Red Hat Directory Server 11

導入ガイド

効果的なディレクトリーサービスを計画するための概念と設定オプション

Last Updated: 2023-10-17

Red Hat Directory Server 11 導入ガイド

効果的なディレクトリーサービスを計画するための概念と設定オプション

Marc Muehlfeld
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

法律上の通知

概要

このガイドは、ディレクトリーサービスを計画するためのものです。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、「Red Hat CTO である Chris Wright のメッセージ」をご覧ください。

はじめに

『Red Hat Directory Server 導入ガイド』は、効果的なディレクトリーサービスを計画するための概念および設定オプションに関する強固な基盤を提供します。ここで提供される情報は、設計者および管理者の両方を対象としています。

1. DIRECTORY SERVER の概要

Red Hat Directory Server には、以下の主な機能があります。

- マルチサプライヤーレプリケーション: 読み取りおよび書き込み両方の操作に高可用性のディレクトリーサービスを提供します。マルチサプライヤーレプリケーションは、簡単なレプリケーションシナリオおよびカスケードレプリケーションシナリオと組み合わせることで、柔軟性が高く、スケーラブルなレプリケーション環境を提供できます。

- チェーンおよび参照: 多数の Directory Server のデータをユーザーに透過的に維持しながら、ディレクトリーの完全な論理ビューを 1 台のサーバーに保存することで、ディレクトリーの能力を向上させます。

- ロールおよびサービスクラス: エントリー間で属性を動的にグループ化および共有するための柔軟なメカニズムを提供します。

- 効率的なアクセス制御メカニズム: ディレクトリーで使用されるアクセス制御ステートメントの数を大幅に削減し、アクセス制御評価のスケーラビリティーを高めるマクロに対応します。

- バインド DN によるリソース制限: クライアントのバインド DN に基づいて検索操作に割り当てられるサーバーリソースの量を制御する権限を付与します。

- 複数のデータベース: ディレクトリーサービスにおけるレプリケーションとチェーンの実装を簡素化するために、ディレクトリーデータをブレークダウンする簡単な方法を提供します。

- パスワードポリシーおよびアカウントのロックアウト: パスワードおよびユーザーアカウントを Directory Server で管理する方法を制御する一連のルールを定義します。

- TLS は、暗号化用の Mozilla Network Security Services (NSS) ライブラリーを使用して、ネットワーク上でのセキュアな認証と通信を提供します。

Directory Server の主要コンポーネントには、以下が含まれます。

- LDAP サーバー: LDAP v3 準拠のネットワークデーモン

- Web コンソール: ディレクトリーサービスのセットアップと保守の作業を軽減するグラフィカル管理コンソール

- SNMP エージェント: Simple Network Management Protocol (SNMP) を使用して Directory Server を監視できます。

第1章 ディレクトリーサービスの概要

Red Hat 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情報に集中化されたディレクトリーサービスを提供します。Directory Server は既存のシステムと統合し、従業員、顧客、サプライヤー、パートナーの情報を統合するための集中化されたリポジトリーとして機能します。Directory Server は、ユーザープロファイル、設定、および認証を管理するように拡張することもできます。

本章では、ディレクトリーサービスの機能を理解しディレクトリーサービスの設計に役立てるために、基本的な概念を説明します。

1.1. ディレクトリーサービスについて

ディレクトリーサービスという用語は、エンタープライズ、サブスクライバー、またはその両方についての情報を格納し、その情報をユーザーが利用できるようにするための、ソフトウェア、ハードウェア、およびプロセスの集合を指します。ディレクトリーサービスは、少なくとも 1 つの Directory Server のインスタンスと、少なくとも 1 つのディレクトリークライアントプログラムで設定されます。クライアントプログラムは、ディレクトリーサービスに保存されている名前、電話番号、住所、その他のデータにアクセスできます。

ディレクトリーサービスの例は、ドメイン名システム (DNS) サーバーです。DNS サーバーは、コンピューターホスト名を IP アドレスにマッピングします。そのため、すべてのコンピューティングリソース (ホスト) は DNS サーバーのクライアントになります。ホスト名のマッピングにより、ユーザーは IP アドレスではなくホスト名を記憶し、ネットワーク上のコンピューターを簡単に特定することができます。DNS サーバーの制限は、名前と IP アドレスという 2 種類の情報のみを保存することです。真のディレクトリーサービスは、ほぼ無制限の種類の情報を保存します。

Directory Server は、すべてのユーザーとネットワーク情報を、1 つのネットワークがアクセス可能なリポジトリーに保存します。さまざまな種類の情報を Directory Server に保存できます。

- 場所、カラーまたは白黒、メーカー、購入日、シリアル番号など、組織内のプリンターのデータ等の物理デバイスの情報
- 名前、メールアドレス、部門などの公開されている従業員情報
- 給与、マイナンバー、自宅住所、電話番号、給与水準など、非公開の従業員情報
- クライアントの名前、最終納期、見積もり情報、契約番号、プロジェクト期日などの契約またはアカウント情報

Directory Server は、さまざまなアプリケーションのニーズに対応します。Directory Server に含まれる情報にアクセスするための、標準のプロトコルおよびアプリケーションプログラミングインターフェイス (API) も提供します。

1.1.1. グローバルディレクトリーサービスについて

Directory Server はグローバルディレクトリーサービスを提供します。つまり、さまざまなアプリケーションに情報を提供します。さまざまなアプリケーションとバンドルするプロプライエタリーデータベースを統合する (管理のための負担が大きい) 代わりに、Directory Server は同じ情報を管理する単一のソリューションとなります。

たとえば、ある会社はプロプライエタリーのディレクトリーサービスを使用して、3 つの異なるプロプライエタリーメールシステムを運用しています。ユーザーが 1 つのディレクトリーでパスワードを変更した場合は、変更が他のディレクトリーで自動的に複製されません。同じ情報の複数のインスタンスを

管理すると、ハードウェアと人事費が増大します。メンテナンスオーバーヘッドの増加は **n+1 ディレクトリーの問題** と呼ばれます。

グローバルディレクトリーサービスは、あらゆるアプリケーションがアクセスできるディレクトリー情報の単一の集中リポジトリーを提供することで、n+1 ディレクトリーの問題を解決します。しかし、ディレクトリーサービスにさまざまな種類のアプリケーションにアクセスするには、アプリケーションとディレクトリーサービス間のネットワークベースの通信が必要です。Directory Server は、アプリケーションがグローバルディレクトリーサービスにアクセスするために LDAP を使用します。

1.1.2. LDAP について

LDAP は、クライアントアプリケーションとサーバーが相互に通信するために使用する共通の言語を提供します。LDAP は、ISO X.500 標準規格で規定される Directory Access Protocol (DAP) の軽量バージョンです。DAP は、拡張可能かつ堅牢な情報フレームワークを使用して、アプリケーションアクセスを提供しますが、管理コストが高まります。DAP は、インターネット標準プロトコルではなく、複雑なディレクトリーの名前規則を持つ通信層を使用します。

LDAP は DAP の最良の機能を維持し、管理コストを削減します。LDAP は、TCP/IP を介して実行するオープンディレクトリーアクセスプロトコルを使用し、簡素化したエンコーディングメソッドを使用します。ハードウェアおよびネットワークインフラストラクチャーへの投資を抑えながら、データモデルを保持し、数百万のエントリーをサポートすることができます。

1.2. DIRECTORY SERVER の概要

Red Hat Directory Server は、複数のコンポーネントで設定されています。ディレクトリー自体の中核は、LDAP プロトコルを実装するサーバーです。Red Hat Directory Server には、LDAP サーバーに加えて、エンドユーザーがディレクトリー内のエントリーを検索および変更できるクライアント側のグラフィカルユーザーインターフェイスがあります。その他の LDAP クライアント (Mozilla LDAP SDK および OpenLDAP SDK を使用して書かれたサードパーティープログラムおよびカスタムプログラムの両方) は、Red Hat Directory Server と一緒に使用することや、他のアプリケーションを Red Hat Directory Server と統合することができます。

Red Hat Directory Server をインストールする場合、以下の要素があります。

- コア Directory Server LDAP サーバー、LDAP v3 準拠のネットワークデーモン(**ns-slapd**)と、サーバーとそのデータベースを管理するための関連プラグイン、コマンドラインツール、その設定およびスキーマファイル。コマンドラインツールの詳細は、『Red Hat Directory Server 設定、コマンド、およびファイルリファレンス』を参照してください。

- Administration Server (LDAP サーバーにアクセスする異なるポータルを制御する Web サーバー)。Administration Server についての詳細は、『Red Hat Directory Server 管理ガイド』を参照してください。

- Web コンソール (ディレクトリーサービスのセットアップと保守の作業を軽減するグラフィカル管理コンソール)Web コンソールについての詳細は、『Red Hat Directory Server 管理ガイド』を参照してください。

- Simple Network Management Protocol (SNMP) を使用して Directory Server を監視する SNMP エージェント。SNMP による監視についての詳細は、『Red Hat Directory Server 管理ガイド』を参照してください。

他の LDAP クライアントプログラムを追加しなくても、Directory Server はイントラネットまたはエクストラネットのベースを提供できます。互換性のあるサーバーアプリケーションは、従業員、顧客、サプライヤー、パートナーデータなどの共有サーバー情報の中央リポジトリーとしてディレクトリーを使用します。

Directory Server は、ユーザー認証を管理し、アクセス制御を作成し、ユーザー設定を定義し、ユーザー管理を一元化することができます。ホストされる環境では、パートナー、顧客、およびサプライヤーは、ディレクトリーの独自の部分を管理し、管理コストを削減できます。

1.2.1. サーバーフロントエンドの概要

Directory Server はマルチスレッドアプリケーションです。つまり、複数のクライアントを同時に同じネットワーク上でサーバーにバインドできます。ディレクトリーサービスが拡張され、多数のエントリーや地理的に分散されたクライアントを含むようになるにつれ、ネットワーク上の戦略的な場所に複数の Directory Server が追加されます。

Directory Server のサーバーフロントエンドは、ディレクトリークライアントプログラムとの通信を管理します。複数のクライアントプログラムは、LDAP over TCP/IP (インターネットトラフィックプロトコル) と LDAP over Unix sockets (LDAPI) の両方を使用してサーバーと通信できます。Directory Server は、クライアントが接続への Transport Layer Security (TLS) の使用をネゴシエートするかどうかに応じて、TLS を使用した **セキュアな** (暗号化された) 接続を確立できます。

TLS を使用して通信が行われる場合、通信は通常暗号化されます。クライアントに証明書が発行されている場合は、Directory Server は TLS を使用して、クライアントにサーバーへのアクセス権限があることを確認できます。メッセージの整合性チェック、デジタル署名、サーバー間の相互認証など、他のセキュリティーアクティビティーを実行するために、TLS が使用されます。

注記

Directory Server はデーモンとして実行され、プロセスは **ns-slapd** です。

1.2.2. サーバープラグインの概要

Directory Server は、コアサーバーに機能を追加するのに **プラグイン** に依存します。たとえば、データベースレイヤーはプラグインです。Directory Server には、レプリケーション、データベースのチェーン化、その他のディレクトリー機能用のプラグインがあります。

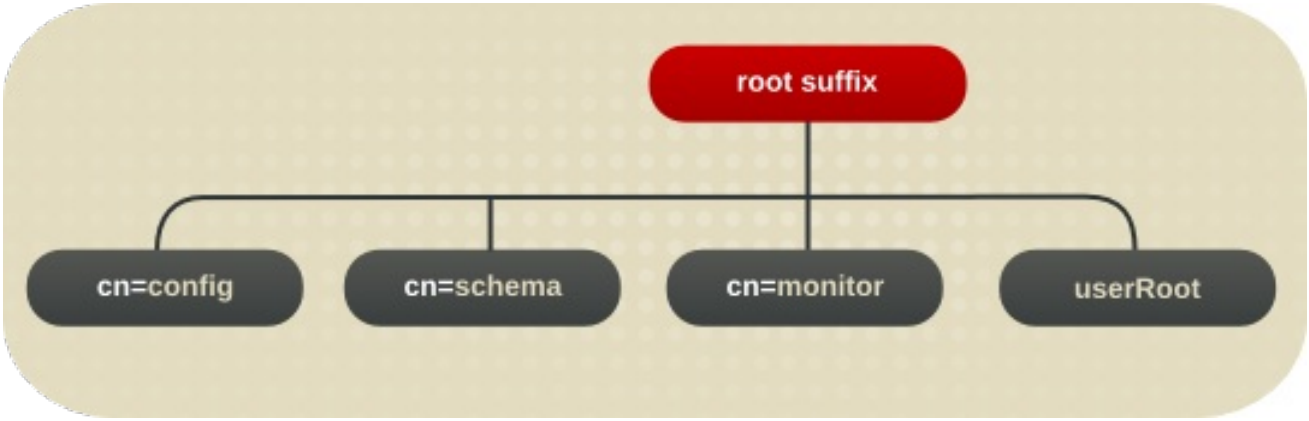
通常、プラグイン (特にサーバー機能を拡張するプラグイン) は無効にできます。無効にしてもプラグインの設定情報はディレクトリー内に残りますが、その機能はサーバーによって使用されません。ディレクトリーが実行する内容に応じて、Directory Server で提供されるすべてのプラグインを有効にして Directory Server 機能を拡張できます(バックエンドデータベースのプラグインなど、コアのディレクトリーサービス操作に関連するプラグインは、当然に無効にすることはできません)。

Directory Server でのデフォルトのプラグインおよびカスタムプラグインの作成に使用できる機能の詳細は、『Red Hat Directory Server Plug-in Guide 』を参照してください。

1.2.3. 基本的なディレクトリーツリーの概要

ディレクトリーツリーはディレクトリー情報ツリー (DIT) とも呼ばれ、ほとんどのファイルシステムで使用されるツリーモデルと同様に、ツリーのルートまたは最初のエントリーが階層の最上部に表示されます。インストール時に、Directory Server はデフォルトのディレクトリーツリーを作成します。

図1.1 デフォルトの Directory Server ディレクトリーツリーのレイアウト



ツリーの root は **root 接尾辞** と呼ばれます。root 接尾辞の命名に関する詳細は、 「接尾辞の選択」 を参照してください。

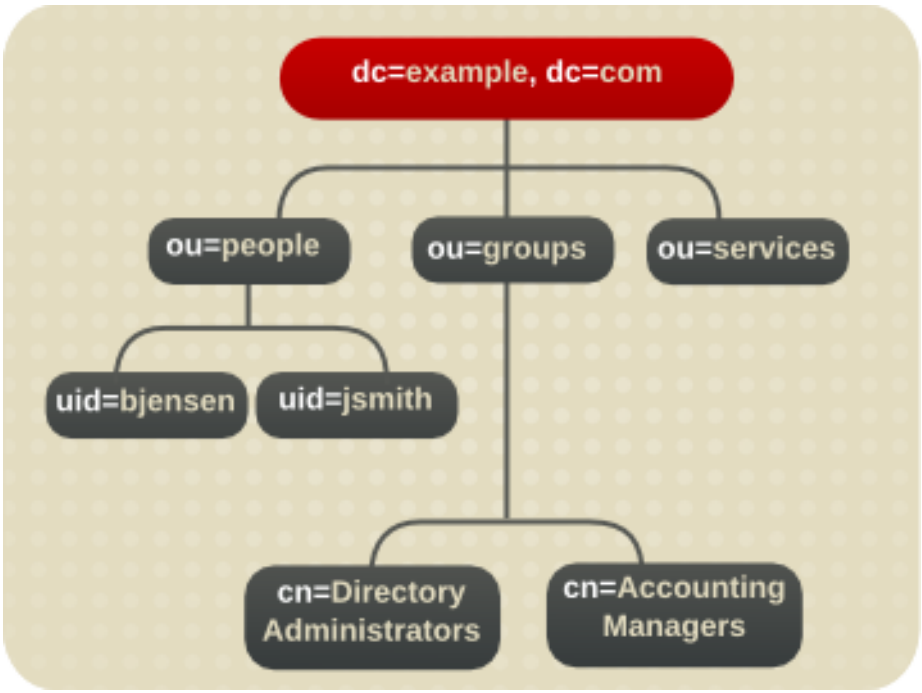
標準のインストール後のディレクトリーには、root 接尾辞の下に 3 つのサブツリーが含まれます。

- **cn=config**: サーバーの内部設定に関する情報が含まれるサブツリー
- **cn=monitor**: Directory Server サーバーおよびデータベース監視統計が含まれるサブツリー
- **cn=schema**: 現在サーバーに読み込まれているスキーマ要素が含まれるサブツリー
- **user_suffix**: Directory Server の設定時に作成されるデフォルトのユーザーデータベースの接尾辞接尾辞の名前は、サーバーの作成時にユーザーが定義します。関連付けられたデータベースの名前は **userRoot** です。データベースには、セットアップ時に LDIF ファイルをインポートしてエントリーを投入することも、後でエントリーを追加することもできます。

  user_suffix 接尾辞には、頻繁に ***dc*** の命名規則があります（例： **dc=example,dc=com** ）。もう 1 つの一般的な命名属性は ***o*** 属性で、組織全体に使用されます（例： **o=example.com** ）。

デフォルトのディレクトリーツリーを拡張して、ディレクトリーのインストールに関連する任意のデータを追加することができます。ディレクトリーツリーの詳細は、*4章 ディレクトリーツリーの設計*を参照してください。

図1.2 Example Corp 用の拡張されたディレクトリーツリー



1.3. DIRECTORY SERVER のデータストレージ

データベースは、ストレージ、パフォーマンス、レプリケーション、およびインデックス化の基本単位です。すべての Directory Server 操作 (エントリーのインポート、エクスポート、バックアップ、復元、およびインデックス化) は、データベース上で実行されます。ディレクトリーデータは LDBM データベースに保存されます。LDBM データベースは、ディレクトリーと共に自動的にインストールされるプラグインとして実装され、デフォルトで有効になります。

デフォルトでは、Directory Server はルート 接尾辞に対して 1 つのバックエンドデータベースインスタンスを使用します。ディレクトリーツリーを含めるには、1 つのデータベースで十分です。このデータベースは、数百万のエントリーを管理できます。

このデータベースは、データに対するリスクを最小限に抑えるために、データのバックアップおよび復元に関する高度な方法をサポートします。

複数のデータベースを使用して、Directory Server デプロイメント全体に対応できます。情報はデータベース全体に分散され、サーバーには 1 つのデータベースに格納できる以上のデータを保存できます。

1.3.1. ディレクトリーエントリーについて

LDAP データ交換形式 (LDIF) は、ディレクトリーエントリーを記述する標準的なテキストベースの形式です。**エントリー** は LDIF ファイルの多数の行 (**スタンザ** とも呼ばれる) で設定され、これには、オブジェクトに関する情報 (組織内の人員やネットワーク上のプリンターなど) が含まれます。

エントリーに関する情報は、属性とその値のセットにより LDIF ファイルで表されます。各エントリーには、エントリーが記述するオブジェクトの種類を指定し、含まれる追加属性のセットを定義するオブジェクトクラス属性があります。各属性は、エントリーの特定の特性を記述します。

たとえば、エントリーは、オブジェクトクラス **organization** openblas、エントリーが組織内のユーザーを表すことを示します。このオブジェクトクラスは、***givenname*** および ***telephoneNumber*** 属性をサポートします。これらの属性に割り当てられている値は、エントリーによって表される人員の名前と電話番号を提供します。

Directory Server は、サーバーによって計算される読み取り専用属性も使用します。これらの属性は、**操作属性** と呼ばれます。管理者は、アクセス制御やその他のサーバー機能に使用できる運用属性を手動で設定できます。

1.3.1.1. ディレクトリーエントリーでのクエリーの実行

エントリーは、ディレクトリーツリーの階層構造に保存されます。LDAP は、データベースのエントリーにクエリーを行い、ディレクトリーツリー内のそのエントリー下のすべてのエントリーを要求するツールをサポートします。このサブツリーのルートは、**ベース識別名** または ベース DN と呼ばれます。たとえば、**ou=people,dc=example,dc=com** のベース DN を指定して LDAP 検索要求を実行すると、検索操作では **dc=example,dc=com** ディレクトリーツリーの **ou=people** サブツリーのみが検査されます。

ただし、LDAP 検索に応答してすべてのエントリーは自動的に返されません。ただし、管理エントリー(**ldapsubentry** オブジェクトクラスを持つ)はデフォルトで LDAP 検索で返されないためです。たとえば、管理オブジェクトは、ロールやサービスクラスを定義するために使用されるエントリーになります。検索応答にこれらのエントリーを含めるには、クライアントが **ldapsubentry** オブジェクトクラスを持つエントリーを検索する必要があります。ロールの詳細は 「ロールの概要」を、サービスクラスの詳細は 「サービスクラスについて」 を、それぞれ参照してください。

1.3.2. ディレクトリーデータの分散

ディレクトリーツリーのさまざまな部分が別のデータベースに保存されると、ディレクトリーは、クライアント要求を並行して処理できるため、パフォーマンスが向上します。データベースを異なるマシンに配置して、パフォーマンスをさらに向上させることもできます。

分散データは、**データベースリンク** と呼ばれるディレクトリーのサブツリーの特別なエントリーによって接続され、このエントリーがリモートで保存されたデータを参照します。クライアントアプリケーションがデータベースリンクからデータを要求すると、データベースリンクはリモートデータベースからデータを取得し、クライアントに返します。このエントリーの下に対して試行された LDAP 操作は、すべてリモートマシンに送信されます。この手法は **チェーン** と呼ばれます。

