

パスワード保存スキーマ

パスワード保存スキーマでは、ディレクトリ内に Directory Server パスワードを格納するときに使用する暗号化のタイプを指定します。次のタイプを指定できます。

- クリアテキスト (暗号化なし)
- SHA (Secure Hash Algorithm):
- SSHA (Salted SHA)。デフォルトの暗号化方式
- UNIX CRYPT アルゴリズム

ディレクトリに格納されているパスワードは ACI (アクセス制御情報) 命令を使用して保護できますが、ディレクトリにクリアテキストでパスワードを格納することはお勧めできません。crypt アルゴリズムは、UNIX のパスワードと互換性があります。SSHA は最も安全な方式で、Directory Server のデフォルトのハッシュアルゴリズムです。

パスワードポリシーの設定

Sun ONE Directory Server 5.2 には、パスワードポリシーについて 4 つのオプションが用意されています。デフォルトでは、Directory Server が提供するデフォルトパスワードポリシーが自動的に適用されます。このポリシーのパラメータは、必要に応じて変更できます。デフォルトパスワードポリシーのバックアップとして、Directory Server にはハードコーディングされたパスワードポリシーも用意されています。これは、デフォルトのパスワードポリシーが見つからない場合や、変更によって有効でなくなった場合に適用されます。ハードコーディングされたパスワードポリシーの属性値は、デフォルトパスワードポリシーの値と同じです。また、独自のパスワードポリシーを定義し、それを特定のユーザーに適用したり、CoS 機能やロール機能を使用してユーザーセットに適用することができます。

ここでは、パスワードポリシーに関するこれらのオプションの詳細と、特定のユーザーエントリに複数のパスワードポリシーが存在する場合にパスワードポリシーの適用を制御する優先順位について説明します。この節は、次の項目で構成されています。

- デフォルトパスワードポリシー
- ユーザーまたはユーザーセットに適用するパスワードポリシーの定義
- 複数のパスワードポリシーとその優先順位

デフォルトパスワードポリシー

Sun ONE Directory Server のデフォルトパスワードポリシーは、その名が示すとおり、ユーザーが独自に設計しない場合に使用されるパスワードポリシーです。

注 デフォルトパスワードポリシーの属性値を変更する場合は、従来リリースの Directory Server とは異なり、パスワードポリシーの属性が `cn=config` の下に直接格納されず、`cn=Password Policy,cn=config` の下に格納されるようになったことに注意してください。このエントリが存在しない場合は、Directory Server のハードコーディングされたパスワードポリシーが適用されます。

パスワードポリシーの属性については、『Sun ONE Directory Server Reference Manual』の「Password Policy Attributes」を参照してください。

デフォルトでは、パスワードポリシーによって次の項目、動作が適用されます。

- SSHA 保管スキーマ
- ユーザーはパスワードを変更できる
- 最初のログイン後、または管理者がパスワードをリセットした後に、ユーザーはパスワードを変更する必要はない
- パスワードの構文検査 (最小文字数の遵守など) を行わない
- パスワードには有効期限は設定されない
- パスワードの最大有効期間は 100 日とするが、パスワードの有効期限を有効化した場合にだけ適用される
- パスワードの変更が可能になるまでの期間は設定されない
- パスワードの有効期間メカニズムを有効化した場合は、パスワードの有効期限が切れる 1 日前の最初のバインド試行時にパスワードの期限切れ警告が送信される
- 使用したパスワードは記録されない
- ユーザーはアカウントをロックアウトされない
- アカウントロックアウトメカニズムを有効にした場合は、最大で 3 回のバインド試行の失敗の後にユーザーはロックアウトされ、ロックアウト期間は 1 時間とする
- パスワード失敗は、600 秒後に失敗カウンタから削除される

デフォルトパスワードポリシーをどのように適用するかは、セキュリティ要件がどれだけ厳しいかによって異なります。パスワードに有効期間が設定されず、構文検査は行われず、アカウントロックメカニズムも有効化されていないデフォルトパスワードポリシーのようなパスワードポリシーには、セキュリティリスクが付きまといます。セキュリティ要件と、厳しいパスワードポリシーによる管理のオーバーヘッドの間で

バランスを取る必要があります。厳密なセキュリティ要件を確立し、特定のパスワードポリシーソリューションを配備すると、長所と短所を分析できるようになります。指定したポリシーが十分なセキュリティを提供していない、または厳しすぎて扱いにくくなっていると判断したときは、ポリシーに変更を加えます。

ユーザーまたはユーザーセットに適用するパスワードポリシーの定義

特定のユーザーエントリまたはユーザーセットに適用するパスワードポリシーを定義するときは、新たに採用された `passwordPolicySubentry` という属性を使用します。この属性の値は、ユーザーのエントリに直接適用するパスワードポリシー属性が含まれる LDAP サブエントリの DN です。この属性は、管理者が入力する実属性、または CoS 定義によって生成される仮想属性のいずれかです。ユーザーセットに適用するパスワードポリシーを定義する一般的な方法は、CoS 定義を設定し、ユーザーエントリが持つロールの機能として、ユーザーエントリ内の `passwordPolicySubentry` 属性に値を指定する方法です。パスワードポリシーを割り当てる上で、ユーザーが持つロールの機能としてユーザーセットに割り当てる方法は唯一の方法ではありませんが、Directory Server コンソールで使用できるのはこの方法です。

ユーザーに適用するパスワードポリシーの定義

特定のユーザーに適用するパスワードポリシーを設定するときは、該当するサブツリーのルートを親に持つ LDAP エントリを追加することで、特定のサブツリーに対して適用されるポリシーを定義します。

警告 ユーザーは、ACI によって制御される `passwordPolicySubentry` 属性を変更できないように設定する必要があります。

Example.com 社のシステム管理者として、`dc=example,dc=com` サブツリーのユーザーにより厳しい `strictPwdPolicy` というパスワードポリシーを適用すると仮定します。適用する `strictPwdPolicy` パスワードポリシーは、デフォルトポリシーとは異なり、構文検査を行い、パスワードの有効期限を 10 日間に設定し、バインド試行に 3 回連続して失敗した場合にアカウントロックアウトを開始します。`dc=example,dc=com` サブツリー内のユーザーエントリに、次の `passwordPolicySubentry` 属性を直接追加します。

```
passwordPolicySubentry:cn=strictPwdPolicy,dc=example,dc=com
```

この passwordPolicySubentry 属性の値は、strictPwdPolicy パスワードポリシーを定義する属性が格納されている LDAP サブエントリの DN です。strictPwdPolicy パスワードポリシーのこの LDAP サブエントリの内容は次のとおりです。

```
dn:cn=strictPwdPolicy,dc=example,dc=com
objectclass:top
objectclass:passwordPolicy
objectclass:LDAPsubentry
passwordStorageScheme:SSHA
passwordChange:on
passwordMustChange:on
passwordCheckSyntax:on
passwordExp:on
passwordMinLength:6
passwordMaxAge:8640000
passwordMinAge:0
passwordWarning:8640000
passwordInHistory:6
passwordLockout:on
passwordMaxFailure:3
passwordUnlock:off
passwordLockoutDuration:3600
passwordResetFailureCount:600
```

ユーザーセットに適用するパスワードポリシーの CoS およびロール機能を使用した定義

前例と同様に、Example.com 社のシステム管理者として、より厳しい veryStrictPwdPolicy パスワードポリシーをすべての契約者に適用し、より緩やかな normalPwdPolicy パスワードポリシーを従業員に適用すると仮定します。Example.com 社では、契約者には contractor という管理されているロールが割り当てられ、社内のすべての従業員には employee という管理されているロールが割り当てられています。

178 ページの表 7-1 は、contractor ロールと employee ロールの定義を示しています。

表 7-1　contractor ロールと employee ロールの定義

contractor ロールの定義	employee ロールの定義
dn:cn=contractorRole,dc=example,dc=com objectclass:LDAPSubentry objectclass:nsRoleDefinition objectclass:nsSimpleRoleDefinition objectclass:nsManagedRoleDefinition cn:contractorRole description:managed role for contractors	dn:cn=employeeRole,dc=example,dc=com objectclass:LDAPSubentry objectclass:nsRoleDefinition objectclass:nsSimpleRoleDefinition objectclass:nsManagedRoleDefinition cn:employeeRole description:managed role for in-house employees

両方のロールに適用するパスワードポリシーを含む CoS 定義エントリと CoS テンプレートエントリを定義すると、各ロールの passwordPolicySubentry 属性 (適切なパスワードポリシーが格納された LDAP サブツリーの DN をポイントします) が生成されます。

注　ロール (のメンバー) にパスワードポリシーを割り当てるときは、passwordPolicySubentry 属性をロール自体に追加するのではなく、そのロールと関連付けられている CoS に割り当ててください。

各ロールの passwordPolicySubentry 属性の値を生成する 1 つの CoS 定義エントリを作成します。178 ページの表 7-2 は、作成しなければならない CoS 定義エントリを示しています。

表 7-2　パスワードポリシーの CoS 定義エントリ

CoS 定義のエントリ
dn:cn=PwdPol_cosDefinition,dc=example,dc=com objectclass:top objectclass:LDAPsubentry objectclass:cosSuperDefinition objectclass:cosClassicDefinition cosTemplateDn:cn=PwdPolTemplContainer,dc=example,dc=com cosSpecifier:nsRole cosAttribute:passwordPolicySubentry operational

次の operational 修飾子を指定することで、

```
cosAttribute:passwordPolicySubentry operational
```

生成される (仮想の) cosAttribute の値が、存在する可能性のあるいかなる実属性に対しても優先して適用されるように指定されます。実際に、ここで operational 修飾子を指定する必要があります。ロールと CoS については、第 4 章「ディレクトリツリーの設計」を参照してください。

CoS 定義エントリを作成したら、仮想 passwordPolicySubentry 属性の値が格納される CoS テンプレートエントリを作成します。この例では、管理されているロール contractor の passwordPolicySubentry 属性の仮想値は strictPwdPolicy LDAP サブエントリの dn で、管理されているロール employee の仮想値は normalPwdPolicy LDAP サブエントリの dn です。179 ページの表 7-3 は、管理されているロール contractor で必要となる CoS テンプレートエントリを示しています。

表 7-3 contractor ロールの CoS テンプレートエントリ

contractor ロールの CoS テンプレートエントリ
dn:cn=¥"cn=ContractorRole,dc=example,dc=com¥", cn=PwdPolTemplContainer,dc=example,dc=com objectclass:top objectclass:extensibleObject objectclass:costemplate objectclass:ldapsubentry cosPriority:1 passwordPolicySubentry:cn=veryStrictPwdPolicy,dc=example,dc=com

179 ページの表 7-4 は、管理されているロール employee で必要となる CoS テンプレートエントリを示しています。

表 7-4 employee ロールの CoS テンプレートエントリ

employee ロールの CoS テンプレートエントリ
dn:cn=¥"cn=EmployeeRole,dc=example,dc=com¥", cn=PwdPolTemplContainer,dc=example,dc=com objectclass:top objectclass:extensibleObject objectclass:costemplate objectclass:ldapsubentry cosPriority:1 passwordPolicySubentry:cn:cn=normalPwdPolicy,dc=example,dc=com

これらのエントリでは、テンプレートエントリを
cn=PwdPolTempContainer,dc=example,dc=com というコンテナに格納することも指定されています。このコンテナは、ldif では次のように表示されます。

```
dn:cn=PwdPolTempContainer,dc=example,dc=com
objectclass:top
objectclass:nsContainer
```

注 パスワードポリシーの管理を容易にするために、パスワードポリシーの適用対象となるユーザーまたはユーザーセットと、パスワードポリシー自体を同じ場所で保管することをお勧めします。これは、パスワードポリシーの LDAP サブエントリをレプリケートし忘れることを防止する上で役立ちます。

複数のパスワードポリシーとその優先順位

Directory Server では、複数のパスワードポリシーを使用できるようになったため、あるユーザーエントリに適用されるパスワードポリシーが存在し、同時に同じユーザーエントリが別のパスワードポリシー対象ロールに属するケースが容易に想像されます。どちらのパスワードポリシーが優先されるのでしょうか。セキュリティ要件に実際に対応するパスワードポリシーを配備するには、適用を制御する優先順位と、CoS テンプレートエントリの定義時にその順序を制御する方法について理解する必要があります。

ユーザーに複数のパスワードポリシーが割り当てられている場合に、パスワードポリシーの適用を制御する優先順位には、主に次の 3 つの規則が適用されます。

1. CoS 定義によって生成されるパスワードポリシーは、同じユーザーエントリに直接割り当てられているパスワードポリシーに優先して適用されます。これは、CoS 定義エントリに定義されている cosAttribute 値には operational 修飾子が必ず含まれているためで、これにより、CoS 生成によるパスワードポリシーは、ユーザーに直接割り当てられたあらゆる実属性に優先して適用されます。ロールと CoS メカニズムについては、第 4 章「ディレクトリツリーの設計」を参照してください。
2. ユーザーエントリに割り当てられているパスワードポリシーは、デフォルトのパスワードポリシーに優先して適用されます。
3. cn=Password Policy,cn=config の下に格納されているデフォルトのパスワードポリシーは、Directory Server に用意されているハードコーディングされたパスワードポリシーの値に優先して適用されます。

警告 CoS を使用してパスワードポリシーを設定するときは、CoS によって生成された複数のパスワードポリシーがユーザーに適用された場合の優先順位を指定する必要があります。優先順位を指定するには、CoS テンポラリエントリを作成するときに cosPriority 属性に適切な値を入力します。最優先を指定するには、値に 0 を割り当てます。cosPriority 属性を持たない CoS テンプレートの優先順位は最低と見なされ、複数のテンプレートの cosPriority 属性の値が同じ (または指定されていない) 場合は、任意の優先順位が適用されます。ロールと CoS については、第 4 章「ディレクトリツリーの設計」を参照してください。

アカウントのロックアウトポリシーの設計

ディレクトリに対するパスワードポリシーを確立すると、アカウントのロックアウトポリシーを設定して、ユーザーのパスワードを潜在的な脅威から保護することができます。

ロックアウトポリシーをパスワードポリシーと組み合わせて使用すると、セキュリティが向上します。パスワードポリシーを設定し、ユーザーが一定の回数バインドに失敗した場合に、そのユーザーをディレクトリからロックアウトすることができます。

アカウントのロックアウト機能を使用すると、ハッカーがユーザーパスワードを繰り返し推測して、ディレクトリに侵入しようとするのを防止できます。アカウントロックアウトのカウンタはディレクトリサーバー固有です。この機能は、ディレクトリサービスからグローバルにロックアウトするようには設定されていません。つまり、レプリケーション環境でもアカウントロックアウトのカウンタはレプリケートされません。

レプリケーション環境でのパスワードポリシーの設計

レプリケーション環境では、パスワードポリシーとアカウントロックアウトポリシーが次のように適用されます。

- パスワードポリシーは、データのマスターコピーに適用される
- アカウントロックアウトポリシーは、レプリケーションの対象となるすべてのサーバーに適用される

ディレクトリ内にあるパスワードポリシーの状態情報の一部はレプリケートされます。次の属性がレプリケートされます。

- passwordHistory
- passwordAllowChangeTime

- passwordExpirationTime

警告 ただし、cn=Password Policy,cn=config に格納されている設定情報はローカルに保持され、レプリケートされません。この情報には、パスワードの構文とパスワードの変更履歴が含まれます。アカウントロックアウトのカウンタもレプリケートされません。

レプリケーション環境でパスワードポリシーを設定するときは、次の点を考慮します。

- 複数のパスワードポリシーを使用する環境では、レプリケートされたエントリに適用するポリシーの定義を含む LDAP サブエントリをレプリケートする必要があります。これを行わない場合、ポリシーの定義を含む LDAP サブエントリが存在しないため、デフォルトのパスワードポリシーが適用されます。
- すべてのレプリカは、パスワードの期限切れが近づくと警告を発します。この情報はローカルの各サーバー上に保持されます。したがって、ユーザーが複数のレプリカに順番にバインドした場合、ユーザーは同じ警告を数回受信することになります。また、ユーザーがパスワードを変更した場合は、この情報がコンシューマのレプリカで更新されるまで時間がかかることがあります。ユーザーがパスワードを変更し、直ちにバインドし直そうとした場合に、マスターレプリカに加えられた変更がコンシューマレプリカに登録されるまでバインドは失敗します。
- サプライヤやコンシューマを含むすべてのサーバーでバインドの動作を一致させたい場合は、各サーバー上でパスワードポリシーの設定情報を一致させます。必ず同じパスワードポリシー設定情報を各サーバーに作成してください。
- マルチマスター環境では、レプリケートされていない場合、アカウントロックアウトのカウンタが予測できない動作をすることがあります。
- レプリケーションのために作成したエントリ (サーバーの識別するレプリカマネージャエントリなど) には、無期限のパスワードを設定する必要があります。これらの特別なユーザーに確実に無期限のパスワードを使用させるには、passwordExpirationTime 属性をエントリに追加し、この属性に 20380119031407Z (有効範囲の最大値) を指定します。

アクセス制御の設計

ディレクトリのクライアントの識別情報を特定する 1 つ以上の認証スキーマを定義したら、ディレクトリ内に含まれる情報を保護するためにスキーマをどのように使用するか決定する必要があります。アクセス制御では、特定の情報へのアクセス権を一部のクライアントに与え、その他のクライアントには与えないように指定することができます。

アクセス制御は、1 つまたは複数のアクセス制御リスト (ACL) を使用して指定します。ディレクトリの ACL は、指定したエントリやその属性についてアクセス権限 (読み取り、書き込み、検索、プロキシ承認、追加、削除、比較など) を許可または拒否する 1 つまたは複数のアクセス制御情報 (ACI) 文から構成されます。

ACL を使用すると、次の項目に対する権限を設定できます。

- ディレクトリ全体
- ディレクトリの特定のサブツリー
- ディレクトリ内の特定のエントリ
- 特定のエントリの属性セット
- 特定の LDAP 検索フィルタにマッチするすべてのエントリ

また、特定のユーザー、特定のグループに属するすべてのユーザー、およびディレクトリのすべてのユーザーに対する権限を設定できます。さらに、IP アドレスや DNS 名など、ネットワークの場所に対してアクセス権を定義することもできます。

ここでは、Sun ONE Directory Server のアクセス制御メカニズムについて説明します。説明する内容は次のとおりです。

- ACI の形式について
- デフォルト ACI
- 権限の設定方法の決定
- 実効権限に関する情報の取得
- ACI の使用に関するヒント
- ACI の制限事項

ACI の形式について

セキュリティポリシーを設計する場合、Directory Server 内での ACI の表現方法を理解していると役立ちます。また、ディレクトリで設定できる権限を理解していることも有用です。ここでは、ACI のメカニズムの概要を簡単に説明します。ACI の形式について詳細は、『Sun ONE Directory Server 管理ガイド』の「Managing Access Control」の章を参照してください。

ACI は、エントリの属性としてディレクトリ内に格納されます。`aci` 属性はオペレーショナル属性です。この属性は、そのエントリのオブジェクトクラス用に定義されたものであるかどうかに関わらず、ディレクトリ内のすべてのエントリで使用できます。aci 属性は、Directory Server がクライアントから LDAP 要求を受け取るときに、どのアクセス権が与えられ、どのアクセス権が拒否されるかを判定するために使用されます。`aci` 属性が `ldapsearch` 処理で返されるように指定することができます。ディレクトリ ACI の一般的な形式は、次のとおりです。

target permission bind_rule

ACI の変数は、次のように定義されます。

- *target*

  ACI の対象となるエントリ (通常はサブツリー)、対象となる属性、または両方を指定します。つまり、*target* は、ACI が適用されるディレクトリの要素を示します。ACI は 1 つのエントリだけを対象にする場合もありますが、複数の属性を対象にすることもできます。また、*target* には LDAP 検索フィルタを含めることもできます。これにより、共通の属性値を持つ多種多様なエントリに権限を設定できます。

- *permission*

  この ACI で設定される実際の権限を示します。*permission* は、指定した *target* に対して読み取り、書き込み、検索、プロキシ承認、追加、削除、比較などの特定のタイプのディレクトリアクセスを、ACI が許可または拒否していることを示します。

- *bind_rule*

  権限が適用されるバインド DN またはネットワークの場所を示します。また、バインド規則で LDAP フィルタを指定することもあり、バインド中のクライアントアプリケーションについてフィルタが true であると評価されると、この ACI がクライアントアプリケーションに適用されます。

つまり、ACI は次のように表現されます。

> 「ディレクトリオブジェクトの *target* に対して *bind_rule* が true の場合、*permission* が許可または拒否される」

すべての属性に対する検索、読み取り、比較をユーザーに許可する ACI の例は、次のようになります。

```
aci: (targetattr = "*")(version 3.0; acl "my aci"; allow
 (search,read,compare) userdn="ldap:///all";)
```

ACI の *permission* と *bind_rule* の部分はペアで設定され、アクセス制御規則 (ACI) とも呼ばれます。すべてのターゲットに複数の *permission bind_rule* ペアを持たせることができます。これにより、特定のターゲットに複数のアクセス制御を効率的に設定できます。たとえば、次のようにします。

target (permission bind_rule) (permissions bind_rule)...

たとえば、Babs Jensen としてバインドしているすべてのユーザーに Babs Jensen の電話番号への書き込み権限を設定できます。この権限のバインド規則 (bind_rule) は、「Babs Jensen としてバインドしている場合は」という部分です。ターゲット (target) は Babs Jensen の電話番号、権限 (permission) は書き込み権限です。

ターゲット

ディレクトリで作成した各 ACI で、どのエントリをターゲットとするのかを決定する必要があります。ディレクトリの分岐点を示すディレクトリエントリをターゲットとする場合は、その分岐点だけでなくその子エントリすべてが権限の適用範囲に含まれます。このように ACI を定義すると、ディレクトリツリーの高いレベルに汎用的な ACI を置き、ツリーの下位に置かれる可能性の高いエントリに対してこの ACI を効果的に適用できます。たとえば、`organizationalUnit` エントリまたは `locality` エントリのレベルで、`inetorgperson` オブジェクトクラスを含むエントリをターゲットとする ACI を作成できます。この機能を使用すると、汎用的な規則を分岐点のできるだけ高いレベルに置くことによって、ディレクトリツリー内の ACI の数を最小限にできます。より限定的な規則の適用範囲を制限するには、できるだけ最下位のエントリに近い位置にその規則を置きます。

ACI がターゲットとするエントリを明示的に指定しない場合は、その ACI 文が含まれているディレクトリエントリが ACI のターゲットとなります。また、ACI のターゲットとなるデフォルトの属性セットは、ターゲットとなったエントリのオブジェクトクラス構造の中で利用可能なあらゆる属性になります。

ACI ごとに、1 つのエントリだけをターゲットにすることも、1 つの LDAP 検索フィルタに一致するエントリだけをターゲットにすることもできます。

注 ルート DSE エントリに置かれた ACI は、そのエントリだけに適用されます。

エントリをターゲットとする設定以外にも、そのエントリの属性をターゲットに含めることができます。これにより、属性値の一部だけに適用される権限を設定できます。属性のセットをターゲットとするには、ターゲットに含まれる属性またはターゲットに含めない属性を ACI で明示的に指定します。オブジェクトクラス構造で許可される数個の属性を除くすべての属性に権限を設定する場合は、後者の方法を使用してください。`aci` 属性には複数の値を設定できます。つまり、同じエントリまたは同じサブツリーに対して、複数の ACI を定義できます。

権限

権限を許可あるいは拒否することができます。一般に、189 ページの「アクセスの許可または拒否」で説明している理由から、権限を拒否することは避けてください。

以下の権限を許可または拒否できます。

- Read (読み取り)

  ディレクトリデータを読み取ることができるかどうかを示します。

- Write (書き込み)

  ディレクトリデータを変更または作成できるかどうかを示します。この権限でディレクトリデータを削除することもできますが、エントリ自体を削除することはできません。エントリ全体を削除するには、削除権限を持っている必要があります。

- Search (検索)

  ディレクトリデータを検索できるかどうかを示します。読み取り権限では、検索操作の一部としてディレクトリデータが返された場合にそれを閲覧できるという点で、この権限は読み取り権限とは異なります。たとえば、共通名の検索と、部屋番号の読み取りを許可している場合、共通名の検索の一部として部屋番号が返されることはありますが、部屋番号自体は検索できません。これにより、ディレクトリを検索しても特定の部屋にだれがいるのかを知ることはできません。

- Compare (比較)

  比較操作にこのデータを使用できるかどうかを示します。比較には検索機能もありますが、検索操作で実際のディレクトリ情報が返されることはありません。代わりに、比較した値がマッチしたかを示す簡単なブール値だけが返されます。これは、ディレクトリの認証中に `userPassword` 属性値を照合するために使用されます。

- Selfwrite (本人による書き込み)

  グループ管理用だけに使用されます。この権限を使用すると、自分をあるグループに追加したり、削除することができます。Selfwrite (本人による書き込み) は、プロキシ承認で使用できます。これは、グループエントリに対して、バインドしたユーザーの DN ではなく、プロキシ DN の追加や削除を行うための権限を与えます。

- Add (追加)

  子エントリを作成できるかどうかを示します。この権限を使用すると、対象のエントリの下に子エントリを作成できます。

- Delete (削除)

  エントリを削除できるかどうかを示します。この権限を使用すると、対象のエントリを削除できます。

- Proxy (プロキシ承認)

  Directory Manager DN 以外の DN を使用して、その DN の権限でユーザーがディレクトリにアクセスできることを示します。

バインド規則

バインド規則は通常、権限が適用されるバインド DN を示します。また、時刻や IP アドレスなどのバインド属性を指定することもできます。

バインド規則を使用すると、ユーザー自身が所有しているエントリだけに ACI が適用されることを簡単に表すことができます。これにより、ユーザーが自分以外のエントリを更新することを防ぎ、各ユーザーが自分のエントリだけを更新できるように設定できます。

バインド規則を使用すると、次の場合に ACI が適用可能であることを示すことができます。

- バインド操作が特定の IP アドレスや DNS ホスト名から行われている場合のみ。これは、ディレクトリへのすべての更新が、特定のマシンまたはネットワークドメインから行われるように強制する場合によく使用される
- ユーザーが匿名でバインドした場合。匿名バインドの権限を設定すると、ディレクトリにバインドするすべてのユーザーにその権限が適用されることになる
- ディレクトリに正常にバインドしたすべてのユーザーについて。これは、匿名アクセスを防ぐ一方、一般的なアクセスを許可する
- クライアントがそのエントリの直接の親としてバインドした場合のみ
- ユーザーがバインドに使用したエントリが、特定の LDAP 検索条件を満たす場合

これらのタイプのアクセスをより簡単に表現するために、次のキーワードを使用できます。

- Parent (親)

  バインド DN が直接の親エントリの場合、バインド規則は true です。たとえば、これにより、あるディレクトリの分岐点ですぐ下の子エントリを管理できるという特定の権限を許可できます。

- Self (本人)

バインド DN がアクセスを要求しているエントリと同じ場合、バインド規則は true です。たとえば、個々のユーザーに自分のエントリを更新できるという特定の権限を許可できます。

- All (すべて):

  ディレクトリに正常にバインドしたすべてのユーザーについて、バインド規則は true です。

- Anyone (全員)

  すべてのユーザーについて、バインド規則は true です。このキーワードによって、匿名アクセスを許可または拒否します。

デフォルト ACI

アクセス制御の設計という観点に立つと、`userRoot` データベースに格納したディレクトリ情報にどのデフォルト ACI が適用されるのかを理解することが重要です。これをどのように変更すれば、配備の要件に合わせることができるのかを決定できます。ディレクトリに新しいデータベースを作成すると、最初のエントリには必ず次のデフォルト ACI が置かれます。

- ユーザーは、ディレクトリ内にある個人のエントリを変更できるが、削除はできない。`aci` 属性と `nsroledn` 属性は変更できない
- ユーザーは、ディレクトリに匿名でアクセスして、検索、比較、および読み込み操作を行うことができる
- 管理者 (デフォルトでは `uid=admin,ou=Administrators, ou=TopologyManagement,o=NetscapeRoot`) には、プロキシ権限以外のすべての権限が与えられる
- 設定管理者グループのすべてのメンバーには、プロキシ権限以外のすべての権限が与えられる
- ディレクトリ管理者グループのすべてのメンバーには、プロキシ権限以外のすべての権限が与えられる
- SIE グループ

`o=NetscapeRoot` サブツリーには、専用のデフォルト ACI が置かれます。

- 設定管理者グループのすべてのメンバーには、プロキシ権限以外のすべての `o=NetscapeRoot` サブツリーでの権限が与えられる
- ユーザーは `o=NetscapeRoot` サブツリーに匿名でアクセスして、検索、比較、および読み込み操作を行うことができる

- o=NetscapeRoot の下のエントリは、uniqueMember 属性を使用して、そのエントリに対する読み取り、検索、および比較のアクセス権がどのグループに与えられるかを定義できる
- 認証されたすべてのユーザーには、管理サーバーを識別する設定属性に対する検索、比較、および書き込みの権限が与えられる