チェーンは、プラグインとしてサーバーに実装され、デフォルトで有効になっています。

1.4. ディレクトリー設計の概要

ディレクトリーを正しく運用するためには、実際のデプロイメントの前にディレクトリーサービスを計画することが最も重要なタスクです。設計プロセスでは、環境やデータソース、ユーザー、ディレクトリーを使用するアプリケーションなど、ディレクトリー要件に関するデータを収集する必要があります。この情報は、設定と必要な機能を特定するのに役立つため、有効なディレクトリーサービスを設計するのに不可欠です。

Directory Server に柔軟性を持たせると、Directory Server のデプロイ後でも、再度ディレクトリー設計を行い、予期しない要件や変動する要件を満たすことができます。

1.4.1. 設計プロセスの概要

1. *2章 ディレクトリーデータのプランニング*

   ディレクトリーには、ユーザー名、電話番号、グループの詳細などのデータが含まれます。本章では、組織内のさまざまなデータソースを分析し、相互関係を理解します。これは、ディレクトリーに格納できるデータのタイプと、Directory Server の内容を設計するための他のタスクを説明します。

2. *3章 ディレクトリースキーマの設計*

ディレクトリーは、1 つ以上のディレクトリー対応アプリケーションをサポートするように設計されています。これらのアプリケーションには、ファイル形式などのディレクトリーに保存されているデータに対する要件があります。ディレクトリースキーマは、ディレクトリーに格納されているデータの特性を決定します。本章では、Directory Server に同梱される標準スキーマに加えて、スキーマをカスタマイズする方法および一貫性のあるスキーマを維持するためのヒントも紹介します。

3. *4章 ディレクトリーツリーの設計*

   Directory Server に **どのような** 情報を含めるかを決定すると共に、その情報を **どのように** 整理して参照するかを決定することが重要です。本章では、ディレクトリーツリーを紹介し、データ階層の設計の概要を説明します。サンプルディレクトリーツリー設計も提供されます。

4. *6章 ディレクトリートポロジーの設計*

   トポロジー設計とは、ディレクトリーツリーを複数の物理 Directory Server に分割する方法、およびこれらのサーバーが相互に通信する方法を決定することを指します。設計の前提となる一般原則、複数のデータベースの使用、分散データの統合に使用できるメカニズム、およびディレクトリー自体が分散データを追跡する方法を、すべて本章で説明します。

5. *7章 レプリケーションプロセスの設計*

   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ーデータを維持し、パフォーマンスが向上し、耐障害性が提供されます。本章では、レプリケーションの仕組み、レプリケーションできるデータタイプ、一般的なレプリケーションシナリオ、および高可用性ディレクトリーサービスを構築するためのヒントを紹介します。

6. *8章 同期の設計*

   Red Hat Directory Server に保存されている情報は、Microsoft Active Directory データベースに保存されている情報と同期でき、複合プラットフォームインフラストラクチャーとの統合を向上させることができます。本章では、同期の仕組み、どのデータの種類を同期できるか、また同期に最適な情報のタイプおよびディレクトリーツリー内の場所についての考慮事項を説明します。

7. *9章 セキュアなディレクトリーの設計*

   最後に、ユーザーおよびアプリケーションのセキュリティー要件を満たすために、ディレクトリー内のデータを保護する方法や、サービスの他の側面を計画します。本章では、一般的なセキュリティーの脅威、セキュリティーメソッドの概要、セキュリティーニーズの分析に関する手順、ならびにアクセス制御の設計およびディレクトリーデータの整合性の保護に関するヒントについて説明します。

1.4.2. ディレクトリーのデプロイ

Directory Server 導入の最初のステップは、テストサーバーインスタンスをインストールし、サービスがユーザーの負荷を処理できるようにすることです。初期設定でサービスが適切でない場合には、デザインを調整して再度テストします。自信を持ってエンタープライズ環境に導入できる堅牢なサービスになるまで、設計を調整します。

パイロット版ディレクトリーの作成と実装に関する包括的な概要は、『Understanding and Deploying LDAP Directory Services』(T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999) を参照してください。

適切なテスト用 Directory Server インスタンスを作成して調整した後に、ディレクトリーサービスを実稼働環境に移行するための計画を作成します。計画には、以下の考慮事項を含めます。

- 必須リソースの推定
- 達成すべき事項および時期に関するスケジュール
- 導入の成功を測定するための基準セット

ディレクトリーサービスのインストールに関する情報は『Red Hat Directory Server インストールガイド』を、ディレクトリーの管理および維持に関する情報は『Red Hat Directory Server 管理ガイド』を、それぞれを参照してください。

1.5. その他の一般的なディレクトリーリソース

以下の出版物には、ディレクトリー、LDAP、および LDIF に関する詳細で有用な情報が記載されています。

- RFC 2849: The LDAP Data Interchange Format (LDIF) Technical Specification, http://www.ietf.org/rfc/rfc2849.txt
- RFC 2251: Lightweight Directory Access Protocol (v3) (http://www.ietf.org/rfc/rfc2251.txt)
- 『Understanding and Deploying LDAP Directory Services』T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999.

https://access.redhat.com/documentation/ja-jp/red_hat_directory_server で利用可能な Red Hat Directory Server ドキュメントには、Directory Server 固有の情報に加えて、LDAP の使用およびディレクトリーサービスの管理に関する基本概念が含まれています。

第2章 ディレクトリーデータのプランニング

ディレクトリーに保存されているデータには、ユーザー名、メールアドレス、電話番号、ユーザーが属するグループに関する情報や、その他のタイプの情報が含まれます。ディレクトリー内のデータタイプによって、ディレクトリーがどのように構造化されるか、だれがデータにアクセスできるか、アクセス権限の要求と付与の方法が決定されます。

本章では、ディレクトリーデータの計画における問題とストラテジーについて説明します。

2.1. ディレクトリーデータの概要

一部の種類のデータは、他よりもディレクトリーに適しています。ディレクトリーに理想的なデータには、以下の特徴があります。

- 書き込みよりも読み取りが頻繁である。
- 属性データ形式で表現可能です（例： **surname=jensen**）。
- 複数の人やグループがそのデータに関心を持っている。たとえば、従業員の名前またはプリンターの場所には、多くの人やアプリケーションが関心を持ちます。
- 複数の物理的な場所からアクセスされる。

たとえば、従業員のソフトウェアアプリケーション設定は、1つのアプリケーションインスタンスのみが情報にアクセスする必要があるため、ディレクトリーに適するとは見なされません。ただし、アプリケーションがディレクトリーから設定を読み取りでき、ユーザーが異なるサイトから設定に従ってアプリケーションと対話したい場合は、ディレクトリーに設定情報を含めると非常に便利です。

2.1.1. ディレクトリーに含める情報

人やアセットに関する説明や有用な情報は、属性としてエントリーに追加することができます。以下に例を示します。

- 電話番号、物理アドレス、メールアドレスなどの連絡先情報
- 従業員番号、ジョブタイトル、マネージャーまたは管理者 ID 、仕事に関する興味などの説明情報
- 組織の連絡先情報 (電話番号、物理アドレス、管理者 ID、ビジネスの説明など)
- プリンターの物理的な場所、プリンターの種類、プリンターが作成できる 1 分あたりのページ数などのデバイス情報
- 企業の取引パートナー、取引先、および顧客に関する連絡先および請求情報
- 顧客の名前、納期、ジョブの説明、課金情報などのコントラクト情報
- 個別のソフトウェア設定またはソフトウェア設定情報
- Web サーバーへのポインター、または特定のファイルまたはアプリケーションのファイルシステムなどのリソースサイト

Directory Server をサーバー管理以外にも使用するには、ディレクトリーに格納する情報のその他のタイプを計画する必要があります。以下に例を示します。

- コントラクトまたはクライアントアカウントの詳細

- 給与データ
- 物理デバイス情報
- 自宅連絡先情報
- エンタープライズ内のさまざまなサイトに関するオフィスの連絡先情報

2.1.2. ディレクトリーから除外する情報

Red Hat Directory Server は、クライアントアプリケーションが読み取りおよび書き込みする大量のデータを管理するのに最適ですが、イメージや、その他のメディアなどの大規模な非構造化オブジェクトを処理することは設計されていません。これらのオブジェクトはファイルシステムで維持する必要があります。ただし、ディレクトリーは FTP、HTTP およびその他のサイトへのポインター URL を使用すると、このような種類のアプリケーションへのポインターを格納できます。

2.2. ディレクトリーニーズの定義

ディレクトリーデータを設計する場合は、現在必須なデータだけでなく、ディレクトリー (および組織) が時間の経過と共にどのように変化するのかも考えてください。設計プロセス中にディレクトリーの今後のニーズを検討すると、ディレクトリー内のデータが構造化され、分散される方法が異なります。

以下の点に注意してください。

- 今ディレクトリーに何を格納すべきか
- ディレクトリーを導入するすることで、どんな直近の問題が解決されるか
- 使用中のディレクトリー対応アプリケーションの直近のニーズは何か
- 近日中にディレクトリーに追加される情報は何か。たとえば、ある企業では、現在 LDAP をサポートしていない会計パッケージを使用しているが、これが数ヶ月後に LDAP 対応になる可能性があります。LDAP 対応アプリケーションで使用されるデータを特定し、技術が利用可能になった時点でデータをディレクトリーに移行する計画を作成します。
- 今後、ディレクトリーに格納されている可能性のある情報は何か。たとえば、ホスト企業は、イメージやメディアファイルを格納する必要があるなど、現行の顧客とは異なるデータ要件を持つ顧客を持つ可能性があります。これは予測が困難な回答ですが、予期せぬ成果を生む場合があります。この種の計画は、少なくとも、考慮されなかった可能性のあるデータソースの特定に役立ちます。

2.3. サイト調査の実行

サイト調査は、ディレクトリーの内容を検出し特徴付けるための正式な方法です。準備がディレクトリーアーキテクチャーのキーであるため、サイト調査の実施に十分な時間を費やします。サイト調査は複数のタスクで設定されます。

- ディレクトリーを使用するアプリケーションを特定する。

  エンタープライズ全体にデプロイされるディレクトリー対応アプリケーションとそのデータのニーズを判断する。

- データソースを特定する。

  企業の調査を行い、Active Directory、その他の LDAP サーバー、PBX システム、人材データベース、電子メールシステムなどのデータソースを特定する。

- ディレクトリーに含まれる必要のあるデータを特徴付ける。

  ディレクトリーに存在すべきオブジェクト (例: 人またはグループ) と、ディレクトリーで維持すべきオブジェクトの属性 (ユーザー名とパスワードなど) を決定します。

- 提供するサービスレベルを決定する。

  ディレクトリーデータをクライアントアプリケーションに提供する際の可用性を決定し、それに応じてアーキテクチャーを設計します。ディレクトリーに要求される可用性が、データのレプリケート方法およびリモートサーバーに保存されたデータを接続するためのチェーンポリシーの設定方法に影響を及ぼします。

  レプリケーションについての詳細は*7章 レプリケーションプロセスの設計*を、チェーンの詳細についての詳細は「トポロジーの概要」を、それぞれ参照してください。

- データサプライヤーを特定する。

  データサプライヤーには、ディレクトリーデータのプライマリーソースが含まれます。このデータは、負荷分散および復元の目的で、他のサーバーにミラーリングされる可能性があります。各データで、データサプライヤーを決定します。

- データの所有者を決定する。

  各データについて、データが最新の状態であることを確認する担当者を決定します。

- データアクセスを決定する。

  データを他のソースからインポートする場合には、一括インポートと増分更新の両方のストラテジーを作成します。このストラテジーの一部として、データを 1 カ所で管理し、データを変更できるアプリケーションの数を制限するようにします。また、任意のデータに書き込みするユーザーの数を制限します。小規模なグループにより、管理のオーバーヘッドを削減しつつ、データの整合性が確保されます。

- サイト調査を文書化します。

ディレクトリーの影響を受ける組織の数により、影響を受ける各組織の代表を含むディレクトリー導入チームを作成して、サイト調査を実行することが役に立つ場合があります。

企業には、一般的に、人事部門、経理部門、製造部門、営業部門、および開発部門があります。各組織からの代表を含めると、調査プロセスに役に立ちます。さらに、影響を受けるすべての組織が直接関与することで、ローカルデータストアから集中ディレクトリーへの移行が受け入れられます。

2.3.1. ディレクトリーを使用するアプリケーションの特定

通常、ディレクトリーにアクセスするアプリケーションおよびこれらのアプリケーションのデータニーズが、ディレクトリーの内容の計画を駆動します。多くの共通アプリケーションはディレクトリーを使用します。

- **オンライン電話帳などのディレクトリーブラウザーアプリケーション**。ユーザーが必要な情報 (メールアドレス、電話番号、従業員名など) を決定し、ディレクトリーに追加します。

- **電子メールアプリケーション (特に電子メールサーバー)**すべての電子メールサーバーには、ディレクトリーで利用可能なメールアドレス、ユーザー名、および一部のルーティング情報が必要です。ただし、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場所や、不在時の自動返信情報、プロトコル情報 (たとえば、IMAP と POP) など、より高度な情報が必要です。

- **ディレクトリー対応人事アプリケーション**。これには、マイナンバー、自宅住所、自宅電話番号、生年月日、所得、およびジョブタイトルなど、より個人的な情報が必要です。

- **Microsoft Active Directory**.Windows User Sync により、Windows ディレクトリーサービスを統合して、Directory Server と連動できます。どちらのディレクトリーでも、ユーザー情報 (ユーザー名とパスワード、メールアドレス、電話番号など) およびグループ情報 (メンバー) を保存できます。ユーザー、グループ、その他のディレクトリーデータがスムーズに同期できるように、既存の Windows サーバー導入の後に Directory Server の導入をスタイルします (またはその逆)。

ディレクトリーを使用するアプリケーションを検証する場合は、各アプリケーションが使用する情報の種類を確認します。以下の表は、アプリケーションの例と、各アプリケーションによって使用される情報を示しています。

表2.1 アプリケーションデータニーズの例

アプリケーション	データのクラス	データ
電話帳	人	名前、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、マネージャー、メール停止
Web サーバー	人、グループ	ユーザー ID、パスワード、グループ名、グループメンバー、グループ所有者
カレンダーサーバー	人、会議室	名前、ユーザー ID、立方数、会議室名

アプリケーションおよび各アプリケーションによって使用される情報の特定後、一部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明確になります。データ計画段階でこのような演習を実行すると、ディレクトリーにおけるデータ冗長性の問題を回避し、ディレクトリー依存アプリケーションが必要とするデータをより明確にすることができます。

ディレクトリーで維持するデータの種類および情報をディレクトリーに移行する時期に関する最終的な決定は、以下の要因の影響を受けます。

- さまざまなレガシーアプリケーションとユーザーが必要とするデータ

- レガシーアプリケーションの LDAP ディレクトリーと通信する能力

2.3.2. データソースの特定

ディレクトリーに含めるすべてのデータを特定するには、既存のデータストアの調査を実行します。調査には以下を含める必要があります。

- 情報を提供する組織を特定する。

  エンタープライズにとって必要とされる情報を管理するすべての組織を見つけます。通常、これには情報サービス、人事、支払い部門、経理部門が含まれます。

- 情報ソースであるツールおよびプロセスを特定する。

  情報の一般的なソースには、ネットワークオペレーティングシステム (Windows、Willell Netware、UNIX NIS)、電子メールシステム、セキュリティーシステム、PBX (電話交換) システム、および人事アプリケーションなどがあります。

- 各データを一元化する方法を判断することが、データの管理に影響を及ぼします。

  集中データ管理では、新しいツールおよび新しいプロセスが必要になる場合があります。集中化により、一部の組織でスタッフを増やす必要があり、他の組織のスタッフが減少する必要があることがあります。

調査の実行中に、表2.2「情報ソースの例」のような、エンタープライズのすべての情報ソースを特定するマトリックスの開発を検討します。

表2.2 情報ソースの例

データソース	データのクラス	データ
人事データベース	人	名前、住所、電話番号、部署番号、マネージャー
E メールシステム	人、グループ	名前、メールアドレス、ユーザー ID、パスワード、電子メール設定
ファシリティーシステム	ファシリティー	ビル名、フロア名、立方数、アクセスコード

2.3.3. ディレクトリーデータの特徴付け

ディレクトリーに含まれるように識別されたすべてのデータは、以下の一般的なポイントに従って特徴付けることができます。

- 形式
- サイズ
- 各種アプリケーションで発生回数
- データの所有者
- 他のディレクトリーデータとの関係

ディレクトリーに含める各種類のデータを調査し、他のデータと共有する特性を決定します。これは、スキーマ設計の段階で時間を節約するのに役立ちます。詳細は、*3章ディレクトリースキーマの設計*で説明されています。

ディレクトリーデータを特徴付ける 表2.3「ディレクトリーデータの特徴付け」のようなテーブルを使用することが良いでしょう。

表2.3 ディレクトリーデータの特徴付け

データ	形式	サイズ	所有者	関連
従業員名	テキスト文字列	128 文字	人事	ユーザーエントリー
Fax 番号	電話番号	14 桁の数字	ファシリティー	ユーザーエントリー

データ	形式	サイズ	所有者	関連
メールアドレス	テキスト	多くの文字	情報システム部門	ユーザーエントリー

2.3.4. サービスレベルの決定

提供されるサービスレベルは、ディレクトリー対応アプリケーションに依存するユーザーの期待値により異なります。各アプリケーションが想定するサービスのレベルを決定するには、まずアプリケーションの使用方法とタイミングを決定します。

ディレクトリーが発展するにつれて、さまざまなサービスレベル (本番環境からミッションクリティカルまで) をサポートする必要がある場合があります。ディレクトリーのデプロイ後にサービスレベルを上げることは困難な場合があるため、初期設計が今後のニーズを満たすようにしてください。

たとえば、合計の失敗のリスクを排除する必要がある場合は、同じデータに対して複数のサプライヤーが存在する、マルチサプライヤーの設定を使用します。

2.3.5. データサプライヤーの検討

データサプライヤー は、データソースのサプライヤーであるサーバーです。同じ情報が複数の場所に保存されるたびに、データの整合性が低下する可能性があります。データサプライヤーにより、複数の場所に保存されている全情報の一貫性が保たれ、正確性が確保されます。データサプライヤーが必要なシナリオはいくつかあります。

- Directory Server 間でのレプリケーション
- Directory Server と Active Directory との間の同期
- Directory Server データにアクセスする独立したクライアントアプリケーション

ディレクトリーに間接的に通信するアプリケーションがある場合には、データソースのサプライヤーについて検討してください。データを変更するプロセスと、データの変更元の場所を可能な限りシンプルに維持します。データを管理する単一のサイトを決定したら、同じサイトを使用して、そこに含まれる他のデータをすべて管理します。単一サイトは、データベースが企業全体で同期を失った場合にトラブルシューティングを簡素化します。

データ提供を実装する方法は複数あります。

- ディレクトリーと、ディレクトリー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を管理する。

  複数のデータサプライヤーを維持する場合、ディレクトリーおよびその他のアプリケーションにデータを出し入れするのにカスタムスクリプトは必要ありません。ただし、データが 1 カ所で変更された場合は、他のすべてのサイトでデータを変更する必要があります。ディレクトリーおよびディレクトリーを使用しないすべてのアプリケーションでサプライヤーデータを維持すると、データが企業全体で同期されないことになります (ディレクトリーはこれを防ぐことが期待されます)。

- ディレクトリー以外の一部のアプリケーションでデータを管理し、スクリプト、プログラム、またはゲートウェイを作成し、そのデータをディレクトリーにインポートする。

データを管理するのにすでに使用されているアプリケーションが 1 つまたは 2 つある場合には、ディレクトリーを使用しないアプリケーションでデータを管理するのが最も妥当です。ディレクトリーは検索にのみ使用されます (例: オンラインの企業電話帳)。

データのメインコピーの維持方法は、特定のディレクトリーのニーズによって異なります。ただし、データサプライヤーがどのように維持されているかに関係なく、単純性と一貫性を維持します。たとえば、複数のサイトでデータの管理を試みないでください。その場合、競合するアプリケーション間でデータを自動的に交換します。これを行うと、last change wins のシナリオが発生し、管理のオーバーヘッドが増加します。

たとえば、ディレクトリーは従業員の自宅電話番号を管理します。LDAP ディレクトリーと人事データベースの両方がこの情報を保存します。人事アプリケーションは LDAP 対応のため、LDAP ディレクトリーから人事データベースへデータを自動的に転送するアプリケーションを作成することができます。その逆の場合も、アプリケーションを作成できます。

ただし、LDAP ディレクトリーと人事データの両方で、従業員の電話番号の変更を管理しようとしていますが、電話番号が変更された最後の場所が他のデータベースの情報を上書きしてしまいます。これは、データを書き込む最後のアプリケーションに正しい情報がある場合に限り許容されます。

人事データがバックアップからリロードされたため、その情報が古くなっていた場合は、LDAP ディレクトリーの正しい電話番号が削除されます。

マルチサプライヤーレプリケーションにより、Directory Server には、複数のサーバーで情報のソースを含めることができます。複数のサプライヤーは変更ログを保持し、競合をより安全に解決できます。限られた数の Directory Server は変更を受け入れるサプライヤーと見なされ、データをレプリカサーバーまたはコンシューマーサーバーに複製します。[1] 複数のデータサプライヤーサーバーあると、サーバーがオフになった場合に安全なフェイルオーバーを提供します。レプリケーションおよびマルチサプライヤーレプリケーションの詳細は、*7章 レプリケーションプロセスの設計*を参照してください。

同期により、Directory Server ユーザー、グループ、属性、およびパスワードを Microsoft Active Directory ユーザー、グループ、属性、パスワードと統合できます。2 つのディレクトリーサービスでは、それらが同じ情報を取り扱うか、共有されるその情報の量、およびその情報のデータサプライヤーとなるサービスを決定します。最適なのは、データを管理する 1 つのアプリケーションを選択し、同期プロセスで他のサービスのエントリーを追加、更新、または削除できるようにすることです。

2.3.6. データオーナーの決定

データの所有者 は、データを最新の状態に維持させる人または組織を指します。データ設計フェーズで、ディレクトリーにデータを書き込むことができるユーザーを決定します。以下は、データの所有者を決定する一般的なストラテジーです。

- ディレクトリーコンテンツマネージャーの小さなグループ以外のすべてのユーザーに対して、ディレクトリーへの読み取り専用アクセスを許可します。
- 個々のユーザーが、自分に関する情報の戦略的サブセットを管理できるようにします。

  この情報のサブセットには、パスワード、自身の説明情報、組織内のロールり、ナンバープレートの番号、電話番号やオフィス番号などの連絡先情報が含まれる可能性があります。
- ユーザーのマネージャーに、連絡先情報やジョブタイトルなど、そのユーザーの情報の戦略的サブセットへの書き込みを可能にします。
- 組織の管理者が、その組織のエントリーを作成して管理できるようにします。

  この方法では、組織の管理者がディレクトリーコンテンツマネージャーとして機能します。
- 読み取りまたは書き込みアクセス権限を持つユーザーグループのロールを作成します。

たとえば、人事、会計、またはアカウンティング向けに作成されたロールがあります。これらの各ロールに、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、書き込みアクセス、またはその両方を許可します。これには、給与情報、マイナンバー、および自宅電話番号および住所が含まれる可能性があります。

ロールおよびエントリーのグループ化の詳細は、「ディレクトリーエントリーのグループ化」を参照してください。

同じ情報への書き込みアクセス権が必要な個人が複数存在する場合があります。たとえば、情報システム (IS) またはディレクトリー管理グループには、おそらく従業員パスワードへの書き込みアクセスが必要になります。また、従業員自体にも自分のパスワードへの書き込みアクセス権があることが望ましい場合があります。一般的には、複数のユーザーが同じ情報への書き込みアクセス権を持つ場合には、このグループを小さくし、簡単に識別できるようにしてください。グループを小さくすると、データの整合性が確保されます。

ディレクトリーのアクセス制御の設定に関する詳細は、9章 *セキュアなディレクトリーの設計*を参照してください。

2.3.7. データアクセスの決定

データの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、従業員の自宅電話番号はディレクトリーに保存できます。このデータは、従業員のマネージャーや人事など、多くの組織に役に立つ場合があります。従業員は、検証目的でこの情報を読み取ることができる必要があります。しかし、自宅の連絡先情報は機密であるとみなされる可能性があるため、企業全体で広く利用するべきではありません。

ディレクトリーに保存されている各情報について、以下を決定します。

- データは匿名で読み取りできるか

  LDAP プロトコルは匿名アクセスをサポートし、オフィスサイト、メールアドレス、ビジネス電話番号などの情報を簡単に検索することができます。ただし、匿名アクセスにより、誰でも共通情報へのディレクトリーアクセスが可能になります。したがって、匿名アクセスの使用は限定的にします。

- データは企業全体で読み取りできるか

  アクセス制御は、クライアントが特定の情報を読み取るためにディレクトリーへのログイン (またはバインド) する必要があるように設定できます。匿名アクセスとは異なり、この形式のアクセス制御により、組織のメンバーのみがディレクトリー情報を表示できます。また、ディレクトリーのアクセスログにログイン情報を取得するので、誰が情報にアクセスしたかが記録に残ります。

  アクセス制御の詳細は、「アクセス制御の設計」を参照してください。

- データの読み取りが必要な人やアプリケーションの特定可能なグループはあるか

  データへの書き込み権限があるユーザーは、通常 (パスワードへの書き込みアクセスを除き) 読み取りアクセスが必要です。特定の組織またはプロジェクトグループに固有のデータがある場合もあります。これらのアクセスのニーズを特定することは、ディレクトリーが必要とするグループ、ロール、およびアクセス制御を把握するのに役立ちます。

  グループおよびロールの詳細は、4章 *ディレクトリーツリーの設計*を参照してください。アクセス制御の詳細は、「アクセス制御の設計」を参照してください。

ディレクトリーデータごとに、これらの決定を行うと、ディレクトリーのセキュリティーポリシーが定義されます。この決定は、サイトの性質と、サイトで利用できるセキュリティーの種類によって異なり

ます。たとえば、ファイアウォールがあったりインターネットに直接アクセスできない場合は、ディレクトリーがインターネットに直接配置される場合よりも匿名アクセスのサポートは安全です。さらに、アクセスを制限する際にアクセス制御と認証手段のみが必要になる場合もあります。その他の機密情報は保存時にデータベース内で暗号化する必要がある場合があります。

多くの国では、データ保護の法律で、企業がどのように個人情報を維持するか、個人情報へのアクセスを制限するかが規定されています。たとえば、法律では、アドレスと電話番号への匿名アクセスを禁止したり、ユーザーがそれらを表すエントリーの情報を表示したり、修正できることを要求したりする場合があります。ディレクトリーの導入が、エンタープライズが活動する国で必要なすべての法に従うように、組織の法務部門に必ず確認してください。

セキュリティーポリシーの作成と、実装する方法の詳細は、9章 *セキュアなディレクトリーの設計*を参照してください。

2.4. サイト調査の文書化

データ設計の複雑性により、サイト調査の結果を文書化します。サイト調査の各ステップでは、単純なテーブルを使用してデータを追跡することができます。決定および未処理の懸念を概説する、サプライヤーテーブルの構築を検討します。テーブルの内容の並べ替えと検索を簡単に行えるように、適切なヒントとしてはスプレッドシートを使用します。

表2.4「例: データ所有者およびアクセスのリスト」は、サイト調査によって識別された各データに対するデータの所有者とデータアクセスを特定しています。

表2.4 例: データ所有者およびアクセスのリスト

データ名	所有者	サプライヤーサーバー/アプリケーション	自己読み取り/書き込み	グローバルな読み取り	人事部門による書き込み	情報システム部門による書き込み
従業員名	HR	PeopleSoft	読み取り専用	○ (匿名)	はい	はい
ユーザーパスワード	IS	Directory US-1	読み取り/書き込み	×	×	○
自宅電話番号	HR	PeopleSoft	読み取り/書き込み	×	○	×
従業員のロケーション	IS	Directory US-1	読み取り専用	○ (ログインが必要)	×	○
オフィスの電話番号	ファシリティー	Phone switch	読み取り専用	○ (匿名)	×	×

表の各行には、評価されている情報、関連する部署、その情報の使用方法、およびアクセス方法が表示されます。たとえば、1行目では、**従業員名** データには以下のような管理上の考慮事項があります。

- **所有者**:人事部門はこの情報を所有しているため、更新や変更を行います。
- **サプライヤーサーバー/アプリケーション**:PeopleSoft アプリケーションは従業員名の情報を管理します。

- **自己読み取り/書き込み**:ユーザーは自分の名前を読み取ることができますが、書き込み (または変更) することはできません。
- **グローバルな読み取り**:従業員名は、ディレクトリーへのアクセスがあるすべてのユーザーが匿名で読み取ることができます。
- **人事部門による書き込み**:人事グループのメンバーは、ディレクトリー内の従業員名を変更、追加、および削除できます。
- **情報システム部門による書き込み**:情報サービスグループのメンバーは、ディレクトリー内の従業員名を変更、追加、および削除できます。

2.5. サイト調査の繰り返し

特に企業に複数の都市や国にオフィスがある場合は、複数のサイト調査が必要になる場合があります。情報に対するニーズが非常に複雑なので、複数の異なる組織では、1 つの中央サイトではなくローカルのオフィスで情報を維持する必要があるかもしれません。

この場合、情報のメインコピーを維持する各オフィスでは、独自のサイト調査を実行する必要があります。サイトの調査プロセスが完了すると、各調査の結果が中央のチーム (各オフィスからの代表で設定される) に返送され、企業全体のデータスキーマモデルとディレクトリーツリーの設計に使用されます。

---

[1] レプリケーションでは、**コンシューマーサーバー** または **レプリカサーバー** は、サプライヤーサーバーまたはハブサーバーから更新を受け取るサーバーです。

第3章 ディレクトリースキーマの設計

2章 *ディレクトリーデータのプランニング*で実施したサイト調査により、ディレクトリーに格納されるデータに関する情報が明確になりました。ディレクトリー **スキーマ** はディレクトリー内のデータの種類を記述します。そのため、使用するスキーマを決定することで、ディレクトリーに保存されているデータを表す方法に関する決定が反映されます。スキーマ設計プロセスで、各データ要素は LDAP 属性にマッピングされ、関連する要素は LDAP オブジェクトクラスに収集されます。適切に設計されたスキーマは、ディレクトリーデータの整合性を維持するのに役立ちます。

本章では、ディレクトリースキーマと、固有の組織ニーズに合わせてスキーマを設計する方法を説明します。

スキーマの複製に関する詳細は、「スキーマレプリケーション」を参照してください。

3.1. スキーマ設計プロセスの概要

スキーマ設計プロセスで、Red Hat Directory Server に保存されているエントリーを表すために使用されるオブジェクトクラスおよび属性を選択して定義します。スキーマの設計には、以下のステップが必要です。

1. 可能な限り多くのデータニーズに合わせて事前定義されたスキーマ要素を選択する。
2. 標準の Directory Server スキーマを拡張して、その他の残りのニーズを満たす新しい要素を定義する。
3. スキーマメンテナンスをプランニングする。

最もシンプルで簡単にメンテナンスされるオプションは、Directory Server で提供され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。標準のスキーマ要素を選択すると、ディレクトリー対応アプリケーションとの互換性が確保されます。スキーマは LDAP 標準をベースとしているため、多くのディレクトリーユーザーによるレビューおよび合意が行われました。

3.2. 標準スキーマ

ディレクトリースキーマは、データ値のサイズ、範囲、およびフォーマットに制約を課すことにより、ディレクトリーに保存されているデータの整合性を維持します。スキーマは、ディレクトリーに含まれるエントリーの種類 (ユーザー、デバイス、組織など) に関する決定を反映しています。

Directory Server に含まれる事前定義スキーマには、標準の LDAP スキーマと、サーバーの機能をサポートする追加のアプリケーション固有のスキーマの両方が含まれています。このスキーマはほとんどのディレクトリーのニーズに対応しますが、ディレクトリーの固有のニーズに対応するために、新しいオブジェクトクラスおよび属性をスキーマに追加できます (**スキーマの拡張**)。スキーマの拡張に関する詳細は、「スキーマのカスタマイズ」を参照してください。

3.2.1. スキーマの形式

Directory Server のスキーマ形式は、LDAP プロトコルのバージョン 3 をベースとしています。このプロトコルでは、ディレクトリーサーバーは LDAP 自体によりそのスキーマを公開する必要があります。これにより、ディレクトリークライアントアプリケーションがプログラム的にスキーマを取得し、その動作を調整することができます。Directory Server のスキーマのグローバルセットは **cn=schema** エントリーにあります。

Directory Server スキーマは、独自のプロプライエタリーオブジェクトクラスおよび属性を使用するため、LDAPv3 スキーマとは若干異なります。さらに、スキーマエントリーが元々定義された場所を記述する **X-ORIGIN** というスキーマエントリーのプライベートフィールドを使用します。

たとえば、スキーマエントリーが標準の LDAPv3 スキーマで定義されている場合、**X-ORIGIN** フィールドは RFC 2252 を参照します。Directory Server の使用のために Red Hat によってエントリーが定義されている場合、**X-ORIGIN** フィールドには **Netscape Directory Server** の値が含まれます。

たとえば、標準の **person** オブジェクトクラスは以下のようにスキーマに表示されます。

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

このスキーマエントリーは、クラス(**2.5.6**.6)、オブジェクトクラスの名前(**person**)、クラスの説明(**Standard Person**)、必要な属性(***objectclass***、***sn***、および ***cn***)、および許可される属性(***description***、***seeAlso***、***telephoneNumber***、および ***userPassword***)を記載します。

LDAPv3 スキーマ形式の詳細は、LDAPv3 属性構文の定義ドキュメント RFC 2252 ならびにその他の標準スキーマ定義 RFC 247、RFC 2927、および RFC 2307 参照してください。これらのスキーマ要素はすべて Red Hat Directory Server でサポートされています。

3.2.2. 標準属性

属性には、名前や fax 番号などの特定のデータ要素が含まれます。Directory Server は、特定の情報に関連付けられた説明的なスキーマ属性である **属性/データのペア** *としてデータを表します。これらは* **属性と値の言明** *または AVA とも呼ばれます。*

たとえば、ディレクトリーは、標準属性を持つペアに人の名前などのデータを保存できます（ここでは ***commonName*** *(****cn****)）。そのため、Babs Jensen という名前のユーザーのエントリーには、属性データペア* ***cn: Babs Jensen*** *があります。*

実際、エントリー全体は属性/データのペアのセットとして表されます。Babs Jensen のエントリー全体は以下のようになります。

```
dn: uid=bjensen,ou=people,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Babs Jensen
sn: Jensen
givenName: Babs
givenName: Barbara
mail: bjensen@example.com
```

*Babs Jensen のエントリーでは、一部の属性に複数の値が含まれます。****givenName*** *属性は、それぞれ一意の値で 2 回表示されます。*

スキーマでは、各属性定義には以下の情報が含まれます。

- *一意な名前*
- *属性のオブジェクト識別子 (OID)*
- *属性のテキストでの説明*
- *属性構文の OID*

- 属性が単一値か複数値かどうか、属性がディレクトリー専用かどうか、属性の元、および属性に関連付けられた追加のマッチングルールを示します。

たとえば、**cn** 属性定義は以下のようにスキーマに表示されます。

```
attributetypes: ( 2.5.4.3 NAME 'cn' DESC 'commonName Standard Attribute'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

属性の構文は属性で許可される値の形式を定義します。これにより、構文は属性に保存できる情報の種類を定義するのに役立ちます。Directory Server は、すべての標準属性構文をサポートします。

サポートされている LDAP 属性の構文は、Red Hat Directory Server 10 Configuration、Command、and File Reference の『Directory ServerAttribute Syntaxes』のセクションで説明されています。

3.2.3. 標準のオブジェクトクラス

オブジェクトクラスは、関連情報をグループ化するために使用されます。通常、オブジェクトクラスはユーザーまたは fax マシンなどの実際のオブジェクトを表します。ディレクトリーでオブジェクトクラスとその属性を使用する前に、スキーマで特定する必要があります。ディレクトリーは、デフォルトでオブジェクトクラスの標準リストを認識します。これらは、『Red Hat Directory Server 設定、コマンド、およびファイルリファレンス』に一覧として記載されています。

各ディレクトリーエントリーは、少なくとも 1 つのオブジェクトクラスに属します。エントリーにスキーマで特定されたオブジェクトクラスを配置すると、エントリーは任意の属性値の特定セットを持ち、さらに必須の属性値のセット (通常は小さい) を持つことを Directory Server に伝えます。

オブジェクトクラス定義には、以下の情報が含まれます。

- 一意な名前
- オブジェクトの名前である **オブジェクト識別子** (OID)
- 必須属性のセット
- 可能な (または任意の) 属性のセット

たとえば、標準の **person** オブジェクトクラスは以下のようにスキーマに表示されます。

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

すべての Directory Server のスキーマの場合と同様に、オブジェクトクラスが定義され、Directory Server に直接保存されます。つまり、ディレクトリーのスキーマは標準の LDAP 操作でクエリーおよび変更できます。

3.3. データのデフォルトスキーマへのマッピング

「サイト調査の実行」で説明されているサイト調査中に特定されたデータは、既存のデフォルトディレクトリースキーマにマッピングされる必要があります。ここでは、既存のデフォルトスキーマを表示する方法を説明し、データを適切な既存のスキーマ要素にマッピングする方法を提供します。

スキーマに、既存のデフォルトスキーマに一致しない要素がある場合は、カスタムオブジェクトクラスおよび属性を作成します。詳細は、「スキーマのカスタマイズ」を参照してください。

3.3.1. デフォルトのディレクトリースキーマの表示

デフォルトのディレクトリースキーマは **/usr/share/dirsrv/schema/** ディレクトリーに保存されます。

このディレクトリーには、Directory Server の共通スキーマがすべて含まれています。LDAPv3 標準ユーザーおよび組織スキーマは **00core.ldif** ファイルにあります。以前のバージョンのディレクトリーで使用される設定スキーマは **50ns-directory.ldif** ファイルにあります。

警告

デフォルトのディレクトリースキーマは **変更しないでください。**

ディレクトリーにある各オブジェクトクラスと属性の詳細は、『Red Hat Directory Server の設定、コマンド、およびファイルリファレンス』を参照してください。また、スキーマファイルおよびディレクトリーの設定属性に関する詳細も説明しています。

3.3.2. スキーマ要素へのデータのマッチング

サイト調査で特定されたデータは、既存のディレクトリースキーマにマッピングされる必要があります。このプロセスには、以下のステップが必要です。

1. データが記述するオブジェクトのタイプを特定する。

   サイト調査で説明されるデータに最もマッチするオブジェクトを選択します。データは、複数のオブジェクトを記述できる場合があります。ディレクトリースキーマで違い記述する必要があるかどうかを判断します。

   たとえば、電話番号は、従業員の電話番号と会議部屋の電話番号を記述できます。ディレクトリースキーマで、これらの異なるデータを異なるオブジェクトとして見なす必要があるかどうかを判断します。

2. 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。

   グループ、ユーザー、組織など、共通のオブジェクトクラスを使用すると良いでしょう。

3. マッチするオブジェクトクラスから類似の属性を選択する。

   マッチするオブジェクトクラスから、サイト調査で特定されたデータと最もマッチする属性を選択します。

4. サイト調査からマッチしないデータを特定する。

   デフォルトのディレクトリースキーマで定義されたオブジェクトクラスおよび属性とマッチしないデータがある場合には、スキーマをカスタマイズします。詳細は、「スキーマのカスタマイズ」を参照してください。

たとえば、以下の表では、ディレクトリースキーマ要素を2章ディレクトリーデータのプランニングのサイト調査中に特定されたデータにマッピングしています。

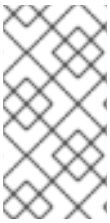
表3.1 デフォルトのディレクトリースキーマにマッピングされるデータ

データ	所有者	オブジェクトクラス	属性
従業員名	HR	person	cn (commonName)
ユーザーパスワード	IS	person	userPassword
自宅電話番号	HR	inetOrgPerson	homePhone
従業員のロケーション	IS	inetOrgPerson	localityName
オフィスの電話番号	ファシリティー	person	telephoneNumber

*表3.1「デフォルトのディレクトリースキーマにマッピングされるデータ」では、従業員名はユーザーを説明しています。デフォルトのディレクトリースキーマには、**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は、複数の属性（そのうちの 1 つは **cn** または **commonName** 属性）がユーザーのフルネームを記述することを許可します。この属性は、従業員名データを含めるのに最もマッチします。*

*ユーザーパスワードは、**person** オブジェクトクラスの要素も記述し、**userPassword** 属性は **person** オブジェクトクラスの許可される属性に一覧表示されます。*

*ホーム電話番号は個人の要素を記述しますが、**person** オブジェクトクラスに関連付けられたリストには関連する属性はありません。自宅電話番号は、組織のエンタープライズネットワークにおける person の要素を記述します。このオブジェクトは、ディレクトリースキーマの **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、**person** オブジェクトクラスから継承する **organizationPerson** 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る属性には、従業員のホーム電話番号を含めるのに適した **homePhone** 属性があります。*



注記

『Red Hat Directory Server 設定、コマンド、およびファイルリファレンス』は、データで利用可能な属性を判断するのに役に立ちます。各属性はそれを受け入れるオブジェクトクラスと共にリスト表示され、各オブジェクトクラスは必要な属性および許可される属性と共に相互リストされます。

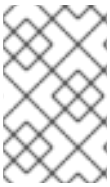
3.4. スキーマのカスタマイズ

ディレクトリーのニーズ用に標準スキーマの制限が多い場合、拡張できます。Directory Server の Web コンソールを使用して、属性とオブジェクトクラスを簡単に追加することにより、スキーマを拡張できます。LDIF ファイルを作成し、スキーマ要素を手動で追加することもできます。詳細は『Red Hat Directory Server 管理ガイド』を参照してください。

Directory Server スキーマをカスタマイズする場合は、以下のルールを念頭に置いてください。

- *スキーマはできるだけシンプルに保ちます。*
- *可能であれば、既存のスキーマ要素を再利用します。*
- *各オブジェクトクラスに定義される必須属性の数を最小限に抑える。*
- *複数のオブジェクトクラスまたは属性を同じ目的 (データ) に定義しないでください。*

- *属性またはオブジェクトクラスの既存の定義は変更しないでください。*



注記

スキーマをカスタマイズする場合、標準スキーマの削除または置き換えは **絶対に行わないでください**。*これを行うと、他のディレクトリーやその他の LDAP クライアントアプリケーションとの互換性の問題が発生する可能性があります。*

カスタムオブジェクトクラスおよび属性は ***99user.ldif*** *ファイルで定義されます。個々のインスタンスは、****/etc/dirsrv/slapd-instance_name/schema/*** *ディレクトリーに独自の* ***99user.ldif*** *ファイルを維持します。カスタムスキーマファイルを作成し、スキーマを動的にサーバーにリロードすることもできます。*

3.4.1. スキーマを拡張するケース

Directory Server により提供されるオブジェクトクラスおよび属性は、ほとんどの一般的な企業のニーズに対応しますが、特定のオブジェクトクラスは組織に関する特殊な情報を格納できない場合があります。また、スキーマは LDAP 対応アプリケーションの固有のデータニーズに必要なオブジェクトクラスおよび属性をサポートするために拡張する必要があることがあります。

3.4.2. 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) を割り当てる必要があります。スキーマが定義されている場合、要素には組織に固有のベース OID が必要です。すべてのスキーマニーズを満たすのに、1 つの OID で十分です。別の階層レベルを追加して、属性とオブジェクトクラスの新規ブランチを作成します。スキーマで OID を取得して割り当てるには、以下のステップが必要です。*

1. *Internet Assigned Numbers Authority (IANA) または国際的な組織から OID を取得します。*

   国によっては、企業に OID がすでに割り当てられています。組織に OID がない場合は、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) にアクセスしてください。

2. *OID 割り当てを追跡する OID レジストリーを作成します。*

   OID レジストリーは、OID と、ディレクトリースキーマで使用される OID の説明のリストです。これにより、OID が複数の目的に使用されないようにします。次に、スキーマに OID レジストリーをパブリッシュします。

3. *スキーマ要素に対応するために OID ツリーでブランチを作成します。*

   属性に **OID.*1*** *を、オブジェクトクラスに OID****.2*** *を使用して、***OID** *ブランチまたはディレクトリースキーマに少なくとも 2 つのブランチを作成します。カスタムマッチングルールまたは制御を定義するには、必要に応じて新規ブランチを追加します（例：***OID.*3***）。*

3.4.3. 命名属性およびオブジェクトクラス

新しい属性とオブジェクトクラスの名前を作成する場合は、名前をできるだけ意味のあるものにしてください。これにより、Directory Server の管理者はスキーマを簡単に使用できます。

すべてのスキーマ要素に一意の接頭辞を含めることで、スキーマ要素と既存のスキーマ要素間の命名の競合を回避してください。たとえば、Example Corp. は、それぞれのカスタムスキーマ要素の前に接頭辞 ***example*** *を追加します。ディレクトリー内で Example Corp. 従業員を特定するために、****examplePerson*** *という特別なオブジェクトクラスを追加する場合があります。*

3.4.4. 新規オブジェクトクラスを定義するストラテジー

新しいオブジェクトクラスを作成する方法は 2 つあります。

- 属性を追加する各オブジェクトクラス構造に対して、多くの新規オブジェクトクラスを作成します。
- ディレクトリーに作成されるすべてのカスタム属性をサポートする単一のオブジェクトクラスを作成します。この種類のオブジェクトクラスは、補助オブジェクトクラスとして定義して作成します。

2 つのメソッドを混在させることが最も簡単な場合があります。

たとえば、管理者が属性 **exampleDateOfBirth**、**examplePreferredOS**、**exampleBuildingFloor**、および **exampleVicePresident** を作成するとします。簡単なソリューションは、これらの属性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することです。

- 1 つのオブジェクトクラス **examplePerson** が作成され、**exampleDateOfBirth** および **examplePreferredOS** を許可します。**examplePerson** の親は **inetOrgPerson** です。
- 2 つ目のオブジェクトクラス **exampleOrganization** は、**exampleBuildingFloor** および **exampleVicePresident** を許可します。**exampleOrganization** の親は、**organization** オブジェクトクラスです。

新しいオブジェクトクラスは、以下のように LDAPv3 スキーマ形式で表示されます。