これらのデフォルト設定の変更方法とアクセス制御ポリシーの実装方法については、『Sun ONE Directory Server 管理ガイド』を参照してください。

権限の設定方法の決定

ディレクトリに ACI が存在しない場合のデフォルトポリシーは、ユーザーにいかなるアクセス権も与えません。Directory Manager は例外です。このため、ユーザーがディレクトリにアクセスできるようにするには、Directory Server にいくつかの ACI を設定する必要があります。次に、Directory Server に用意されているアクセス制御のメカニズムについて説明します。ACI の設定方法については、『Sun ONE Directory Server 管理ガイド』の「Managing Access Control」の章を参照してください。

優先規則

ユーザーがディレクトリエントリに対してどのようなタイプのアクセスを試みた場合でも、Directory Server はディレクトリ内のアクセス制御セットを検査します。アクセスを確定するために、Directory Server は優先規則を適用します。この規則は、2 つの競合する権限が存在する場合、アクセス拒否の権限がアクセス許可の権限よりも常に優先されることを規定しています。

たとえば、ディレクトリのルートレベルで書き込み権限を拒否して、Directory Server にアクセスするすべてのユーザーにこの権限を適用した場合、ユーザーに許可されているほかの権限に関係なく、だれもディレクトリに書き込むことはできません。特定のユーザーに Directory Server への書き込み権限を許可するには、元の書き込み拒否の適用範囲を限定してそのユーザーを除外しておく必要があります。また、対象となるユーザーに対して書き込みを許可する権限を作成し追加しておく必要があります。

アクセスの許可または拒否

ディレクトリツリーへのアクセスは、明示的に許可または拒否できます。Directory Server へのアクセスを明示的に拒否する場合は、慎重に行ってください。優先規則が適用されるため、明示的にアクセスを禁止する規則がディレクトリにある場合、アクセスを許可する権限の有無にかかわらず、アクセスは拒否されることになります。

アクセスを許可する規則の適用範囲を限定して、最低限のユーザーまたはクライアントアプリケーションだけが含まれるようにしてください。たとえば、ユーザーのディレクトリエントリにあるすべての属性への書き込み権限を許可し、ディレクトリ管理者を除くすべてのユーザーに uid 属性への書き込み権限を拒否するように権限を設定できます。あるいは、次の方法で書き込み権限を許可する 2 つのアクセス規則を作成するという方法もあります。

- uid 属性を除くすべての属性への書き込み特権を許可する規則を 1 つ作成する。この規則はすべてのユーザーに適用する必要がある
- uid 属性への書き込み特権を許可する規則を 1 つ作成する。この規則は、ディレクトリ管理者グループのメンバーだけに適用する必要がある

許可特権だけを作成することにより、明示的な拒否特権を設定する必要はなくなります。

アクセスを拒否する場合

明示的な拒否を設定する必要は、ほとんどありません。ただし、次のような状況では明示的な拒否が有効である場合もあります。

- 複雑な ACL が全体的に適用されている大きなディレクトリツリーがある場合

  セキュリティ上の理由から、特定ユーザー、特定グループ、または物理的な場所へのアクセスを拒否する必要性が突然生じる場合があります。許可の権限の適切な制限方法を把握するために既存の ACL を詳しく調べるよりも、検討の余裕ができるまでの間、明示拒否を一時的に設定することができます。ACL がこのように複雑になってしまった場合、拒否の ACI 設定は、長期的には管理の負担を増大させるだけです。できるだけ早期に ACI を修正して、明示的な拒否を取り除きアクセス制御スキーマ全体を簡略化します。

- 曜日または時間帯に基づいてアクセス制御を限定する場合

  たとえば、日曜日の午後 11:00 (2300) から月曜日の午前 1:00 (0100) まで、すべての書き込み操作を禁止します。管理上の観点からは、ディレクトリを検索してすべての ACI 書き込み権限を特定し、この時間帯における許可範囲を制限するよりも、この種の時間に基づいたアクセスを明示的に制限する ACI を管理した方が簡単な場合があります。

- ディレクトリの管理権限を複数の管理者に与えるときに、特権を制限する場合

  個人またはグループのメンバーにディレクトリツリーの管理を部分的に許可し、その上でそのツリーの一部を変更しないようにする場合は、明示的な拒否を使用します。たとえば、メール管理者に共通名属性への書き込み権限を許可しないようにする場合、共通名属性への書き込み権限を明示的に拒否する ACI を設定します。

アクセス制御規則の格納場所

アクセス制御規則は、ディレクトリの任意のエントリに置くことができます。多くの場合、管理者は country、organization、organizationalUnit、inetOrgPerson、または group のタイプのエントリにアクセス制御規則を置きます。

ACL の管理を簡単にするために、この規則はできるだけグループ化します。通常、規則はターゲットエントリとそのエントリのすべての子に適用されるため、最下位にある個々のエントリ (ユーザーなど) にアクセス制御規則を分散させるよりも、ディレクトリのルートポイントまたは分岐点にアクセス制御規則を配置すると良いでしょう。

注

パフォーマンスを考えると、ACI はメモリ内に維持されるため、メモリ消費のパフォーマンスに大きな影響が生じることに注意する必要があります。サーバーを起動したとき、すべてのアクセス制御情報は、キャッシュの制限を受けることなくメモリに読み込まれます。このため、繰り返しのディレクトリツリー構造を持つ企業では、可能であればマクロ ACI を使用して、Directory Server で使用される ACI の数を最適化することをお勧めします。

マクロは、ACI の中で DN、または DN の一部を表現するために使用される可変部分です。マクロを使用すると、ACI のターゲット部分またはバインド規則部分、あるいはその両方の DN を表すことができます。実際の処理では、Directory Server が LDAP 操作を受け取ると、LDAP 操作のターゲットとなるリソースに対して ACI マクロの一致が比較されます。一致した場合、マクロは対象となるリソースの DN の値に置き換えられます。次に Directory Server は ACI を通常どおりに評価します。マクロ ACI については、『Sun ONE Directory Server 管理ガイド』の「Managing Access Control」の章を参照してください。

フィルタが適用されたアクセス制御規則の使用

Directory Server の ACI モデルの強力な機能の 1 つに、LDAP 検索フィルタを使用してアクセス制御を設定できるという機能があります。LDAP 検索フィルタでは、定義した条件に一致するすべてのディレクトリエントリへのアクセス権限を設定できます。

たとえば、マーケティングに設定されている organizationalUnit 属性を含むすべてのエントリへの読み取り権限を許可できます。

フィルタが適用されたアクセス制御規則では、事前にアクセスレベルを定義することもできます。たとえば、ディレクトリに自宅の住所と電話番号に関する情報が含まれているとします。これを公開したいと考える人と、リストからの情報の削除を求める人がいます。この情報は、次のように解決できます。

- 各ユーザーのディレクトリエントリに publishHomeContactInfo という属性を作成する

- publishHomeContactInfo 属性が TRUE (有効を意味する) に設定されているエントリだけに、homePhone と homePostalAddress 属性への読み取り権限を許可するアクセス制御規則を設定する。LDAP 検索フィルタを使用して、この規則のターゲットを示す
- ディレクトリユーザーが自分の publishHomeContactInfo 属性の値を TRUE または FALSE に変更できるようにする。このようにすると、この情報を公開するかどうかをディレクトリユーザーが決定できる

LDAP 検索フィルタの使用方法と、ACI での LDAP 検索フィルタの使用については、『Sun ONE Directory Server 管理ガイド』を参照してください。

実効権限に関する情報の取得

Directory Server が提供する機能が豊富なアクセス制御モデルは強力で、さまざまなメカニズムを使用してユーザーにアクセス権を与えることができます。しかし、この豊富な機能は、アクセス制御ポリシーを設定する際はとても有用ですが、後日、そのポリシーが実際にどのように機能するのかを調べる作業は面倒なことになりかねません。たとえば、IP アドレスとマシン名、時刻、認証方法、追加する属性の種類など、ユーザーのセキュリティは、いくつかのパラメータを使って定義することができます。このため、エントリと属性の権限をリスト化する機能が必要になります。ディレクトリエントリと属性に対する特定ユーザーの権限情報を取得する機能がなければ、ユーザーの管理、アクセス制御ポリシーの検証、デバッグが一層困難になります。

Sun ONE Directory Server 5.2 には、クライアントがディレクトリエントリおよび属性に対してどのようなアクセス制御権を持つのかを照会できるように、実効権限という機能が新たに追加されました。

注

実効権限機能によって得られる実効権限情報は次の項目に対応する情報です。

- 照会時に有効な ACI
- 適用される認証方法
- 照会元ホストマシンの名前とアドレス

特定のユーザーが特定のデータに対するアクセス権を持つ、または持たない理由を特定するには、実効権限を照会するシステム管理者などのユーザーが、実効権限の検索を開始するときにすべてのユーザーのパラメータを理解している必要があります。

ここでは、実効権限機能の詳細について説明します。説明する内容は次のとおりです。

- 実行権限機能について

- 実効権限機能に対するアクセス制御
- 実効権限検索結果の内容

実行権限機能について

実効権限機能は、ldapsearch 操作によって実効されます。この機能を実行するには、DSRK (Directory Server リソースキット) に含まれる ldapsearch ユーティリティが必要です。

照会する権限情報の指定には、次に説明する特別なオプションを使用します。権限に関する情報は、ldapsearch の結果として返されます。照会する権限の情報を指定するには、ldapsearch 操作に次の要素を指定します。

- 照会する実効権限情報を指定する実行権限制御。ldapsearch 操作では、-J オプションを使用してこの制御を指定する
- エントリレベルと属性レベルの両方でアクセス制御実効権限情報を取得するには、aclRights 操作属性を照会する
- 権限照会の対象となるユーザーを指定するには、ldapsearch の -c オプションを使用する
- 権限情報に属性タイプのリストが含まれない場合、その情報を取得するには、ldapsearch の -X オプションを使用する

注 実効権限制御の指定が必要になるのは、ldapsearch 操作に -c オプションまたは -X オプションを指定していない場合だけです。これらのオプションを指定した場合は、検索対象の制御が自動的に追加されます。

また、実効権限制御に NULL 値が含まれる場合、Directory Server は、対象ユーザーの権限と、その ldapsearch 操作によって返される属性とエントリの権限の両方が要求されているものと解釈します。

実効権限機能を使用するときに、最初に決定しなければならないことは、どのユーザーに関する有効アクセス制御権を調べるのか、ということです。どのユーザーの権限を照会するのかが明らかになれば、ldapsearch の -c オプションでそのユーザーを指定できます。次のように指定した場合は、

```
-c "dn:"
```

匿名ユーザーの実効権限の照会を ldapsearch に指定したことになります。-c "" のように -c オプションに NULL (空の文字列) を指定した場合は、バインドされているユーザーの権限が照会されます。

次に決定しなければならないことは、どの属性の実効権限を照会するのか、ということです。実効権限の照会対象となる属性は、エントリ内に存在している必要はありません。たとえば、エントリには description 属性は存在しないが、この属性に対して特定のユーザーがどのような権限を持っているのかを調べる場合などには、このような照会が行われます。権限情報の照会対象となる属性タイプのリストを指定するには、-X description のように、-X ldapsearch オプションと属性名を続けて指定します。

注 時刻、認証方法、マシンのアドレスと名前など、アクセス制御に影響するその他のパラメータについては、この機能は検索を行うユーザーからこれらのパラメータを継承します。

Directory Server の実効権限機能では、アクセス制御情報のさまざまなレベルを指定することもできます。権限情報だけ、ログ情報だけ、または両方の取得を選択できます。ログ情報 (権限情報の概要が含まれます) には、許可レベルまでの詳細が記録されているため、どのアクセス制御ポリシーが有効であるかを調べるだけでなく、特定の操作がなぜ拒否または許可されたのかを理解することもできます。照会する情報の指定に使用する aclRights オペレーショナル属性のサブタイプは、次のとおりです。

aclRights	権限情報だけを照会する
aclRightsInfo	ログ情報だけを照会する
aclRights aclRightsInfo	権限情報とログ情報の両方を照会する

警告 検索フィルタでは、aclRights 属性を使用することはできません。

実効権限機能に対するアクセス制御

実効権限情報を正しく取得するには、実効権限制御を使用するためのアクセス制御権が必要です。また、特定エントリのユーザーに権限情報を返すには、aclRights 属性に対する読み取りアクセス権も必要です。実効権限機能に対する基本的な保護は、この二重のアクセス制御によって得られ、これを必要に応じてさらに調整することができます。プロキシ承認のように、aclRights に対する読み取りアクセス権があれば、エントリと属性に対するどのユーザーの権限でも調べることができます。このアクセス制御設定は、リソース管理ユーザーが、実際にはリソースに対する権限を管理しないとしても、どのユーザーがリソースに対して権限を持つのかを調べるのは当然である、という考え方に基づいています。

権限情報を照会するユーザーが実効権限制御を使用する権限を持たない場合、操作は失敗し、エラーメッセージが送信されます。一方、権限情報を照会するユーザーがこの制御を使用する権限は持つが、aclRights 属性に対する読み取りアクセス権を持たない場合は、返されるエントリから aclRights 属性が省略されるだけです。この動作は、Directory Server の通常の検索動作を反映しています。

注 実効権限制御に基づく ldapsearch 操作にプロキシ制御が関わっている場合、実行権限の照会操作はプロキシユーザーとして承認されます。つまり、実効権限制御を使用して権限を照会するのはプロキシユーザーであり、返されるエントリは、そのプロキシユーザーが検索と表示の権限を持つエントリとなります。

実効権限検索結果の内容

ここでは、実効権限の検索結果について説明します。説明する内容は次のとおりです。

- 権限情報
- Write、Selfwrite_add、Selfwrite_delete 権限
- ログ情報

権限情報

実行権限情報は、次のサブタイプに基づいて提供されます。

aclRights;entrylevel	エントリレベルの権限情報を提供する
aclRights;attributelevel	属性レベルの権限情報を提供する
aclRightsInfo;entrylevel	エントリレベルのログ情報を提供する
aclRightsInfo;attributelevel	属性レベルのログ情報を提供する

このレベルまで提供された情報は、権限と属性の名前によってタイプ別に分けられます。

Sun ONE Directory Server 5.2 では、aclRights 文字列は次のように表現されます。

*permission:value(permission:value)**

エントリレベルの権限には、add、delete、read、write、proxy があり、属性レベルの権限には read、search、compare、write、selfwrite_add、selfwrite_delete、proxy があります。

警告 Directory Server の将来のリリースでは、この文字列に新しい権限が追加される可能性があります。

照会した権限情報の結果を含む aclRights 文字列では、権限に次のマークが付けられます。

- 0 : 権限が与えられていない
- 1 : 権限が与えられている
- ? : 追加、削除、置換する属性の値に応じて権限が与えられる。? が表示される場合は、ログ情報を調べ、権限が与えられる、または与えられない具体的な理由を確認する
- - : 属性は仮想属性であり、更新することはできない。仮想属性を変更する唯一の方法は、それを生成するメカニズムを変更することである

例として、cn=peopletestResource サブツリーのユーザー cn=justread が cn 属性に対して持つ実効権限 (ログ情報は含まれません) を照会すると仮定します。次の ldapsearch コマンドを実行します。

```
./ldapsearch -D "cn=directory manager" -w secret12
-c "dn:cn=justread,dc=france,dc=sun,dc=com" -p 5200
-b cn=peopletestResource,dc=france,dc=sun,dc=com
objectclass=* cn "aclRights"
```

ユーザー cn=justread が cn 属性に対して読み取りアクセス権だけを持つ場合は、照会した実効権限情報は、次のように出力されます。

```
dn: cn=peopleTestResource,dc=france,dc=sun,dc=com
aclRights;attributeLevel;cn:search:0,read:1,compare:0,write:0,
selfwrite_add:0,selfwrite_delete:0,proxy:0
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
```

Write、Selfwrite_add、Selfwrite_delete 権限

Sun ONE Directory Server 5.2 では、書き込み属性レベルの権限だけが「?」とマークされます (将来のリリースで変更される可能性もあります)。追加と削除の権限では、追加、削除できるエントリは、そのエントリの属性値によって異なります。しかし、ldapsearch の実行結果としては、? と表現しなければならない場合を除き、権限の状態 (「0」または「1」) が返されます。

write 権限の値が「1」であれば、承認 DN 値を除くすべての値について、追加と削除の両方の ldapmodify 操作が許可されます。値が「0」であれば、承認 DN の値を除くすべての値について、追加、削除いずれの ldapmodify 操作も許可されません。承認 DN の値に適用される権限は、いずれかの selfwrite 権限によって明示的に返されます。この権限には、selfwrite_add と selfwrite_delete があります。

実効権限機能の属性レベルの権限では、selfwrite-add 属性と selfwrite-delete 属性を明確に区別する必要があります。これらの権限は ACI のコンテキストには実際には存在しませんが、現実には ACI を組み合わせることで、追加だけ、または削除だけの変更操作を selfwrite 権限としてユーザーに与えることができます。write 権限にはこの区別は適用されません。これは、変更される属性の値が定義される selfwrite 権限とは異なり、この権限では承認 DN が定義され、変更する属性の値に対する書き込み権限は定義されないためです。実行権限が targattrfilters ACI に依存する場合は、これ以上の分析が不可能であるため、「?」がマークされ、権限の詳細を確認するにはログ情報を参照する必要があることが示されます。write、selfwrite_add、selfwrite_delete 権限の相互依存関係は複雑です。197 ページの表 7-5 は、この 3 つの権限の考えられる組み合わせと、それぞれの意味を示しています。

表 7-5 write、selfwrite_add、selfwrite_delete 権限の相互依存関係と実効権限

write	selfwrite_add	selfwrite_delete	実効権限の説明
0	0	0	この属性のどの値も追加、削除できない
0	0	1	承認 DN 値の削除だけが可能
0	1	0	承認 DN 値の追加だけが可能
0	1	1	承認 DN 値の追加、削除だけが可能
1	0	0	承認 DN 値を除くすべての値の追加、削除が可能
1	0	1	承認 DN 値を含むすべての値の削除と、承認 DN 値を除くすべての値の追加が可能
1	1	0	承認 DN 値を含むすべての値の追加と、承認 DN 値を除くすべての値の削除が可能
1	1	1	この属性のすべての値の追加、削除が可能
?	0	0	承認 DN 値を追加、削除することはできないが、その他の値を追加、削除できる可能性がある。書き込み権限の詳細を確認するには、ログ情報を調べる
?	0	1	承認 DN 値の削除は可能であるが、追加はできない。また、その他の値を追加、削除できる可能性がある。書き込み権限の詳細を確認するには、ログ情報を調べる
?	1	0	承認 DN 値の追加は可能であるが、削除はできない。また、その他の値を追加、削除できる可能性がある。書き込み権限の詳細を確認するには、ログ情報を調べる
?	1	1	承認 DN 値の追加、削除が可能。また、その他の値を追加、削除できる可能性がある。書き込み権限の詳細を確認するには、ログ情報を調べる

ログ情報

実効権限のログ情報は、難しいアクセス制御を理解するヒントとなり、デバッグに役立ちます。ログ情報には、`acl_summary` というアクセス制御の概要文が含まれ、アクセス制御が許可または拒否される理由を確認することができます。アクセス制御の概要文によって示される内容は、次のとおりです。

- アクセスが許可されるか、拒否されるか
- 与えられている権限
- 権限のターゲットエントリ
- ターゲット属性の名前
- 照会した権限のサブジェクト
- 照会がプロキシとして行われたかどうか (行われた場合はプロキシの認証 DN)
- デバッグで最も重要となる、アクセスが許可または拒否された理由。考えられる理由については 198 ページの表 7-6 を参照

表 7-6 実効権限のログ情報に表示される理由とその説明

ログ情報に表示される理由	説明
`no reason available`	アクセスがどのような理由で許可または拒否されたのかは不明である
no allow acis	許可 ACI が存在しないため、アクセス拒否が返された
`result cached deny`	キャッシュされた情報を使用してアクセスの拒否が決定された
`result cached allow`	キャッシュされた情報を使用してアクセスの許可が決定された
`evaluated allow`	ACI の評価によってアクセスの許可が決定された。決定に関与した ACI の名前はログ情報に含まれる
`evaluated deny`	ACI の評価によってアクセスの拒否が決定された。決定に関与した ACI の名前はログ情報に含まれる
`no acis matched the resource`	リソースまたはターゲットと一致する ACI が存在しないため、アクセス拒否が返された
`no acis matched the subject`	アクセス制御を照会するサブジェクトと一致する ACI が存在しないため、アクセス拒否が返された
`allow anyone aci matched anon user`	userdn = `"ldap:///anyone"` サブジェクトが指定された ACI が存在するため、匿名ユーザーによるアクセスが許可された

表 7-6　実効権限のログ情報に表示される理由とその説明 (続き)

ログ情報に表示される理由	説明
no matching anyone aci for anon user	userdn = "ldap:///anyone" サブジェクトが指定された ACI が存在しないため、匿名ユーザーによるアクセスが拒否された
user root	ユーザーがルート DN であるため、アクセスが許可された

注　仮想属性は更新不可能であるため、仮想属性に対する書き込み権限は与えられません。また、それに関連するログ情報も記録されません。

実際のログファイルの形式については、『Sun ONE Directory Server Reference Manual』を参照してください。

ACI の使用に関するヒント

次に示すヒントは、ディレクトリのセキュリティモデルを管理する際の負担を軽減し、ディレクトリのパフォーマンス向上にも役立ちます。

一部のヒントについては、この章ですでに説明しました。リスト化のために、ここに含められています。

- ディレクトリ内の ACI の数を最小化し、可能であればマクロ ACI を使用する

  Directory Server は 50,000 以上の ACI を評価できますが、多数の ACI 文を管理することは困難です。また、ACI はメモリに格納されるため、ACI の数が多すぎると、ACI メモリが不足する可能性があります。ACI の数が多すぎると、特定のクライアントが利用できるディレクトリオブジェクトを即時に判断するのが難しくなります。ディレクトリツリーの ACI の数を減らし、可能であればマクロ ACI を使用することで、アクセス制御ポリシーの管理が容易になるだけでなく、ACI メモリの使用も効率化されます。

- 許可権限と拒否権限のバランスを図る

  デフォルトの規則ではアクセスを与えられていないすべてのユーザーに対してアクセスを拒否しますが、アクセスを許可する 1 つの ACI をツリーのルートの近くで使用し、アクセスを拒否する少数の ACI を最下位のエントリの近くで使用することで、ACI の数を減らすことができます。このようにすると、最下位のエントリの近くでアクセスを許可する複数の ACI を使用する必要がなくなります。

- ACI では最小の属性セットを指定する

これは、オブジェクトの属性の一部でアクセスを許可または制限している場合、その最小の属性リストが、許可される属性セットであるか、拒否される属性セットであるかを判定することを意味します。次に、最小の属性リストを管理するように ACI を表します。

たとえば、ユーザーオブジェクトクラスに多くの属性が含まれている場合を考えます。これらの属性のうち、1 つか 2 つだけをユーザーが更新できるようにする場合、その少数について書き込み権限を許可する ACI を作成します。逆に、1 つか 2 つの属性以外のすべてをユーザーが更新できるようにする場合は、それらの特定の属性を除くすべてについて書き込み権限を許可する ACI を作成します。

- LDAP 検索フィルタは慎重に使用する

  検索フィルタではアクセスを管理するオブジェクト名が直接指定されないため、特にディレクトリが複雑になる場合などは、検索フィルタを使用すると、予期しない状況が発生することがあります。ACI の中で検索フィルタを使用する場合、同じフィルタを使用して `ldapsearch` 操作を実行し、変更の結果がディレクトリに及ぼす影響を確認します。

- ディレクトリツリーの別の部分で ACI を重複させない

  ACI が重複しないよう注意してください。たとえば、あるグループに `commonName` および `givenName` 属性への書き込み権限を許可する ACI がディレクトリのルートポイントにあり、同じグループに `commonName` 属性だけへの書き込み権限を許可する別の ACI がある場合は、ACI を作り直して 1 つの制御でそのグループに書き込み権限を許可するようにしてください。

  ディレクトリが複雑になるにつれ、偶発的に ACI が重複する事態が発生しやすくなります。ACI の重複を避けることにより、セキュリティ管理が容易になる上、ディレクトリに含まれる ACI の総数も減少します。

- ACI に名前を付ける

  ACI に名前を付けるかどうかは任意ですが、各 ACI にわかりやすい短い名前を付けておくと、特に Directory Server コンソールから ACI を検査するときなど、セキュリティ モデルの管理に役立ちます。

- ユーザーエントリに標準属性を使用して、アクセス権限を決める

  可能な限り、すでに標準ユーザーエントリの一部となっている情報を使用して、アクセス権限を決めてください。特別な属性を作成する必要がある場合は、ロールまたはサービスクラス (CoS) の定義の一部として作成することを検討します。ロールと CoS については、70 ページの「ディレクトリエントリのグループ化と属性の管理」を参照してください。

- ディレクトリ内のできるだけ近い場所に ACI をグループ化する

  ACI の配置をディレクトリのルートポイントとディレクトリの主な分岐点に限定します。ACI をグループ化すると、ACI を総合したリストの管理に役立つだけでなく、ディレクトリ内の ACI の総数を最小限に留めるのにも役立ちます。

- 二重否定は使用しないようにする (バインド DN が cn=Joe と等しくない場合の書き込みの拒否など)

  サーバーはこの構文を完全に理解できますが、管理者にとっては混乱の元となります。

ACI の制限事項

ディレクトリサービスに対するアクセス制御ポリシーを決定するときは、次の制限事項に注意してください。

- ディレクトリツリーが連鎖機能によって複数のサーバー上に分散されている場合は、アクセス制御文で使用できるキーワードにいくつかの制約が適用されます。
  - グループエントリ (groupdn キーワード) に依存する ACI は、グループエントリと同じサーバー上に置く必要がある。ダイナミックグループである場合は、そのメンバーすべても同じサーバー上にエントリを持つ必要がある。スタティックグループである場合は、リモートサーバー上にメンバーのエントリを置くことができる
  - ロール定義 (roledn キーワード) に依存する ACI は、ロール定義エントリと同じサーバー上に置く必要がある。ロールを持たせる予定のエントリも、すべて同じサーバー上に置く必要がある

  ただし、ターゲットエントリに格納された値と、バインドユーザーのエントリに格納された値のマッチングは可能です (userattr キーワードなどを使用)。ACI を持つサーバー上にバインドユーザーがエントリを持っていない場合も、通常どおりにアクセスに対する評価が行われます。

  アクセス制御を連鎖させる方法については、『Sun ONE Directory Server 管理ガイド』の「Database Links and Access Control Evalutation」を参照してください。
- CoS によって作成された属性を、すべての ACI キーワードで使用できるとは限りません。特に、userattr キーワードによって CoS で作成した属性を使用しないでください。アクセス制御規則が機能しなくなります。このキーワードについては、第 4 章「ディレクトリツリーの設計」を参照。
- アクセス制御規則の評価は、常にローカルサーバー上で行われるます。したがって、ACI キーワードで使用される LDAP URL で、サーバーのホスト名やポート番号を指定する必要はありません。指定しても、LDAP URL は無視されます。LDAP URL の詳細については、『Sun ONE Directory Server Reference Manual』の「LDAP URL」を参照してください。
- サーバーが使用するメモリが利用可能な物理メモリに収めるためのキャッシュ設定は ACI キャッシュには適用されません。つまり、ACI の数が多すぎると、利用可能メモリが不足する可能性があります。

プロキシ権限を与える場合、ユーザーに Directory Manager となるプロキシ権限を与えたり、Directory Manager にプロキシ権限を与えたりすることはできません。

SSL による接続のセキュリティ保護

特定ユーザーの認証スキーマと、情報を保護するためのアクセス制御スキーマを設計した後は、サーバーとクライアントアプリケーションの間のネットワーク上でやり取りされる情報の完全性を保護する必要があります。

ネットワーク上で安全な通信を可能にするために、SSL (Secure Sockets Layer) 上で LDAP プロトコルと、DSML over HTTP の両方を使用することができます。SSL を設定して有効化すると、クライアントはセキュリティ保護された専用ポートに接続します。このポートとの SSL 接続が確立されると、すべての通信内容が暗号化されます。Directory Server は、Start TLS (Start Transport Layer Security) 制御もサポートしています。この制御を使用することで、クライアントは標準の LDAP ポート上での暗号化通信を開始できます。

それぞれのポートを独立させることで、Directory Server は SSL によるセキュリティ保護された接続と、SSL が適用されていない接続を同時にサポートします。

SSL は暗号化によって機密情報を保護し、チェックサムのハッシュによってデータの完全性を確保します。SSL 接続を確立すると、クライアントアプリケーションと Directory Server は、両者が設定内に共通して持つ最も強力な暗号化アルゴリズムを選択します。このアルゴリズムを暗号化方式と呼びます。Directory Server で使用できる暗号化方式は次のとおりです。

- DES : 56 ビットのブロック暗号化方式
- 3DES (「トリプル DES」) : 156 ビットのブロック暗号化方式
- RC2 : 128 ビットのブロック暗号化方式 (または 40 ビットのエクスポート暗号化方式)
- RC4 : 128 ビットのストリーム暗号化方式 (または 40 ビットのエクスポート暗号化方式)

暗号化方式は、次のいずれかのハッシュアルゴリズムと組み合わせて使用できます。

- MD5
- SHA-1

暗号化方式とハッシュアルゴリズムについては、213 ページの「その他のセキュリティ関連資料」を参照してください。

通信の暗号化が確立されると、SSL プロトコルによってサーバーはクライアントに証明書を送信します。公開鍵暗号化方式を使用して、クライアントは、それがクライアントが信頼する認証局によって発行されたものであるかどうかを検証し、証明書の確実性を判断することができます。証明書の検証によって、クライアントは、第三者への成りすましによる介入者攻撃を防止することができます。