```
objectclasses: ( 2.16.840.1.117370.999.1.2.3 NAME 'examplePerson' DESC 'Example Person
Object Class'
    SUP inetorgPerson MAY (exampleDateOfBirth $ examplePreferredOS) )

objectclasses: ( 2.16.840.1.117370.999.1.2.4 NAME 'exampleOrganization' DESC 'Organization
Object Class'
    SUP organization MAY (exampleBuildingFloor $ exampleVicePresident) )
```

または、これらの属性をすべて許可するオブジェクトクラスを 1 つ作成し、これらの属性を必要とするエントリーと共に使用ます。単一のオブジェクトクラスは以下のようになります。

```
objectclasses: (2.16.840.1.117370.999.1.2.5 NAME 'exampleEntry' DESC 'Standard Entry Object
Class' SUP top
    AUXILIARY MAY (exampleDateOfBirth $ examplePreferredOS $ exampleBuildingFloor $
exampleVicePresident) )
```

新しい **exampleEntry** オブジェクトクラスには **AUXILIARY** というマークが付けられます。つまり、構造のオブジェクトクラスに関係なく、任意のエントリーと併用できます。

注記

この例の新しいオブジェクトクラスの OID (**2.1.117370**)は、以前の Netscape OID 接頭辞に基づいています。カスタムオブジェクトクラスを作成するには、「オブジェクト識別子の取得と割り当て」の説明に従って OID を取得します。

組織環境に応じて、新規オブジェクトクラスを整理する方法はいくつかあります。新しいオブジェクトクラスの実装方法を決定する際に、以下を考慮してください。

- 複数のオブジェクトクラスによって、作成および維持するスキーマ要素がより多くなります。

通常、要素の数が小さいため、メンテナンスはほとんど必要ありません。ただし、スキーマに 2 つ以上のオブジェクトクラスが追加されている場合は、単一のオブジェクトクラスを使用するのがより簡単です。

- *複数のオブジェクトクラスの場合には、より注意深い厳密なデータ設計が必要です。*

  厳密なデータ設計により、すべてのデータが置かれるオブジェクトクラス構造に注意を払うことが強制されます。これには、便利な面と面倒な面があります。

- *人とアセットエントリーの両方など、複数のタイプのオブジェクトクラスに適用できるデータがある場合に、単一のオブジェクトクラスはデータ設計を簡素化します。*

  *たとえば、カスタムの **preferredOS** 属性は、ユーザーエントリーとグループエントリーの両方に設定できます。単一のオブジェクトクラスは両方のタイプのエントリーでこの属性を許可できます。*

- *新規オブジェクトクラスには、必要な属性は使用しないでください。*

  *新規オブジェクトクラスの属性に **許可** ではなく **必要** を指定すると、スキーマが柔軟ではなくなります。新規オブジェクトクラスを作成する場合には、できるだけ **必要** でなく **許可** を使用します。*

  新しいオブジェクトクラスを定義したら、許可および必要な属性、ならびに属性を継承するオブジェクトクラスを決定します。

3.4.5. 新規属性を定義する際のストラテジー

アプリケーションの互換性と長期的なメンテナンスの両方のために、可能な場合は常に標準の属性を使用します。デフォルトのディレクトリースキーマにすでに存在する属性を検索し、新しいオブジェクトクラスと関連する属性を使用するか、『Directory Server Schema Guide』を確認してください。ただし、標準スキーマに必要なすべての情報が含まれていない場合は、新しい属性と新しいオブジェクトクラスを追加します。

*たとえば、**person エントリーでは、デフォルトで person、organization** openblas、または **inetOrgPerson** オブジェクトクラスがサポートするよりも多くの属性が必要になる場合があります。たとえば、標準の Directory Server スキーマには、生年月日を格納する属性はありません。新しい属性 **dateOfBirth** を作成し、新しい補助オブジェクトクラス **examplePerson** 内で許可される属性として設定できます。*

```
attributetypes: ( dateofbirth-oid NAME 'dateofbirth' DESC 'For employee birthdays'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'Example defined')

objectclasses: ( 2.16.840.1.117370.999.1.2.3 NAME 'examplePerson' DESC 'Example Person
Object Class'
    SUP inetorgPerson MAY (exampleDateOfBirth $ cn) X-ORIGIN 'Example defined')
```

*重要事項: カスタム属性を標準のスキーマ要素に追加したり、削除したり **しないでください**。ディレクトリーにカスタム属性が必要な場合は、カスタムオブジェクトクラスを追加してそれらを含めます。*

3.4.6. スキーマ要素の削除

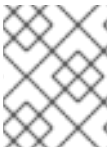
Directory Server にデフォルトで含まれるスキーマ要素を削除しないでください。未使用のスキーマ要素は、操作や管理のオーバーヘッドを表しません。標準の LDAP スキーマの一部を削除すると、Directory Server およびその他のディレクトリー対応アプリケーションの今後のインストールとの間で互換性の問題が発生する可能性があります。

ただし、未使用のカスタムスキーマ要素を削除できます。スキーマからオブジェクトクラス定義を削除する前に、オブジェクトクラスを使用して各エントリーを変更します。最初に定義を削除すると、オブジェクトクラスを使用するエントリーが後で変更されなくなる可能性があります。未知のオブジェクトクラス値がエントリーから削除されない限り、変更されたエントリーのスキーマチェックも失敗します。

3.4.7. カスタムスキーマファイルの作成

管理者は、Directory Server で提供される **99user.ldif** ファイルに加えて、Directory Server が使用するカスタムスキーマファイルを作成できます。これらのスキーマファイルは、組織に固有の新しいカスタム属性とオブジェクトクラスを保持します。新しいスキーマファイルは、スキーマディレクトリー **/etc/dirsrv/slapd-instance_name/schema/** に配置する必要があります。

標準属性とオブジェクトクラスはすべて、カスタムスキーマ要素が読み込まれた後にのみロードされます。



注記

カスタムスキーマファイルは、**99user.ldif** より数字またはアルファベット順で高くしないでください。そうしないと、サーバーで問題が発生する可能性があります。

カスタムスキーマファイルの作成後、全サーバー間でスキーマの変更を分散する方法が 2 つあります。

- このカスタムスキーマファイルをインスタンスのスキーマディレクトリー **/etc/dirsrv/slapd-instance/schema** に手動でコピーします。スキーマをロードするには、サーバーを再起動するか、**schema-reload.pl** スクリプトを実行してスキーマを動的にリロードします。
- Web コンソールまたは **ldapmodify** などの LDAP クライアントでサーバーのスキーマを変更します。
- サーバーがレプリケートされたら、レプリケーションプロセスがスキーマ情報を各コンシューマーサーバーにコピーするのを許可します。

  レプリケーションでは、レプリケートされたスキーマ要素はすべてコンシューマーサーバーの **99user.ldif** ファイルにコピーされます。**90example_schema.ldif** などのカスタムスキーマファイルにスキーマを保持するには、ファイルを手動でコンシューマーサーバーにコピーする必要があります。レプリケーションは、スキーマファイルをコピーしません。

これらのカスタムスキーマファイルがすべてのサーバーにコピーされていない場合、Web コンソールまたは **ldapmodify** などの LDAP クライアントを使用してサプライヤーサーバーのスキーマに変更が加えられた場合に限り、スキーマ情報はレプリカ（コンシューマーサーバー）に複製されます。

スキーマ定義が存在しないコンシューマーサーバーに複製されると、**99user.ldif** ファイルに保存されます。このディレクトリーは、スキーマ定義の保存場所を追跡しません。コンシューマーの **99user.ldif** ファイルにスキーマ要素を保存しても、スキーマがサプライヤーサーバーでのみ維持されている限り、問題はありません。

カスタムスキーマファイルが各サーバーにコピーされた場合、スキーマファイルへの変更を各サーバーに再度コピーする必要があります。ファイルが再びコピーされない場合、変更がコンシューマー上の **99user.ldif** ファイルに複製され保存される可能性があります。**99user.ldif** ファイルに変更を加えると、スキーマ管理が困難になる可能性があります。これは、一部の属性がコンシューマー上の 2 つの別個のスキーマファイルに表示されるためです。1 度は、サプライヤーからコピーされた元のカスタムスキーマファイルで、レプリケーション後に **99user.ldif** ファイルに再度表示されるためです。

スキーマの複製の詳細は、「スキーマレプリケーション」を参照してください。

3.4.8. カスタムスキーマのベストプラクティス

スキーマファイルを使用する場合は、互換性があり、管理しやすいスキーマを必ず作成してください。

3.4.8.1. スキーマファイルの命名

カスタムスキーマファイルの命名時には、以下の命名形式を使用します。

> [00-99]yourName.ldif

99user.ldif よりも低いもの（数字とアルファベット順）のカスタムスキーマファイルに名前を付けます。これにより、LDAP ツールと Web コンソールの両方で、Directory Server が **99user.ldif** への書き込みが可能になります。

99user.ldif ファイルには、**X-ORIGIN** 値が **'user defined'** の属性が含まれます。ただし、Directory Server は、すべての **ユーザー定義のスキーマ要素を、数値順でアルファベット順に名前の高いファイルに書き込みます。99zzz.ldif** という名前のスキーマファイルがある場合、次にスキーマが更新されるとき(LDAP コマンドラインツールまたは Web コンソールを使用)、**'user defined'** の値を持つ **X-ORIGIN** 属性はすべて **99zzz.ldif** に書き込まれます。結果は、重複情報が含まれる 2 つの LDIF ファイルであり、**99zzz.ldif** ファイルの一部の情報が消去される可能性があります。

3.4.8.2. Origin として 'user defined' の使用

'user defined' は、LDAP でスキーマが追加されるときに Directory Server によって内部で使用されるため、カスタムスキーマファイル（例： **60example.ldif**）の **X-ORIGIN** フィールドに **'user defined'** を使用し **ない** でください。カスタムスキーマファイルで、**'Example Corp. defined'** などの説明的なものを使用します。

ただし、カスタムスキーマ要素を手動で **99user.ldif** に直接追加する場合は、**'user defined'** を **X-ORIGIN** の値として使用します。別の **X-ORIGIN** 値を設定すると、サーバーはそれを上書きする可能性があります。

'user defined' の値の **X-ORIGIN** を使用すると、**99user.ldif** ファイルのスキーマ定義が Directory Server によってファイルから削除されないようにします。Directory Server は、**'user defined'** の値の **X-ORIGIN** を使用して、**99user.ldif** ファイルに存在する必要がある要素を指示するため、これらを削除しません。

以下に例を示します。

```
attributetypes: ( exampleContact-oid NAME 'exampleContact'
DESC 'Example Corporate contact'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN 'Example defined')
```

Directory Server がスキーマエントリーを読み込むと、以下のように表示されます。