これで接続のセキュリティ保護は完了し、サーバーはクライアントに認証されます。さらに、クライアントからの認証を要求するようにサーバーを設定することもできます。Directory Server は、証明書ベースと SASL ベースのクライアント認証をサポートしています。これらのメカニズムについては、164 ページの「適切な認証方法の選択」を参照してください。サーバーへのクライアント認証により、クライアントとサーバーとの間の通信が第三者によって傍受または妨害される可能性がなくなり、最大レベルのセキュリティが提供されます。

証明書ベースの認証を使用する SSL プロトコル通信のパフォーマンスを向上できるように、Directory Server 5.2 は Sun Crypto アクセラレータボードをサポートしています。このボードは、SSL キー関連の計算を高速化します。クライアントが SSL 経由のバインド、検索、バインド解除を何度も繰り返す配備で役立つかもしれません。キー関連の計算がパフォーマンスのボトルネックとならない場合、SSL アクセラレータボードを使用しても Directory Server のパフォーマンスが向上しない可能性があります。SSL アクセラレータボードが最も効果的なのは、クライアントが別のマシンから通信を確立する場合です。システムが SSL ベースの複数の接続を確立する場合、SSL キャッシングセッションによって RSA 操作の数が限定される可能性が高くなります。この場合、アクセラレータボードを利用するメリットも限定されてしまいます。Sun Crypto アクセラレータボードのインストール方法と設定方法については、『Sun ONE Directory Server インストールおよびチューニングガイド』の付録 B 「Using a Sun Crypto Accelerator Board」を参照してください

Directory Server およびクライアントでの SSL の設定と有効化については、『Sun ONE Directory Server 管理ガイド』の第 11 章「Implementing Security」を参照してください。

属性の暗号化

ここでは、Sun ONE Directory Server 5.2 の新機能である属性暗号化について説明します。説明する内容は次のとおりです。

- 属性暗号化とは
- 属性暗号化の制限
- 属性の暗号化とパフォーマンス
- 属性暗号化の使用に関する注意点

属性暗号化とは

Directory Server では、簡易パスワード認証、証明書ベースの認証、SSL、プロキシ承認など、アクセスレベル (ディレクトリへの読み取りおよび書き込みアクセス時) でデータを保護するためのさまざまな機能が用意されています。しかし、データベースファイル、バックアップファイル、ldif ファイルに記録されているデータの保護が必要になることも少なくありません。ディレクトリ内に 4 桁の暗証番号を記録している銀行を考えてみてください。保護されていないデータベースファイルにダンプが行われる場合、未認証のユーザーがこの機密情報にアクセスすることも考えられます。属性を暗号化する新機能を使用することで、格納されている機密情報にユーザーがアクセスできないように保護できます。

属性暗号化では、どの属性を暗号化形式で格納するかを指定できます。これは、データベースレベルで設定されます。つまり、属性の暗号化を決定すると、データベース内のすべてのエントリでその属性は暗号化されます。属性の暗号化はエントリレベルではなく属性レベルで行われるため、エントリ全体を復号化するには、すべての属性を復号化する必要があります。

注

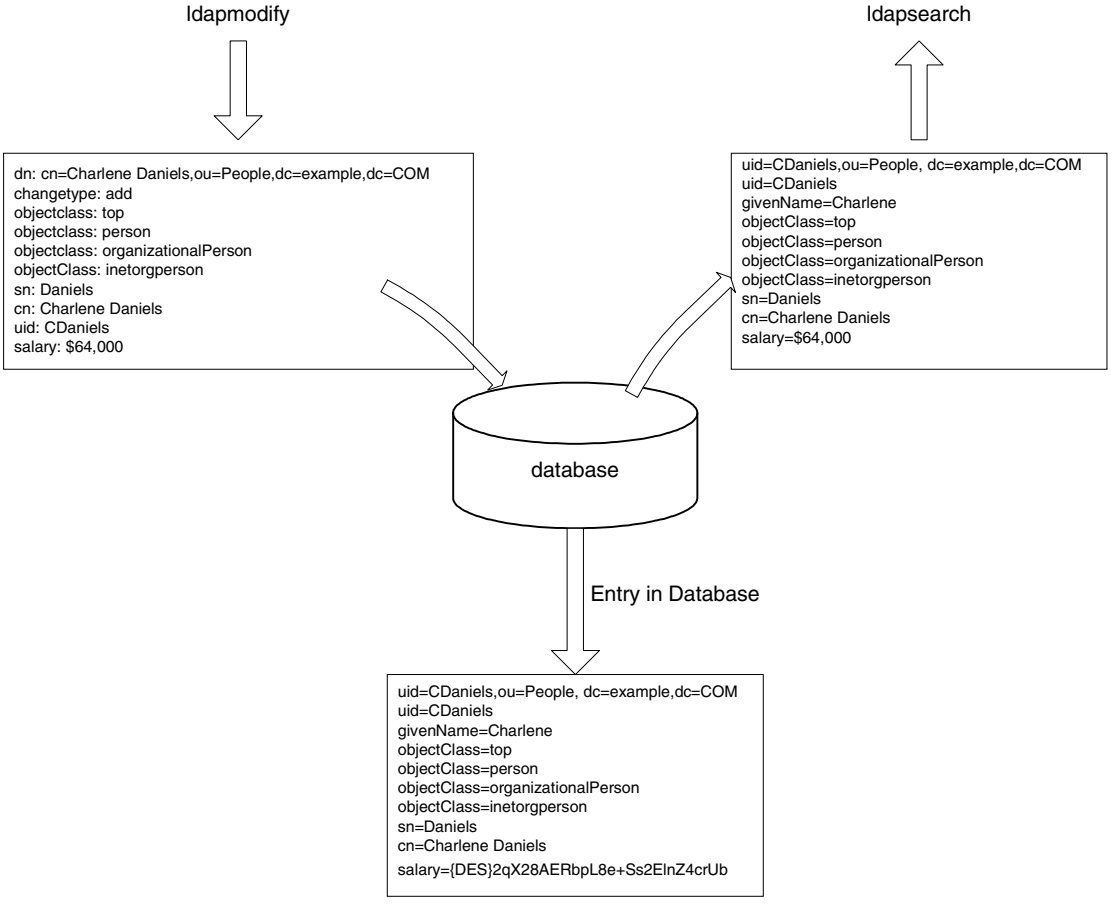
1. 属性暗号化は `userPassword` 属性の暗号化をサポートしていますが、ユーザーパスワードの保護を目的にこの機能を利用することはお勧めできません。`userPassword` 属性に対するアクセス制御が十分に保護されていることを確認してください。また、パスワード長、有効性、期限切れの警告、検査、履歴、リセットなどのオプションによるセキュリティ上の利点を得られるように、`userPassword` 属性に適切なパスワードポリシーを割り当てる必要があります。
2. RDN 属性の暗号化を指定しても、DN は暗号化されません。属性暗号化は DN には適用されません。

暗号化された上で格納されている属性には、適用された暗号化アルゴリズムを示す暗号化方式タグが最初につけられます。DES 暗号化アルゴリズムを使用して暗号化された属性は、次のように表示されます。

```
{CKM_DES_CBC}3hakc&jla+=snda%
```

属性暗号化機能を使用することで、保管中のデータを保護できるだけでなく、データを別のデータベースに暗号化された状態でエクスポートすることができます。ただし、属性を暗号化する目的は、保管中またはエクスポート中の機密データを保護することであり、暗号化は常に可逆的です。このため、検索要求の結果として返された場合は、暗号化された属性は復号化されます。206 ページの図 7-1 は、データベースに追加されるユーザーエントリを示しています。ここで設定されている属性暗号化は、`salary` 属性を暗号化します。

図 7-1　属性暗号化のロジック



属性暗号化の制限

Directory Server の属性暗号化機能は、さまざまな暗号化アルゴリズムをサポートしているため、異なるプラットフォーム間での可搬性が確保されます。必要な暗号化アルゴリズムを指定するには、`dsEncryptionAlgorithm` 属性を使用します。属性暗号化の設定属性については、『Sun ONE Directory Server Reference Manual』を参照してください。

追加のセキュリティ対策として、属性暗号化はサーバーの SSL 証明書の秘密鍵を使用して専用の鍵を作成します。この鍵は、暗号化と復号化を行うときに使用されます。つまり、属性を暗号化するには、サーバーが SSL 経由で実行されている必要があります。SSL 証明書とその秘密鍵はデータベース内にセキュリティ保護された状態で格納されており、パスワードによって保護されています。サーバーへの認証では、この鍵データベースのパスワードが必要となります。つまり、この鍵データベースのパスワードにアクセスできるユーザーであれば、誰もが暗号化されたデータのエクスポートを認証されます。

データを暗号化するためにオンラインでインポートする場合、サーバーへの認証に使用される鍵データベースのパスワードはすでに指定されているため、2 回目には要求されなくなります。データをオフラインでインポートする場合、インポートするデータを暗号化するたびに Directory Server はパスワードを要求します。より機密性の高い操作であるデータの復号化では、エクスポートをオンライン、オフラインのどちらで行うかに関係なく、Directory Server は常に鍵データベースのパスワードを要求します。これにより、セキュリティはさらに高まります。

注 証明書が変更されない限り、サーバーは同じ鍵を生成し続けます。このため、1 つのサーバーインスタンスから別のサーバーインスタンス (両者が同じ証明書を使用する場合) データを転送 (エクスポート後にインポート) することができます。

属性の暗号化とパフォーマンス

属性を暗号化することでデータのセキュリティは向上しますが、パフォーマンスに特定の影響が生じます。どの属性を暗号化するかを決定するときは、これを念頭に慎重に行い、特に機密性が高いと考えられる属性だけを暗号化してください。

インデックスファイルから機密性の高いデータに直接アクセスすることができるので、属性を完全に保護するために、暗号化された属性に対応するインデックスキーを暗号化する必要があります。インデックス付け自体がすでに Directory Server のパフォーマンスに影響しているため (インデックスキーの暗号化による負荷は含まれません)、データをインポートする、またはデータベースに初めて追加する前に属性暗号化を設定してください。こうすることで、暗号化された属性には最初からインデックスが付けられます。

属性暗号化の使用に関する注意点

次に、属性暗号化機能を使用するときに考慮すべき事項について簡単に説明します。

- 一般に、属性暗号化の設定を変更するときは、データをエクスポートし、変更を加えた上で、新たに設定されたデータをインポートすることをお勧めします。

  このような方法で行うことで、機能が喪失することなく、すべての設定変更が全体として適用されます。このような方法で行わなかった場合、一部の機能が喪失し、データのセキュリティが危険にさらされる可能性があります。

- 既存のデータベースに対する属性暗号化の設定を変更すると、パフォーマンスに大きく影響することがあります。

  たとえば、既存のデータが格納されたデータベースインスタンスについて考えてみましょう。このデータベースには、機密性の高い `mySensitiveAttribute` という属性を持つエントリが格納されています。この属性の値は、クリアテキストとしてデータベースとインデックスファイルに格納されています。`mySensitiveAttribute` 属性の暗号化を後から決定した場合、属性暗号化の設定を適用するためにサーバーがデータベースとインデックスファイルを更新する必要があり、データベースインスタンス内のすべてのデータはエクスポートされ、データベースに再びインポートされます。これにより、パフォーマンスに大きな影響が生じます。これは、`mySensitiveAttribute` 属性を最初から暗号化していた場合には回避できる影響です。

- 復号化された形式でデータをエクスポートするときに、誤ったパスワードを指定するとエクスポートが拒否されます。

  セキュリティ対策として、復号化された形式でデータをエクスポートするユーザーに対してサーバーはパスワードを要求します。ユーザーが誤ったパスワードを指定した場合、復号化されたデータのエクスポートは拒否されます。

- ユーザーは、パスワードを直接指定するだけでなく、SSL パスワードファイルと同じ構文を持つパスワードが記録されたファイルへのパスを指定できます。

  属性暗号化の詳細な手順については、『Sun ONE Directory Server 管理ガイド』の第 2 章「Encrypting Attribute Values」を参照してください。

- アルゴリズムの変更はサポートされていますが、正しく作成されていないインデックス付け機能は失われる可能性があります。

  データの暗号化に使用するアルゴリズムを変更するときに、インデックス付けに関連する機能が喪失しないようにするには、データをエクスポートし、属性暗号化の設定を変更してから、データをインポートし直します。この手順どおりに行わない場合、内部暗号化アルゴリズムに基づいて作成されたインデックスは機能しなくなります。

  暗号化された属性の前には、適用された暗号化アルゴリズムを指定する暗号化方式が付けられ、データのインポートはサーバーの内部処理として行われます。このため、アルゴリズムを変更する前にデータを暗号化された形式でエクスポートできるようにすることで、セキュリティがさらに提供されます。

- サーバーの SSL 証明書を変更すると、暗号化されたデータを復号化できなくなります。

  属性暗号化機能は、専用鍵の生成にサーバーの SSL 証明書を使用し、暗号化と復号化の実行にはこの専用鍵が使用されます。このため、暗号化されたデータを復号するには、この証明書が必要となります。事前にデータを復号化せずにサーバーの SSL 証明書を変更すると、そのデータを復号化できなくなります。復号化された形式でデータをエクスポートし、証明書を変更してからデータをインポートし直すことをお勧めします。

- 暗号化されたデータを伝送する、つまり、サーバーインスタンス間でエクスポートとインポートを行うには、両方のサーバーインスタンスが同じ証明書を使用する必要があります。

エントリの安全なグループ化

ここでは、エントリのグループ化に関するセキュリティ上の問題について説明します。説明する内容は次のとおりです。

- ロールの安全な使い方
- CoS の安全な使い方

ロールの安全な使い方

セキュリティの状況によっては、ロールの使用が適していない場合があります。新しいロールを作成するときは、エントリへのロールの割り当てやエントリからのロールの削除がどの程度簡単にできるかを考慮します。ロールへのユーザーの追加やロールからの削除をユーザー自身が簡単に実行できることが望ましい場合もあります。たとえば、Mountain Biking という名前の同好会のロールがある場合は、興味のあるユーザーが自身を簡単に追加または削除できるようにする必要があります。

ただし、セキュリティの状況によっては、このような開放的なロールが適していない場合があります。たとえば、アカウントの無効化に関するロールがあるとします。デフォルトでは、アカウントの無効化に関するロールには、そのサフィックスに対して定義された ACI が含まれています。アカウントの無効化については、『Sun ONE Directory Server 管理ガイド』の「Inactivating Users and Roles」を参照してください。サーバー管理者は、ロールを作成するときに、ロールへのユーザーの追加やロールからの削除をユーザー自身が実行できるようにするかどうかを決定します。

たとえば、ユーザー A が、管理されているロール MR を持っているとします。さらに、MR ロールが、コマンド行からアカウントの無効化によってロックされたとします。つまり、ユーザー A の `nsAccountLock` 属性は「true」として計算されるので、ユーザー A はサーバーにバインドできません。ただし、ユーザーがバインド済みで、MR ロールに関して現在ロックされているという通知を受けたとします。ユーザーの行為を禁止する ACI がない場合は、ユーザーは、自分のエントリから `nsRoleDN` 属性を削除し、自分でロックを解除できます。

ユーザーが `nsRoleDN` 属性を削除できないようにするには、ACI を適用する必要があります。フィルタを適用したロールを使用する場合、ユーザーが属性を削除してフィルタが適用されたロールを放棄することを防ぐために、フィルタの一部を保護する必要があります。フィルタを適用したロールで使用される属性の追加、削除、変更をユーザーが実行できないようにし、フィルタの属性値が計算によって決定する場合は、フィルタの属性値に影響する可能性のあるすべての属性も保護する必要があります。入れ子のロールは、フィルタを適用したロールと管理されているロールから構成されることがあるため、入れ子のロールを構成する各ロールについても上記注意点を考慮する必要があります。

CoS の安全な使い方

読み取り用のアクセス制御は、エントリの実際の属性と仮想属性の両方に適用されます。サービスクラスメカニズムによって生成された仮想属性は、通常の属性として読み取ることができるので、読み取り保護も同様の方法で指定します。

ただし、CoS 値をセキュリティで保護するには、定義エントリ、テンプレートエントリ、ターゲットエントリなど、使用するすべての情報のソースを保護する必要があります。これは更新処理でも同様です。情報のソースから生成された値を保護するために、各情報のソースに対する書き込みを制御する必要があります。

次に、各 CoS エントリのデータに読み取りおよび書き込み保護を設定する際の一般的な原則について説明します。個々の ACI (Access Control Instruction) の定義手順については、『Sun ONE Directory Server 管理ガイド』の「Managing Access Control」を参照してください。

CoS 定義のエントリの保護

CoS 定義のエントリには、生成された属性の値は含まれません。このエントリは、値を検索するための情報を提供します。CoS 定義のエントリを読み取ると、値を含むテンプレートエントリの検索方法がわかり、このエントリへの書き込みによって、仮想属性の生成方法を変更できます。

したがって、CoS 定義のエントリに読み取りと書き込みの両方のアクセス制御を定義する必要があります。

CoS テンプレートエントリの保護

CoS テンプレートエントリには、生成された CoS 属性の値が含まれます。したがって、少なくともテンプレートの CoS 属性の読み取りと更新を保護する必要があります。

ポインタ CoS の場合は、名前の変更が禁止されているテンプレートエントリが 1 つ存在します。通常、テンプレートエントリ全体を保護するのがもっとも簡単な方法です。

クラシック CoS では、すべてのテンプレートエントリは、定義エントリで指定された共通の親を持ちます。この親エントリにテンプレートを格納するだけで、親エントリに対するアクセス制御によってテンプレートが保護されます。ただし、親の下のほかのエントリにアクセスする場合は、テンプレートエントリを個別に保護する必要があります。

間接 CoS の場合は、アクセスする必要があるユーザーエントリを含む、ディレクトリ内の任意のエントリにテンプレートを指定できます。必要に応じて、ディレクトリ全体の CoS 属性に対するアクセスを制御するか、またはテンプレートとして使用される各エントリの CoS 属性のセキュリティを保護するかを選択できます。

CoS のターゲットエントリの保護

仮想 CoS 属性が生成される、CoS 定義の適用範囲内のすべてのエントリも値の算出に役立ちます。

CoS 属性がターゲットエントリにすでに存在する場合は、デフォルトでは、CoS メカニズムはこの値を上書きしません。この動作を変更する場合は、ターゲットエントリを上書きするように CoS を定義するか、すべてのターゲットエントリで CoS 属性を保護します。これらの手順については、『Sun ONE Directory Server 管理ガイド』を参照してください。

間接 CoS とクラシック CoS は、ターゲットエントリの指示子属性に依存します。この属性は、使用するテンプレートエントリの DN または RDN を指定します。この属性を保護する場合は、CoS の適用範囲全体でグローバルに保護するか、または各ターゲットエントリで必要に応じて個別に保護する必要があります。

その他の従属関係の保護

最後に、生成されたその他の CoS 属性およびロールに関して仮想 CoS 属性を定義することができます。仮想 CoS 属性を確実に保護するために、これらの従属関係を理解し保護する必要があります。

たとえば、ターゲットエントリの CoS 指示子属性には `nsRole` を指定できます。したがってロール定義も保護する必要があります。詳細は、210 ページの「エントリの安全なグループ化」を参照してください。

一般に、仮想属性値の算出に関係する属性またはエントリには、読み取りおよび書き込みアクセス制御を設定します。このため、複雑な従属関係は、十分に計画してから設定するか、以後のアクセス制御の実装の複雑さを軽減できるように簡素化する必要があります。その他の仮想属性との従属関係を最小限に抑えると、ディレクトリのパフォーマンスを向上させ、管理作業を削減することができます。

設定情報のセキュリティ保護

ほとんどの配備では、ルート DSE エントリ (長さがゼロの DN が指定されたベースオブジェクト検索で返されるエントリ)、または cn=config、cn=monitor、または cn=monitor の下のサブツリーでは、追加のアクセス制御は必要ありません。ルート DSE エントリとこれらのサブツリーには、Directory Server が自動的に生成する属性が含まれ、LDAP クライアントはディレクトリサーバーの機能と設定を判断するときにこの属性を使用します。

しかし、namingContexts というルート DSE エントリ属性には、各 Directory Server データベースのベース DN のリストが含まれます。このリストに加え、これらの DN は cn=config および cn=monitor の下のマッピングツリーエントリにも格納されます。セキュリティ上の理由から 1 つまたは複数のサブツリーを非表示に設定して設定情報を保護するには、次のような配置が必要です。

- ACI 属性を、非表示にするサブツリーのベースにあるエントリに配置する
- ルート DSE エントリに含まれる ACI を namingContexts 属性に配置する
- ACI を cn=config サブツリーと cn=monitor サブツリーに配置する

その他のセキュリティ関連資料

安全なディレクトリの設計方法については、以下の資料を参照してください。

- Sun ONE ミドルウェア開発者向けのセキュリティ関連資料
  http://developer.iplanet.com/tech/security/
- 『Understanding and Deploying LDAP Directory Services』
  (T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年)
- SecurityFocus.com
  http://www.securityfocus.com/
- コンピューター緊急事態対策チーム (CERT) 管理センター
  http://www.cert.org/
- CERT セキュリティ向上モジュール
  http://www.cert.org/security-improvement/

第 8 章

ディレクトリの監視

Directory Server の配備を成功させるには、効果的な監視とイベント管理の戦略が必要です。この戦略は、監視対象となるイベント、使用するツール、そのイベントが発生した場合に実行するアクションを定義します。発生しがちなイベントについて対策を用意しておくことで、サービスの停止やレベル低下を防ぎ、可用性とサービスの品質を向上することができます。

監視とイベント管理の戦略には、レプリケーション設定などの具体的なアーキテクチャコンポーネントを含める必要があり、また、システムとネットワークの監視も含める必要があります。この章では、効果的な監視戦略に何を含めるかについて説明し、Sun ONE Directory Server の監視機能について解説します。

注	システムとネットワークの監視は Sun ONE Directory Server に固有の話題ではないので、この章では詳しく触れません。

この章は、次の節から構成されています。

- 監視およびイベント管理戦略の定義
- Directory Server の監視ツール
- Directory Server の監視
- SNMP による監視

監視およびイベント管理戦略の定義

ここでは、監視とイベント管理の戦略を定義するための手順の概要について説明します。効果的な監視の定義手順は、次の段階に分けることができます。

1. 適切な管理ツールを選択します。オペレーティングシステムのツール、Sun ONE Directory Server の監視ツール、サードパーティ製の監視ツールを使用できます。
2. 監視対象となるパフォーマンス指標をディレクトリアーキテクチャから選択します (多くの場合、これはサイズ設定属性と調整属性です)。
3. 監視ツールを使用してパフォーマンス指標を監視する場合に、イベントまたはアラーム状態を始動する条件を定義します。つまり、パフォーマンスの受容可能レベル、または各パフォーマンス指標に対する処理を定義します。
4. アラーム状態が発生したときに実行するアクションを決定します。

Directory Server の監視ツール

ここでは、Directory Server で使用できる監視ツールと、Directory Server のアクティビティの監視に利用できるその他のツールについて、概要を説明します。次節で説明する主要パフォーマンス指標は、これらのツールを使用して、またはツールを組み合わせて、すべて監視できます。

- コマンド行ツール

  コマンド行監視ツールには、ディスク使用量などのパフォーマンスを監視するオペレーティングシステムに固有のツール、ディレクトリに格納されているサーバー統計を収集する `ldapsearch` などの LDAP ツール、サードパーティ製ツール、カスタムシェル、Perl スクリプトが含まれます。

- Directory Server ログ

  Sun ONE Directory Server で使用できるアクセスログ、監査ログ、エラーログには、監視に利用できる情報が豊富に含まれます。これらのログを手動で監視したり、カスタムスクリプトを使用して解析することで、配備に関連する監視情報を抽出できます。ログ情報へのアクセスに使用する Sun ONE スクリプトについては、『Sun ONE DSRK Tools Reference』を参照してください。ログファイルの表示と設定については、『Sun ONE Directory Server 管理ガイド』の「Managing Log Files」を参照してください。

- Directory Server コンソール

  Sun ONE Directory Server コンソールでは、ディレクトリの動作をリアルタイムにグラフィカルユーザーインタフェースで確認できます。このコンソールは、リソースサマリー、現在のリソース使用状況、接続状態、グローバルデータベースキャッシュ情報など、一般的なサーバー情報を提供します。また、データベース

のタイプと状態、エントリキャッシュ統計、キャッシュ情報、データベース内の各インデックスに関連する情報など、一般的なデータベース情報も提供します。さらに、コンソールには接続と各連鎖サフィックスで実行される操作に関する情報も表示されます。

- レプリケーション監視ツール

  Sun ONE Directory Server のレプリケーション監視ツールを使用することで、次の操作を実行できます。

  - マスターレプリカと 1 つまたは複数のコンシューマレプリカの間の同期状態を監視する
  - 複数の異なるレプリカの間で同じエントリを比較することで、レプリケーションの状態を確認する
  - 完全なレプリケーショントポロジを描写する (これは複雑なディレクトリ配備で特に有用です)

- SNMP (Simple Network Management Proocol)

  Directory Server は、SNMP (Simple Network Management Proocol) による監視をサポートしています。SNMP は、グローバルネットワーク制御と監視のための標準メカニズムで、ネットワーク管理者はネットワーク管理作業を一元的に行えます。

  SNMP と Directory Server がサポートする SNMP 管理対象オブジェクトについては、225 ページの「SNMP による監視」を参照してください。SNMP の設定方法については、『Sun ONE Directory Server 管理ガイド』の「Monitoring Directory Server Using SNMP」の章を参照してください。

Directory Server の監視

監視とイベント管理の戦略を定義する上で最も重要な手順は、ディレクトリアーキテクチャ上の 1 つまたは複数のコンポーネントで監視する、主要パフォーマンス指標を決定することです。何を監視対象とし、それをどの程度監視するかは、配備によって大きく異なります。

ここでは、監視対象となるパフォーマンス指標について説明します。説明する内容は次のとおりです。

- Directory Server アクティビティの監視
- データベースアクティビティの監視
- ディスクの状態の監視
- レプリケーションアクティビティの監視
- インデックス付けの効率の監視
- セキュリティの監視

注

- ディレクトリの `cn=monitor` 分岐内の監視情報に対して `ldapsearch` コマンドを実行するときは、ユーザーは認証を受け、情報にアクセスするための適切な権限を持っている必要があります。監視戦略を定義するには、このようなアクセス権限の取得が前提条件となります。
- システムが効率的に稼動し、Directory Server に悪影響を生じていないことを確認する上で、Sun ONE Directory Server が稼動するオペレーティングシステム環境を監視することは重要ですが、これは Sun ONE Directory Server に固有の話題ではないので、この章では詳しく触れません。詳細は、オペレーティングシステムのマニュアルを参照してください。

Directory Server アクティビティの監視

Directory Server では、さまざまな方法でサーバーの状態を監視できます。たとえば、次のような方法があります。

- Sun ONE コンソールの「サーバーおよびアプリケーション」タブには、インストール日、バージョン、サーバーの状態 (起動されているかどうか)、ポート番号など、サーバーに関する一般的な情報が表示されます。
- Directory Server コンソールでは、これ以外の情報も確認することができます。このコンソールの「状態」タブには、次の情報が表示されます。
  - 起動時刻とサーバーの現在の時刻

- 接続、開始および終了された捜査、クライアントに送信されたエントリとバイト数の詳細を示すリソースサマリー
- アクティブスレッド、開いている接続と利用できる接続、クライアントからの読み取りを待つスレッドの数、使用中のデータベースの数など、現在のリソース使用状況に関する情報
- 接続が開かれた日時、開始および終了された接続の数、クライアントがサーバーへのバインドに使用する識別名、接続の状態 (ブロック、または非ブロック)、接続の種類 (LDAP または DSML) など、開いているすべての接続に関する情報

Directory Server コンソールで利用できるパフォーマンスカウンタについては、『Sun ONE Directory Server 管理ガイド』の「Monitoring Your Server From the Directory Server コンソール」を参照してください。

- `cn=monitor,cn=config` エントリに対して `ldapsearch` コマンドを実行すると、Directory Server コンソールの「状態」タブに表示される情報と同じ情報を取得できます。一部の情報は、`ldapsearch` コマンドを実行するユーザーがディレクトリ管理者としてバインドしている場合にだけ表示されます。このアクセス制限を解除するには、この情報に関連付けられているアクセス制御を設定し直す必要があります。`cn=monitor,cn=config` の下に格納されているパフォーマンスカウンタについては、『Sun ONE Directory Server 管理ガイド』の「Monitoring Your Server From the Command Line」を参照してください。
- UNIX システムでは、`ps` コマンドを実行すると、現在稼動しているプロセスが表示されます。これにより、Directory Server `slapd` デーモンの稼動状態を確認することができます。詳細については、`ps(1)` マンページ、またはオペレーティングシステムに付属するコマンド行ツールのマニュアルを参照してください。
- `ldapsearch` コマンド行ユーティリティを使用して、Directory Server が要求に応答しているかどうかを確認することができます。時間のかかる、インデックス付けされていない検索を避け、ベースレベルの検索を行います。複数のデータベースがあるときは、各データベースサフィックスへの LDAP クエリーを作成し、データベースがオンライン状態にあり、応答しているかどうかを確認することができます。
- サーバーの動作を監視に Directory Server のアクセスログを使用して、Directory Server が稼動しているかどうかを確認することができます。アクセスログの内容と接続コードについては、『Sun ONE Directory Server Reference Manual』の「Access Logs and Connection Codes」を参照してください。
- Directory Server のエラーログには、サーバーの起動と停止の状態が記録されます。この情報に基づいて、サーバーが稼動しているかどうかを確認することができます。ログファイルの表示と設定については、『Sun ONE Directory Server 管理ガイド』の「Managing Log Files」を参照してください。

データベースアクティビティの監視

データベースのアクティビティを監視することは、データベースがオンラインの状態にあり、必要になったときに確実にアクセスできることを確認する上で有効です。データベースの監視情報にアクセスするには、cn=config 分岐の特定の領域に対して ldapsearch コマンドを実行します。表 8-1 は、提供される監視情報の種類と、cn=config 分岐の対応領域を示しています。

表 8-1 cn=config 内のデータベース監視情報ソース

情報の領域	対応する cn=config の分岐
データベースの一般情報	cn=database,cn=monitor,cn=ldbm database, cn=plugins,cn=config
データベースのキャッシュ情報	cn=monitor,cn=ldbm database,cn=plugins,cn=config
特定のデータベースインスタンスの情報	cn=monitor,cn=*suffixName*,cn=ldbm database, cn=plugins,cn=config
連鎖サフィックスの情報	cn=monitor,cn=*suffixName*,cn=chaining database, cn=plugins,cn=config

次に、データベース監視情報の領域について、さらに詳しく説明します。

- cn=database,cn=monitor,cn=ldbm database,dn=plugins,cn=config 分岐は、キャッシュへのアクセス、トランザクション、ロック、ログの情報を提供します。Directory Server の設定属性の完全なリストについては、『Sun ONE Directory Server Reference Manual』の「Core Server Configuration」の章および「Plug-in Implemented Server Functionality」の章を参照してください。

  監視する一般データベース情報は、配備するディレクトリに固有の要件によって異なります。たとえば、Directory Server が複数のトランザクションを同時に処理することが多い場合は、一度に並行して処理できるトランザクションの最大数の監視が必要になるかもしれません。この値 (nsslapd-db-max-txns 属性によって定義されます) が、許容されている最大トランザクション数 (nsslapd-db-configured-txns 属性によって定義されます) に達した場合に操作の失敗を防ぐには、許容される最大トランザクション数を増やす必要があります。

- データベースのキャッシュとインデックス付けのパフォーマンスを監視するときは、Directory Server コンソールの「状態」タブを使用するか、次の分岐で ldapsearch コマンドを実行します。

  cn=monitor,cn=ldbm database,cn=plugins,cn=config および cn=monitor,cn=*suffixName*,cn=ldbm database,cn=plugins,cn=config

関連する設定属性の完全なリストについては、『Sun ONE Directory Server Reference Manual』の「Core Server Configuration Attributes」の章および「Plug-in Implemented Server Functionality」の章を参照してください。

- `cn=monitor,cn=`*suffixName*`,cn=chaining database,cn=plugins,cn=config` 分岐は、接続、LDAP、および実行中のバインド操作、バインド解除操作に関する情報を提供します。この情報は、Directory Server コンソールの「状態」タブにも表示されます。

ディスクの状態の監視

ディスク容量を効果的に監視することで、ディスクリソースの不足に関連する問題を防止できます。`cn=disk,cn=monitor` エントリにより、次の監視情報が提供されます。

- データベースインスタンスへのパス。同一ディスクに複数のデータベースインスタンスが存在する場合、またはインスタンスが同一ディスク上の複数のディレクトリを参照する場合は、短いパス名が表示される
- サーバーが利用できるディスク容量 (M バイト単位)
- ディスクの状態。正常 (normal)、残量低下 (low)、残量なし (full) で表される。この状態は、利用可能容量、および「low」、「full」の警告を始動するためのしきい値によって決定される

`cn=disk,cn=monitor` 属性、およびディスク容量の 2 つのしきい値については、『Sun ONE Directory Server Reference Manual』の「Core Server Configuration Attributes」および「Plug-in Implemented Server Functionality」を参照してください。

レプリケーションアクティビティの監視

レプリケーション状態の監視は、グローバルな監視戦略では重要な要素です。レプリケーションの問題の可能性を早く認識できるほど、問題を迅速に解決し、正しいレプリケーション動作を再開できます。

Sun ONE Directory Server 5.2 には、レプリケーション機能のさまざまな側面を監視する 3 つの監視ツールが用意されています。レプリケーション監視ツールは LDAP クライアントとして機能し、標準接続またはセキュリティ保護された接続 (LDAPS) を介して使用できます。次の 3 種類のレプリケーション監視ツールは次のとおりです。

- insync
- entrycmp
- repldisc

insync

`insync` ツールは、マスターレプリカと 1 つまたは複数のコンシューマレプリカの間の同期状態を示します。競合の可能性を管理する場合は、同期のレベルに注意する必要があります。

entrycmp

`entrycmp` ツールを使用することで、複数の異なるサーバー上の同一エントリを比較することができます。マスターレプリカからエントリが取得され、エントリの `nsuniqueid` を使用して指定コンシューマから同じエントリを取得します。エントリの属性と値が比較され、どちらも同じであれば、これらのエントリは同一であると見なされます。

注 `insync` ツールと `entrycmp` ツールを実行するマシンは、指定されたすべてのホストにアクセスできる必要があります。ファイアウォール、VPN、またはネットワーク設定上のその他の理由からホストにアクセスできない場合、これらのツールを使用することは困難になります。同じ理由で、レプリケーション監視ツールを実行する前に、すべてのサーバーが起動され、稼動していることを確認してください。

---

repldisc

`repldisc` ツールを使用することで、レプリケーショントポロジを推測することができます。トポロジの推測は、1 つのサーバーから始まり、トポロジ内のすべての既知のサーバーをグラフ化することで行われます。次に、`repldisc` ツールはトポロジを示す隣接マトリクスを出力します。このレプリケーショントポロジ推測ツールは、配備したグローバルトポロジを思い出すことが難しい、大規模で複雑な配備で便利です。

注

- レプリケーション監視ツールを使用するときは、ホストをすべてを記号名で指定するか、またはすべてを IP アドレスで指定します。2 つを組み合わせた場合、問題が生じる可能性が高くなります。
- SSL が有効な状態でレプリケーション監視ツールを実行するときは、ツールの実行対象サーバーには、トポロジ内の他のサーバーが使用するすべての証明書のコピーが保持されている必要があります。
- これらのツールは LDAP クライアントベースであるため、サーバーへの認証と、`cn=config` に対する読み取りアクセス権を持つバインド DN が必要となります。これらのツールの詳細な設定方法と、SSL が有効な状態での使用については、『Sun ONE Directory Server 管理ガイド』の「Monitoring Replication Status」を参照してください。

レプリケーション監視ツールについては、『Sun ONE Directory Server Reference Manual』の「Replication Monitoring Tools」を参照してください。

インデックス付けの効率の監視

インデックス付けは、書き込み (インデックス作成時) と読み取り (ディレクトリ検索時) のパフォーマンスに影響します。このため、書き込みと読み取りのパフォーマンスのバランスを適切に保つには、インデックス付けの効率を監視することが重要です。効果的なインデックス付け戦略により、不要なインデックスをなくし、クライアントアプリケーションで必要なインデックスだけを維持することができます。

インデックス付けの効率は、次の方法で監視できます。

- アクセスログを調べ、インデックスのない検索が完了するまでの時間を監視することで、不釣合いな時間を消費した、インデックスのない検索を識別することができます。アクセスログには、検索とそのフィルタに関する情報も記録されています。これに基づいて、インデックスを作成することでパフォーマンスを向上できるかどうかを判断できます。
- Directory Server コンソールの「状態」タブでは、サフィックスまたは連鎖サフィックスごとに最も頻繁に使用されたインデックスを監視できます。これは、インデックスの使用が何回試みられ、何回成功したかを示します。`cn=monitor,cn=`*suffixName*`,cn=ldbm database,cn=plugins,cn=config` 分岐に対して `ldapsearch` コマンドを実行しても同じ監視情報が得られます。

設定されているインデックスのリストは、Directory Server コンソールの「設定」タブに表示されます (「Data」> 「*suffixName*」ノードの下)。前述の頻繁に使用されるインデックスと、設定されているインデックスのリストを比較することで、リソースを不必要に消費しているインデックスを特定し、それを削除するかどうかを決定できます。エントリにインデックス付けされた属性が含まれ、インデックスが使用されていない場合、そのインデックスを削除するとパフォーマンスが向上します。

アクセスログの内容と接続コードについては、『Sun ONE Directory Server Reference Manual』の「Access Logs and Connection Codes」を参照してください。Directory Server の設定属性の完全なリストについては、『Sun ONE Directory Server Reference Manual』の「Core Server Configuration」の章および「Plug-in Implemented Server Functionality」の章を参照してください。

セキュリティの監視

配備のセキュリティの監視は、安全にアクセスできるディレクトリにとって不可欠です。受容可能なセキュリティレベルを維持する上で、Directory Server を次のように監視することをお勧めします。

- バインド試行の失敗数の監視は、ディレクトリへの侵入試行に関する警告となります。SNMP エージェントが稼動している場合は、`cn=snmp,cn=config` の下にある SNMP 管理対象オブジェクトカウンタ `dsBindSecurityErrors` に対して `ldapsearch` を実行することで、バインド試行の失敗を監視できます。
- アクティビティが行われていない、開いている接続の数の監視は、サービス拒否攻撃の可能性に関する警告となります。現在の接続数と完了した操作の数は、Directory Server コンソールの「状態」タブ、または `cn=monitor` の下にある属性を検索することで確認できます。
- 新しい実効権限機能を使用することで、クライアントはディレクトリのエントリと属性に対して持つアクセス制御権を照会できます。ユーザーがアクセス権を照会できるようになったことで、ユーザーの管理、アクセス制御ポリシーの検証、設定内容の決定が容易になりました。

  実効権限機能は、日々の操作としてではなく、定期的に使用されることが多いと考えられます。実効権限については、192 ページの「実効権限に関する情報の取得」を参照してください。

SNMP による監視

SNMP はグローバルなネットワーク制御と監視のための標準メカニズムです。SNMP を使用することで、ネットワーク管理者はネットワーク管理作業を一元的に行い、さまざまなデバイスをリアルタイムで監視することができます。ここでは、SNMP を使用して Directory Server の操作を監視する方法について説明します。説明する内容は次のとおりです。

- SNMP について
- Sun ONE Directory Server での SNMP 監視

SNMP について

SNMP は、ネットワークアクティビティに関するデータを交換するために使用されるプロトコルです。SNMP を使用すると、管理対象デバイスと、ユーザーがネットワークをリモート管理する NMS (ネットワーク管理ステーション) の間で、データが移動します。管理対象デバイスは、ホスト、ルータ、Directory Server など、SNMP が稼動するすべてのデバイスです。NMS とは通常、1 つ以上のネットワーク管理アプリケーションが稼動する強力なワークステーションを指します。ネットワーク管理アプリケーションは、管理対象デバイスに関する情報 (どのデバイスが有効または無効か、どのエラーメッセージをいくつ受け取ったか、など) を、多くの場合はグラフィカルに表示します。

情報は、サブエージェントとマスターエージェントの 2 種類のエージェントを使用して、NMS と管理対象デバイスの間で転送されます。サブエージェントは、管理対象デバイスに関する情報を収集し、その情報をマスターエージェントに渡します。Sun ONE Directory Server にはサブエージェントがあります。マスターエージェントは、各種サブエージェントと NMS の間で情報を交換します。マスターエージェントは、通信先のサブエージェントと同じホストマシン上で動作します。

ホストマシンには、複数のサブエージェントをインストールできます。たとえば、Directory Server、Enterprise Server、および Messaging Server をすべて同じホスト上にインストールした場合、これらの各サーバーのサブエージェントは、同一のマスターエージェントと通信します。Windows 環境では、マスターエージェントは Windows オペレーティングシステムによって提供される SNMP サービスです。UNIX 環境では、マスターエージェントは Sun ONE 管理サーバーと一緒にインストールされます。

照会可能な SNMP 属性の値は、管理対象デバイス上に保持され、必要に応じて NMS に報告されます。各属性は管理対象オブジェクトと呼ばれ、エージェントはこのオブジェクトにアクセスし、これを NMS に送ることができます。管理対象オブジェクトはすべて、ツリーのような階層を持つデータベースである MIB (管理情報ベース) に定義されています。この階層の最上位には、ネットワークに関する一般的な情報が含まれています。下位の各階層には、個々のネットワーク領域に関するより詳細な情報が含まれています。

SNMP は、PDU (プロトコルデータ単位) の形式でネットワーク情報を交換します。PDU には、管理対象デバイス上に格納された変数に関する情報が含まれています。これらの変数は管理対象オブジェクトとも呼ばれ、必要に応じて NMS に報告される値と名前を保持しています。NMS と管理対象デバイスとの間の通信は、次の 2 つのどちらかの方法で行われます。

- NMS 主導の通信
- 管理対象デバイス主導の通信

次に、Sun ONE Directory Server がサポートしている NMS 主導の通信について説明します。

NMS 主導の通信

これは、NMS と管理対象デバイス間の通信で、最も一般的なタイプの通信です。このタイプの通信では、NMS は管理対象デバイスから情報を要求するか、または管理対象デバイス上に格納された変数の値を変更します。

NMS によって開始される SNMP セッションは、次のように実行されます。

1. NMS が、どの管理対象デバイスおよびオブジェクトに監視が必要であるかを判断します。
2. NMS が、マスターエージェントを介して、PDU を管理対象デバイスのサブエージェントに送ります。この PDU は、管理対象デバイスから情報を要求するか、または管理対象デバイス上に格納された変数の値を変更するようサブエージェントに指示します。
3. 管理対象デバイスのサブエージェントが、マスターエージェントから PDU を受け取ります。
4. NMS から送られた PDU が変数に関する情報を要求するものである場合、サブエージェントはマスターエージェントにその情報を送り、マスターエージェントは別の PDU として NMS に情報を送り返します。次に、NMS が、この情報を文字またはグラフィック形式で表示します。

   NMS から送られた PDU が、変数に値を設定するようサブエージェントに要求するものである場合、サブエージェントは変数値を要求のとおりに設定します。

Sun ONE Directory Server での SNMP 監視

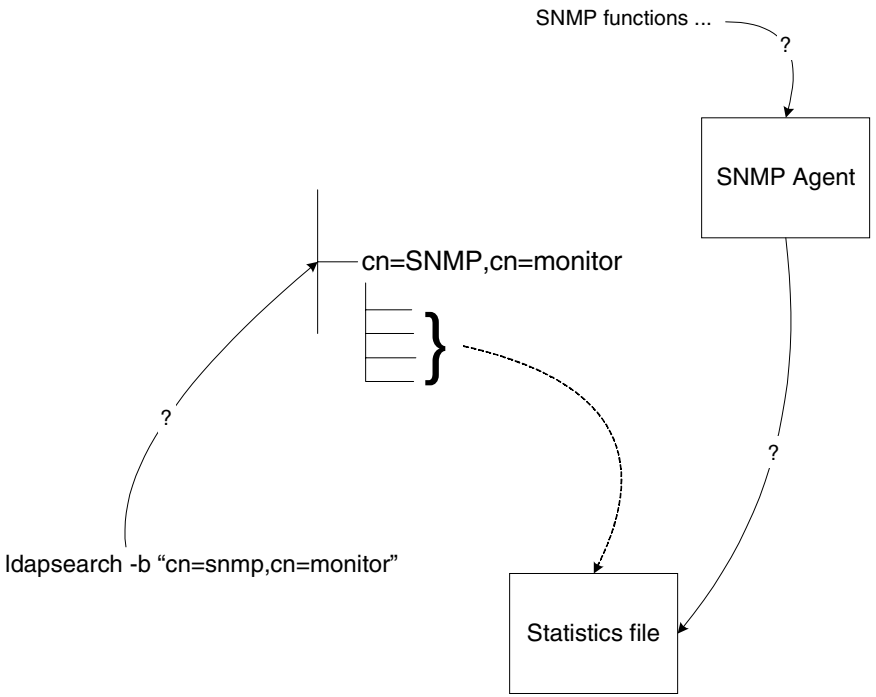
Directory Server は、次の 2 つの方法による SNMP 監視をサポートしています。

- SNMP エージェント経由の監視。SNMP 属性は統計ファイルにマッピングされ、SNMP エージェントが照会されるたびにこのファイルが読み取られる。Directory Server が稼動していない場合は、この統計ファイルは存在しない
- `ldapsearch` コマンド行ユーティリティによる監視。SNMP 属性は `cn=snmp,cn=monitor` エントリの下に格納される。次の `ldapsearch` コマンドを実行すると、Directory Server のすべての SNMP 属性がリスト表示される

```
ldapsearch -h host -p port -s base -b "cn=snmp,cn=monitor"
 "objectclass=*"
```

図 8-1 は、Directory Server から SNMP 監視情報を取得する 2 つの方法を示しています。

図 8-1 Directory Server での SNMP 監視



MIB の定義場所および SNMP の使用方法については、『Sun ONE Directory Server 管理ガイド』の「Monitoring Directory Server Using SNMP」の章を参照してください。

Sun ONE Directory Server がサポートする SNMP 管理対象オブジェクトは、Directory Server Monitoring MIB RFC 2605 の初期草案に基づいています。SNMP エージェントから返される SNMP 操作管理オブジェクトは、`ldapsearch` コマンドによって返される SNMP 監視属性と同じです。これらの属性については、『Sun ONE Directory Server Reference Manual』の「Monitoring Attributes」を参照してください。SNMP エージェントから返される属性の名前には、`ds` という接頭辞が付けられます。

操作管理オブジェクトに加え、Sun ONE Directory Server は監視対象の Directory Server とピア Directory Server の間の対話に関連する管理対象オブジェクトもサポートしています。表 8-2 は、これらのオブジェクトを示しています。

表 8-2 サポートされている対話関連の SNMP 管理対象オブジェクト

管理対象オブジェクト	説明
dsTimeOfCreation	Directory Server とピア Directory Server の間の (試行された) 対話の詳細を含むエントリが作成された時点の、システム起動からの経過時間の値。エントリが、管理ネットワークサブシステムの初期化以前に作成された場合、このオブジェクトの値はゼロとなる
dsTimeOfLastAttempt	この Directory Server へのアクセスが最後に試みられた時点の、システム起動からの経過時間の値。最後の試行が、管理ネットワークサブシステムの初期化以前に行われていた場合、このオブジェクトの値はゼロとなる
dsTimeOfLastSuccess	この Directory Server へのアクセスが最後に成功した時点の、システム起動からの経過時間の値。すべての試行が成功しなかった場合、このオブジェクトの値はゼロとなる。最後に成功した試行が、管理ネットワークサブシステムの初期化以前に行われていた場合、このオブジェクトの値はゼロとなる
dsFailuresSinceLastSuccess	この Directory Server への最後に成功した接続試行以後の失敗回数。成功した試行がない場合は、このオブジェクトの値は、このエントリの作成以後の失敗回数となる
dsFailures	このエントリの作成以後に行われたピア Directory Server へのアクセス失敗の累積回数
dsSuccesses	このエントリの作成以後に行われたアクセス成功の累積回数
dsURL	ピア Directory Server の URL

Sun ONE Directory Server は、Directory Server の現在のインストールに関する情報を含む、エンティティ関連の管理対象オブジェクトもサポートしています。表 8-3 は、これらの管理対象オブジェクトを示しています。

表 8-3 サポートされているエンティティ関連の SNMP 管理対象オブジェクト

管理対象オブジェクト	説明
dsEntityDescr	インストールされている Directory Server の一般的な説明テキスト
dsEntityVers	Directory Server のバージョン
dsEntityOrg	Directory Server のこのインストールの責任組織
dsEntityLocation	この Directory Server の物理的な場所。ホスト名、ビル名、番号、研究所番号など
dsEntityContact	インストールされている Directory Server の責任者と連絡先情報
dsEntityName	インストールサイトによって Directory Server のインストールに割り当てられた名前

第 2 部

Directory Server の配備例と参照アーキテクチャ

第 2 部では、Sun ONE Directory Server 5.2 の配備例と、いくつかのアーキテクチャ戦略を紹介します。特定のビジネス目標を達成するソリューションを提供するために、Sun ONE Directory Server 5.2 をどのように配備するかを理解できるように、ビジネス上の観点から配備例を紹介します。企業がディレクトリを持つデータセンター (サイト) の数によって分類される各アーキテクチャ戦略では、データの物理的な格納場所に関するガイドラインや適切なレプリケーショントポロジだけでなく、フェイルオーバー、スケーラビリティ、バックアップの戦略についても説明します。

Directory Server 配備の実装にあたっては、Sun Professional Service からの支援を受けることをお勧めしますが、次に紹介する配備例とアーキテクチャ戦略を参照することは、知っておくべき主な留意点や問題点の理解に役立つはずです。

第 9 章

銀行での配備例

ビジネス上の課題

ExampleBank 銀行は国際的な銀行で、顧客と従業員に電子バンキングと電話バンキングの機能を提供しようとしています。顧客は、たとえば、口座残高の確認、同一銀行内または他行への振り替え、投資ポートフォリオの確認、管理者行員との打ち合わせの設定、オンラインニュースの表示、プロファイル情報の変更などを期待しており、行員は、自宅から業務へのログイン、銀行電話帳の検索、プロファイル情報の変更、電話バンキングサービスを使用する顧客から依頼される振り替えの実行を望んでいます。ExampleBank 銀行のビジネス上の課題は、特定範囲の銀行サービスへのアクセスを次の条件でユーザー (顧客と従業員) に提供することです。

- 危険にさらされることのないセキュリティ
- 優れた性能
- スケーラビリティ
- 管理容易性

ExampleBank 銀行は、このビジネス上の課題を克服するために、銀行業務配備の基盤として Sun ONE Directory Server 5.2 を配備しました。ExampleBank のユーザーが、ポータル経由または電話経由で銀行のオンラインシステムへの認証を行うたびに使用される、ユーザー認証データとプロファイルデータは、すべて Directory Server に保持されます。次の配備例は、ExampleBank 銀行が電子バンキングと電話バンキングのビジネス目標を達成するために、Directory Server をどのように実装したかを詳細に説明しています。説明する内容は次のとおりです。

- 配備コンテキストとレプリケーショントポロジ
- パフォーマンス要件
- スキーマ、データ、ディレクトリ情報ツリーの設計
- セキュリティ上の注意点
- 実装

配備コンテキストとレプリケーショントポロジ

配備コンテキスト

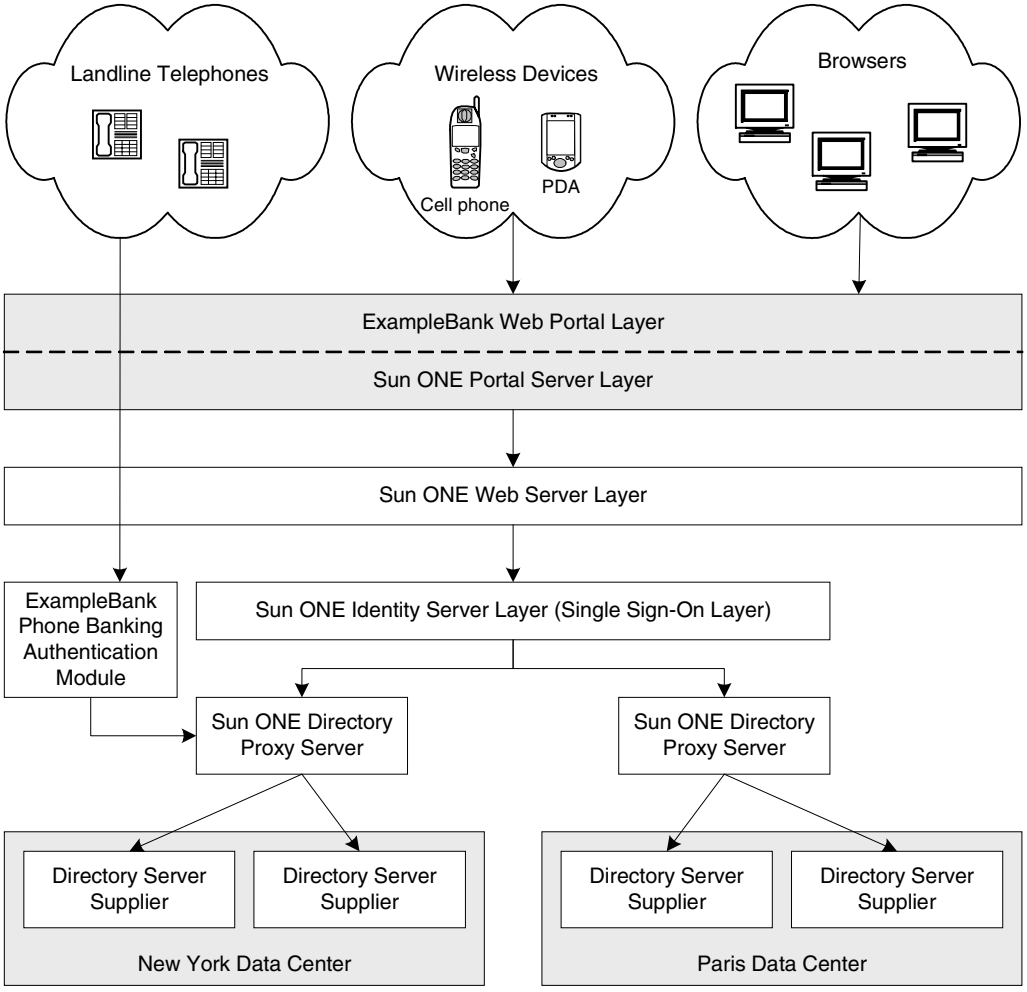
ExampleBank の配備では、電子バンキングサービスと電話バンキングサービスの両方を提供し、両サービスへの認証はセキュリティ保護され、管理が容易で、迅速かつスケーラブルである必要があります。ExampleBank は、両タイプのサービスへのアクセスに、次に説明する同一の製品群を使用します。

ExampleBank の電子バンキング機能にアクセスするユーザーは、主に Sun ONE Portal Server を使用します。ポータルは、ExampleBank の容易なログイン管理に必要なシングルサインオンソリューションを提供する Sun ONE Identity Server と組み合わされた Sun ONE Web Server で実行されます。ExampleBank は、Directory Server の操作フローの負荷分散と透過的なフェイルオーバーを行うサーバー群へのリフェラルを管理できる Sun ONE Directory Proxy Server を、配備の次のスタックにディレクトリプロキシサーバーとして配備しました。配備スタックの最後に Directory Server が配置されます。ここには、銀行業務に必要なユーザープロファイルと認証データが格納されます。

電話バンキング機能を利用する ExampleBank ユーザーは、指定の番号に電話をかけ、口座番号と電話バンキングの暗証番号を入力します。特別な電話バンキング認証モジュールが Directory Server にアクセスして口座番号を検索し、暗証番号が正しいかどうかを検証します。正しい場合は、電子バンキング担当者を呼び出します。

235 ページの図 9-1 は、ExampleBank の銀行業務向け配備スタックを示しています。

図 9-1　ExampleBank の銀行業務向け配備スタック



レプリケーショントポロジ

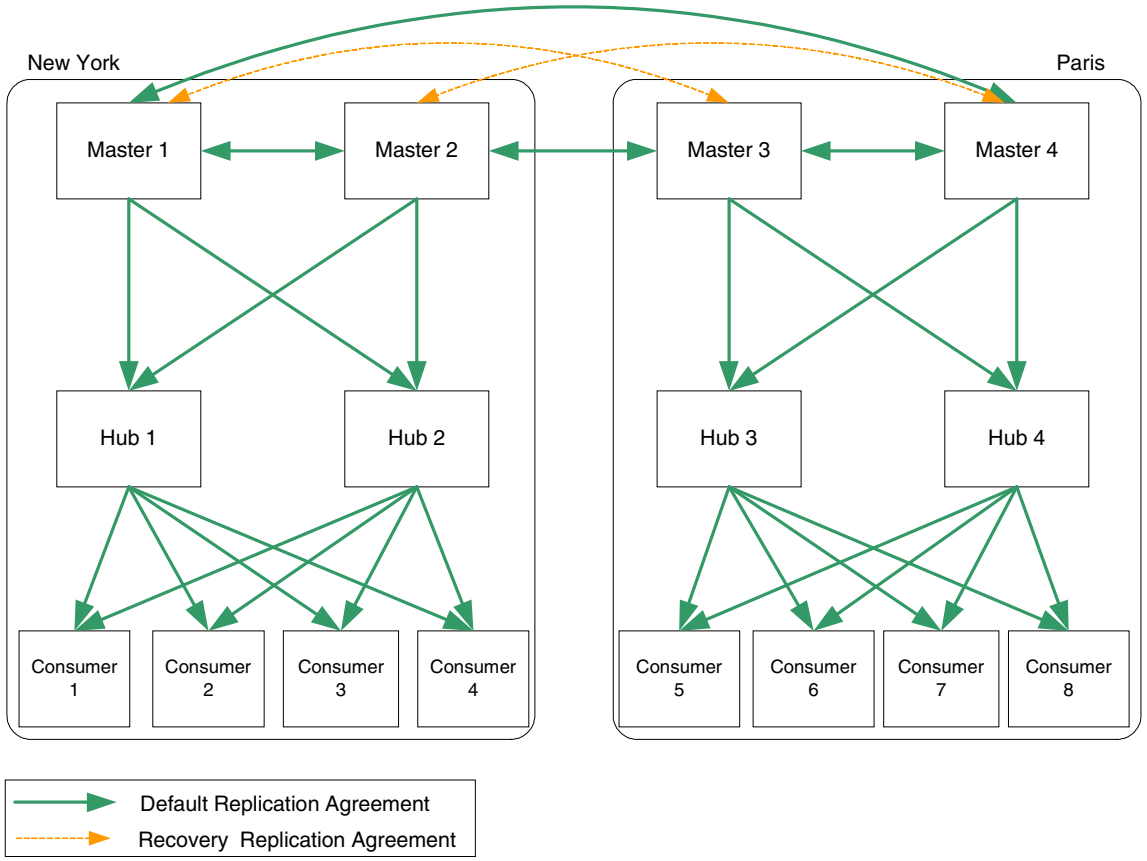
ここでは、ExampleBank のレプリケーション戦略について次の内容を説明します。