```
attributetypes: ( exampleContact-oid NAME 'exampleContact'
DESC 'Example Corporate contact'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
X-ORIGIN ('Example defined' 'user defined') )
```

3.4.8.3. オブジェクトクラスの前の属性の定義

新しいスキーマ要素を追加する場合、すべての属性をオブジェクトクラスで使用できる前に定義する必要があります。属性とオブジェクトクラスを同じスキーマファイルに定義できます。

3.4.8.4. 単一ファイルでのスキーマの定義

カスタム属性またはオブジェクトクラスは、それぞれ 1 つのスキーマファイルでのみ定義する必要があります。これにより、最近作成されたスキーマを読み込む際に、サーバーが以前の定義をオーバーライドしなくなります (サーバーは最初に番号順、その後にアルファベット順でスキーマを読み込むため)。重複するファイルにスキーマを確保しない方法を決定します。

- 各スキーマファイルにどのスキーマ要素が含まれているかに注意してください。
- スキーマファイルの命名および更新には注意が必要です。スキーマ要素が LDAP ツールを使用して編集されると、変更が自動的に最後のファイル (アルファベット順) に書き込まれます。ほとんどのスキーマの変更は、デフォルトのファイル **99user.ldif** に書き込みます。これは、**60example.ldif** などのカスタムスキーマファイルではありません。また、**99user.ldif** のスキーマ要素は、他のスキーマファイルの重複要素を上書きします。
- すべてのスキーマ定義を **99user.ldif** ファイルに追加します。これは、Web コンソールからスキーマを管理している場合に便利です。

3.5. 一貫性のあるスキーマの維持

Directory Server 内で一貫したスキーマは、LDAP クライアントアプリケーションがディレクトリーエントリーを見つけるのに役立ちます。一貫性のないスキーマを使用すると、ディレクトリーツリーの情報を効率的に特定することは困難になります。

一貫性のないスキーマは、異なる属性またはフォーマットを使用して同じ情報を保存します。スキーマの一貫性を以下の方法で維持します。

- スキーマチェックを使用して、属性とオブジェクトクラスがスキーマルールに準拠していることを確認します。
- 構文検証を使用して、属性値が必要な属性構文と一致するようにします。
- 一貫性のあるデータ形式を選択して適用します。

3.5.1. スキーマチェック

スキーマチェックにより、すべての新規または変更されたディレクトリーエントリーがスキーマルールに準拠していることを確認します。ルールに違反すると、ディレクトリーは要求された変更を拒否します。

注記

スキーマチェックは、適切な属性が存在することのみを確認します。属性の値が正しい構文にあることを確認するには、「構文の検証」の説明に従って構文の検証を使用します。

デフォルトでは、ディレクトリーはスキーマチェックを有効にします。Red Hat は、この機能を無効にしないことを推奨します。スキーマチェックの有効化および無効化に関する情報は、『Red Hat Directory Server 管理ガイド』を参照してください。

スキーマチェックを有効にすると、オブジェクトクラスで定義されるように必要な属性および許可される属性に注意を払います。オブジェクトクラス定義には、通常 1 つ以上の必要な属性と 1 つまたは複数

の任意の属性が含まれます。任意の属性は、ディレクトリーエントリーに追加することが可能な属性ですが、必須ではありません。属性を、エントリーのオブジェクトクラス定義に従って必要でも許可でもないエントリー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、**organization** openblas オブジェクトクラスを使用するようにエントリーが定義されている場合は、エントリーに共通名(**cn**)および姓(**sn**)属性が必要です。つまり、エントリーの作成時にこれらの属性の値を設定する必要があります。さらに、**telephoneNumber**、**uid**、**streetAddress**、および **userPassword** などの説明的な属性など、エントリーでオプションで使用できる属性のリストがあります。

3.5.2. 構文の検証

構文の検証 とは、Directory Server が属性の値が、その属性に必要な構文と一致することを確認することを意味します。たとえば、構文の検証では、新しい **telephoneNumber** 属性に、実際にその値に有効な電話番号が指定されていることを確認します。

3.5.2.1. 構文の検証の概要

デフォルトでは、構文の検証が有効になっています。これは最も基本的な構文検証です。スキーマチェックと同様に、ディレクトリーの変更を検証し、構文ルールに違反する変更を拒否します。オプションとして追加の設定を行い、構文検証により構文違反に関する警告メッセージをログに記録し、変更を拒否したり、変更プロセスを正常に実行できるようにしたりすることもできます。

構文検証は、新規属性が追加されるか、属性値が変更されたために新しい属性値が追加される LDAP 操作をチェックします。構文の検証は、既存の属性やレプリケーションなどのデータベース操作で追加された属性を処理しません。既存の属性は、特別なスクリプト **syntax-validate.pl** を使用して検証できます。

この機能は、バイナリー構文 (検証できない) および標準以外の構文 (定義された必要な形式がない) を除き、すべての属性構文を検証します。構文は RFC 4514 に対して検証されますが、DN は厳格ではない RFC 1779 または RFC 2253 に対して検証されます(厳格な DN 検証を設定することもできます)。

3.5.2.2. 構文の検証およびその他の Directory Server 操作

構文の検証は、エントリーの作成 (add) や属性の編集 (modify) などの標準の LDAP 操作に主に関係します。ただし、属性の構文の検証は他の Directory Server 操作に影響を及ぼす可能性があります。

データベース暗号化

通常の LDAP 操作では、値がデータベースに書き込まれる直前に属性は暗号化されます。これは、属性構文の検証 **後に** 暗号化が実行されることを意味します。

暗号化されたデータベース (「データベースの暗号化」で説明) をエクスポートおよびインポートすることができます。通常、これらのエクスポート操作およびインポート操作は **db2ldif** および **ldif2db** とともに **-E** フラグを使用して行うことが強く推奨されます。これにより、インポート操作で構文の検証が問題になる可能性もあります。ただし、**-E** フラグを使用せずに暗号化されたデータベースをエクスポートする場合は (サポートされていない)、暗号化された値で LDIF が作成されます。この LDIF をインポートすると、暗号化された属性を検証できず、警告がログに記録され、インポートされたエントリーで属性検証はスキップされます。

同期

Windows Active Directory エントリーと Red Hat Directory Server エントリーでは、属性の許容構文または強制構文に違いがある場合があります。この場合、構文の検証により Directory Server エントリーの RFC 標準が強制されるため、Active Directory の値を適切に同期できませんでした。

レプリケーション

Directory Server 11.0 インスタンスがコンシューマーに変更を複製するサプライヤーである場合は、構文検証の使用に問題はありません。ただし、レプリケーションのサプライヤーが古いバージョンの Directory Server であったり、構文の検証が無効になっていたりする場合は、Directory Server 11.0 コンシューマーは、サプライヤーが許可する属性値を拒否する可能性があるため、構文の検証を 11.0 コンシューマーで使用しないでください。

3.5.3. 一貫性のあるデータフォーマットの選択

LDAP スキーマを使用すると、任意の属性値に任意のデータを配置できます。ただし、LDAP クライアントアプリケーションおよびディレクトリーユーザーに適した形式を選択して、一貫性を維持してディレクトリーツリーでデータを保存することが重要になります。

LDAP プロトコルおよびDirectory Server では、データはRFC 2252 で指定されたデータ形式で示される必要があります。たとえば、電話番号に関する正しいLDAP 形式は、2 つのITU-T 推奨ドキュメントで定義されます。

- ITU-T Recommendation E.123:国内およびおよび国際電話番号の記法
- ITU-T Recommendation E.163:国際的な電話サービスの番号付けプランたとえば、米国の電話番号は **+1 555 222 1717** としてフォーマットされます。

別の例として、**postalAddress** 属性は、ドル記号($) を行区切り文字として使用する複数行の文字列の形式の属性値を想定します。適切にフォーマットされたディレクトリーエントリーは以下のように表示されます。

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

属性には、文字列、バイナリー入力、整数その他の形式が必要になる場合があります。使用できる形式は、属性のスキーマ定義に設定されます。

3.5.4. 複製されたスキーマでの一貫性の維持

ディレクトリースキーマを編集すると、変更は変更ログに記録されます。レプリケーション中、変更ログをスキャンして変更の有無を確認し、変更はすべて複製されます。複製されたスキーマで一貫性を維持することで、レプリケーションがスムーズに維持されます。複製された環境で一貫したスキーマを維持するために、以下のポイントを考慮してください。

- 読み取り専用レプリカでスキーマを変更しないでください。

  読み取り専用レプリカでスキーマを変更すると、スキーマで不整合が生じ、レプリケーションが失敗します。

- 異なる構文を使用する同じ名前の属性を 2 つ作成しないでください。

  読み取り/書き込みレプリカで作成された属性がサプライヤーレプリカ上の属性と同じ名前を持ち、サプライヤー上の属性とは異なる構文を持つ場合、レプリケーションは失敗します。

3.6. その他のスキーマリソース

標準のLDAPv3 スキーマの詳細は、以下のリンクを参照してください。

- RFC 2251: Lightweight Directory Access Protocol (v3) (http://www.ietf.org/rfc/rfc2251.txt)

- *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)*
- *Internet Engineering Task Force (IETF) (http://www.ietf.org/)*
- 『Understanding and Deploying LDAP Directory Services』 *T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999.*

第4章 ディレクトリーツリーの設計

ディレクトリーツリー により、ディレクトリーサービスにより保存されたデータを確認することができます。ディレクトリーに保存されている情報の種類、企業の物理的な特性、ディレクトリーで使用するアプリケーション、および実装されるレプリケーションのタイプにより、ディレクトリーツリーの設計が形作られます。

本章では、ディレクトリーツリーを設計するための手順を説明します。

4.1. ディレクトリーツリーの概要

ディレクトリーツリーを使用することで、ディレクトリーデータはクライアントアプリケーションによって名前が付けられ、参照されます。ディレクトリーツリーは、ディレクトリーのデータの分散、複製、またはアクセス制御で利用可能な選択肢など、他のデザインの意思決定と密接に対話します。ディレクトリーツリーをデプロイする前に、時間をかけて適切に設計します。適切に設計されたディレクトリーツリーは、デプロイメントフェーズ中とディレクトリーサービスの実運用中に、かなりの時間を節約できます。

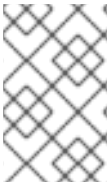
適切に設計されたディレクトリーツリーは、以下を提供します。

- ディレクトリーデータのメンテナンスの簡素化
- レプリケーションポリシーおよびアクセス制御の作成の柔軟性
- ディレクトリーサービスを使用するアプリケーションのサポート
- ディレクトリーユーザーの簡素化されたディレクトリーナビゲーション

ディレクトリーツリーの構造は、階層 LDAP モデルに従います。ディレクトリーツリーは、グループ、人、または場所など、さまざまな論理方法でデータを整理する方法を提供します。また、複数のサーバー間でのデータのパーティション設定方法も決定します。たとえば、各データベースは接尾辞レベルでデータをパーティション化する必要があります。適切なディレクトリーツリー構造がないと、データを複数のサーバーに効率的に分散できない場合があります。

さらに、レプリケーションは、使用されるディレクトリーツリー構造の種類によって制限されます。レプリケーションを機能させるため、パーティションを慎重に定義します。ディレクトリーツリーの一部のみを複製するには、設計プロセス中にそのことを考慮してください。

分岐点でアクセス制御を使用する場合にも、ディレクトリーツリー設計でそのことを検討してください。



注記

Directory Server は、仮想ディレクトリー情報ツリービューと呼ばれる、ディレクトリー情報の階層ナビゲーションと組織の概念をサポートします。ディレクトリーツリーを設計する前に、「仮想ディレクトリー情報ツリービュー」を参照してください。

4.2. ディレクトリーツリーの設計

ディレクトリーツリー設計には、いくつかの主要な決定があります。

- データを含めるための接尾辞を選択する。
- データエントリー間での階層関係を決定する。

- ディレクトリーツリー階層のエントリーに名前を付ける。

4.2.1. 接尾辞の選択

接尾辞は、ディレクトリーツリーの root にあるエントリーの名前で、ディレクトリーデータはその下に保存されます。ディレクトリーには、複数の接尾辞を含めることができます。自然共通の root を持たない情報のディレクトリーツリーが 2 つ以上ある場合には、複数の接尾辞を使用できます。

デフォルトでは、標準の Directory Server のデプロイメントには複数の接尾辞が含まれています。1 つはデータの保管用で、残りは内部ディレクトリー操作に必要なデータ (例: 設定情報、ディレクトリースキーマなど) 用です。これらの標準ディレクトリー接尾辞の詳細は、『Red Hat Directory Server 管理ガイド』を参照してください。

4.2.1.1. 接尾辞の命名規則

ディレクトリーのすべてのエントリーは、共通のベースエントリー (**root 接尾辞**) 下に置く必要があります。root ディレクトリー接尾辞の名前を選択する際には、名前を効率的にするために以下の 4 つのポイントを検討してください。

- グローバルに一意であること
- 静的であること、したがって、ほとんど変更されないこと。
- ルートディレクトリーの下にあるエントリーが画面で読みやすいように、短いこと
- 入力と覚えておくのが簡単なこと

単一のエンタープライズ環境では、企業の DNS 名またはインターネットドメイン名に対応するディレクトリー接尾辞を選択します。たとえば、エンタープライズが **example.com** のドメイン名を所有する場合、ディレクトリーの接尾辞は論理的に **dc=example,dc=com** になります。

dc 属性は、ドメイン名をコンポーネント部分に分割することで接尾辞を表します。

通常、任意の属性を使用して root 接尾辞に名前を付けることができます。ただし、ホスト組織では、root 接尾辞を以下の属性に制限します。

- **dc** ドメイン名のコンポーネントを定義します。
- **c** ISO で定義されている国名を表す 2 桁のコードが含まれます。
- **l** エントリーが配置されている、またはエントリーに関連付けられている国、都市、またはその他の地理的エリアを識別します。
- **st** エントリーが存在する州または地区を識別します。
- **o** エントリーが属する組織の名前を識別します。

これらの属性が存在すると、サブスクライバーアプリケーションとの相互運用が可能になります。たとえば、ホスト組織はこれらの属性を使用して、そのいずれかのクライアント **example_a** に root 接尾辞を作成します（例： **o=example_a, st=Washington,c=US** ）。

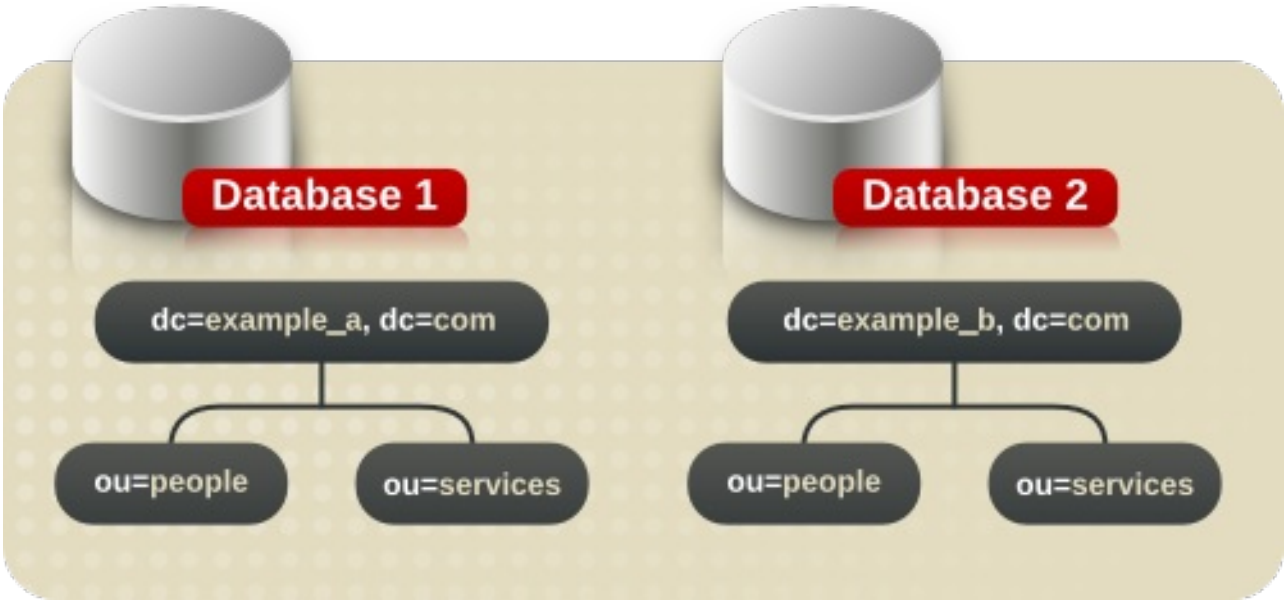
組織名の後に国の指定を続けて使用することは、接尾辞の X.500 命名規則の典型です。

4.2.1.2. 複数接尾辞の命名

ディレクトリーと一緒に使用される各接尾辞は、一意のディレクトリーツリーです。ディレクトリーサービスに複数のツリーを含む方法は複数あります。1 つ目は、Directory Server が提供する個別のデータベースに保存される複数のディレクトリーツリーを作成することです。

たとえば、**example_a** と **example_b** の個別の接尾辞を作成し、それらを別のデータベースに保存します。

図4.1 データベースへの複数のディレクトリーツリーの追加



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

4.2.2. ディレクトリーツリー構造の作成

フラットなツリー構造または階層ツリー構造を使用するかどうかを決定します。一般的なルールとして、ディレクトリーツリーを可能な限りフラットとして作成してみてください。ただし、情報を複数のデータベース間でパーティション化する場合、レプリケーションを準備する場合、またはアクセス制御を設定する場合に、ある程度の階層化が重要になります。

ツリーの構造では、以下の手順および考慮事項が必要です。

- 「ディレクトリーの分岐」
- 「分岐点の特定」
- 「レプリケーションに関する考慮事項」
- 「アクセス制御に関する考慮事項」

4.2.2.1. ディレクトリーの分岐

問題のある名前の変更を避けるように、階層を設計します。名前空間がフラットであるほど、名前を変更する可能性は低くなります。名前変更の可能性は、変更する可能性のある名前のコンポーネントの数に概ね比例します。ディレクトリーツリーが階層的であるほど、名前のコンポーネントが多くなり、名前が変更される可能性が高くなります。

ディレクトリーツリー階層を設計するためのガイドラインを以下に示します。

- 企業内の最大下部組織を表すツリーだけを分岐します。