- レプリケーショントポロジの概要
- 障害と復元

レプリケーショントポロジの概要

ExampleBank には、ニューヨークとパリに 2 つの主要データセンターがあります。サービスには 1 日 24 時間アクセスできる必要があり、ローカル書き込みフェイルオーバーも必要であるため、各データセンターでマスターのペアを持つレプリケーション設計が採用されました。ExampleBank の 2 つの大陸にまたがる 4 方向のマスターレプリケーショントポロジは、Sun ONE Directory Server 5.2 の新機能である WAN 経由の 4 方向マルチマスターレプリケーションによって可能になりました。負荷を分散するために、このレプリケーショントポロジにはハブも含まれ、各データセンターの 4 つのコンシューマによって読み取り (検索) 操作のスケーラビリティが確保されます。237 ページの図 9-2 は、別の大陸にある ExampleBank の 2 つのデータセンターのレプリケーショントポロジを示しています。

図 9-2　ExampleBank の 2 つのデータセンターのレプリケーショントポロジ



237 ページの図 9-2 は、マスターとハブ、ハブとコンシューマの間のレプリケーションアグリーメント (デフォルトで有効化されます) を示しています。また、マスター 1 と 3、マスター 2 と 4 の間の復元レプリケーションアグリーメントも示されます。これは、マスターがオフラインになった場合に有効化されます。Sun ONE Directory Server 5.2 では、最初にレプリケーションアグリーメントを設定し、後から必要に応じて有効化または無効化できるので、トポロジに柔軟性が得られます。これにより、復元の準備が可能になるだけでなく、必要であればネットワーク使用量の最適化や、一時に送信される変更の数の最小化も可能になります。

注　データセンター間のリンクには、可用性を最適化するために複数の WAN 接続を使用することをお勧めします。

障害と復元

配備に影響するいくつもの障害が発生する可能性があります。ネットワークリンクに関連するものに限っても、Directory Server プロセス (`slapd`)、ハードウェア、レプリケーション障害、過度の読み取りまたは書き込み操作によるシステムへの過度な負荷などが挙げられます。必要な復元ソリューションが提供されるようにレプリケーショントポロジを設定することが重要です。

1 つのマスターがオフラインになった場合であれば、237 ページの図 9-2 に示されるように、適切な復元レプリケーションアグリーメントを有効化するだけで解決できます。1 つのデータセンターで 2 つのマスターが同時にオフラインになった場合は、いくつかのオプションが考えられます。1 つの復元ソリューションは、障害が発生していないデータセンターの 1 つのマスターと、障害が発生しているデータセンターの 1 つのハブの間で、事前に設定されているレプリケーションアグリーメントを有効化する方法です。もう 1 つの復元ソリューションは、Sun ONE Directory Server 5.2 のオンラインでのレプリカ昇格、降格機能を使用する方法です。障害が発生しているデータセンターの 1 つのハブをマスターに昇格させ、障害が発生したマスターがオンラインに戻るまでローカルな書き込み要求を担当させます。レプリケーショントポロジの例、障害からの復元、バックアップ戦略については、第 10 章「アーキテクチャ戦略」を参照してください。レプリケーションの手順に関する情報については、『Sun ONE Directory Server 管理ガイド』の「Managing Replication」の章を参照してください。

パフォーマンス要件

電子バンキングと電話バンキングの実行可能性は、操作の処理速度に大きく依存するため、ExampleBank の配備では、パフォーマンスは非常に重要です。優れたパフォーマンスに加え、ExampleBank では年率 6% の予定成長率を前提にオンラインバンキングのスタックをスケーラブルにする必要もあります。Directory Server は、両方の要件に対応できます。

ここでは、ExampleBank のパフォーマンス要件について説明します。説明する内容は次のとおりです。

- ユーザーからの要求
- ハードウェアのガイドライン

ユーザーからの要求

ExampleBank の現在のユーザーベースには 1,000 万人のユーザーが登録され、そのうち 10,000 人が従業員です。表 9-1 は、従業員ユーザーと顧客ユーザーの要件を示しています。これらの要件に基づいて、ExampleBank はオンラインバンキングが 1 秒間に処理する必要のある書き込みと読み取りの回数を計算します。

表 9-1 ExampleBank のユーザーからの要求

ユーザーからの要求	関連するユーザーの割合	要求の頻度	1 秒あたりの要求数
顧客のログイン	70%	1 週間に 4 回	195 ログイン
従業員のログイン	100%	1 週間に 3 回	2 ログイン
顧客による銀行トランザクション	100%	1 週間に 1 トランザクション	70 トランザクション
顧客による残高照会	70%	1 か月に 3 回	25 トランザクション
ユーザープロファイルの変更	100%	1 か月に 1 回	20 回のプロファイル変更
従業員による検索	100%	1 日に 1 回	1 回の検索
従業員による顧客トランザクションの代行	50%	1 日に 10 回	8 トランザクション

上の表で説明したトランザクションは、表 9-2 に示される LDAP 操作に変換されます。

表 9-2 ユーザーの要求と LDAP 操作の関係

ユーザーからの要求	対応する LDAP 操作
ログイン	検索 + バインド + 検索
銀行トランザクション	検索 + 変更
残高照会	LDAP 操作ではない
プロファイル変更	検索 + 変更
検索	検索

上記条件から、ExampleBank は次の検索、バインド、変更操作に対応する必要があることがわかります。

- 687 回の検索操作
- 197 回のバインド操作
- 98 回の変更操作

ハードウェアのガイドライン

この例のように、中規模から大規模の配備として位置付けられる配備では、基本的に次のような構成のハードウェアをお勧めします。

- 8 CPU
- 32 ～ 64G バイトの RAM
- RAID 構成の それぞれが 655G バイトの 3 ディスクアレイで、たとえば、次のように設定する
  - 最初のアレイにデータベース
  - 第 2 アレイにトランザクションログ
  - 第 3 アレイに変更ログ、アクセスログ、エラーログ

スキーマ、データ、ディレクトリ情報ツリーの設計

ここでは、ExampleBank のスキーマ、データ、ディレクトリ情報ツリーの設計について詳しく説明します。説明する内容は次のとおりです。

- スキーマ
- データ
- ディレクトリ情報ツリー

注 ここで紹介するスキーマとディレクトリ情報ツリーは実装例であり、完全または確実な情報ではありません。Sun ONE Identity Server と Sun ONE Portal Server はいくつかのオブジェクトクラスと属性を必要とします。詳細については、Sun ONE Identity Server と Sun ONE Portal Server のマニュアルを参照してください。

スキーマ

ExampleBank では、ユーザープロファイル、電子バンキングサービス、電話バンキングサービスを表わすスキーマが必要です。ここでは、このスキーマについて説明します。説明する内容は次のとおりです。

- 属性
- オブジェクトクラス

属性

ExampleBank は、`ebStatus` 属性を使用して属性レベルだけで顧客と従業員を区別します。次の 2 つの表は、ユーザープロファイル、バンキングサービス、追加のポートフォリオ管理バンキングサービスに関連する属性を示しています。

表 9-3 は、基本的なユーザープロファイル属性を示しています。

表 9-3 ExampleBank の基本的なユーザープロファイル属性

属性名	属性の説明	構文	複数値属性
`ebID`	ExampleBank のユーザーの一意の識別子 (顧客と従業員の両方)	整数	No
`ebStatus`	ユーザーが顧客、従業員、契約業者のいずれであるかを指定する	ディレクトリ文字列	Yes

表 9-3　ExampleBank の基本的なユーザープロファイル属性 (続き)

ebPreferredLanguage	ExampleBank ユーザーの指定言語	ディレクトリ文字列	No
ebSecondaryLanguage	ExampleBank ユーザーの第 2 言語	ディレクトリ文字列	No
ebAreaCode	ExampleBank ユーザーの地域コード	ディレクトリ文字列	No
ebCurrentAccountNumber	ExampleBank ユーザーの当座口座番号	整数	No
ebSavingsAccountNumber	ExampleBank ユーザーの普通口座番号	整数	No
ebPhoneBankingPin	電話バンキングの暗証番号を指定する	整数	No
ebNewsLetterSubscription	ExampleBank のニュースレターをユーザーに配信するかどうかを指定する	ディレクトリ文字列	No
ebNewsLetterType	ユーザーに配信するニュースレターの種類を指定する	ディレクトリ文字列	No
ebEBankingPreferencesFont	電子バンキングのユーザー指定フォントを指定する	ディレクトリ文字列	No
ebEbankingPreferencesFontSize	電子バンキングのユーザー指定フォントサイズを指定する	ディレクトリ文字列	No
ebPhoneBankingSafeNumber	安全でよく知られている地上回線の電話番号を指定する	電話番号	No

ExampleBank のユーザープロファイル属性に加え、表 9-4 に示される属性は、特定のユーザーでどの ExampleBank サービスが有効化されているかどうかを指定します。

表 9-4　ExampleBank のサービス有効化属性

属性名	属性の説明	構文	複数値属性
ebPhoneBankingEnabled	電話バンキングサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingEnabled	電子バンキングサービスが有効であるかどうかを指定する	ディレクトリ文字列	No

表 9-4 ExampleBank のサービス有効化属性 (続き)

ebPhoneBankingLoanServicesEnabled	電話バンキングローンサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingLoanServicesEnabled	電子バンキングローンサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingCheckBalanceEnabled	電子バンキング残高照会サービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingIntraBankTransfersEnabled	電子バンキング行内振り替えサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingInterBankTransfersEnabled	電子バンキング銀行間振り替えサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingChangeProfileEnabled	電子バンキングプロファイル変更サービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebEBankingPortfolioManagementEnabled	電子バンキングポートフォリオ管理サービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebPhoneBankCheckBalanceEnabled	電話バンキング残高照会サービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebPhoneBankIntraBankTransfersEnabled	電話バンキング行内振り替えサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebPhoneBankInterBankTransfersEnabled	電話バンキング銀行間振り替えサービスが有効であるかどうかを指定する	ディレクトリ文字列	No
ebPhoneBankChangeProfileEnabled	電話バンキングプロファイル変更サービスが有効であるかどうかを指定する	ディレクトリ文字列	No

表 9-4　ExampleBank のサービス有効化属性 (続き)

ebPhoneBankPortfolioManagementEnabled	電話バンキングポートフォリオ管理サービスが有効であるかどうかを指定する	ディレクトリ文字列	No

オブジェクトクラス

ディレクトリ内のエントリは、inetOrgPerson オブジェクトクラスの次に ebPerson オブジェクトクラスから継承されます。ExampleBank は、次の表に示されるオブジェクトクラスを使用します。

表 9-5　ebPerson オブジェクトクラス

オブジェクトクラス名	ebPerson
オブジェクトクラスのタイプ	structural
上位クラス	inetOrgPerson
必須の属性	ebid
許可された属性	ebStatus、ebCurrentAccountNumber、ebSavingsAccountNumber、ebPreferredLanguage、ebSecondaryLanguage、ebCustomField1、ebCustomField2、ebCustomField3、ebCustomField4、ebAreaCode、ebNewsletterSubscription、ebNewsLetterType、ebCheckBalanceEnabled、ebEBankingEnabled、ebPhoneBankingEnabled

表 9-6　ebEBankingUser オブジェクトクラス

オブジェクトクラス名	ebEBankingUser
オブジェクトクラスのタイプ	auxiliary
上位クラス	ebPerson

表 9-6　ebEBankingUser オブジェクトクラス (続き)

許可された属性	サービスに固有の追加属性のほかに、 ebEBankingCheckBalanceEnabled、 ebEBankingIntraBankTransferEnabled、 ebEBankingInterBankTransferEnabled、 ebEBankingChangeProfileEnabled、 ebEbankingPortfolioManagementEnabled、 ebEBankingLoanServicesEnabled

表 9-7　ebPhoneBankingUser オブジェクトクラス

オブジェクトクラス名	ebPhoneBankingUser
オブジェクトクラスのタイプ	auxiliary
上位クラス	ebPerson
許可された属性	サービスに固有の追加属性のほかに、 ebPhoneBankingPin、 ebPhoneBankCheckBalanceEnabled、 ebPhoneBankIntraTransfersEnabled、 ebPhoneBankInterTransfersEnabled、 ebPhoneBankChangeProfileEnabled、 ebPhoneBankPortfolioManagementEnabled、 ebEBankingLoanServicesEnabled

データ

前述のスキーマに基づくサンプルコンテナは、次のようになります。

```
dn: dc=eb,dc=com
objectclass:top
objectclass:organization
dc: eb
o:ExampleBank


dn: ou=people, dc=eb,dc=com
ou:people
description: Customers and employees of ExampleBank
objectclass:top
objectclass: organizationalunit
objectclass: example-am-managed-org-unit
```

ExampleBank 銀行の従業員である Bill Smith は、電子バンキングサービスと電話バンキングサービスの両方が有効化されていると仮定します。しかし、口座へのアクセスは主に Web 経由で行うため、Bill Smith は電話による残高確認と住所変更を残してその他の電話バンキングサービスを無効にするように求めました。この場合のサンプルエントリは次のようになります。

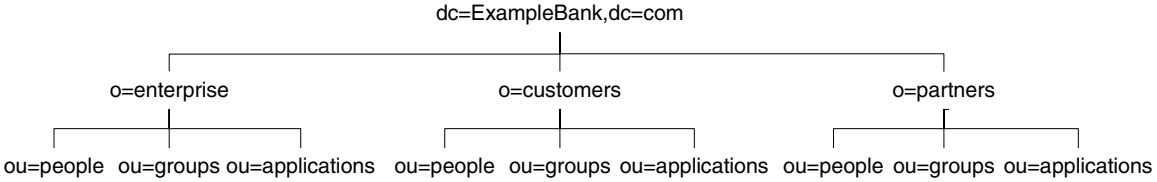
```
dn:ebid=123456789,ou=people,dc=eb,dc=com
objectclass:top
objectclass:person
objectclass:organizationalPerson
objectclass:inetOrgPerson
objectclass:ebPerson
objectclass:ebEBankingUser
objectclass:ebPhoneBankingUser
objectclass:example-am-web-agent-service
objectclass:example-am-managed-person
objectclass:example-am-user-device
objectclass:inetuser
objectclass:examplePreferences
objectclass:inetOrgPerson
objectclass: sunPortalDesktopPerson
objectclass: sunPortalNetmailPerson
ebid:123456789
displayname:Bill Smith
userPassword:{SSHA}Ek12JHYZ87op9645==
uid: 123456789
inetuserstatus: active
sn:Smith
givenname:William
cn:William F. Smith
mail:Bill.Smith@eb.com
telephonenumber:+1-256-556-5896
facsimiletelephonenumber:+1-256-556-5897
ebStatus:employee
ebAreaCode:CA-17B
ebPreferedLanguage:en
ebSecondaryLanguage:fr
ebCheckingsAccountNumber:133003300
ebSavingsAccountNumber:233003300
ebPhoneBankingEnabled:active
ebPhoneBankingPin:123456
ebEBankingEnabled:active
ebNewsletterSubscription:active
ebNewsletterType:email
ebEBankingCheckBalanceEnabled:active
ebEBankingIntraBankTransfersEnabled:active
ebEBankingInterBankTransfersEnabled:active
```

```
ebEBankingChangeProfileEnabled:active
ebEBankingPortfolioManagementEnabled:active
ebEBankingLoanServicesEnabled: inactive
ebPhoneBankCheckBalanceEnabled:active
ebPhoneBankingLoanServicesEnabled: inactive
ebPhoneBankIntraBankTransfersEnabled:inactive
ebPhoneBankInterBankTransfersEnabled:inactive
ebPhoneBankChangeProfileEnabled:active
ebPhoneBankPortfolioManagementEnabled:inactive
```

ディレクトリ情報ツリー

組織変更が頻繁に行われる ExampleBank 銀行では、データの階層構造を連鎖反応から保護するために、比較的フラットなディレクトリ情報ツリー構造を採用しました。図 9-3 に示されるように、情報ツリーの異なる部分に従業員、顧客、パートナー企業を配置したことで、それぞれの違いが明確になりました。

図 9-3 ExampleBank のディレクトリ情報ツリー



それぞれの分岐に異なるセキュリティポリシーを適用できるため、顧客、従業員、パートナー企業を分けて配置することは、ExampleBank にとってセキュリティ上の理由からも求められます。また、この区分は検索の負荷が少なく、操作も容易で、アクセス制御管理の向上と合わせてパフォーマンスの最適化と使い勝手の向上にも役立ちます。

注 図 9-3 のディレクトリ情報ツリーは、既存のディレクトリ構造がないことが前提で選択された設計を反映しています。多くの企業では、`o=enterprise`、`o=customer`、`o=partner` レベルを含まないディレクトリ構造を保有している可能性があり、このようなディレクトリツリーを作成するには、ディレクトリ構造を改造するための多大な労力と関連オーバーヘッドが必要になるかもしれません。しかし、それ以後のパフォーマンスおよび使い勝手の向上を考えると、これは価値のある投資であるといえます。

ディレクトリ情報ツリーとグループメカニズムによるユーザー管理だけでなく、ExampleBank は Sun ONE Directory Server 5.2 がサポートするロール機能の実装も選択しました。Directory Server のロール機能を使用することで、ユーザーを意味のあるユーザーセットに割り当て、アクセス制御、アカウントロックアウト、パスワードポリシーなどの Directory Server のその他の機能で内部的に使用するだけでなく、電話バンキングや ExampleBank が実装する人事アプリケーションなどの外部アプリケーションでも使用できます。

表 9-8 は、ユーザー管理のために ExampleBank が実装できるロールを示しています。もちろんこれは完全なリストではありません。管理機能を活用しようとする ExampleBank が、電話バンキングサービスと電子バンキングサービスに関連するユーザーをどのようにロールとしてグループ化するかについて考えるための出発点です。

表 9-8 ExampleBank のユーザー管理を活用するために実装されるロール

ロール名	ロールのメンバー	ロールの特徴
顧客ロール	ExampleBank のすべての顧客	顧客をグループ化する管理されているロール
契約業者ロール	ExampleBank のすべての契約業者	すべての契約業者をグループ化する管理されているロール。このロールは、従業員と顧客の特定の機密属性に対するアクセスを制限する
従業員ロール	ExampleBank のすべての従業員	従業員をグループ化する管理されているロール
電話バンキング担当者ロール	電話バンキングサービスの実行に関連するすべての電話バンキング担当者	すべての電話バンキング担当者をグループ化し、ユーザーエントリの `ebPhoneBankingLoanServicesEnabled` 属性以外のすべての電話バンキング属性に対するアクセスを許可する管理されているロール
信頼されている貢献者ロール	ExampleBank のすべての従業員と、従業員電話帳情報へのアクセスを必要とする特定の契約事業者	従業員電話帳データへのアクセスを必要とする契約業者を含めるように従業員ロールの範囲を拡張する入れ子のロール。このロールのメンバーは、従業員電話帳データに対する読み取り、検索、比較アクセス権を持つ
電子バンキング担当者ロール	電子バンキングサービスの実行に関連するすべての電子バンキング担当者	すべての電子バンキング担当者をグループ化し、ユーザーエントリの `ebEBankingLoanServicesEnabled` 属性以外のすべての電子バンキング属性に対するアクセスを許可する管理されているロール

表 9-8 ExampleBank のユーザー管理を活用するために実装されるロール (続き)

電話バンキングローン責任者ロール	ローンの配分を決定するすべての電話バンキング責任者	電話バンキングのすべての上級管理者をグループ化し、ebPhoneBankingLoanServicesEnabled 属性を含むすべての電話バンキング属性へのアクセスを許可する管理されているロール
電子バンキングローン責任者ロール	ローンの配分を決定するすべての電子バンキング責任者	電子バンキングのすべての上級管理者をグループ化し、ebEBankingLoanServicesEnabled 属性を含むすべての電子バンキング属性へのアクセスを許可する管理されているロール
ローン管理者ロール	ローンの配分を決定するすべての電話バンキング責任者と電子バンキング責任者	電話バンキングローン責任者ロールと電子バンキングローン責任者ロールを統合する入れ子のロール。このロールのメンバーは、ebPhoneBankingLoanServicesEnabled 属性と ebEBankingLoanServicesEnabled 属性に対するアクセス権を持つ

表 9-8 に示されるロールは、それぞれの所属に応じてディレクトリ情報ツリーの o=enterprise、o=customer、o=partner 分岐の ou=people サブツリーの下に配置されます。これは、ロールとロール内の属性に対するアクセス権を持つロールマネージャエントリによって管理されます。このロールマネージャエントリは、o=enterprise の下に配置されます。各ロールとロールマネージャのアクセス権は、アクセス制御によって管理されます。

コード例 9-1 は、ldif ファイル内の信頼されている貢献者ロールのエントリを示しています。このロールの機能がどのように実装されるのかをより詳細に示しています。

コード例 9-1 信頼されている貢献者ロールエントリの ldif

```
dn:cn=TrustedContributorRole,ou=people,o=enterprise,
dc=ExampleBank,dc=com
objectclass:top
objectclass:LDAPsubentry
objectclass:nsRoleDefinition
objectclass:nsComplexRoleDefinition
objectclass:nsNestedRoleDefinition
cn: TrustedContributorRole
nsRoleScopeDN: o=partners,dc=ExampleBank,dc=com
nsRoleDN: cn=EmployeeRole,o=enterprise,dc=ExampleBank,dc=com
nsRoleDN: cn=ContractorRole,o=partners,dc=ExampleBank,dc=com
```

この信頼されている貢献者ロールのすべてのメンバーに、電話帳データに対する読み取り、検索、比較アクセス権を与えるアクセス制御は、コード例 9-2 に示されるように、o=enterprise,dc=ExampleBank,dc=com の下に指定されます。

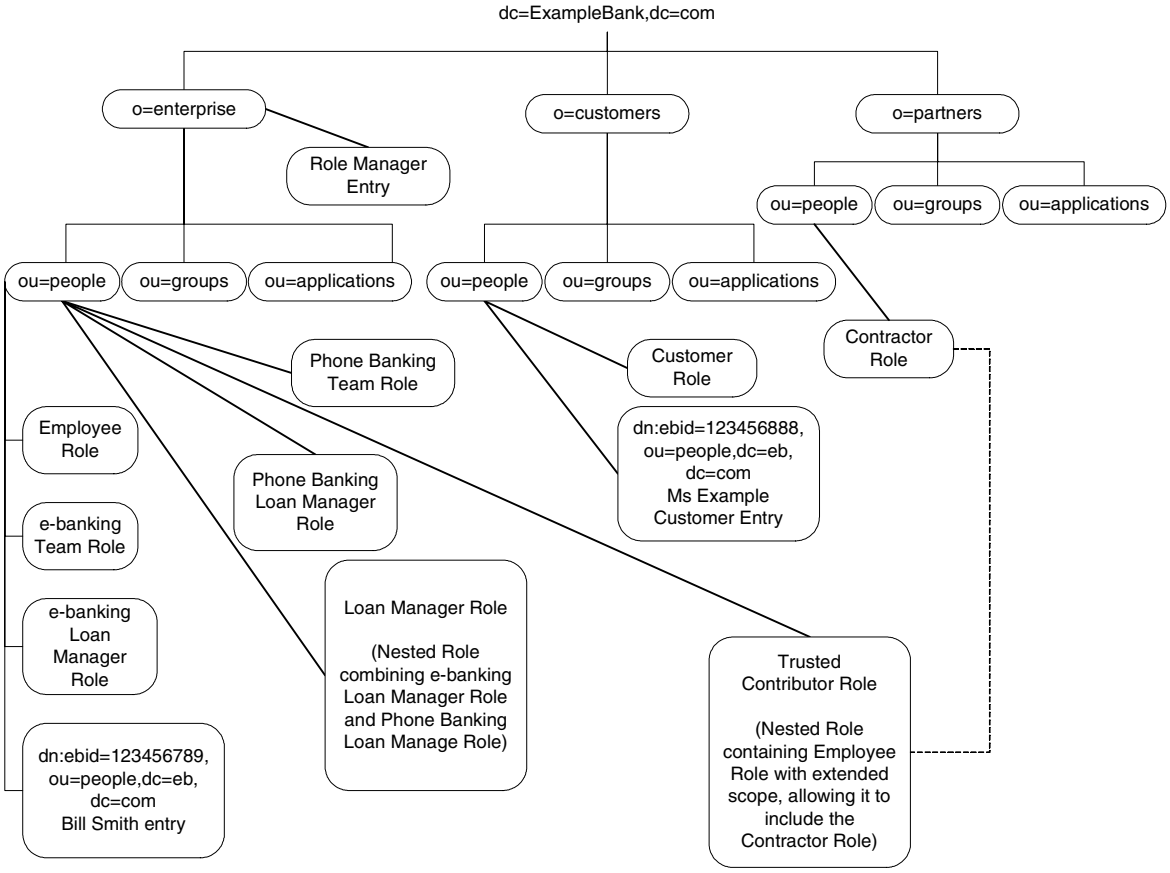
コード例 9-2 信頼されている貢献者ロールのすべてのメンバーに電話帳データに対する読み取り、検索、比較アクセス権を与えるアクセス制御

```
aci:(targetattr="telephoneNumber || mail  ||facsimileTelephonenumber")(version
 3.0; aci "authorize for search,read,compare";allow(search,read,compare)
 roledn = "ldap:///cn=TrustedContributorRole,ou=people,o=enterprise,
 dc=ExampleBank,dc=com";)
```

注 このロール機能の範囲拡張は、Directory Server 5.2 の新機能です。Sun ONE Identity Server 6.0 はこの新機能を認識しません。

251 ページの図 9-4 は、ロール設定のグローバルビューと、それがどのようにディレクトリ情報ツリー構造に組み込まれているかを示しています。

図 9-4　ExampleBank のロール、ロールマネージャ、およびサンプルエントリ



セキュリティ上の注意点

金融業界では、危険にさらされないセキュリティを提供することが成功の鍵となります。オンラインバンキング配備スタックの一部に Directory Server を使用することで、ExampleBank はチャンネルの保護、データに対するアクセスの制御、保管中機密データの暗号化、柔軟なパスワードポリシーの提供、SSL 接続の最適化において、最適なセキュリティを提供できます。

ここでは、ExampleBank のセキュリティポリシーについて次の内容を説明します。

- 通信チャンネルのセキュリティ保護
- 保管中データのセキュリティ保護
- パスワード認証のセキュリティ保護

通信チャンネルのセキュリティ保護

ExampleBank の第一の責務は、配備スタック内要素間のすべての通信チャンネルを確実にセキュリティ保護することです。このために、ExampleBank はデータ転送を確実にセキュリティ保護する SSL を実装します。さらに、証明書ベースの認証を行う SSL (Secure Sockets Layer) プロトコルを使用する通信のパフォーマンスを向上するために、ExampleBank は Directory Server 5.2 の新機能である Sun Crypto アクセラレータボードを利用します。Sun Crypto アクセラレータボードのインストール方法と設定方法については、『Sun ONE Directory Server インストールおよびチューニングガイド』の付録 B 「Using the Sun Crypto Accelerator Board」を参照してください

保管中データのセキュリティ保護

第二の責務は、暗証番号などの、保管中の機密データを最大限に保護することです。ExampleBank は、Sun ONE Directory Server 5.2 の属性暗号化機能を使用してデータを保護します。この属性暗号化機能では、どの属性を暗号化形式で格納するかを ExampleBank が決定できます。この機能は、データベースレベルで設定されます。つまり、ExampleBank が属性の暗号化を決定すると、データベース内のすべてのエントリでその属性が暗号化されます。暗号化された上で格納されている属性には、適用された暗号化アルゴリズムを示す暗号化方式タグが最初につけられます。DES 暗号化アルゴリズムを使用して暗号化された属性は、次のように表示されます。

```
{DES}3hakc&jla+=snda%
```

セキュリティの面では、実際の暗号化がエントリレベルではなく属性レベルで行われるため、エントリ全体を暗号化するには、ExampleBank はそのエントリのすべての属性を暗号化しなければならない、ということが重要です。また、属性暗号化の目的は保管中の機密データの保護であるため、属性暗号化は常に可逆的であることにも注意してください。つまり、暗号化された属性は、検索要求の結果として返されるときは復号化されるため、通信チャンネルを SSL でセキュリティ保護する必要があります。

Sun ONE Directory Server 5.2 の属性暗号化機能、ExampleBank が必要とするレベルのセキュリティを提供します。この機能では、サーバーの SSL 証明書の秘密鍵を使用して専用の鍵が作成され、暗号化と復号化にはこの鍵が使用されます。このため、ExampleBank は暗号化とさらに重要な復号化を行う前に、鍵を生成するために SSL 設定を行う必要があります。Directory Server の属性暗号化機能が NSS ライブラリに基づいていることは、ExampleBank にさらに有利に働きます。さまざまな暗号化アルゴリズムを選択することが可能となり、異なるプラットフォーム間での可搬性も保証されます。

パスワード認証のセキュリティ保護

ExampleBank のユーザーは従業員、顧客、契約業者、パートナー企業で、全員がサードパーティ製のシングルサインオンアプリケーションを通じてオンラインバンキングスタックへの認証を行います。ExampleBank の希望は、契約業者などの一部のユーザーのパスワードポリシーをより厳しくし、顧客と従業員に適用されるパスワードポリシーをより緩いものとすることでした。サードパーティ製シングルサインオンアプリケーションが Sun ONE Directory Server 5.2 の複数パスワードポリシー機能を利用することで、この希望はかなえられます。指定したユーザーまたはユーザーセットに個別にパスワードポリシーを定義することができます。ユーザーセットに適用するパスワードポリシーを割り当てる一般的な方法は、CoS 定義を設定し、ユーザーエントリが持つロールの機能として、ユーザーエントリ内の `passwordPolicySubentry` 属性に値を指定する方法です。

このため、ExampleBank はユーザーのセキュリティ要件に合わせてパスワードポリシーを調整できるだけでなく、すでに定義されているロールを使用することでパスワードポリシーを簡単に管理できます。

実装

Sun ONE Directory Server オンラインバンキングスタックを世に送り出すことは、大規模な事業であり、広範な計画、分析、組織的なサポートを必要とします。

注

ユーザープロファイルのニーズ情報を作成するマーケティング担当者からシステム設計者にいたるまで、関連するすべての担当者の間に緊密なコーディネーションが必要であることは、いくら強調しても足りません。共通のアーキテクチャ戦略を描き、集中型の意思決定構造の利点を得ることができる、この緊密なコーディネーションこそが実装を成功させる鍵となります。実質的な現状維持の再作成を避けるには、作業全体を通じてこの緊密なコーディネーションが最優先であり続ける必要があります。

Sun ONE サポート担当者とお客様の間で適切なトレーニングと文書作成を行うことは、納入物の一貫性を保ち、この緊密なコーディネーションの基礎を築く上で役立ちます。

実装前に、試験的なディレクトリ作成と実装の概要について、『Understanding and Deploying LDAP Directory Services』(T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年) を参照することを強くお勧めします。

実装は複雑であるため、段階的なアプローチが必要です。次に、このアプローチの概要を説明します。実装では、時間の経過とともに論理段階が切り替わります。これらの段階は、およそ次のようになります。

- ディレクトリインフラストラクチャの分析と計画

  この段階では、Directory Server インフラストラクチャの機能要件とビジネス要件を評価、分析します。この分析は、システムのカスタマイズまたは拡張に必要な機能の特定に役立ちます。既存の実装の運用環境を技術面から見直すことで、提案されているハードウェアおよびネットワーク環境でのスケーラビリティ、パフォーマンス、信頼性に関する潜在的な問題を特定できます。

  上の分析に基づいて高度なアーキテクチャ定義が行われ、推奨されるサイズ、スケーリング、パフォーマンス、物理的な分散、レプリケーションとリフェラル、セキュリティ、フェイルオーバー、バックアップ、他のデータソースとの同期、認証メカニズムはこの段階で決定されます。

- ディレクトリインフラストラクチャの設計と構築

  第 2 段階では、Sun ONE と ExampleBank プロジェクトチームの設計者が共同でコアオンラインバンキング配備スタックを設計、構築します。この段階で重要なアクティビティには、サーバーのサイズと配置の決定、サーバーの設定、既存のバックエンドアプリケーションとディレクトリインフラストラクチャの統合、配備プロセスと管理プロセスの計画、追加の必要機能の特定が含まれます。

- コアディレクトリインフラストラクチャの実装

  この段階では、Sun ONE はコアディレクトリインフラストラクチャを運用環境に実装、配備します。設定と管理に関する適切な技術指導および教育が提供されることが重要です。プロジェクトチームは、Sun ONE と共同でエンドユーザーに与える影響を評価し、必要な連絡とトレーニングの計画を策定します。

- E2E (Enterprise to Employee) 機能の有効化

  最初に有効にする機能は、E2E (Enterprise to Employee) 機能です。通常、この準備では次の作業が行われます。

  - LDAP アクセス可能ユーザーリストの作成
  - 認証ソースのマッピングとユーザーデータのクリーンアップ
  - 標準 ID の自動生成
  - フラグによるアカウントの停止
  - LDAP アプリケーションの有効化
  - イントラネットアプリケーションによる認証

  上のような操作環境が整ったら、E2E 機能の詳細を設定します。別のデータリポジトリとのリンクを自動化し、段階的な拡張を行います。パスワード同期などのセキュリティ要件も解決します。

  最終段階直前のこの段階では、すべての Web アプリケーションとバックエンドアプリケーションで、スケーラブルなシングルサインオンアーキテクチャを利用できるようにすることを目指します。次に、実装テストを行います。このテストには、統合、パフォーマンス、E2E 機能の否定テストとユーザー承認テストを含める必要があります。適切なトレーニングと連絡を準備するために、シングルサインオンアーキテクチャに関連するエンドユーザーへの影響をもう一度評価します。

  プラットフォームの Web アプリケーションとバックエンドアプリケーションのシングルサインオンが Directory Server インフラストラクチャで有効化されたら、企業内の他のアプリケーションとビジネスグループの同様の E2E 機能の実装に役立つように、プロジェクトチームは手順を文書化します。

- コア Directory Server インフラストラクチャの拡張による追加共通サービスのサポート

  目的は、追加の共通サービスおよび機能のサポートを拡張し続け、さらなる E2E、B2B (Business to Business)、B2C (Business to Consumer) などの要件に対応することです。各部署でビジネス要件が決定、承認された時点で、B2B や B2C に固有の機能を定義、設計、実装します。

このように、Directory Server の配備スタックを公開する作業は簡単なものではありません。最適な実装のためにも、Sun ONE Professional Service との連携を強くお勧めします。Sun ONE Professional Service の連絡先は、
`http://jp.sun.com/service/sunps/sunone/` です。

第 10 章

アーキテクチャ戦略

ディレクトリの配備を計画するときは、いくつかの事項を考慮する必要があります。最も重要な事項には、データの物理的な配置場所、このデータをどこに、また、どのようにレプリケートするか、障害を最小化するにはどうすればよいか、障害が発生した場合にどのように対応するか、などが含まれます。この章で説明するアーキテクチャ戦略は、これらの事項について考える上でのガイドラインとなります。

この章は、次の節から構成されています。

- 障害と復元について
- バックアップ戦略の策定
- レプリケーショントポロジの例

障害と復元について

障害発生時にサービスの中断を最小限にとどめるための戦略を準備しておくことが重要です。ここでいう障害とは、必要とされる最小限のサービスを Directory Server が提供できなくなる原因として定義されます。ここでは、配備で発生する障害の原因を特定し、障害に迅速に対応できるように、障害のさまざまな発生原因について説明します。

障害は、主に次の 2 つに分けられます。

- システムが使用できなくなる
- システムが信頼できなくなる

システムが使用できなくなることには、次のような原因があります。

- ネットワークの問題 : ネットワークが停止している、速度が低下している、または断続的になる
- プロセス (slapd) の問題 : プロセスが停止している、ビジーである、再起動している、または動作不良を起こす

- ハードウェアの問題 : ハードウェアが稼動していない、障害が発生した、または再起動している

システムが信頼できなくなることには、次のような原因があります。

- レプリケーションに失敗または遅延が生じ、データが古くなったり、同期がとれなくなる
- システムが過度のビジー状態になる : 読み取りまたは書き込み操作が過剰に行われ、データの信頼性が失われる

書き込み可能なサーバーが利用できなくなった場合に、ディレクトリ内のデータを追加、変更する機能を維持するには、書き込み操作が代替サーバーを経由する必要があります。書き込み操作のルーティングにはさまざまな方法があり、Sun ONE Directory Proxy Server の使用もそれに含まれます。

ディレクトリ内のデータを読み取る機能を維持するには、適切なロードバランス戦略を導入する必要があります。読み取りの負荷を複数のコンシューマレプリカに分散するには、ソフトウェアとハードウェアの両方のロードバランスソリューションを利用できます。それぞれのソリューションには、各レプリカの状態を特定し、ロードバランストポロジの中でどのような役割を果たすべきかを管理する機能 (完全性と精度はそれぞれ異なります) があります。

ディレクトリの内容をレプリケートすると、Directory Server の可用性とパフォーマンスが向上します。信頼できるレプリケーショントポロジを構築することで、障害が発生した場合でも、データセンターにアクセスするすべてのクライアントは最新のデータに確実にアクセスできます。

次に、読み取り操作と書き込み操作のための障害戦略について説明します。これは、レプリケーショントポロジにも関連します。

バックアップ戦略の策定

データの破損や喪失をともなう障害では、データの最新のバックアップが不可欠になります。最新のバックアップが得られない場合、障害が発生したマスターを別のマスターから初期化し直す必要があります。データをバックアップするための包括的な手順については、『Sun ONE Directory Server 管理ガイド』の「Backing Up Data」を参照してください。

ここでは、バックアップと復元の戦略を計画する上で考慮すべき事項について簡単に説明します。

バックアップ方法の選択

Sun ONE Directory Server では、バイナリバックアップ (`db2bak`) と、ldif ファイルへのバックアップ (`db2ldif`) という 2 つの方法でデータをバックアップできます。どちらの方法にも利点と制限があるため、効果的なバックアップ戦略を計画するには、それぞれの方法について理解することが役立ちます。

バイナリバックアップ (db2bak)

バイナリバックアップは、ファイルシステムレベルで行われます。バイナリバックアップの出力は、すべてのエントリ、インデックス、更新履歴ログ、トランザクションを含むバイナリファイルのセットです。

注 バイナリバックアップでは、`dse.ldif` 設定ファイルはバックアップされません。失われた設定情報を復元するには、このファイルを手動でバックアップする必要があります。

バイナリバックアップには、次のような利点があります。

- すべてのサフィックスを一度にバックアップできる
- ldif へのバックアップと比較して、バイナリバックアップは格段に高速である

バイナリバックアップには、次のような制約があります。

- バイナリバックアップからの復元は、同じ設定のサーバーだけで実行できる。つまり、次の制約が適用される
  - 両方のマシンが同じハードウェア、同じオペレーティングシステム (サービスパックやパッチも含まれる) を使用している必要がある

- 両方のマシンに同じバージョンの Directory Server (バイナリ形式 (32 ビットまたは 64 ビット)、サービスパック、パッチも含まれる) がインストールされている必要がある
- 両方のサーバーは、同じサフィックスに分岐する同じディレクトリツリーを持つ必要がある。すべてのサフィックスのデータベースファイルをまとめてコピーする必要があり、サフィックスを個別にコピーすることはできない
- 両方のサーバーの各サフィックスには同じインデックス (VLV (仮想リスト表示) インデックスも含まれる) が設定されている必要がある。サフィックスのデータベースの名前は同じである必要がある
- コピーされる Directory Server は、o=NetscapeRoot サフィックスを保持していてはならない。つまり、Sun ONE 管理サーバーの設定ディレクトリであってはならない
- 各サーバーでは、同じサフィックスがレプリカとして設定されている必要があり、両方のサーバーでレプリカに同じ役割 (マスター、ハブ、コンシューマ) が設定されている必要がある。部分レプリケーションが設定されている場合は、すべてのマスターサーバーが同じように設定されている必要がある
- どちらのサーバーでも、属性の暗号化を使用していてはならない

バイナリ復元機能によるデータの復元については、『Sun ONE Directory Server 管理ガイド』の「Initializing a Replica Using Binary Copy」を参照

互いに連携するマシンの各セット (上で定義した同じ設定を持つマシン) で、少なくとも定期的にバイナリバックアップを行う必要があります。

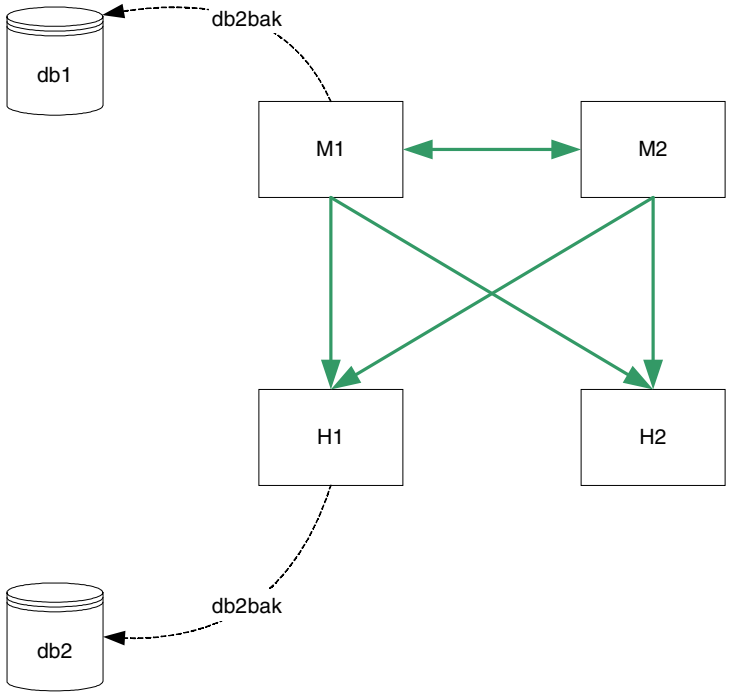
注 ローカルバックアップからの復元のほうが簡単なので、各サーバーでバイナリバックアップを行うことをお勧めします。

この章に示される図で使用される略語の意味は次のとおりです。

- M = マスター
- H = ハブ
- C = コンシューマ
- RA = レプリケーションアグリーメント

図 10-1 は、M1 と M2 が同じ設定を持ち、H1 と H2 が同じ設定を持つことを前提としています。この例では、M1 と H1 でバイナリバックアップが行われます。障害が発生した場合、いずれかのマスターを M1 (db1) のバイナリバックアップから復元し、いずれかのハブを H1 (db2) のバイナリバックアップから復元することができます。H1 のバイナリバックアップからマスターを復元したり、M1 のバイナリバックアップからハブを復元することはできません。

図 10-1　バイナリバックアップ



LDIF (db2ldif) へのバックアップ

ldif へのバックアップはサフィックスレベルで行われます。`db2ldif` の出力は、フォーマットされた ldif ファイルです。このため、このプロセスはバイナリバックアップと比較して時間がかかります。

注

`db2ldif` の実行時に `-r` オプションを指定しない限り、レプリケーション情報はバックアップされません。

ldif へのバックアップでは、`dse.ldif` 設定ファイルはバックアップされません。失われた設定情報を復元するには、このファイルを手動でバックアップする必要があります。

ldif へのバックアップには、次のような利点があります。

- ldif へのバックアップは、設定に関係なくどのサーバーからも実行できる

- ldif バックアップからの復元は、設定に関係なくどのサーバーからも実行できる (-r オプションを指定してレプリケーション情報がエクスポートされている場合)

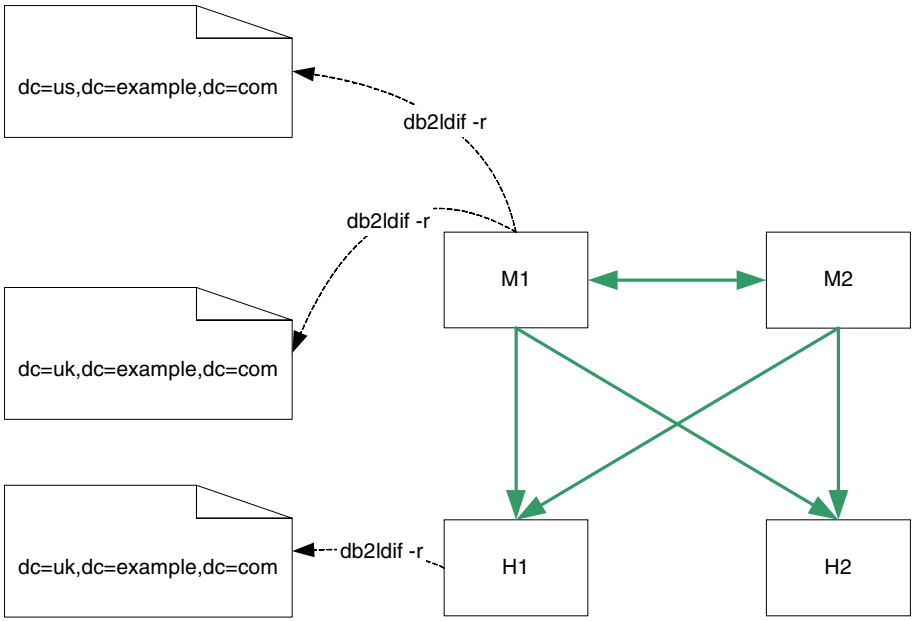
ldif へのバックアップには、次のような制約があります。

- 迅速なバックアップと復元が必要な状況では、ldif へのバックアップでは時間がかかり過ぎる可能性がある

トポロジの単一マスターで、レプリケートされた各サフィックスを ldif に定期的にバックアップすることをお勧めします。

次の図では、M1 だけ、または M1 と H1 のレプリケートされた各サフィックスに対して `db2ldif -r` を実行しています。

図 10-2　db2ldif -r によるバックアップ



警告 パージ遅延より頻繁にバックアップを行うことが重要です。`nsDS5ReplicaPurgeDelay` 属性によって指定されるパージ遅延は、更新履歴ログに対して内部パージ操作を開始するまでの期間 (秒単位) です。デフォルトのパージ遅延は 604800 秒 (1 週間) です。更新履歴ログは、レプリケートが完了している、またはレプリケートが完了していない更新の記録を保持しています。

更新の頻度がパージ遅延より低い場合、バックアップを行う前に更新履歴ログの内容がクリアされてしまう可能性があります。この場合、バックアップからデータを復元しようとしても、変更は失われています。

復元方法の選択

Sun ONE Directory Server では、バイナリ復元 (`bak2db`) と、ldif ファイルからの復元 (`ldif2db`) という 2 つの方法でデータを復元できます。すでに説明したバックアップ方法と同様に、どちらの方法にも利点と制限があります。

バイナリ復元 (bak2db)

バイナリ復元では、データベースレベルでデータがコピーされます。このため、バイナリ復元によるデータの復元には、次のような利点があります。

- すべてのサフィックスを一度に復元できる
- ldif ファイルからの復元と比較して、バイナリ復元は格段に高速である

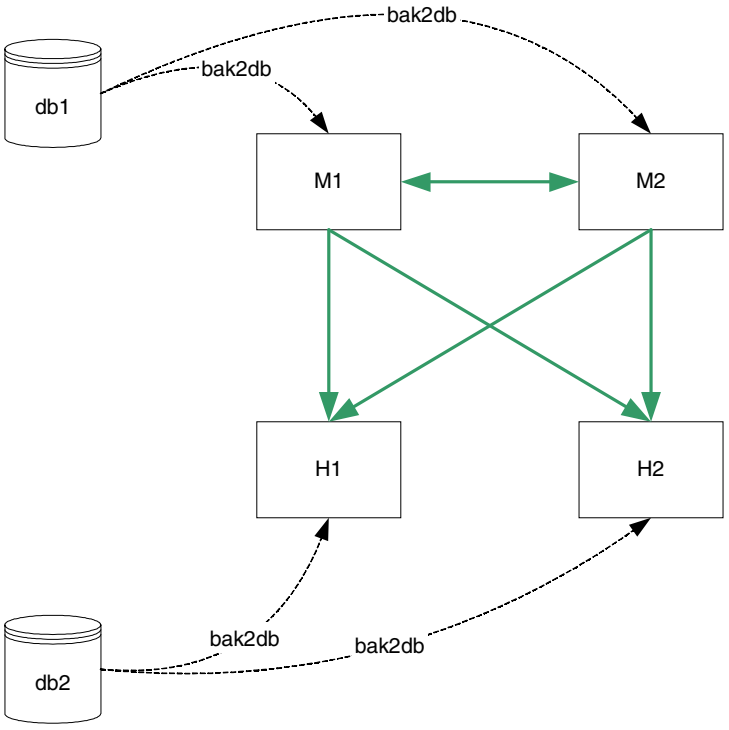
バイナリ復元によるデータの復元には、次のような制約があります。

- 復元は、同じ設定を持つサーバー (259 ページの「バイナリバックアップ (db2bak)」を参照) だけで実行できる。バイナリ復元機能によるデータの復元については、『Sun ONE Directory Server 管理ガイド』の「Initializing a Replica Using Binary Copy」を参照
- バイナリバックアップではデータベースの同一コピーが作成されるため、データベースが破損していることに気付かずにバイナリバックアップを行った場合、破損したデータベースが復元される危険性がある

マシンの設定が同一であり、実行時間に特に考慮が必要な場合、推奨される復元方法はバイナリ復元になります。

図 10-3 は、M1 と M2 が同じ設定を持ち、H1 と H2 が同じ設定を持つことを前提としています。この例では、いずれかのマスターを M1 (db1) のバイナリバックアップから復元し、いずれかのハブを H1 (db2) のバイナリバックアップから復元することができます。

図 10-3　バイナリ復元



LDIF からの復元 (ldif2db)

ldif ファイルからの復元は、サフィックスレベルで行われます。このため、このプロセスはバイナリ復元と比較して時間がかかります。ldif ファイルからの復元には、次のような利点があります。

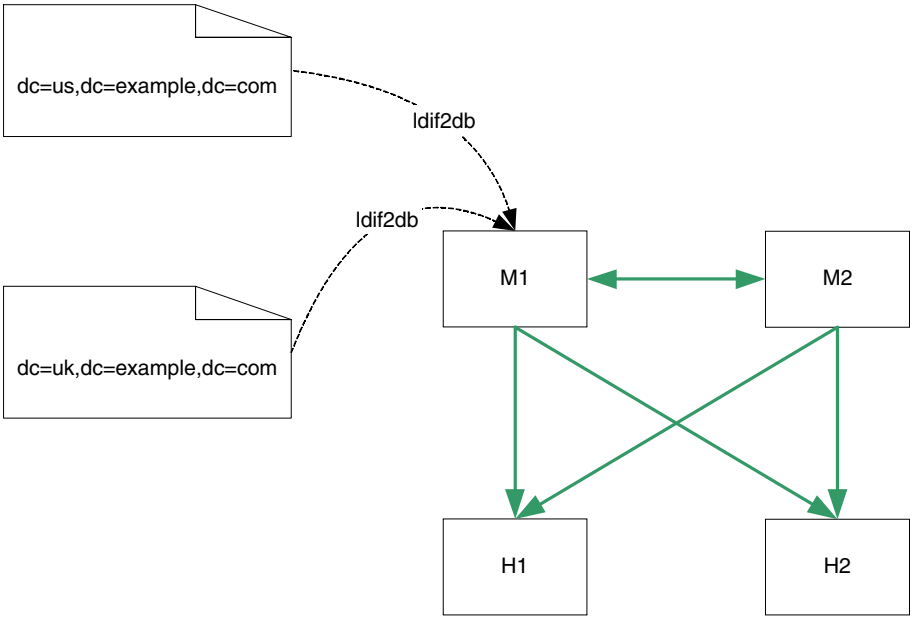
- ldif からの復元は、設定に関係なくどのサーバーからも実行できる
- レプリケーショントポロジに関係なく、ディレクトリサービス全体の配備に単一の ldif ファイルを使用できる。予定されているビジネスニーズに合わせてディレクトリサービスをダイナミックに拡張または縮小する場合に、これは特に便利である

ldif ファイルからの復元には、次のような制限があります。

- 迅速な復元が必要な状況では、`ldif2db` の実行は時間がかかり過ぎる場合がある

次の図では、M1 だけ、または M1 と H1 のレプリケートされた各サフィックスに対して `ldif2db` を実行しています。

図 10-4　ldif2db による復元



レプリケーショントポロジの例

レプリケーショントポロジは、企業の規模と、データセンターの物理的な場所によって決定されます。このため、ここで紹介するレプリケーショントポロジの例は、企業がディレクトリを維持するデータセンター (サイト) の数によって分けられています。

ディレクトリを最初に配備するときは、エントリの現在の数と、ディレクトリへの読み取りおよび書き込み操作の現在のボリュームに基づいて配備を行います。エントリ数が増えると、読み取りパフォーマンス向上のためにディレクトリのスケーリングが必要になる場合があります。それぞれの企業に合わせたスケーラビリティが提案されます。

これらのトポロジは、1 つのコンポーネントで障害が発生した場合にも、迅速な人的対応なしでサービスを提供し続けることを目的としています。1 つまたは 2 つのデータセンターで、読み取りと書き込みのフェイルオーバーもローカルに行われます。

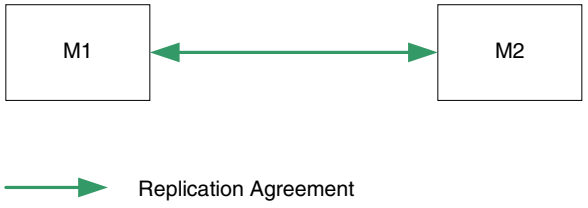
1 つのデータセンター

1 つのデータセンターのトポロジには、ディレクトリの想定パフォーマンス要件が大きく影響します。提案される基本的なトポロジでは、読み取りと書き込みの操作を処理するために、少なくとも 2 つのサーバーの動作が保証される配備が想定されています。2 つのマスターによって、高可用性ソリューションも提供されます。

1 つのデータセンターの基本トポロジ

図 10-5 に示されるトポロジは、読み取りと書き込みのパフォーマンスのために 2 つのマスターを持つ 1 つのデータセンターを示しています。この基本例では、クライアントはいずれかのマスターに書き込みを行い、いずれかのマスターから読み取りを行います。

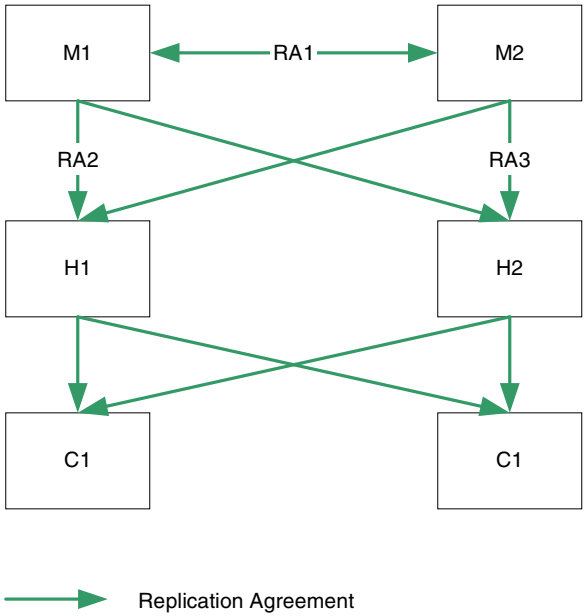
図 10-5　1 つのデータセンター : 基本トポロジ



読み取りパフォーマンスのための 1 つのデータセンターのスケーリング

図 10-6 に示されるように、ハブとコンシューマを追加することで、読み取りパフォーマンスが向上します。第 3 レベルのコンシューマを簡単に追加できるように、まず、マスターの下にハブを追加します。第 2 レベルのサーバーをハブとして設定することで、マシンを再設定する必要なく、その下にコンシューマを追加できます。

図 10-6　読み取りパフォーマンスのための 1 つのデータセンターのスケーリング



1 つのデータセンターの障害マトリックス

図 10-6 の例では、257 ページの「障害と復元について」に示されるいずれかの原因により、さまざまなコンポーネントが使用不可能になる可能性があります。表 10-1 は、これらの障害と、それに対応する復元処理を示しています。

表 10-1　1 つのデータセンター : 障害マトリックス

障害が発生したコンポーネント	対策
M1	Sun ONE Directory Proxy Server、クライアントサーバーリスト、ハードウェアまたはソフトウェアのロードバランサにより、ローカル書き込みは M2 にルーティングされる。M2 は、H1、H2 へのレプリケーションを継続する。
M2	Sun ONE Directory Proxy Server、クライアントサーバーリスト、ハードウェアまたはソフトウェアのロードバランサにより、ローカル書き込みは M1 にルーティングされる。M1 は、H1、H2 へのレプリケーションを継続する。

表 10-1　1 つのデータセンター : 障害マトリックス (続き)