このような分岐点は、法務サービス、カスタマーサポート、セールス、エンジニアリングなどの部門に制限する必要があります。ディレクトリーツリーの分岐に使用される部門を一定にいます。企業が頻繁に再編成する場合は、このような分岐は実行しないでください。

- *分岐点には、実際の組織名ではなく、機能または汎用名を使用します。*

  名前の変更サブツリーの名前を変更できますが、多くの子エントリーを持つ大きな接尾辞では、長くリソースを必要とするプロセスになります。組織の機能を表す汎用名（例： ***Widget Research and Development****の代わりに* ***Engineering*** *を使用）を使用すると、組織またはプロジェクトの変更後にサブツリーの名前を変更する必要がはるかに低くなります。*

- *同様の機能を実行する組織が複数になる場合は、部門をベースに分岐するのではなく、その機能に対して分岐点を1つ作成してみてください。*

  たとえば、特定の製品ラインを担当するマーケティング組織が複数存在する場合でも、 ***ou=Marketing*** *サブツリーを1つ作成します。すべてのマーケティングエントリーはそのツリーに属します。*

エンタープライズ環境における分岐

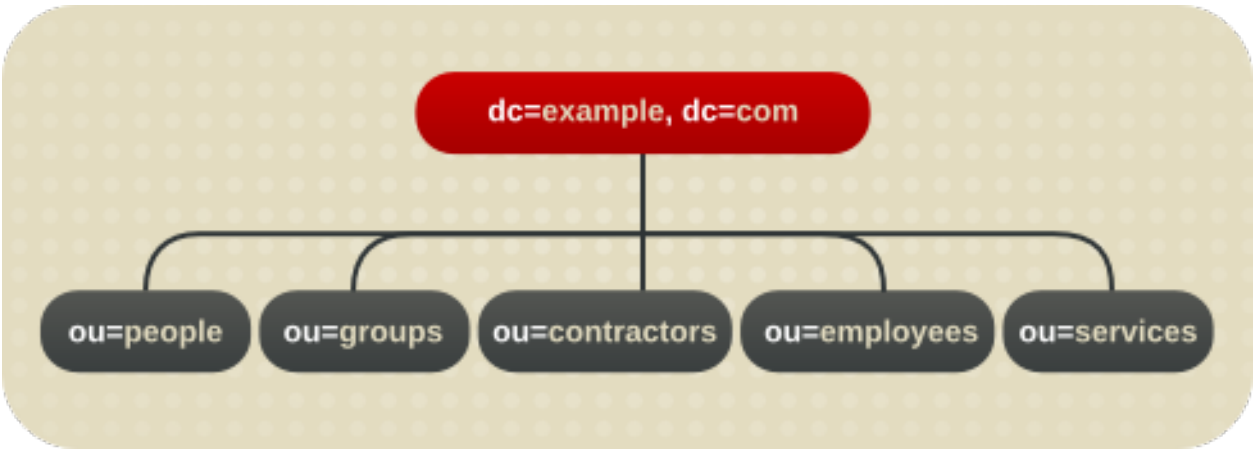
ディレクトリーツリー構造が変更されない情報に基づいている場合、名前の変更は回避できます。たとえば、組織ではなくツリー内のオブジェクトのタイプを構造のベースとします。これは、組織ユニット間でエントリーをシャッフルするのを回避するのに役立ちます。シャッフルする場合、コストのかかる操作である識別名 (DN) の変更が必要になります。

構造の定義するのに使用すると良い、便利な共通のオブジェクトがあります。

- ***ou=people***
- ***ou=groups***
- ***ou=services***

このオブジェクトを使用して整理されたディレクトリーツリーは、以下のように表示されます。

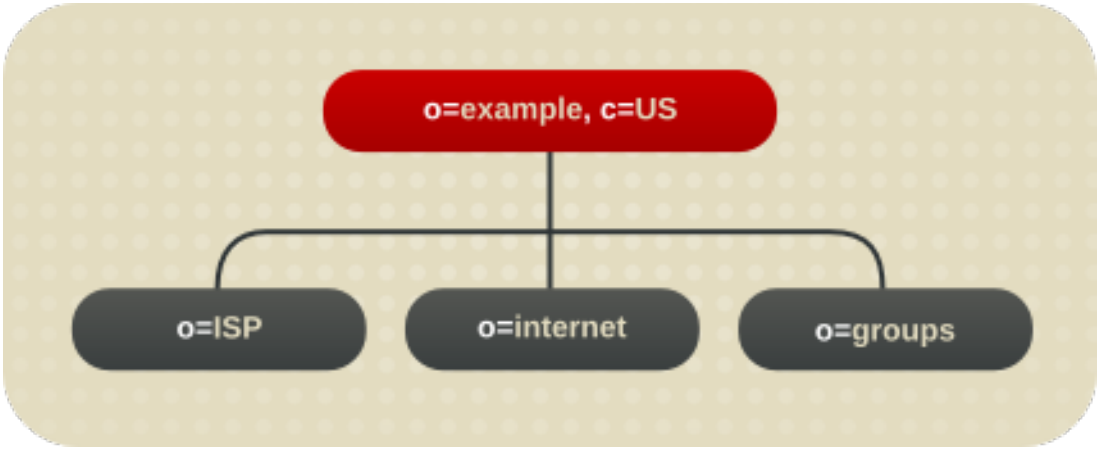
図4.2 環境ディレクトリーツリーの例



ホスト環境でのブランチング

ホスト環境の場合は、ルート接尾辞の下に、オブジェクトクラス ***organization o****の2つのエントリーとオブジェクトクラスの1つのエントリーを含むツリーを作成します。****organizationalUnit ou****たとえば、Example ISP は、ディレクトリーを以下のように分岐します。*

図4.3 ホストディレクトリーツリーの例



4.2.2.2. 分岐点の特定

ディレクトリーツリー内のブランチをプランニングする際に、分岐点の特定に使用する属性を決定します。DN は属性/データのペアで設定される一意の文字列です。たとえば、Example Corp. の従業員である Barbara Jensen のエントリーの DN は ***uid=bjensen,ou=people,dc=example,dc=com*** *です。*

各属性とデータのペアは、エンタープライズ Example Corp のディレクトリーツリーの 図4.4「Example Corp のディレクトリーツリー」 に示すように、ディレクトリーツリー内のブランチポイントを表します。

図4.4 Example Corp のディレクトリーツリー

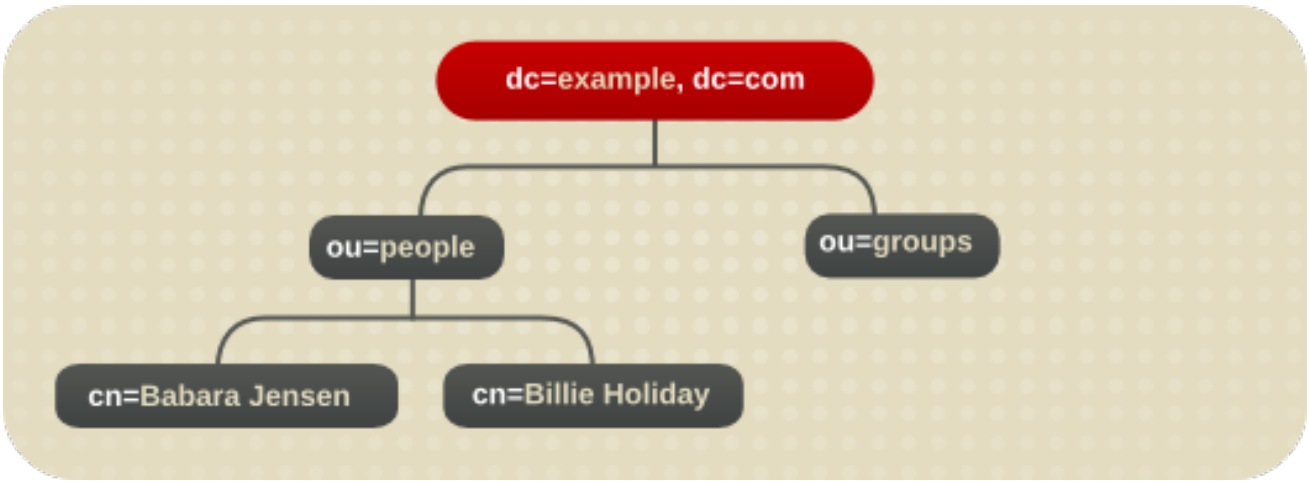
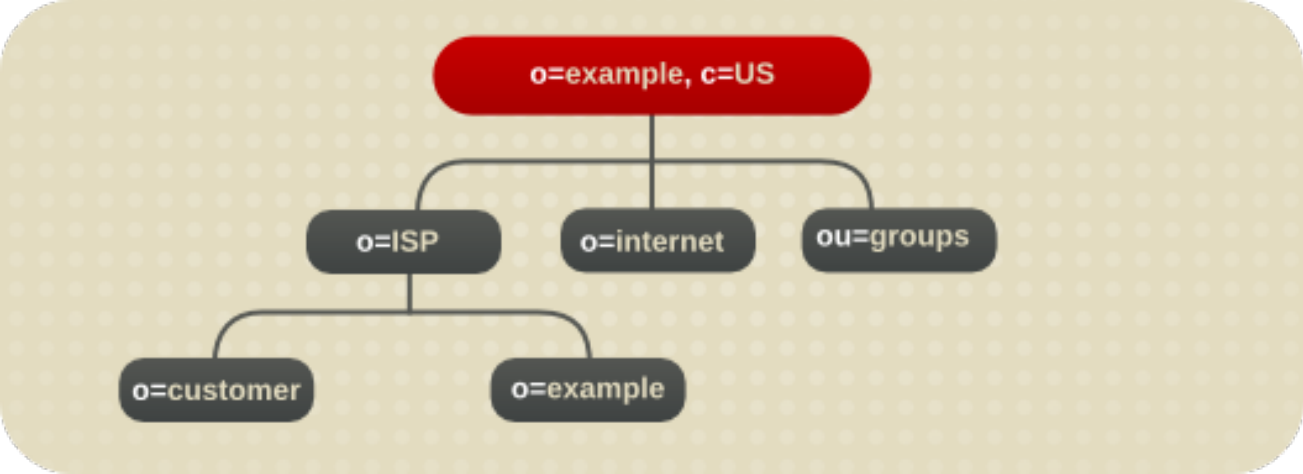


図4.5「Example ISP のディレクトリーツリー」 インターネットホストの Example ISP のディレクトリーツリーを表示します。

図4.5 Example ISP のディレクトリーツリー



*接尾辞エントリー **c=US,o=example** の下には、ツリーは 3 つのブランチに分割されます。ISP ブランチには、Example ISP の顧客データおよび内部情報が含まれます。インターネットブランチは、ドメインツリーです。groups ブランチには、管理グループに関する情報が含まれます。*

分岐点の属性を選択する場合は、以下を考慮してください。

- *一貫性を持たせる必要があります。*

  *ディレクトリーツリー全体で識別名 (DN) 形式に一貫性がない場合、一部の LDAP クライアントアプリケーションは混同する可能性があります。つまり、ディレクトリーツリーの 1 つで **l** が **ou** に従属する場合、ディレクトリーサービスの他の部分で **l** が **ou** に従属するようにします。*

- *従来の属性 (「分岐点の特定」に示す) のみの使用を試みます。*

  従来の属性を使用すると、サードパーティーの LDAP クライアントアプリケーションとの互換性を維持する可能性が高まります。従来の属性を使用すれば、デフォルトのディレクトリースキーマに認識されるため、ブランチ DN のエントリーの構築が容易になります。

表4.1 従来の DN 分岐点属性

属性	定義
dc	***dc=example*** *などのドメイン名の要素。これは、dc=example,dc=com や dc=* ***mtv****,* ***dc=example,dc=com*** *など、ドメインに応じてペアまたはそれ以上で指定されます。*
c	*国名。*
o	*組織名。この属性は、通常、「接尾辞の命名規則」のように、企業の部門、学問 (人間、サイエンス)、子会社、または企業内のその他の主要なブランチなど、大規模な部門の分岐を表すために使用されます。*
ou	*組織単位。この属性は、通常、組織よりも小さな企業部門のブランチを表すために使用されます。通常、組織単位は前述の組織に従属します。*
st	*州または地区名。*

属性	定義
l または、以下を実行します。 **locality**	都市、国、オフィス、またはファシリティー名などのローカリティー。

注記

一般的な間違いは、識別名で使用される属性に基づいてディレクトリーを検索することを仮定することです。識別名はディレクトリーエントリーの一意の識別子であり、検索キーとして使用できません。代わりに、エントリー自体に保存されている属性とデータのペアに基づいてエントリーを検索します。したがって、エントリーの識別名が **uid=bjensen,ou=People,dc=example,dc=com** の場合、**dc=example** の検索は、そのエントリーの属性として **dc:example** が明示的に追加されない限り、そのエントリーと一致しません。

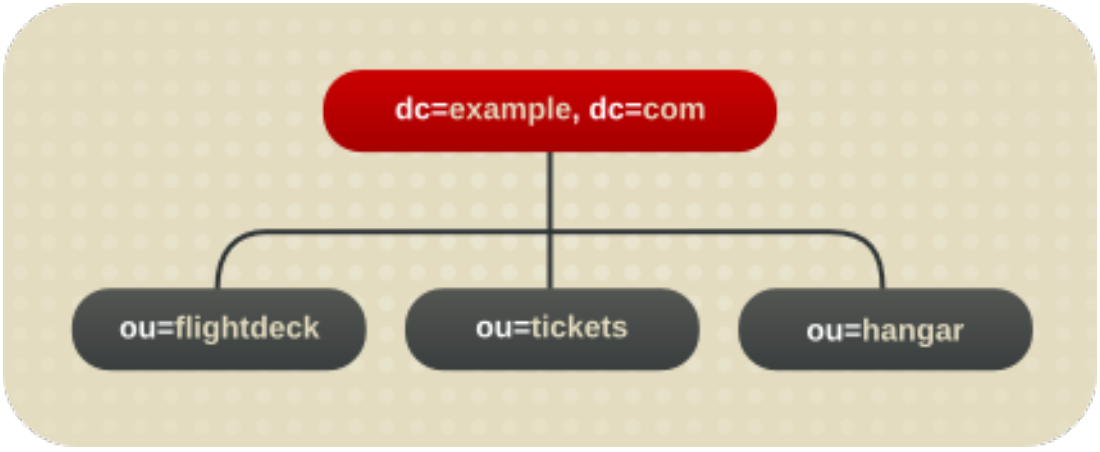
4.2.2.3. レプリケーションに関する考慮事項

ディレクトリーツリーの設計プロセスで、どのエントリーがレプリケートされるかを考慮してください。レプリケートするエントリーセットを記述する自然な方法は、サブツリーの上部に DN を指定し、その下のすべてのエントリーを複製することです。このサブツリーはデータベース (ディレクトリーデータの一部が含まれるディレクトリーパーティション) にも対応しています。

たとえば、エンタープライズ環境では、1 つの方法は、企業のネットワーク名に対応するようにディレクトリーツリーを整理することです。ネットワーク名は変更されないため、ディレクトリーツリー構造は安定します。さらに、ネットワーク名を使用してディレクトリーツリーのトップレベルブランチを作成するのは、レプリケーションを使用して異なる Directory Server を連携させる場合に役立ちます。

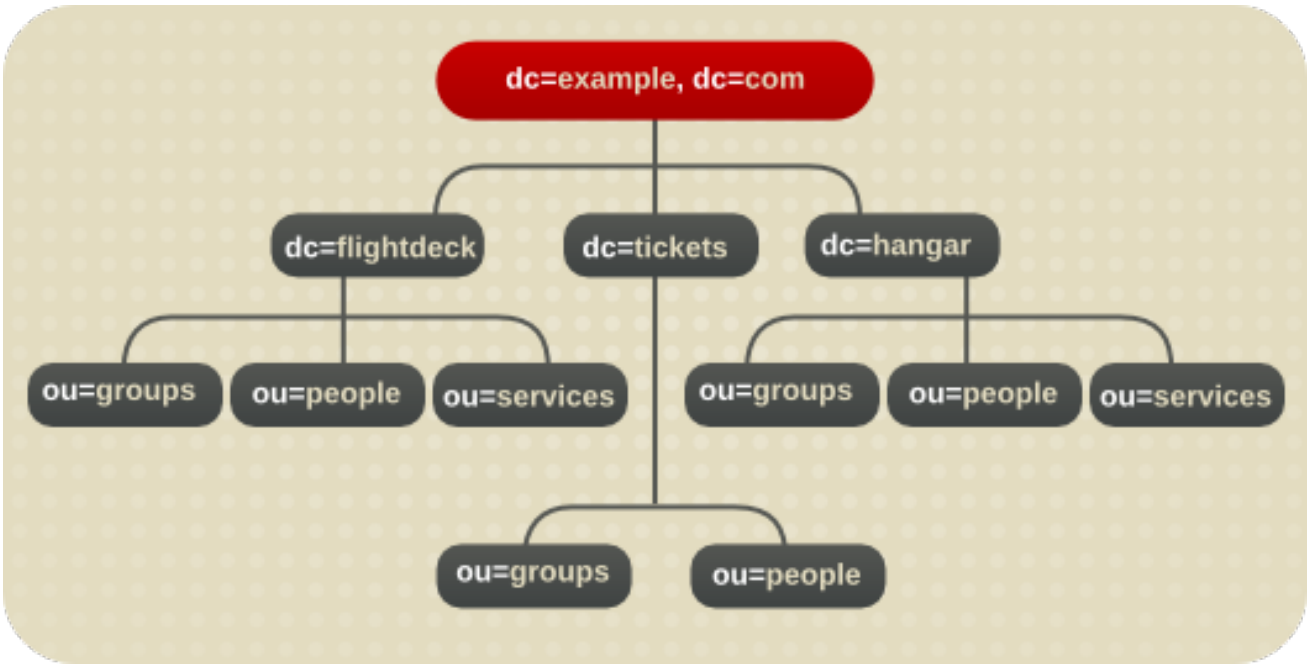
たとえば、Example Corp. には **flightdeck.example.com**、**tickets.example.com**、および **hangar.example.com** として知られる 3 つの主要ネットワークがあります。主要な組織部門用に、当初はディレクトリーツリーを 3 つのメイングループに分岐します。

図4.6 Example Corp. のディレクトリーツリーの初期分岐



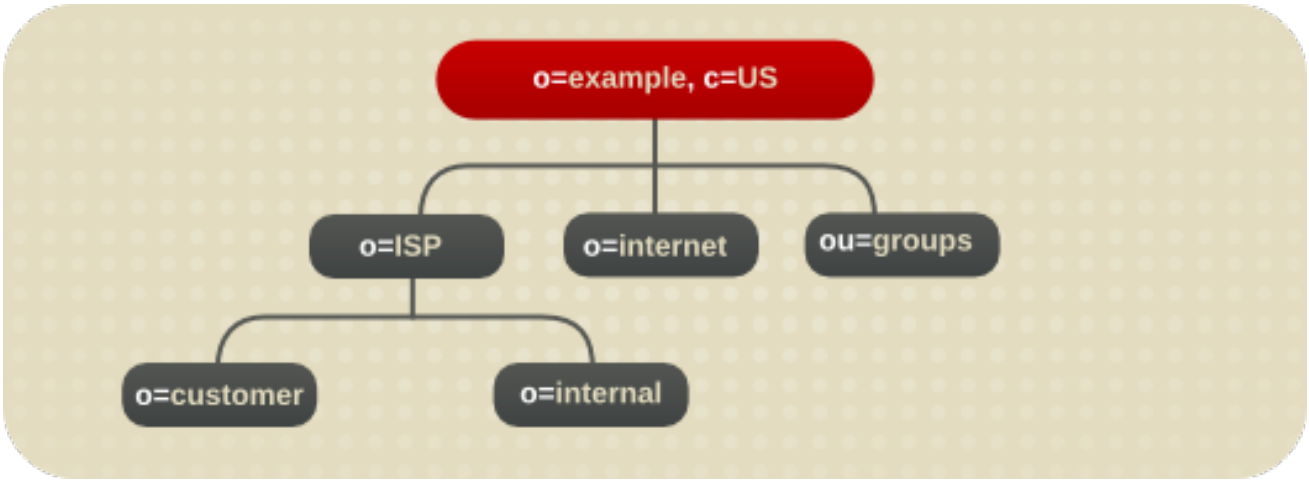
ツリーの初期構造を作成した後、各組織グループの詳細を表示する追加のブランチを作成します。

図4.7 Example Corp. の拡張ブランチ



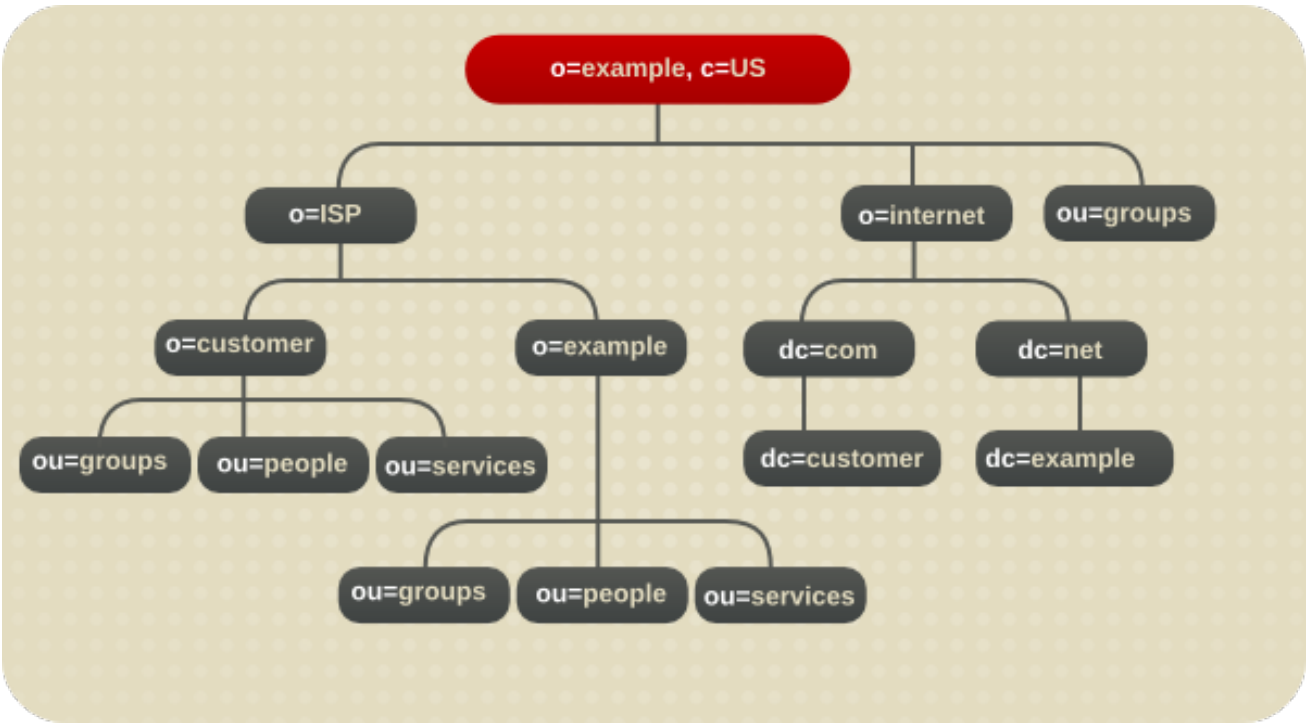
Example ISP は、組織をミラーリングする非対称ツリーのディレクトリーを分岐します。

図4.8 Example ISP のディレクトリーの分岐



ディレクトリーツリーの初期構造を作成したら、論理サブグループ用に追加のブランチが作成されます。

図4.9 Example ISP の拡張ブランチ



エンタープライズとホスト組織の両方が、頻繁に変更されない情報に基づいて、データ階層をデザインします。

4.2.2.4. アクセス制御に関する考慮事項

ディレクトリーツリーに階層を導入すると、特定タイプのアクセス制御が可能になります。レプリケーションと同様に、類似したエントリーをグループ化して、1 つのブランチから管理するのが簡単になります。

階層ディレクトリーツリーを通じて、管理の分配を有効にすることもできます。たとえば、マーケティング部門の管理者にマーケティングエントリーへのアクセス権を付与し、営業部門の管理者に営業のエントリーへのアクセス権を付与するには、これらの部門に従ってディレクトリーツリーを設計します。

*アクセス制御は、ディレクトリーツリーではなくディレクトリーコンテンツに基づいている可能性があります。フィルターされたメカニズムは、ディレクトリーエントリーが特定の属性値を含むすべてのエントリーにアクセスできることを示す単一のアクセス制御ルールを定義できます。たとえば、営業管理者に属性値 **ou=Sales** を含むすべてのエントリーへのアクセス権を付与する ACI フィルターを設定します。*

ただし、ACI フィルターは管理が困難な場合があります。ディレクトリーツリー階層組織分岐、ACI フィルター、またはこの組み合わせなど、ディレクトリーに最も適しているアクセス制御の方法を決定します。

4.2.3. エントリーの命名

ディレクトリーツリーの階層を設計したら、構造内のエントリーに名前を付ける時に使用する属性を決定します。通常、1 つ以上の属性値を選択して **相対識別名 (RDN)** *を形成することで、名前は作成されます。RDN は DN 内の単一のコンポーネントです。これは最初に表示されるコンポーネントであるため、そのコンポーネントに使用される属性は、エントリーに一意の名前を設定するため、* **命名属性** *になります。使用する属性は、名前が付けられるエントリーのタイプによって異なります。*

エントリー名は、以下のルールに従う必要があります。

- 命名用に選択される属性は、変更されてないけません。
- この名前は、ディレクトリー全体で一意でなければなりません。

  一意の名前により、DN は最大でも 1 つのディレクトリー内のエントリーを確認できるようになります。

エントリーの作成時に、エントリー内で RDN を定義します。エントリー内で最小限の RDN を定義することで、エントリーを簡単に見つけることができます。これは、検索は実際の DN に対しては実行されませんが、エントリー自体に保存されている属性値に対して実行されるためです。

属性名には意味があります。したがって、表現するエントリーのタイプとマッチする属性名を使用します。たとえば、組織を表すのに **l** を使用しないでください。また、組織単位を表すのに **c** を使用しないでください。

- 「人物エントリーの命名」
- 「グループエントリーの命名」
- 「組織エントリーの命名」
- 「その他の種類のエントリーの命名」

4.2.3.1. 人物エントリーの命名

人物エントリーの名前、DN は一意である必要があります。従来、識別名は **commonName** または **cn** 属性を使用して人物エントリーに名前を付けます。つまり、Babs Jensen という名前のユーザーのエントリーは、**cn=Babs Jensen,dc=example,dc=com** の識別名になります。

共通名を使用すると、エントリーと人物の関連付けが容易になりますが、同じ名前のユーザーを除外するほど十分に一意ではない可能性があります。これにより、直ぐに **DN 名の競合** と呼ばれる、同じ識別名を持つ複数のエントリーが存在する問題が発生します。

cn=Babs Jensen+employeeNumber=23,dc=example,dc=com など、一意識別子を共通名に追加して、一般的な名前の競合を回避します。

ただし、これにより、大規模なディレクトリーでは共通名が不便になり、管理が困難になる可能性があります。

より良い方法は、**cn** 以外の属性を持つ個人のエントリーを特定することです。以下の属性のいずれかを使用することを検討してください。

- **uid**

  uid 属性を使用して、個人の一意の値を指定します。可能性としては、ユーザーログイン ID または従業員番号が含まれます。ホスト環境のサブスクライバーは、**uid** 属性で識別する必要があります。

- **mail**

  mail 属性には、常に一意のユーザーのメールアドレスが含まれます。このオプションを使用すると、重複した属性値（例： **mail=bjensen@example.com,dc=example,dc=com**）を含む awkward DN が発生する可能性があるため、**uid** 属性で使用する他の一意の値がない場合に限りこのオプションを使用します。たとえば、企業が一時的または契約社員に従業員番号やユーザー ID を割り当てない場合は、**mail** 属性の代わりに **uid** 属性を使用します。

- **employeeNumber**

inetOrgPerson オブジェクトクラスの従業員の場合は、**employeeNumber** などの勤務担当者が割り当てた属性値を使用することを検討してください。

人物エントリー RDN に使用される属性/データのペアに関係なく、それらが一意の永続的値であることを確認してください。人物エントリー RDN も読み取り可能でなければなりません。たとえば、**uid=bjensen,dc=example,dc=com** は、**uid=b12r56A,dc=example,dc=com** よりも推奨されます。これは、認識可能な DN が、識別名に基づいてディレクトリーエントリーを変更するなど、一部のディレクトリータスクを簡素化するためです。また、一部のディレクトリークライアントアプリケーションは、**uid** および **cn** 属性に人間が判読できる名前が使用されることを想定しています。

ホストされる環境の人物エントリーに関する考慮事項

人物がサービスへのサブスクライバーである場合、エントリーはオブジェクトクラス **inetUser** になり、エントリーに **uid** 属性が含まれている必要があります。属性は、顧客サブツリー内で一意である必要があります。

人物がホスト組織の一部である場合は、**nsManagedPerson** オブジェクトクラスを持つ **inetOrgPerson** として表現します。

DIT への人物エントリーの配置

以下は、ディレクトリーツリーに人物エントリーを配置するためのガイドラインです。

- エンタープライズ内のユーザーは、組織のエントリー下のディレクトリーツリーに置く必要があります。
- ホスティング組織をお持ちのお客様は、ホストされた組織の **ou=people** ブランチの下になければなりません。

4.2.3.2. グループエントリーの命名

グループを表す 4 つの方法があります。

- **静的グループ** は、明示的にメンバーを定義します。**groupOfNames** または **groupOfUniqueNames** オブジェクトクラスには、グループのメンバーに名前を付ける値が含まれます。静的グループは、ディレクトリー管理者のグループなど、メンバーの少ないグループに適しています。静的グループは、数千のメンバーを持つグループには適していません。

  uniqueMember は **groupOfUniqueNames** オブジェクトの必須の属性であるため、静的グループエントリーには **uniqueMember** 属性値が含まれている必要があります。このオブジェクトクラスは、グループエントリーの DN を形成するために使用できる **cn** 属性を必要とします。

- **動的グループ** は、検索フィルターとサブツリーを含むグループを表すエントリーを使用します。フィルターにマッチするエントリーはグループのメンバーです。

- **ロール** は、静的グループおよび動的グループの概念を統一します。詳細は、「ディレクトリーエントリーのグループ化」を参照してください。

ホストされている組織を含むデプロイメントでは、**groupOfUniqueNames** オブジェクトクラスを使用して、ディレクトリー管理で使用されるグループのメンバーに名前を付ける値を含めることを検討してください。ホストされる組織では、ディレクトリー管理に使用されるグループエントリーを **ou=Groups** ブランチの下に配置することが推奨されます。

4.2.3.3. 組織エントリーの命名

他のエントリー名と同様に、組織エントリー名は一意である必要があります。他の属性値と共に組織の有効な名前を使用すると、名前が一意になるようにすることができます（例： ***o=example_a+st=Washington,o=ISP,c=US*** *）。*

商標も使用できますが、一意である保証はありません。

ホスト環境では、 ***organization*** *(****o****)属性を命名属性として使用します。*

4.2.3.4. その他の種類のエントリーの命名

このディレクトリーには、地域、州、国、デバイス、サーバー、ネットワーク情報など、その他の種類のデータ等を表すエントリーが含まれます。

このようなエントリータイプでは、可能であればRDN で ***cn*** *属性を使用します。次に、グループエントリーに名前を付けるために、* ***cn=administrators,dc=example,dc=com*** *のように名前を付けます。*

ただし、エントリーのオブジェクトクラスが ***commonName*** *属性をサポートしない場合があります。代わりに、エントリーのオブジェクトクラスがサポートする属性を使用します。*

エントリーのDN に使用される属性は、エントリーで実際に使用されている属性に対応する必要はありません。ただし、DN 属性とエントリーで使用される属性の間で相関がある場合、ディレクトリーツリーの管理が簡素化されます。

4.2.4. エントリーおよびサブツリーの名前変更

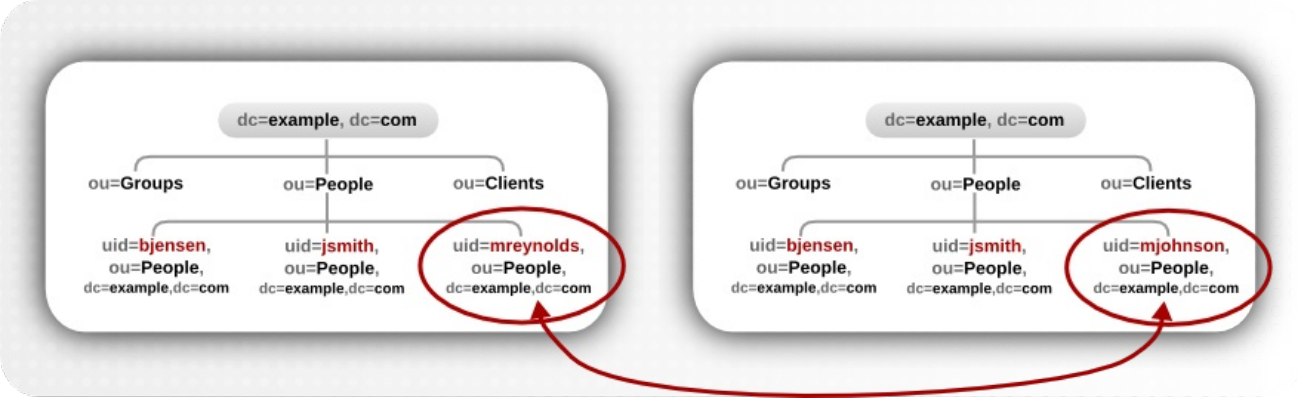
「エントリーの命名」では、Red Hat Directory Server でエントリーの命名が重要であることについて説明します。エントリー名は、ある意味ディレクトリーツリーの構造を定義します。各分岐点 (その下にエントリーがあるエントリー) は、階層に新しいリンクを作成します。

例4.1 エントリーDN のビルド