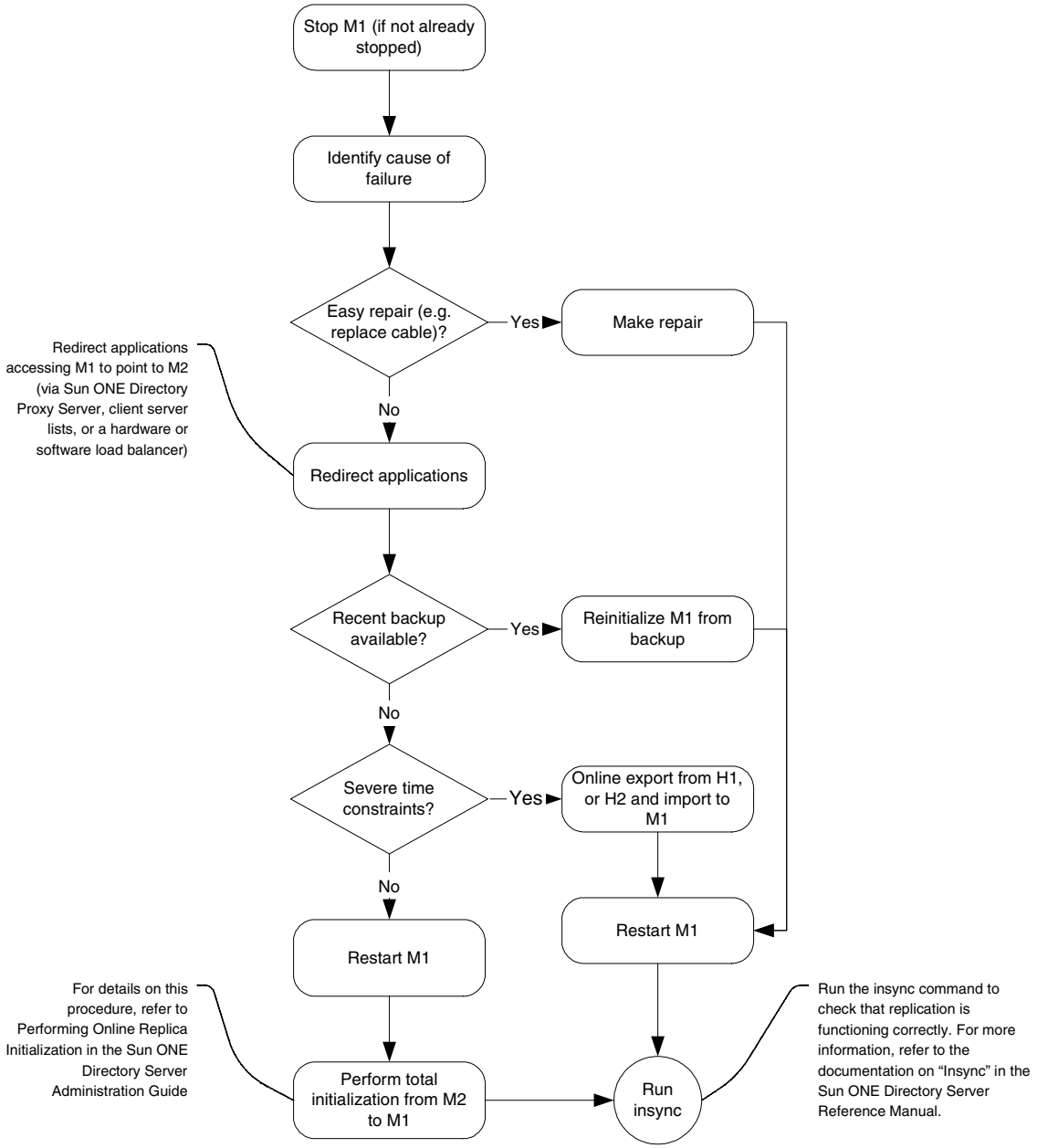
障害が発生したコンポーネント	対策
RA1 の LAN リンク	どちらのマスターのローカル書き込みも受け付ける。コンシューマに同じデータが含まれるように、競合の解決はハブレベルで行われる
H1 または H2	どちらのマスターのローカル書き込みも受け付ける。コンシューマに同じデータが含まれるように、競合はマスターレベルで解決され、正常に稼動するほうのハブを通じてすべてのコンシューマにレプリケートされる
RA2 の LAN リンク	どちらのマスターのローカル書き込みも受け付ける。H1 へのレプリケーションは M2 から行われ、ハブからコンシューマへのレプリケーショントラフィックは正常に機能し続ける

1 つのデータセンターの復元手順 (1 つのコンポーネント)

2 つのマスターを持つ 1 つのデータセンターでは、1 つのマスターで障害が発生しても読み取りと書き込みの機能は維持されます。ここでは、障害が発生したコンポーネントの復元に適用できる復元戦略の例について説明します。

図 10-7 のフローチャートとその後の手順は、1 つのマスター (M1) で障害が発生したことを前提としています。

図 10-7　1 つのデータセンターの復元手順の例 (1 つのコンポーネント)



1. M1 を停止します (すでに停止していない場合)。
2. 障害の原因を特定します。たとえばネットワークケーブルの交換などによって簡単に修復できる場合は、修復します。
3. 問題がより重大で修復に時間がかかる場合は、M1 にアクセスするアプリケーションが、Sun ONE Directory Proxy Server、クライアントサービスリスト、ハードウェアまたはソフトウェアのロードバランサを通じて M2 にリダイレクトされることを確認します。
4. 最新のバックアップがあれば、バックアップから M1 を初期化し直します。
5. 最新のバックアップを利用できない場合は、M1 を再起動し、M2 から M1 への完全初期化を行います。詳細な手順については、『Sun ONE Directory Server 管理ガイド』の「Performing Online Replica Initialization」を参照してください。
6. 最新のバックアップを利用できず、完全初期化を行う時間もない場合は、H1 または H2 からオンラインエクスポートを行い、M1 に `ldif2db` をインポートします。
7. M1 を起動します (すでに起動していない場合)。
8. M1 のモードが読み取り専用モードであれば、読み書き有効モードに設定します。
9. `insync` コマンドを使用して、レプリケーションが正常に機能していることを確認します。詳細については、『Sun ONE Directory Server Reference Manual』の「Insync」を参照してください。

注 オンラインエクスポートの実行はサーバーのパフォーマンスに影響します。このため、エクスポートにはマスター (M2) ではなく、その時点で書き込み操作を実行できる唯一のサーバーであるハブを使用することをお勧めします。

1 つのデータセンターの復元手順 (2 つのコンポーネント)

この例で 2 つのマスターに障害が発生した場合、書き込み機能は失われます。障害が重大で修復に時間がかかる場合は、できるだけ早急に書き込み機能を提供するための戦略を実装する必要があります。

次の手順は、M1 と M2 の両方に障害が発生し、すぐには復元できない状況を前提としています。最も迅速で、できるだけ簡単な復元方法を考える必要があります。この手順では、最も簡単な方法の例として、サーバーの昇格を紹介します。

1. H1 を書き込み可能なマスターに昇格させます。具体的な手順については、『Sun ONE Directory Server 管理ガイド』の「Promoting or Demoting Replicas」を参照してください。
2. M1 または M2 にアクセスしていたアプリケーションが新しいマスターにリダイレクトされることを確認します。

3. 変更がコンシューマにレプリケートされ続けるように、新しいマスターと H2 の間に新しいレプリケーションアグリーメントを追加します。

2 つのデータセンター

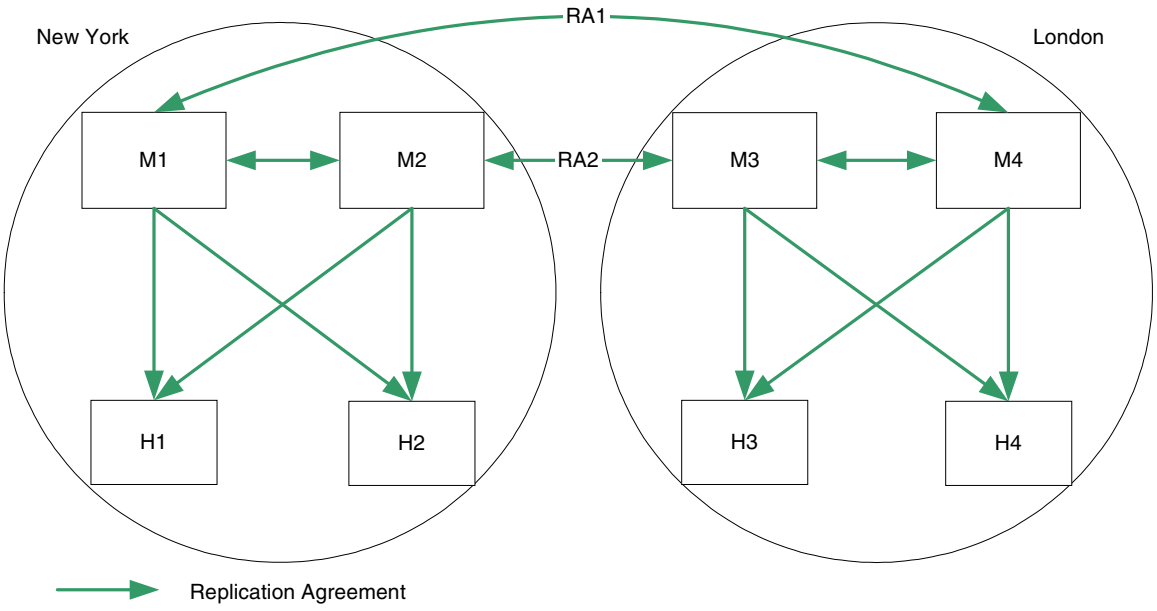
複数のサイトでデータが共有される場合は、パフォーマンスとフェイルオーバーの両方の面で効果的なレプリケーショントポロジが欠かせません。

2 つのデータセンターの基本トポロジ

図 10-8 に示されるトポロジは、最適な読み取り、書き込みパフォーマンスのために、各データセンターが 2 つのマスターと 2 つのハブを持つことを前提としています。第 2 レベルのサーバーをハブとして設定することで、マシンを再設定する必要なく、その下にコンシューマを追加できます。

この例では、レプリケーションアグリーメント RA1 と RA2 を異なるネットワーク経由で設定することをお勧めします。この設定であれば、いずれかのネットワークリンクが使用不可能になったり、信頼できなくなった場合でも、データセンター間のレプリケーションを行えます。

図 10-8　2 つのデータセンターの基本トポロジ

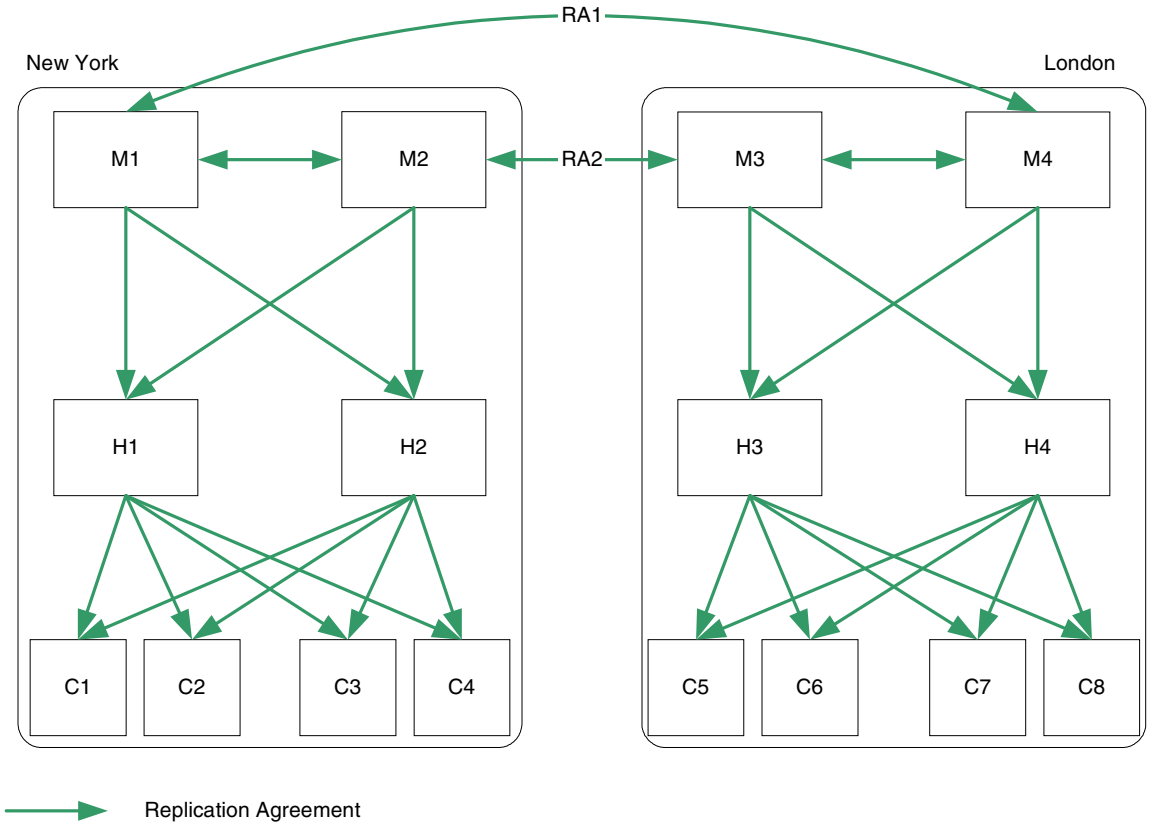


読み取りパフォーマンスのための 2 つのデータセンターのスケーリング

図 10-6 に示される 1 つのデータセンターの例のように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。

この例では、レプリケーションアグリーメント RA1 と RA2 を異なるネットワーク経由で設定することをお勧めします。この設定であれば、いずれかのネットワークリンクが使用不可能になったり、信頼できなくなった場合でも、データセンター間のレプリケーションを行えます。

図 10-9 読み取りパフォーマンスのための 2 つのデータセンターのスケーリング



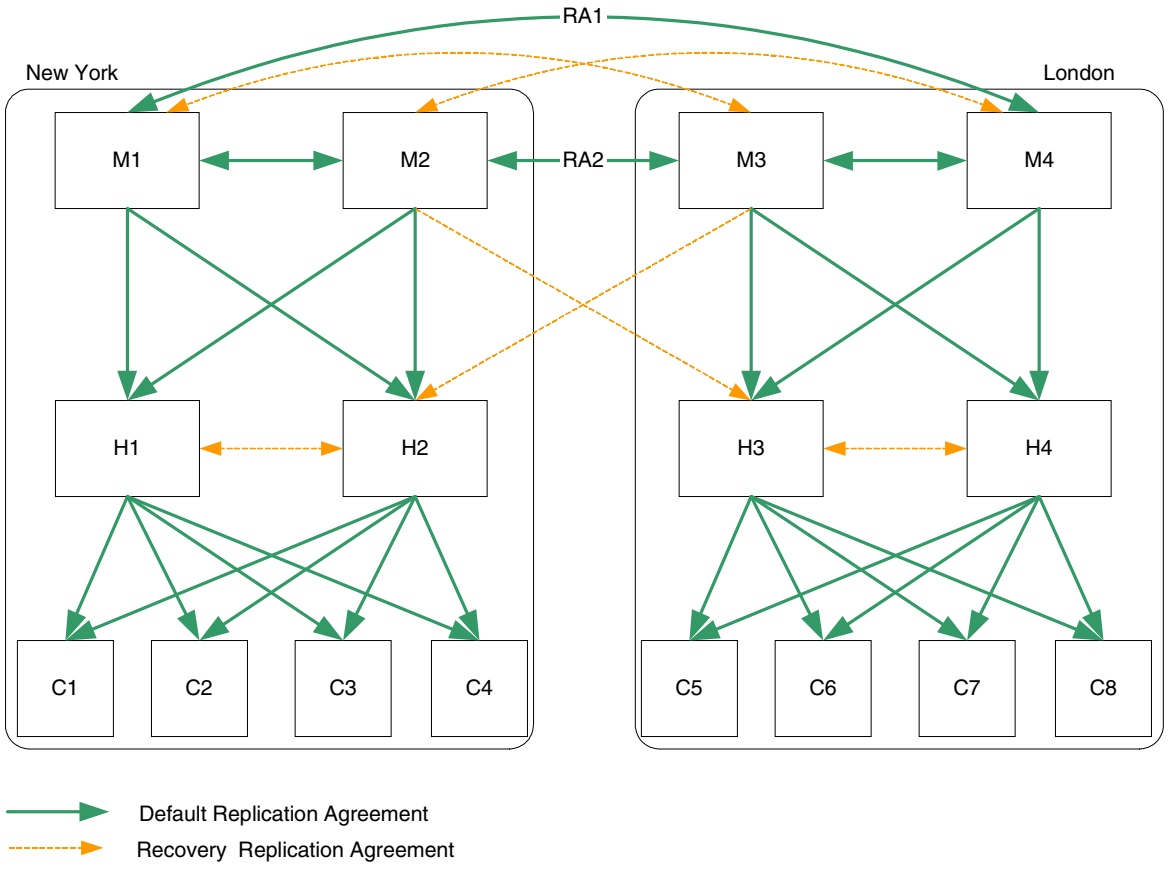
2 つのデータセンターの復元例

図 10-9 に示される配備では、1 つのマスターで障害が発生した場合に、1 つのデータセンターに適用した復元戦略をそのまま適用できます。いずれかのデータセンターの 1 つのデータセンターが使用不可能な状態になった場合でも、M1 と M4 の間、および M2 と M3 の間のレプリケーションアグリーメントにより、どちらのデータセンターもレプリケートされた更新を受け取り続けることができます。

しかし、複数のマスターで障害が発生した場合は、高度な復元戦略が必要となります。これには、復元アプリケーションアグリーメントの作成が関連します。このアグリーメントは、デフォルトでは無効化されていますが、障害が発生した場合には迅速に有効化できます。

図 10-10 は、この復元戦略を示しています。

図 10-10 2 つのデータセンターの復元レプリケーションアグリーメント



適用される復元戦略は、どのような組み合わせでコンポーネントに障害が発生するかによって異なります。しかし、複数の障害に対する基本的な戦略を準備しておけば、その他のコンポーネントに障害が発生した場合にもそれを適用できます。

図 10-10 のトポロジ例は、ニューヨークデータセンターの両方のマスターに障害が発生したことを前提としています。この場合の復元戦略は、次のようになります。

1. M3 と H2 の間の復元レプリケーションアグリーメントを有効化します。

   これにより、ロンドンサイトへのリモート書き込みが、引き続きニューヨークサイトにレプリケートされます。

2. H2 を書き込み可能なマスターに昇格させます。具体的な手順については、『Sun ONE Directory Server 管理ガイド』の「Promoting or Demoting Replicas」を参照してください。

   これにより、ニューヨークサイトでの書き込み機能が維持されます。

3. 新たに昇格したマスター (それまでの H2) と M3 の間にレプリケーションアグリーメントを作成します。

   これにより、ニューヨークサイトへのリモート書き込みが、引き続きロンドンサイトにレプリケートされます。

4. H2 と H1 の間の復元レプリケーションアグリーメントを有効化します (1 方向のみ)。

   これにより、H1 は、引き続きレプリケーショントポロジ全体からの更新を受け取ることができます。

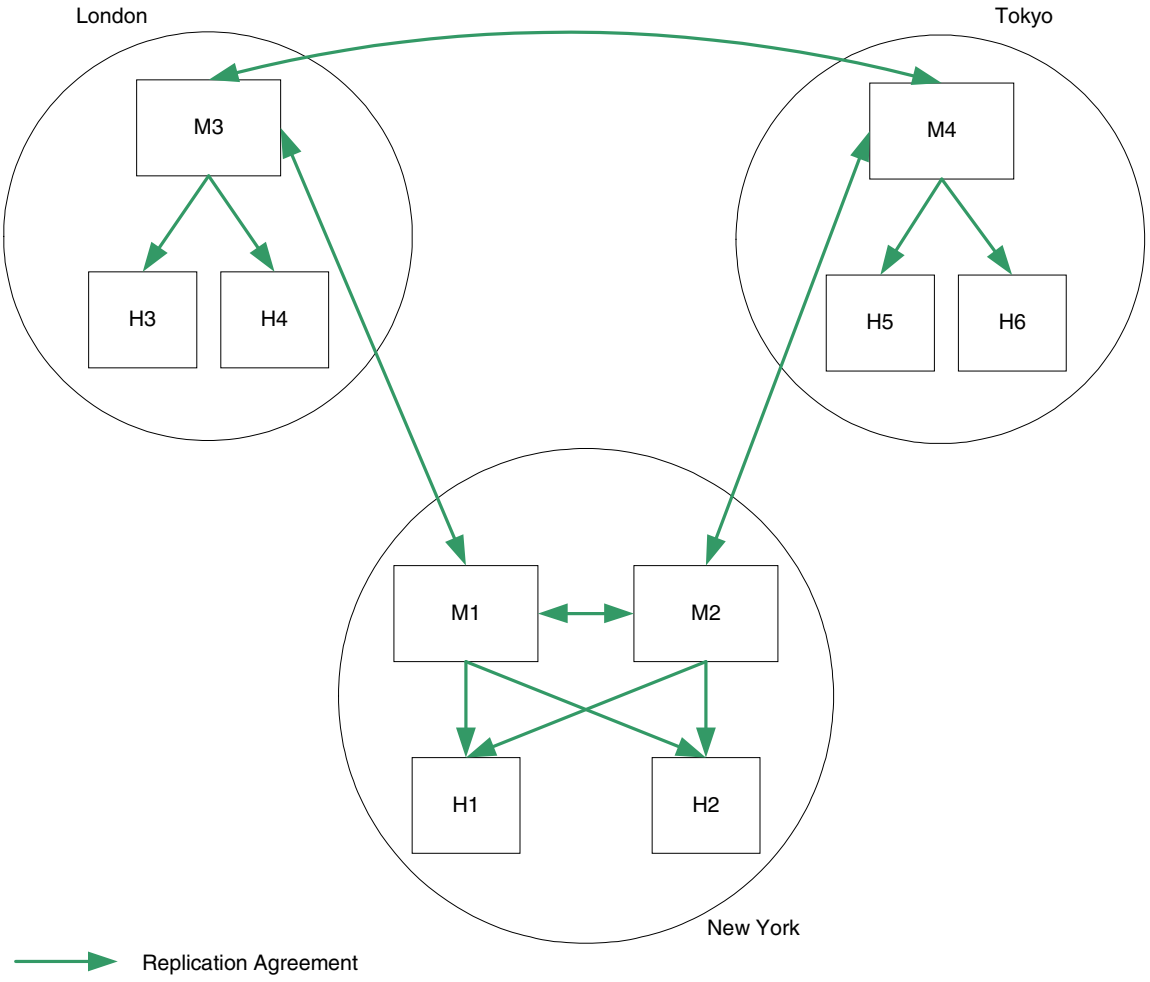
3 つのデータセンター

Directory Server 5.2 は、4 方向のマルチマスターレプリケーションをサポートしています。地理的に離れた 3 つの拠点にまたがる企業では、それぞれの地域で 2 つのマスターを持つ 1 つのデータセンターが必要になる可能性があります。このディレクトリの機能をどのように分割するかは、各データセンターで行われる読み取りおよび書き込み操作の相対的なトラフィックボリューム (など) によって異なります。

3 つのデータセンターの基本トポロジ

図 10-11 に示されるトポロジは、ニューヨークのデータセンターに対する読み取り、書き込み要求の数が最大で、3 つのデータセンターのそれぞれでローカルな読み取り、書き込み要求が可能であることを前提としています。

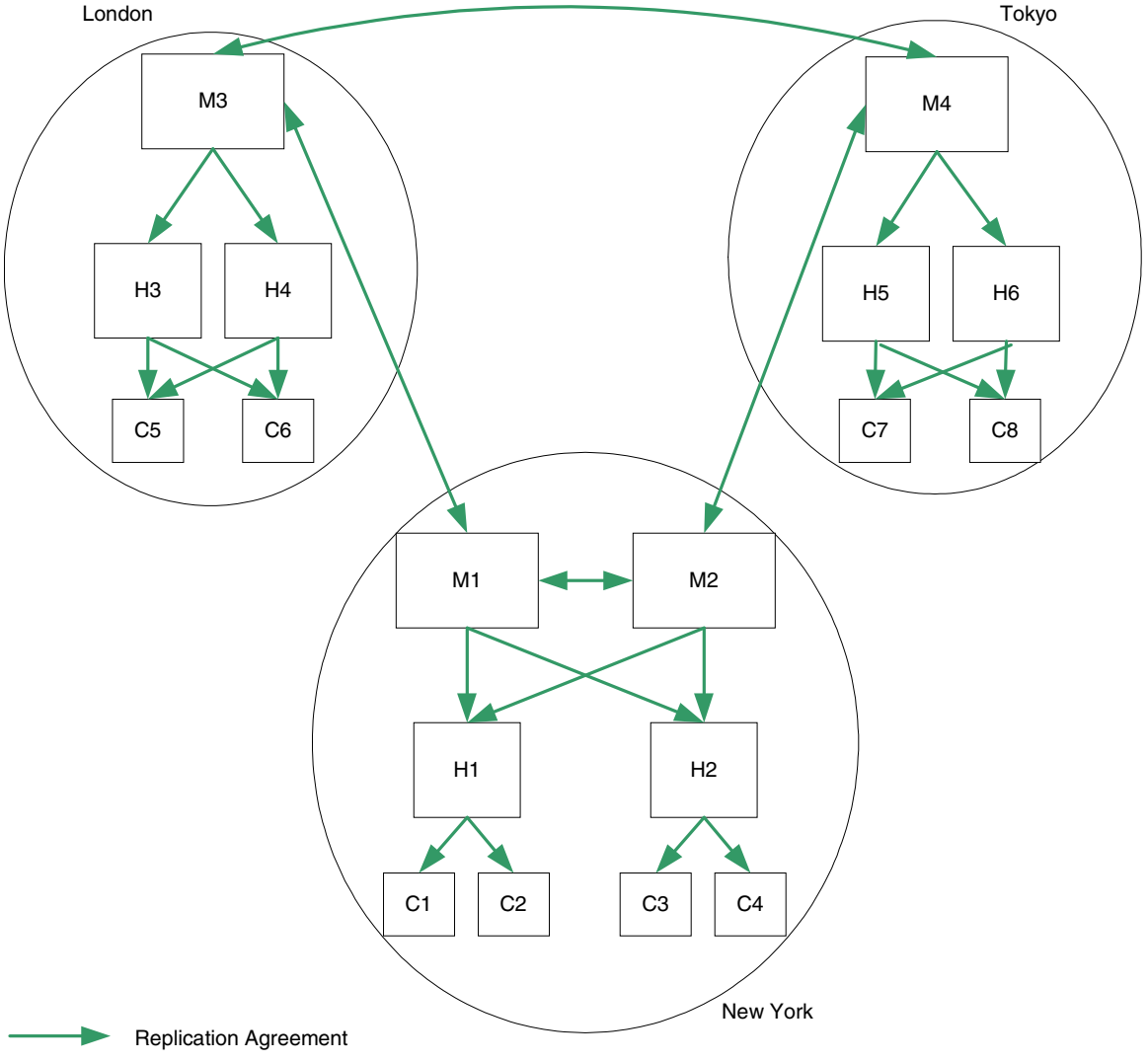
図 10-11　3 つのデータセンターの基本トポロジ



読み取りパフォーマンスのための 3 つのデータセンターのスケーリング

これまでの例のように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。ただし、それぞれのデータセンターでの想定パフォーマンス要件を考慮する必要があります。図 10-12 は、推奨されるトポロジを示しています。

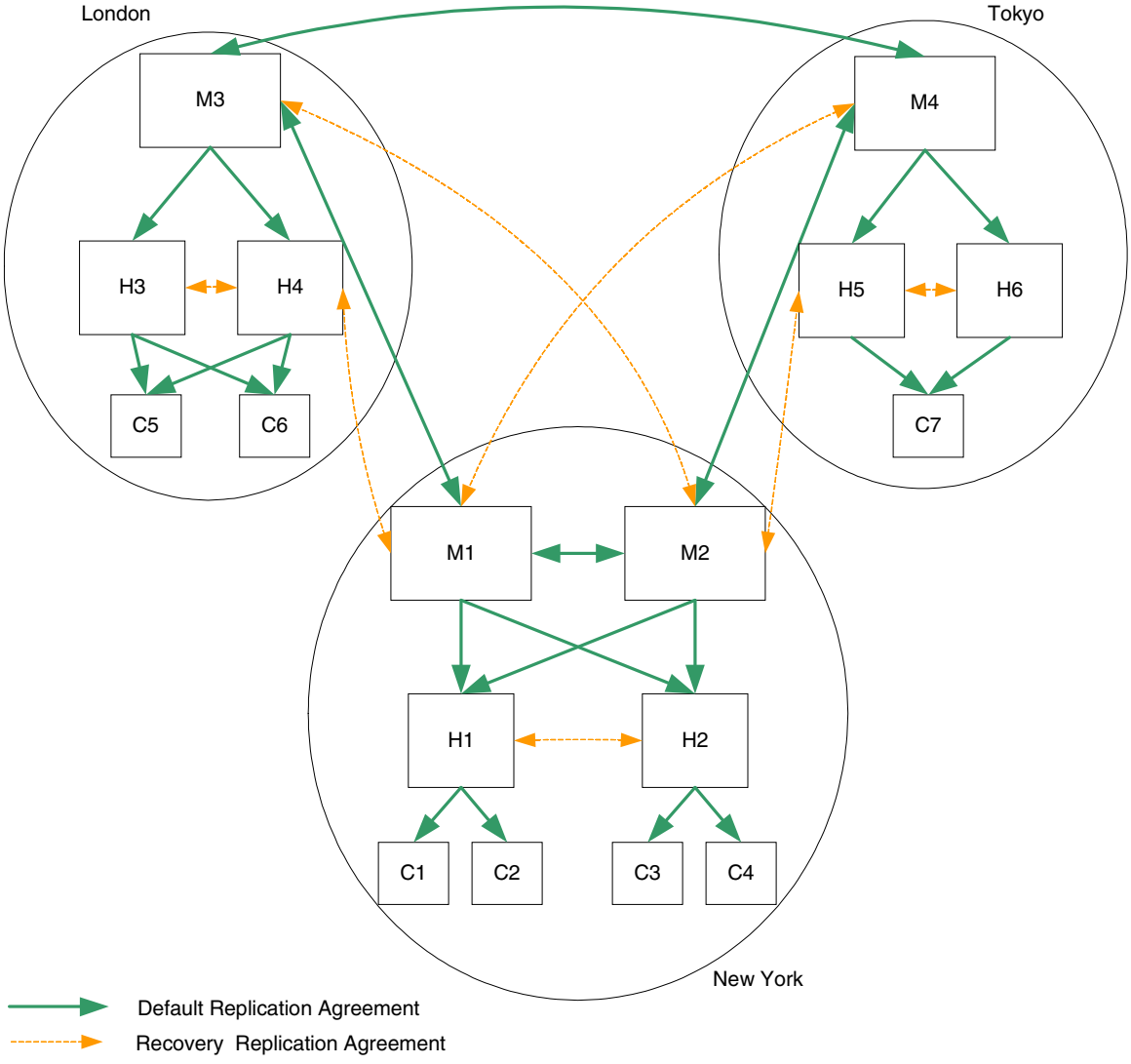
図 10-12 読み取りパフォーマンスのための 3 つのデータセンターのスケーリング