```
dc=example,dc=com  =>  root suffix
ou=People,dc=example,dc=com  =>  org unit
st=California,ou=People,dc=example,dc=com  =>  state/province
l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  city
ou=Engineering,l=Mountain View,st=California,ou=People,dc=example,dc=com  =>  org
unit
uid=jsmith,ou=Engineering,l=Mountain View,st=California,ou=People,dc=example,dc=com  =>
leaf entry
```

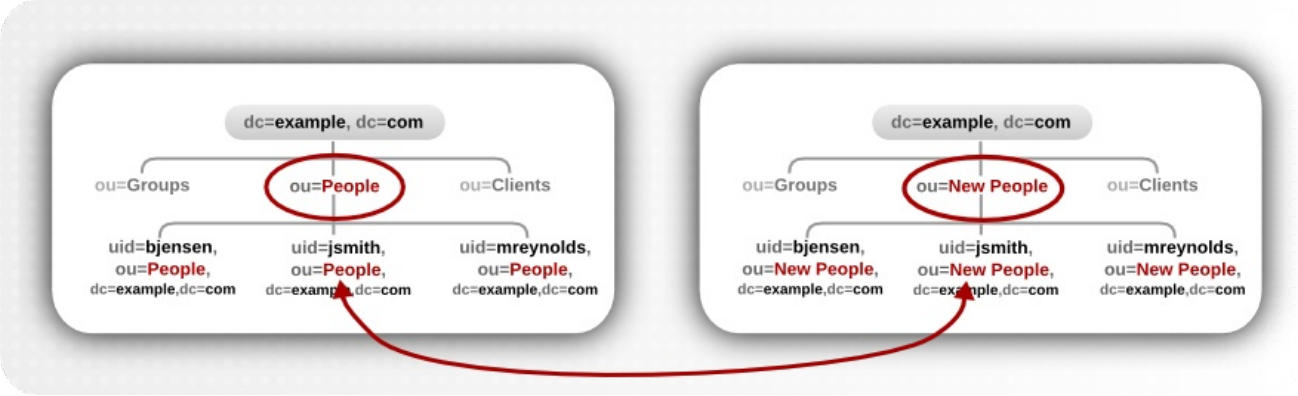
エントリーの命名属性 (DN の左端にある要素) が変更されると、これは **modrdn 操作** *になります。ディレクトリーツリー内のエントリーを移動するため、ある意味特別な変更操作です。リーフエントリー (子を持たないエントリー) の場合、modrdn 操作は横方向の移動です。エントリーは同じ親を持ち、名前が新しいだけです。*

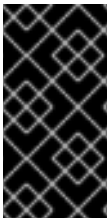
図4.10 リーフエントリーに対する modrdn 操作



サブツリーエントリーの場合、modrdn 操作はサブツリーエントリー自体の名前を変更するだけでなく、サブツリーの **下にある** *すべての子エントリーの DN コンポーネントも変更します。*

図4.11 サブツリーエントリーに対する modrdn 操作



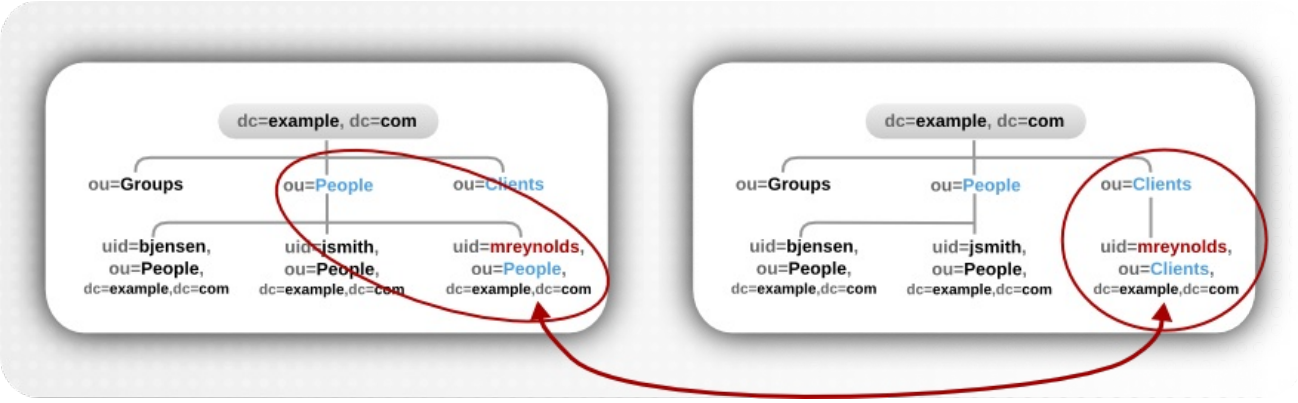


重要

サブツリーの modrdn 操作も、サブツリーエントリーの下にあるすべての子エントリーを移動し、名前を変更します。サブツリーが大きい場合、これは時間とリソースを必要とするプロセスになります。ディレクトリーツリー階層の命名構造を計画し、サブツリーの名前変更操作が頻繁に要求されないようにします。

サブツリーの名前を変更する同様のアクションは、エントリーをあるサブツリーから別のサブツリーに移動することです。これは modrdn 操作の拡張タイプで、同時にエントリーの名前を変更し（同じ名前の場合でも）、 ***newsuperior*** *属性を設定して、エントリーを別の親に移動します。*

図4.12 新しい親エントリーに対する modrdn 操作



エントリーが **entryrdn.db** インデックスに保存される方法が原因で、新しい上位操作とサブツリーの名前変更操作の両方が可能です。各エントリーは、独自のキー (**自己リンク**) で、続いてその親 (**親リンク**) と子を特定するサブキーで識別されます。これには、親と子をエントリーに対する属性として処理することでディレクトリーツリー階層を配置する形式があり、すべてのエントリーが完全な DN ではなく一意の ID とその RDN によって記述されます。