3 つのデータセンターの復元例

2 つのデータセンターの場合と同様に、複数のマスターに障害が発生した場合の復元戦略には、復元レプリケーションアグリーメントの作成が必要となります。図 10-13 に示されるように、これらのアグリーメントは、デフォルトでは無効化されていますが、障害が発生した場合には迅速に有効化できます。

図 10-13 3 つのデータセンターの復元レプリケーションアグリーメント



3 つのデータセンターの復元手順 (1 つのコンポーネント)

図 10-13 の例で、ロンドンまたは東京のマスターに障害が発生し、ローカル書き込み機能が失われたと仮定します。次の手順は、M3 (ロンドン) で障害が発生したことを前提としています。

1. H4 を書き込み可能なマスターに昇格させます。具体的な手順については、『Sun ONE Directory Server 管理ガイド』の「Promoting or Demoting Replicas」を参照してください。
2. H4 から H3 への復元レプリケーションアグリーメントを有効化し、ローカル書き込みがすべてのコンシューマにレプリケートされるようにします。
3. M1 と H4 の間の復元レプリケーションアグリーメントを有効化し、ローカル書き込みがリモートデータセンターにレプリケートされ、リモート書き込みがローカルコンシューマにレプリケートされるようにします。
4. M3 にアクセスしていたアプリケーションが新しいマスターにリダイレクトされることを確認します。

注 この手順は、M3 の修復中に、直接的なローカルの読み取りおよび書き込み機能を提供するための中間ソリューションです。

---

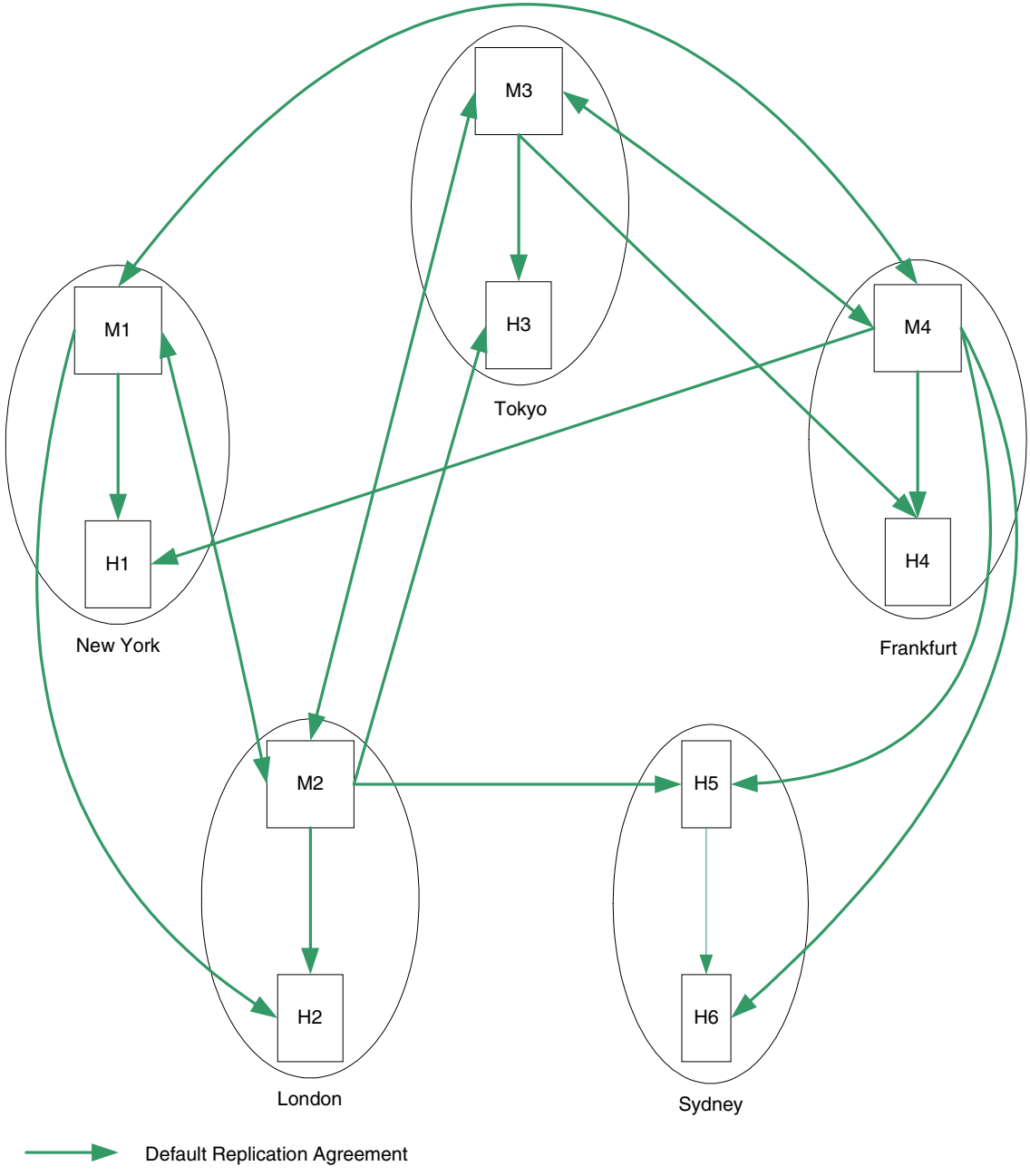
5 つのデータセンター

Sun ONE Directory Server 5.2 は、4 方向のマルチマスターレプリケーションをサポートしています。地理的に離れた 5 つの拠点にまたがる企業では、どの地域のローカル更新パフォーマンス要件が最小であるかを考慮する必要があります。この地域にはマスターサーバーを置かず、書き込みをいずれかの地域のマスターにリダイレクトします。

5 つのデータセンターの基本トポロジ

図 10-14 に示されるトポロジは、シドニーデータセンターが最も少ない書き込み要求を受け取ることを前提としています。ローカル読み取り要求は、5 つのそれぞれのデータセンターで可能です。

図 10-14　5 つのデータセンターの基本トポロジ



読み取りパフォーマンスのための 5 つのデータセンターのスケーリング

これまでの例のように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。ただし、それぞれのデータセンターでの想定パフォーマンス要件を考慮する必要があります。

5 つのデータセンターの復元例

2 つのデータセンターの場合と同様に、複数のマスターに障害が発生した場合の復元戦略には、復元レプリケーションアグリーメントの作成が必要となります。281 ページの図 10-15 に示されるように、これらのアグリーメントは、デフォルトでは無効化されていますが、障害が発生した場合には迅速に有効化できます。

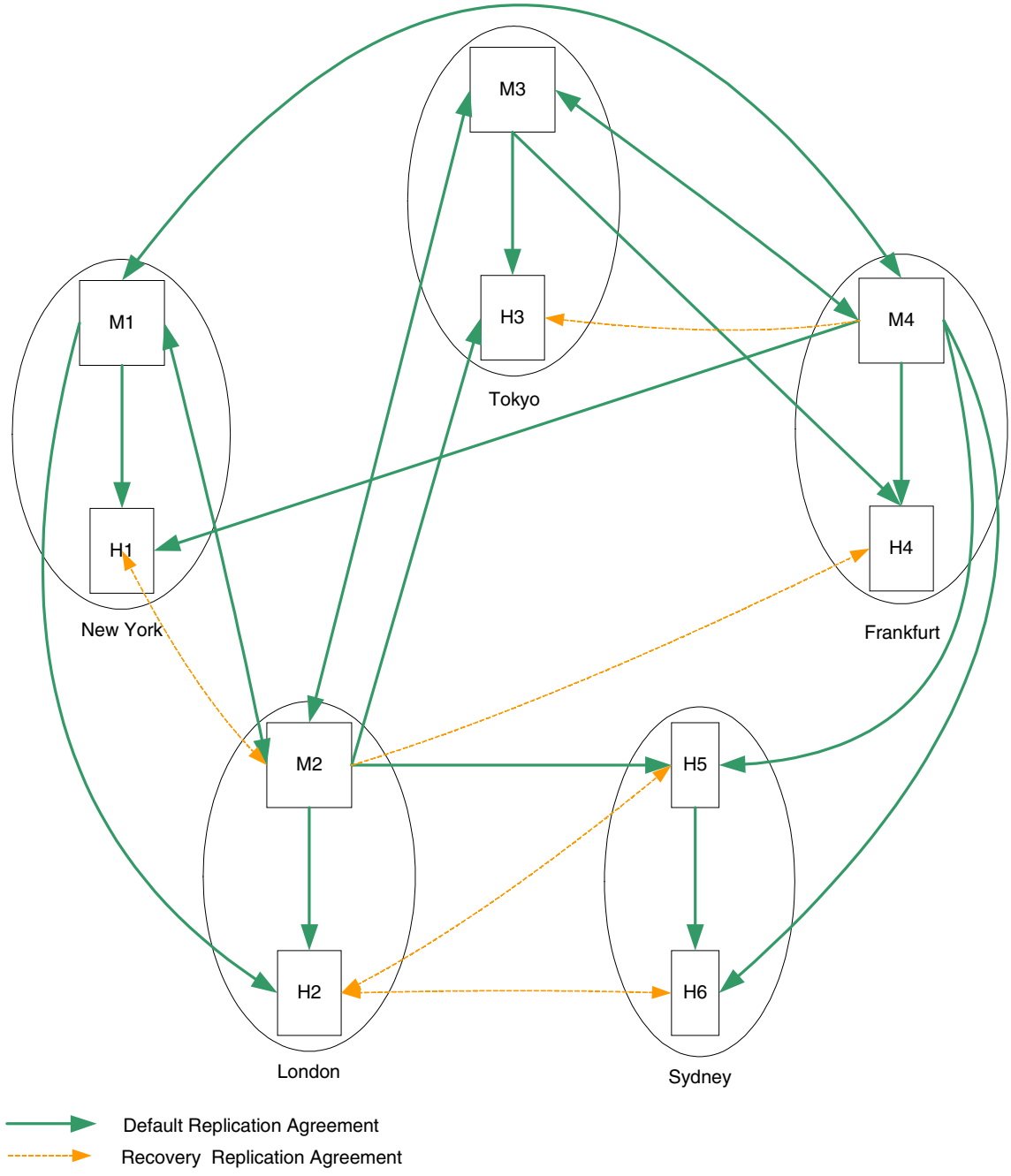
5 つのデータセンターの復元手順 (1 つのコンポーネント)

図 10-15 の例で、いずれかのマスターに障害が発生し、ローカル書き込み機能が失われたと仮定します。次の手順は、M1 (ニューヨーク) で障害が発生したことを前提としています。

1. H1 を書き込み可能なマスターに昇格させます。具体的な手順については、『Sun ONE Directory Server 管理ガイド』の「Promoting or Demoting Replicas」を参照してください。
2. M2 から H1 への復元レプリケーションアグリーメントを有効化し、ローカル書き込みがリモートデータセンターにレプリケートされ、リモート書き込みがローカルコンシューマにレプリケートされるようにします。
3. M1 にアクセスしていたアプリケーションが新しいマスターにリダイレクトされることを確認します。

注 この手順は、M1 の修復中に、直接的なローカルの読み取りおよび書き込み機能を提供するための中間ソリューションです。

図 10-15　5 つのデータセンターの復元レプリケーションアグリーメント



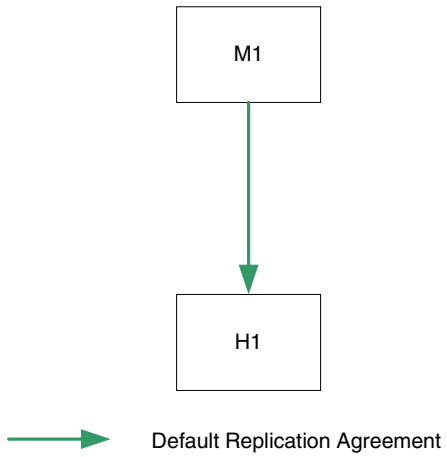
旧バージョン形式の更新履歴ログプラグインを使用する 1 つのデータセンター

すでに説明した 1 つのデータセンターのトポロジは、旧バージョン形式の更新履歴プラグインの使用を考慮していませんでした。旧バージョン形式の更新履歴に依存するほとんどのアプリケーションは、更新が順序よくリストされていることを前提としており、マルチマスターレプリケーションモデルの旧バージョン形式の更新履歴で一貫性が失われることによって失敗してしまいます。一般に、配備するアプリケーションが旧バージョン形式の更新履歴を必要とする場合は、その配備ではマルチマスターレプリケーショントポロジを採用するべきではありません。旧バージョン形式の更新履歴プラグインについては、『Sun ONE Directory Server 管理ガイド』の「Using the Retro Change Log Plug-In」を参照してください。

旧バージョン形式の更新履歴プラグインの基本トポロジ

マルチマスターレプリケーションを配備できない場合は、図 10-16 に示される基本トポロジが推奨されます。

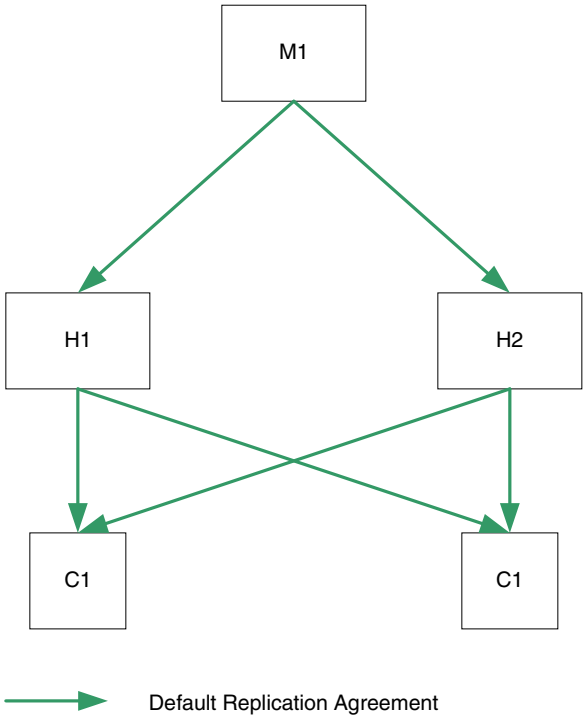
図 10-16 旧バージョン形式の更新履歴ログプラグインを使用する 1 つのデータセンター



読み取りパフォーマンスのための旧バージョン形式の更新履歴プラグインのスケーリング

1 つのデータセンターのトポロジと同様に、図 10-17 に示されるように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。

図 10-17 旧バージョン形式の更新履歴ログプラグインを使用する 1 つのデータセンター (スケーリング)



旧バージョン形式の更新履歴プラグインの復元手順

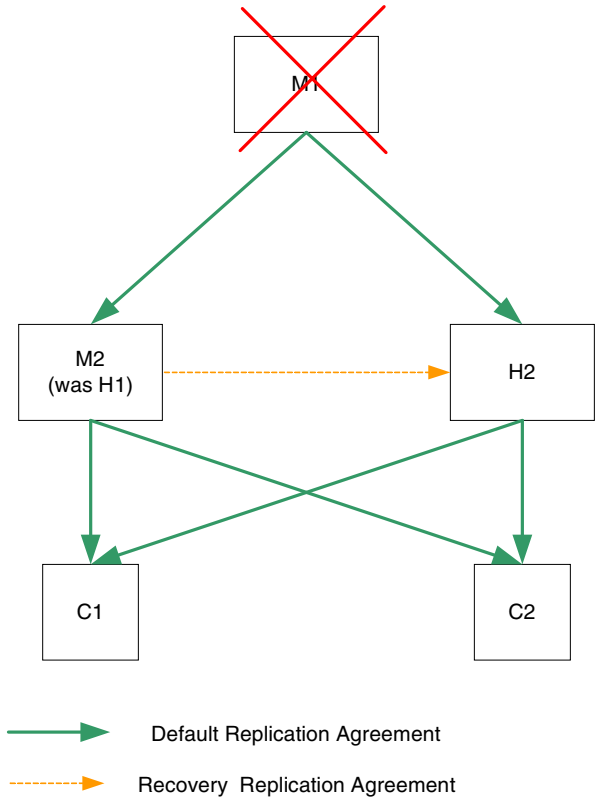
図 10-17 に示される配備では、マスターサーバーに障害が発生した場合に次の戦略を適用できます。

1. M1 を停止します (すでに停止していない場合)。
2. H1 または H2 をマスターサーバーに昇格させます。具体的な手順については、『Sun ONE Directory Server 管理ガイド』の「Promoting or Demoting Replicas」を参照してください。
3. 新しいマスター (M2) で旧バージョン形式の更新履歴プラグインを有効化します。

4. 旧バージョン形式の更新履歴のバックアップを M2 に復元します。
5. サーバーを再起動します。
6. 変更がハブにレプリケートされ続けるように、M2 と残りのハブの間に新しいレプリケーションアグリーメントを追加します。

   図 10-18 は、この復元戦略を示しています。

図 10-18 旧バージョン形式の更新履歴ログプラグインを使用する 1 つのデータセンター (復元)



DSMLv2 を使用した HTTP/SOAP 経由のデータへのアクセス

Sun ONE Directory Server 5.2 は、DSMLv2 をサポートしています。DSMLv2 は、LDAP バージョン 3 のほとんどのクエリー、更新機能を XML、SOAP、HTTP にマッピングします。ディレクトリ操作を行うために、DSMLv2 はディレクトリデータへのアクセスを XML と Web サーバーの世界に直接広げます。ここでは、次の例を参考に、DSMLv2 を使用して SOAP/HTTP 経由でディレクトリ操作を行う方法について詳細に説明します。

- 空の匿名 DSML Ping 要求
- ユーザーバインドを発行する DSML 要求
- DSML 検索要求

警告 次の例の `content-length:` ヘッダーには、DSMLv2 要求の具体的な長さが指定されています。これらの例が正常に機能するためには、このコンテンツ長を正しく理解するエディタを使用するか、必要に応じて値を手動で変更する必要があります。

DSML 関連の属性の完全なリストと、DSMLv2 標準については、『Sun ONE Directory Server Reference Manual』を参照してください。DSML に関連する手順については、『Sun ONE Directory Server 管理ガイド』を参照してください。

空の匿名 DSML Ping 要求

HTTP/SOAP 経由の DSML 要求を発行する前に、空の DSML バッチ要求をディレクトリに送信し、DSML フロントエンドが有効化されていることを確認する必要があります。空の DSML バッチ要求は、次のようになります。

```
POST /dsml HTTP/1.1
content-length: 451
HOST: hostMachine
SOAPAction: ""
Content-Type: text/xml
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>

   <soap-env:Body>
      <batchRequest
          xmlns='urn:oasis:names:tc:DSML:2:0:core'
          requestID='Ping!'>
          <!-- empty batch request -->
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

この DSML の最初のセクションは、HTTP セクションです。

```
POST /dsml HTTP/1.1
content-length: 451
HOST: hostMachine
SOAPAction: ""
Content-Type: text/xml
Connection: close
```

これは、次の HTTP メソッド行から構成されます。

```
POST /dsml HTTP/1.1
```

HTTP ヘッダーの番号が続きます。HTTP メソッド行は、HTTP メソッド要求と、DSML フロントエンドが使用する URL を指定します。`POST` は、DSML フロントエンドが受け付ける唯一の HTTP メソッド要求を示し、`/dsml` URL には Directory Server のデフォルト URL 以外の任意の有効 URL を設定できます。これ以後の HTTP ヘッダーは、DSML 要求の残りの詳細を指定します。次のヘッダーは、

```
content-length: 451
```

SOAP/DSML 要求の実際の長さを指定し、次のヘッダーは、

```
HOST: hostMachine
```

Directory Server が接続するホストの名前を指定します。次のヘッダーは、

```
SOAPAction:
```

必須ヘッダーです。これは、HTTP/SOAP スタックでの DSML 要求の実行をディレクトリに知らせます。ただし、このヘッダーは空のまま残すことができます。次のヘッダーは、

```
Content-Type: text/xml
```

値として `text/xml` を持つ必要があります。これは、コンテンツが XML であることを定義します。最後のヘッダーは、

```
Connection: close
```

要求が成功した場合に接続を閉じることを指定しています。デフォルトの HTTP/1.1 の動作では、接続は開いた状態で維持されます。

次のセクションは、要求の SOAP/DSML セクションを構成します。

```
<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>

   <soap-env:Body>
      <batchRequest
          xmlns='urn:oasis:names:tc:DSML:2:0:core'
          requestID='Ping!'>
          <!-- empty batch request -->
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

DSML 要求は、次の XML 序言ヘッダーから始まります。

```
<?xml version='1.0' encoding='UTF-8'?>
```

これは、要求が UTF-8 文字セットで符号化されなければならないことを示します。次に、SOAP エンベロープと、必須の XML スキーマ、XML スキーマインスタンス、SOAP ネームスペースを含める本文要素が次のように続きます。

```
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>

   <soap-env:Body>
```

次の DSML バッチ要求要素は、

```
      <batchRequest
```

DSML バッチ要求の開始を示し、直後に必須 DSMLv2 ネームスペースを含める要素が続きます。

```
xmlns='urn:oasis:names:tc:DSML:2:0:core'.
```

さらに、オプションとして次の要求 ID が指定されます。

```
requestID='Ping!'>
```

空のバッチ要求は、

```
<!-- empty batch request -->.
```

このようにコメントアウトされた XML です。SOAP/DSML バッチ要求は、続くバッチ要求の終了、SOAP 本文の終了、SOAP エンベロープ要素の終了によって閉じられます。

```
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

警告 ディレクトリに同時に接続できるクライアント数と、DSML 要求のサイズには、最大制限が適用されます。クライアント数の制限は `ds-dsml-poolsize` 属性と `ds-dsml-poolmaxsize` 属性によって管理され、要求サイズの制限は `ds-dsmll-requestmaxsize` 属性によって管理されます。DSML 関連の属性の詳細は、『Sun ONE Directory Server Reference Manual』を参照してください。

DSML フロントエンドが有効化されていれば、空の DSML 要求に対して次のような応答が返されます。

```
HTTP/1.1 200 OK
Cache-control: no-cache
Connection: close
Date: Mon, 09 Sep 2002 13:56:49 GMT
Accept-Ranges:none
Server: Sun-ONE-Directory/5.2
Content-Type: text/xml; charset="utf-8"
Content-Length: 500

<?xml version='1.0' encoding='UTF-8' ?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'
   >
<soap-env:Body>
<batchResponse
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns='urn:oasis:names:tc:DSML:2:0:core'
   requestID='Ping!'
   >
</batchResponse>
</soap-env:Body>
</soap-env:Envelope>
```

なにも返されない場合は、フロントエンドが無効であることがわかります。

注 DSML フロントエンドは、デフォルトでは無効化されています。有効化するには、`dse.ldif` ファイルの `cn=DSMLv2-SOAP-HTTP,cn=frontends,cn=plugins,cn=config` エントリの下にある `nsslapd-pluginEnabled` 属性の値を `on` に変更する必要があります。また、`ds-hdsml-port` 属性と `ds-hdsml-root` 属性を変更することもできますが、これはオプションです。DSML フロントエンドの有効化について詳細は、『Sun ONE Directory Server 管理ガイド』を参照してください。

ユーザーバインドを発行する DSML 要求

LDAP プロトコルの場合と同様に、DSML 要求を発行するには、特定のユーザーとしてディレクトリにバインドするか、または匿名でアクセスします。特定のユーザーとしてディレクトリにバインドするには、dn にマッピングされる uid とパスワードを含む HTTP 承認ヘッダーを DSML 要求に含めます。

注　　ID のマッピングを設定する方法については、『Sun ONE Directory Server 管理ガイド』を参照してください。

HTTP 承認要求は、次のようになります。

```
POST /dsml HTTP/1.1
content-length: 578
Content-Type: text/xml; charset="utf-8"
HOST: hostMachine
Authorization: Basic ZWFzdGVyOmVnZw==
SOAPAction: ""
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'>
   <soap-env:Body>
     <batchRequest
        xmlns='urn:oasis:names:tc:DSML:2:0:core'
        <extendedRequest>
          <requestName>1.3.6.1.4.1.4203.1.11.3</requestName>
        </extendedRequest>
     </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

この例は、プレーンテキスト形式で easter という uid と egg というパスワードが `easter:egg` に変換され、base64 で次のようにコード化される HTTP 承認ヘッダーを示しています。

```
Authorization: Basic ZWFzdGVyOmVnZw==
```

匿名でアクセスする場合は、uid とパスワードを含む HTTP 承認ヘッダーは必要ありません。しかし、匿名アクセスは厳密なアクセス制御の多くで拒否対象となるため、データアクセスが制約される可能性が高いことに注意してください。同様に、LDAP プロキシとして LDAP 操作を実行する DSML 要求を発行することもできます。DSML 要求はバッチベースで管理されるため、LDAP プロキシとして実行するための要求を発行する場合に必要となる DSML プロキシ承認要求は、その要求バッチの最初に指定する必要があることに注意してください。

DSML 検索要求

DSML フロントエンドが有効化されたら、ディレクトリ操作を開始できます。次に、`namingContexts`、`supportedLDAPversion`、`vendorName`、`vendorVersion`、`supportedSASLMechanisms` 属性について、ルート DSE エントリに対して実行される DSML ベースのオブジェクト検索要求の例を示します。

```
POST /dsml HTTP/1.1
HOST: hostMachine
Content-Length: 1081
Content-Type: text/xml
SOAPAction: ""
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<soap-env:Envelope
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:soap-env='http://schemas.xmlsoap.org/soap/envelope/'
   >
   <soap-env:Body>
      <batchRequest
        xmlns='urn:oasis:names:tc:DSML:2:0:core'
        requestID='Batch of search requests'
        >
        <searchRequest
            dn=""
            requestID="search on Root DSE"
            scope="baseObject"
            derefAliases="neverDerefAliases"
            typesOnly="false"
            >
            <filter>
               <present name="objectClass"/>
            </filter>
            <attributes>
               <attribute name="namingContexts"/>
               <attribute name="supportedLDAPversion"/>
               <attribute name="vendorName"/>
               <attribute name="vendorVersion"/>
               <attribute name="supportedSASLMechanisms"/>
            </attributes>
        </searchRequest>
      </batchRequest>
   </soap-env:Body>
</soap-env:Envelope>
```

属性とデータの次のペアを指定することで、

```
dn=""
requestID="search on Root DSE"
```

この検索操作は、ルート DSE エントリの下にあり、オプションの要求 ID 属性によって特定されるデータを要求します。次の属性とデータのペアは、

```
scope="baseObject"
```

検索がベースオブジェクト検索であることを指定し、次の属性と値のペアは、

```
derefAliases="neverDerefAliases"
```

ベースオブジェクトの検索または特定時に、エイリアスが間接参照されないことを指定します。実際に、Directory Server がサポートするのは `derefAliases` 値だけです。

次の属性とデータのペアは、

```
typesOnly="false"
```

属性名とその値の両方が返されることを指定します。反対に、`typesOnly="true"` と指定した場合は、属性名だけが返されます。この属性のデフォルト値は `false` です。

適用するフィルタには、オブジェクトクラスフィルタが次のように使用されます。

```
<filter>
   <present name="objectClass"/>
</filter>
```

次に、目的の属性のリストが続きます。

```
<attributes>
   <attribute name="namingContexts"/>
   <attribute name="supportedLDAPversion"/>
   <attribute name="vendorName"/>
   <attribute name="vendorVersion"/>
   <attribute name="supportedSASLMechanisms"/>
</attributes>
```

索引

P

S

T

U

あ

い

し

す

せ

そ