```
numeric_id:RDN => self link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
P#:RDN => parent link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
C#:RDN => child link
    ID: #; RDN: "rdn"; NRDN: normalized_rdn
```

たとえば、**ou=people** サブツリーには、親の **dc=example,dc=com** と子の **uid=jsmith** があります。

```
4:ou=people
  ID: 4; RDN: "ou=People"; NRDN: "ou=people"
P4:ou=people
  ID: 1; RDN: "dc=example,dc=com"; NRDN: "dc=example,dc=com"
C4:ou=people
  ID: 10; RDN: "uid=jsmith"; NRDN: "uid=jsmith"
```

名前変更操作を実行する際に留意すべき事項があります。

- root 接尾辞の名前を変更することはできません。
- サブツリー名前変更操作によるレプリケーションへの影響は最小限に抑えられます。レプリカ合意は、データベースのサブツリーではなく、データベース全体に適用されます。したがって、サブツリーの名前変更操作ではレプリカ合意の再設定は必要ありません。サブツリーの名前変更操作後のすべての名前の変更は、通常どおり複製されます。
- サブツリーの名前を変更し、同期合意を再設定する必要がある **場合があります**。同期合意は、接尾辞またはサブツリーレベルで設定されるため、サブツリーの名前を変更すると、同期が破損してしまう可能性があります。
- サブツリーの名前を変更するには、サブツリーに設定されたサブツリーレベルの ACI を手動で再設定し、サブツリーの子エントリーに設定されたエントリーレベルの ACI (エントリーレベルの ACI) を手動で再設定する **必要があります**。
- 子を持つサブツリーの名前を変更できますが、子を持つサブツリーを削除できません。
- **ou** から **dc** への移行など、サブツリーのコンポーネントを変更しようとすると、スキーマ違反で失敗する可能性があります。たとえば、**organizationalUnit** オブジェクトクラスには **ou** 属性が必要です。サブツリーの名前変更の一部としてその属性を削除すると、操作は失敗します。

4.3. ディレクトリーエントリーのグループ化

必要なエントリーを作成したら、管理を容易にするためにそれらをグループ化します。Directory Server は、エントリーをグループ化するための複数の方法をサポートします。

- グループの使用
- ロールの使用

4.3.1. グループについて

グループは、名前が示すように、ユーザーのコレクションです。Directory Server にはいくつかの異なるタイプのグループがあり、証明書グループ、URL グループ、および一意のグループ (各メンバーが一意でなければならない) など、許可されるメンバーシップのタイプを反映します。それぞれのグループタイプは、オブジェクトクラス(**groupOfUniqueNames** など) および対応するメンバー属性(**uniqueMember** など) で定義されます。

グループのタイプは、メンバーのタイプを識別します。グループの設定は、それらのメンバーをグループに追加する方法によって異なります。Directory Server には 2 種類のグループがあります。

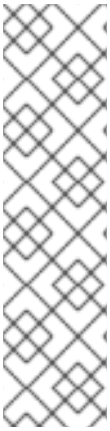
- **静的グループ** には、グループエントリーに手動で追加されるメンバーの、有限な定義済みのリストがあります。
- **動的グループ** は、フィルターを使用してグループのメンバーであるエントリーを認識します。したがって、グループメンバーシップはグループフィルターに一致するエントリーが変わるにつれて絶えず変更されます。

グループは、Directory Server におけるエントリーを整理する最も単純な形態です。これらは主に手動で設定され、組織の方法以外に機能や動作がありません。(実際、操作を実行するために LDAP クライアントはグループを操作することはできますが、グループはディレクトリーエントリーに対してなにも実行しません。)

4.3.1.1. ユーザーエントリーにおけるグループメンバーシップのリスト表示

グループは基本的にユーザー DN のリストです。デフォルトでは、グループメンバーシップはユーザーエントリーにではなく、グループエントリー自体にのみ反映されます。ただし、memberOf プラグインでは、グループメンバーエントリーを使用してユーザーエントリーを動的に更新し、**ユーザーエントリーで** ユーザーが所属するグループを反映させます。MemberOf プラグインは、指定されたメンバー属性を持つグループエントリーを自動的にスキャンし、すべてのユーザー DN のトレースを行い、グループ名でユーザーエントリーに対応する **memberOf** 属性を作成します。

グループメンバーシップはグループエントリーの member 属性によって **決定** されますが、ユーザーのすべてのグループのグループメンバーシップは、**memberOf** 属性のユーザーのエントリーに **反映され** ます。ユーザーが属するすべてのグループの名前は、**memberOf** 属性として一覧表示されます。これらの **memberOf** 属性の値は、Directory Server によって管理されます。



注記

「複数のデータベースの使用について」で概説するように、異なるデータベースに異なる接尾辞を保存することができます。

デフォルトでは、MemberOf プラグインは、グループと同じデータベースにあるユーザーのメンバーのみを検索します。ユーザーがグループとは異なるデータベースに保存されている場合、プラグインはそれらの関係を保証できないため、ユーザーエントリーは **memberOf** 属性で更新されません。

memberOfAllBackends 属性を有効にすると、設定されたすべてのデータベースを検索するように MemberOf プラグインを設定できます。

プラグインエントリーに複数値の **memberofgroupattr** を設定することで、複数のメンバー属性を識別するように MemberOf プラグインの 1 つのインスタンスを設定できます。これにより、MemberOf プラグインは複数のグループタイプを管理できます。

4.3.1.2. グループへの新規エントリーの自動追加

グループ管理は、特にDirectory Server データおよび組織を使用するクライアントや、グループを使用してエントリーに機能を適用するクライアントなど、ディレクトリーデータを管理する上で重要な要素となります。グループにより、ディレクトリー全体で一貫して、信頼できるポリシーの適用が容易になります。パスワードポリシー、アクセス制御リスト、その他のルールはすべてグループメンバーシップに基づいて設定できます。

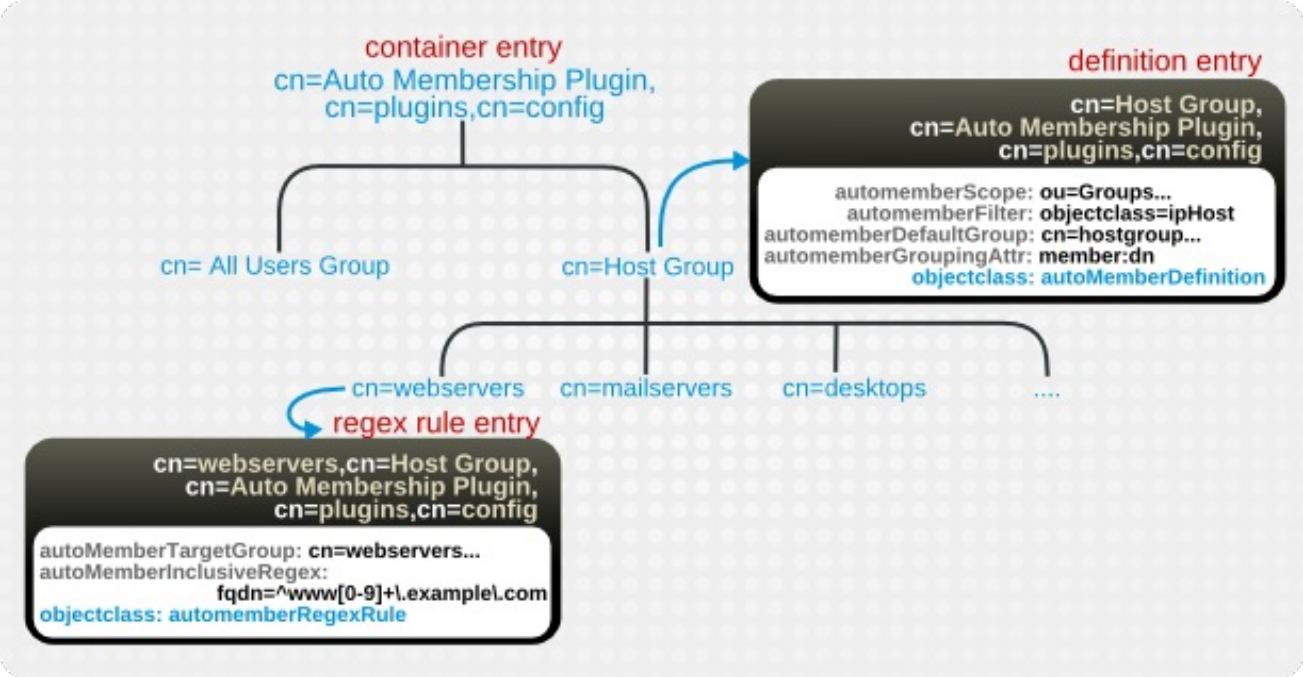
アカウントの作成時に、新しいエントリーをグループに自動的に割り当てることができるため、管理者の介入なしに、適切なポリシーと機能がそれらのエントリーに即座に適用されるようにします。

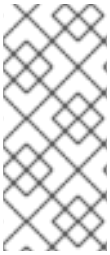
自動メンバーシッププラグイン *により、基本的に、静的グループが動的グループのように動作できるようにします。これは、(エントリー属性、ディレクトリーの場所、正規表現に基づく) ルールのセットを使用して、ユーザーを自動的に指定されたグループに割り当てます。*

他の属性の値によっては、LDAP 検索フィルターに一致するエントリーを異なるグループに追加する必要がある場合があります。たとえば、IP アドレスや物理的な場所に応じて、異なるグループにマシンを追加しないといけない場合があります。ユーザーは、従業員 ID 番号に応じて異なるグループに置かなければならない場合があります。

自動メンバー定義は、自動メンバーシッププラグインコンテナー、次に自動メンバー定義、次にその定義の正規表現条件による、ネスティングされたエントリーのセットです。

図4.13 正規表現の条件





注記

自動メンバーシップの割り当ては、エントリーがDirectory Server に追加される場合にのみ自動的に行われます。

自動メンバールールを満たすように編集される既存のエントリーの場合、適切なグループメンバーシップを割り当てるために実行できる修正タスクがあります。

4.3.2. ロールの概要

ロール *はハイブリッドグループで、静的グループと動的グループの両方として機能します。グループを使用すると、エントリーはメンバーとしてグループエントリーに追加されます。ロールを使用すると、role 属性がエントリーに追加され、その属性はロールエントリー内のメンバーを自動的に識別するため*

に使用されます。

*ロールは、効果的に、**そして自動的に** さまざまな方法でユーザーを整理します。*

- **ロールメンバーを明示的にリスト表示します。***ロールを表示すると、そのロールのメンバーの完全なリストが表示されます。ロール自体にクエリーを実行してメンバーシップを確認することができます (動的グループでは不可能です)。*

- **エントリーが属するロールの表示します。***ロールのメンバーシップはエントリーの属性によって決定されます。エントリーを表示するだけで、所属するロールがすべて表示されます。これはグループの* memberOf *属性と似ていますが、この機能が機能するためにプラグインインスタンスを有効化または設定する必要がないだけです。これは自動的です。*

- **適切なロールを割り当てます。***ロールのメンバーシップは、ロールからではなく、**エントリー**から割り当てられます。そのため、ユーザーが所属するロールは、1 つの手順でエントリーを編集して簡単に割り当てられ、削除することができます。*

マネージドロールは、通常、静的グループで実行可能なものをすべて実行できます。ロールメンバーは、動的グループによるフィルタリングと同様に、フィルターされたロールを使用してフィルタリングできます。ロールはグループよりも使いやすく、実装に柔軟性が高まり、クライアントの複雑さが軽減されます。

*ロール **メンバー** は、ロールを持つエントリーです。メンバーは、明示的に、または動的に指定できます。ロールのメンバーシップの指定方法は、ロールのタイプによって異なります。Directory Server は、以下の 3 種類のロールをサポートします。*

- **マネージドロール** *には、メンバーの明示的な列挙リストがあります。*

- **フィルターされたロール** *には、LDAP フィルターで指定される各エントリーに含まれる属性に応じて、エントリーがロールに割り当てられます。フィルターに一致するエントリーはロールを持ちます。*

- **ネストされたロール** *は、他のロールが含まれるロールです。*

ロール。ロールをアクティブ/非アクティブにするコンセプトにより、エントリーのグループ全体を一度の操作でアクティブ/非アクティブにすることができます。つまり、あるロールのメンバーを、そのメンバーが属するロールを非アクティブにすることで、一時的に無効にすることができます。

*ロールが非アクティブになっても、そのロールエントリーを使用してユーザーをサーバーにバインドできなくなるわけではありません。非アクティブ化されたロールの意味は、そのロールに属するエントリーを使用してユーザーをサーバーにバインドできないことです。非アクティブ化されたロールに属するエントリーは、**nsAccountLock** 属性が **true** に設定されます。*

*ネスティングされたロールが非アクティブになると、ネスティングされたロール内の任意のロールのメンバーである場合、ユーザーをサーバーにバインドできません。ネスティングされたロールに直接的または間接的であるロールに属するすべてのエントリーでは、**nsAccountLock** が **true** に設定されています。何層にもなったネスティングロールが可能です。ネスティング内のどのネスティングされたロールを非アクティブにしても、その下にあるすべてのロールとユーザーが非アクティブになります。*

4.3.3. ロールとグループの使い分け

ロールとグループは同じ目標を達成することができます。管理されたロールは、静的グループができることをすべて行うことができ、フィルタリングされたロールは、動的グループのようにメンバーをフィルタリングして識別することができます。ロールとグループの両方に、メリットとデメリットがあります。ロールとグループのどちらを使用するか (または混在させるか) は、クライアントのニーズとサーバーのリソースのバランスによって決まります。

*ロールはクライアント側の複雑さを軽減することができ、それがロールの主な利点です。ロールを使用すると、クライアントアプリケーションはエントリーの **nsRole** 操作属性を検索して、ロールのメンバーシップを確認できます。この多値属性は、エントリーが属するすべてのロールを識別します。クライアントアプリケーションの観点からは、メンバーシップを確認する方法は統一されており、サーバー側で実行されます。*

しかし、このようなクライアントにとっての使い勝手の良さは、サーバーの複雑化という代償を伴います。 サーバーがクライアントアプリケーションに対して機能するため、Directory Server ではロールの評価がグループを評価するよりもリソース集約されます。

グループは、サーバーにとっては簡単ですが、効果的に使用するには、よりスマートで複雑なクライアントが必要です。たとえば、ダイナミックグループは、アプリケーションの観点から見ると、グループメンバーのリストを提供するサーバーからのサポートを提供しません。代わりに、アプリケーションはグループ定義を取得してから、フィルターを実行します。グループメンバーシップは、適切なプラグインが設定されている場合にのみ、ユーザーエントリーに反映されます。結局のところ、グループのメンバーシップを決定する方法は一様ではなく、予測もできません。

注記

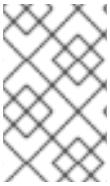
*グループメンバーシップの管理をバランスよく行うことができるものとして、MemberOf プラグインがあります。**memberOf** を使用すると、クライアントにとっての使いやすさと、サーバーによる計算効率の良さのバランスがとります。*

*MemberOf プラグインは、ユーザーがグループに追加されるたびに、ユーザーエントリーに **memberOf** 属性を動的に作成します。クライアントは、グループエントリーを 1 回検索することで、そのグループのすべてのメンバーのリストを取得できます。ユーザーエントリーを 1 回検索することで、そのユーザーが属するすべてのグループの完全なリストを取得できます。*

*サーバーには、メンバーシップが変更されたときにのみメンテナンスのオーバーヘッドが発生します。指定されたメンバー（グループ）属性と **memberOf**（ユーザー）属性の両方がデータベースに格納されるため、検索に必要な追加の処理がないため、クライアントからの検索は非常に効率的になります。*

4.4. 仮想ディレクトリー情報ツリービュー

*Directory Server は、**仮想ディレクトリー情報ツリービュー** あるいは **仮想 DIT ビュー** と呼ばれる、ディレクトリー情報の階層ナビゲーションと組織の概念をサポートします。*



注記

ビューが返すエントリーが同じバックエンドに存在しなければならないないという点で、仮想ビューは、複数のバックエンドと完全に互換性があるわけではありません。検索は 1 つのバックエンドに限定されます。

4.4.1. 仮想 DIT ビューについて

ディレクトリーの名前空間を設定するには 2 つの方法があります。

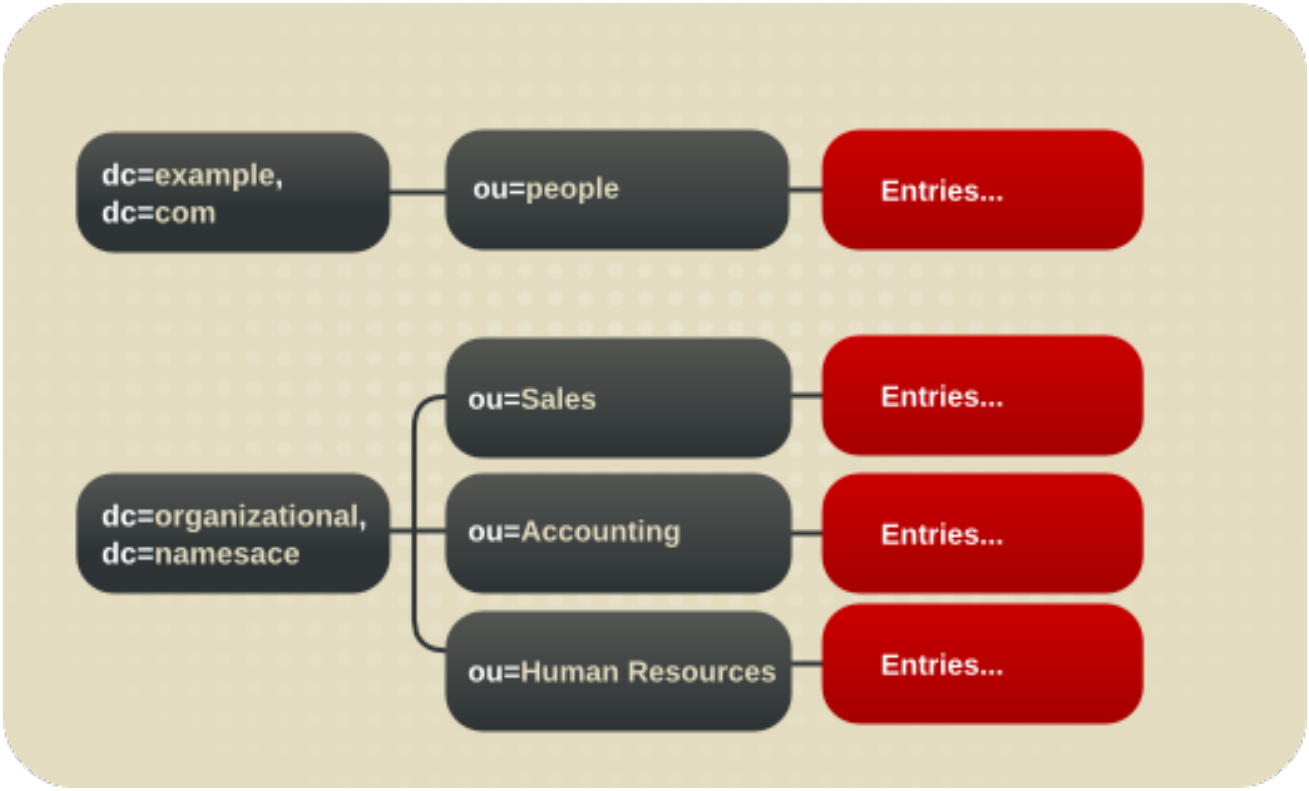
- *階層構造のディレクトリー情報ツリー。*
- *フラットなディレクトリー情報ツリー。*

階層構造の DIT は、ディレクトリーのナビゲーションに便利ですが、変更するのは面倒で時間がかかります。階層構造の DIT への大規模な組織変更は、通常、かなりのサービスの中断を伴うため、費用と時

間のかかる作業になります。通常、営業時間外やトラフィックの少ない時間帯に変更を行うことでしか、このデメリットを最小限に抑えることはできません。

フラットなDIT は、ほとんど変更を必要としませんが、ディレクトリーサービス内でエントリーをナビゲーションしたり、管理するための便利な方法を提供しません。また、フラットなDIT は、自然な階層構造のグルーピングを持たないことにより管理が複雑になるため、多くの管理上の課題があります。

図4.14 フラットなDIT と組織ベースのDIT の例



階層構造のDIT を使用する場合、デプロイメントは階層のサブジェクトドメインを決定する必要があります。選択肢は1 つしかなく、自然な傾向は、組織の階層を選択することです。

組織のこのビューは多くの場合に適しますが、1 つのビューしかないことが、ディレクトリーのナビゲーションや管理に対する大きな制約となります。例えば、経理部門の人に属するエントリーを探すには、組織の階層が適しています。しかし、この表示では、Mountain View, California のような地理的な場所にいる人に属するエントリーを探すのにはあまり役に立ちません。2 番目のクエリーも1 番目のクエリーと同様に有効ですが、エントリーに含まれる属性の知識と追加の検索ツールが必要になります。このような場合、DIT を使用してナビゲーションを行うことは適切ではありません。

同様に、DIT が管理機能の要件に合致していれば、ディレクトリーの管理は非常に容易になります。また、DIT の設定は、レプリケーションやマイグレーションへの考慮など、他の要因にも影響されます。これらによりDIT がこれらのアプリケーションの機能ユーティリティーを持ちますが、他のケースでは実用性がほとんどないということもあります。

明らかに、階層はナビゲーションと管理のための便利なメカニズムです。しかし、既存のDIT に変更を加える際の負担を回避するために、デプロイメントではフラットなDIT のメリットにより階層をすべて廃止する選択も可能です。

もし、ディレクトリーが、対象となるエントリーを移動させることなくエントリーにマッピングされる階層を任意の数作成する方法を提供する場合は、デプロイメントにとって有用になるでしょう。Directory Server の **仮想 DIT ビュー** *機能は、ディレクトリーのデプロイメントに使用するDIT の種類を決定する際の迷いを解消します。*

仮想 DIT ビュー *では、エントリーが特定の場所に物理的に存在していなくても、それらのエントリー*

を階層的にナビゲートすることができます。仮想 DIT ビューは、エントリーの情報を使用してビュー階層に置きます。クライアントアプリケーションでは、仮想 DIT ビューは通常のコンテナー階層として表示されます。ある意味、仮想 DIT ビューは、エントリーがフラットな名前空間にあるか、独自の別の階層にあるかに関わらず、DIT 階層をそれらのエントリーのセットに重ね合わせます。

*通常の DIT 階層と同じように、仮想 DIT ビュー階層を作成します。同じエントリー（組織単位エントリーなど）を作成しますが、追加のオブジェクトクラス(**nsview**)およびビューを記述するフィルター属性(**nsviewfilter**)を作成します。追加属性の追加後、ビューフィルターにマッチするエントリーが即座にビューを反映させます。対象となるエントリーは、ビューの中に存在している **ように見える** だけで、実際の場所は変わりません。仮想 DIT ビューは、サブツリーまたは 1 レベルの検索が、想定された結果で返されることで、通常の DIT と同様に動作します。*

*エントリーの追加および修正については、『*Red Hat Directory Server Administration Guide*』の Creating Directory Entries を参照してください。*

図4.15 ビューを使用した DIT の組み合わせ

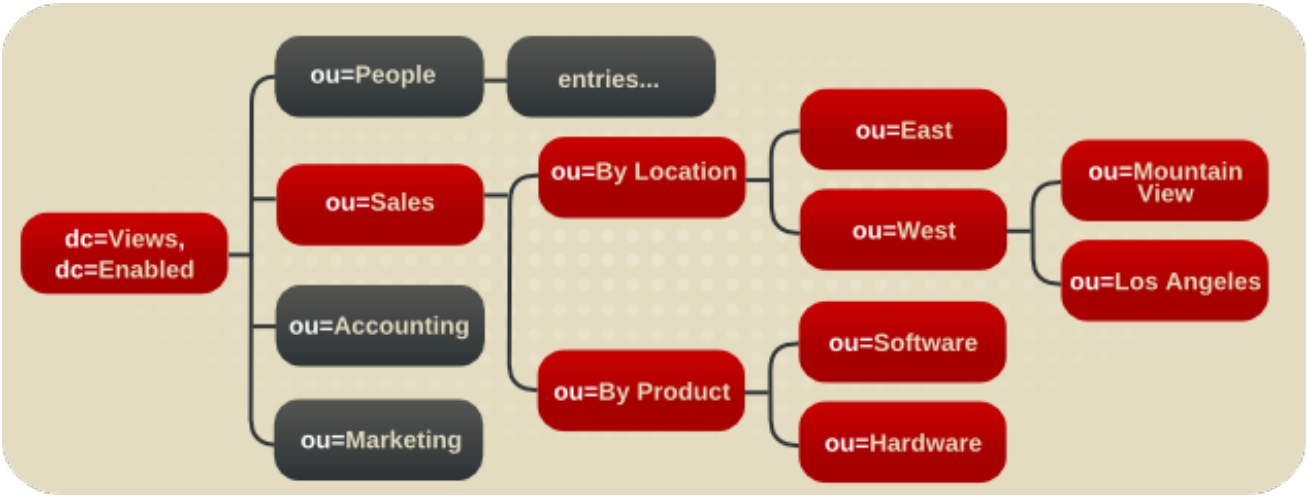
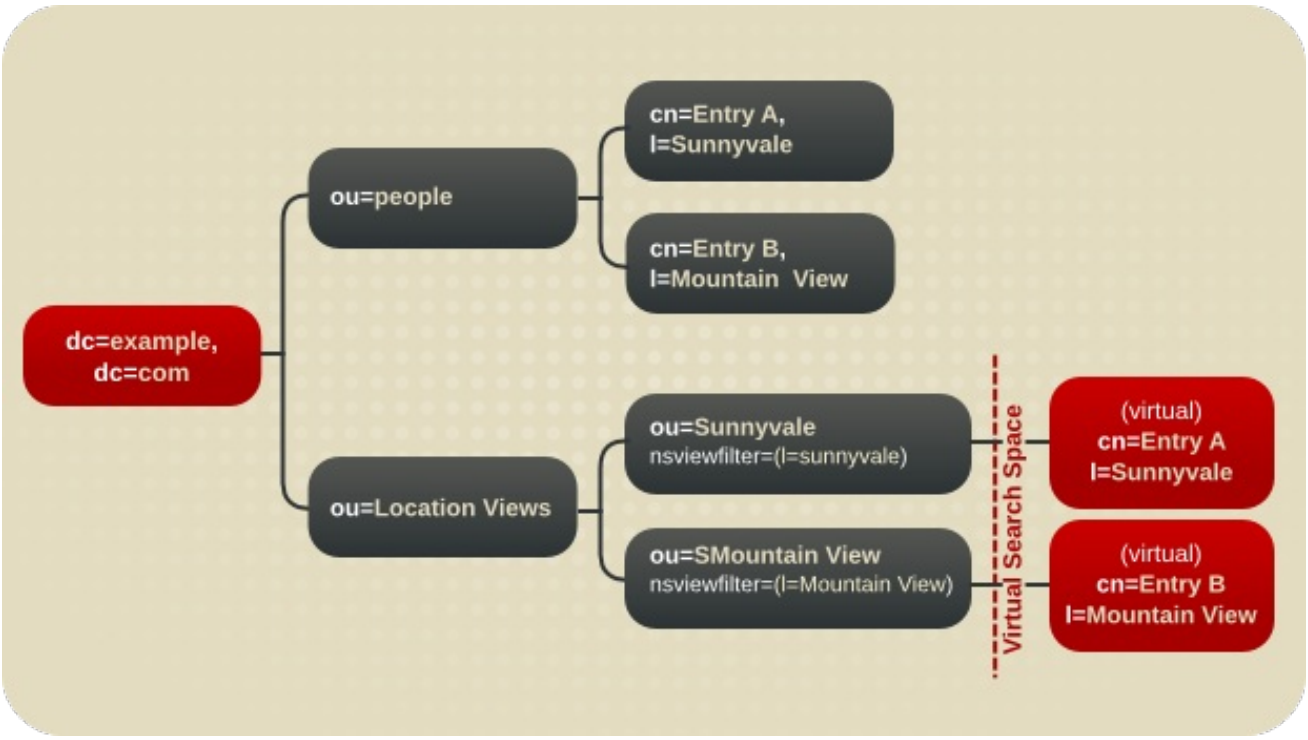


*図4.15「ビューを使用した DIT の組み合わせ」の DIT は、図4.14「フラットな DIT と組織ベースの DIT の例」で示した 2 つの DIT がビューを使用して組み合わされたときの様子を示しています。ビューでは本来、エントリーがビュー階層の複数の場所に表示されることを許可するため、この機能を使用して **ou=Sales** エントリーを拡張し、営業エントリーを場所別または製品別に表示できるようになりました。*

仮想 DIT ビュー階層のセットが与えられた場合、ディレクトリーユーザーは、必要なエントリーにナビゲートするのに最も適したビューを使用することができます。たとえば、対象のエントリーが Mountain View に住んでいる人であれば、場所ベースの情報を利用したナビゲーションにより始まるビューが最適です。組織的な質問であれば、組織ビューの方が適しているでしょう。この 2 つのビューは、同時に Directory Server に存在し、同じエントリーに対して操作を行いますが、それぞれのビューはディレクトリー構造のバージョンを表示する際の目的が異なります。

図4.15「ビューを使用した DIT の組み合わせ」のビュー対応ディレクトリーのエントリーは、階層内の最上位ビューの親のすぐ下にあるフラットな名前空間に含まれています。これは必須ではありません。エントリーはそれ専用の階層に存在することができます。エントリーの配置に関してビューが持つ唯一の懸念は、ビュー階層の親の子孫でなければならないということです。

図4.16 仮想 DIT ビュー階層を含む DIT



- *サブツリー **ou=People** には、実際の エントリー **A** および エントリー **B** エントリーが含まれています。*
- *サブツリー **ou=Location Views** は、ビュー階層です。*
- *リーフノード **ou=Sunnyvale** および **ou=Mountain View** には、ビューを記述する属性 **nsviewfilter** が含まれています。*

  *実際のエントリーを含まないため、これらはリーフノードです。ただし、クライアントアプリケーションがこれらのビューを検索すると、**ou=Mountain View** の下で **ou=Sunnyvale** および **Entry B** の下にエントリー **A** が見つかります。この仮想検索領域は、すべての上位ビューの **nsviewfilter** 属性によって記述されます。ビューからの検索では、仮想検索空間からのエントリーと実際の検索空間からのエントリーの両方が返されます。これにより、ビュー階層を従来の DIT として機能させたり、従来の DIT をビュー階層に変更したりすることができます。*

4.4.2. 仮想 DIT ビュー活用のメリット

仮想 DIT ビューを利用することで、導入の意思決定が容易になります。その理由は以下のとおりです。

- *仮想 DIT ビューは、従来の階層構造が提供するのと同様のナビゲーションと管理のサポートを提供するため、ビューではエントリーにフラットな名前空間を使用することができます。*

  また、DIT に変更があった場合でも、エントリーを移動する必要はなく、仮想 DIT ビュー階層のみが変更されます。これらの階層は実際のエントリーを含まないため、シンプルで素早く変更することができます。

- *導入計画中の見落としは、仮想 DIT ビューを使用することで、壊滅的ではなくなります。最初の段階で階層が正しく開発されていなくても、サービスに支障をきたすことなく、簡単かつ迅速に変更することができます。*

- *ビュー階層を数分で完全に修正し、その結果を即座に実現できるため、ディレクトリーのメンテナンスコストを大幅に削減できます。*

仮想 DIT 階層の変更は瞬時に実現されます。組織変更があっても、新しい仮想 DIT ビューを迅速に作成することができます。新しい仮想 DIT ビューは古いビューと同時に存在できるので、エントリー自体およびそれらを使用するアプリケーション向けに、より段階的な変更が可能です。ディレクトリーの組織の変更は、1 か 0 かの操作ではないため、一定期間サービスを中断せずに実行できます。

- *ナビゲーションと管理に複数の仮想 DIT ビューを使用すると、ディレクトリーサービスをより柔軟に利用することができます。*

  仮想 DIT ビューで提供される機能により、組織は古いメソッドと新しいメソッドの両方を使用して、DIT の特定のポイントにエントリーを配置する必要なしにディレクトリーデータを編成できます。

- *仮想 DIT ビュー階層は、共通的に必要とされる情報の取得を容易にするために、レディーメイドのクエリーとして作成できます。*

- *ビューは、作業の柔軟性を高め、ディレクトリーユーザーが通常は知る必要のない属性名や値を使用して複雑な検索フィルターを作成する必要性を軽減します。*

  ディレクトリー情報を表示およびクエリーする手段が複数ある柔軟性により、エンドユーザーとアプリケーションは階層ナビゲーションを通じて必要とされるものを直感的に把握できます。

4.4.3. 仮想 DIT ビューの例

*以下の LDIF エントリーは、場所に基づいた仮想 DIT ビュー階層を示しています。**dc=example,dc=com** の下に存在し、ビューの説明に適合するエントリーは、すべて場所別に整理されたこのビューに表示されます。*

```
dn: ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Location Views
description: views categorized by location


dn: ou=Sunnyvale,ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Sunnyvale
nsViewFilter: (l=Sunnyvale)
description: views categorized by location


dn: ou=Santa Clara,ou=Location Views,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Santa Clara
nsViewFilter: (l=Santa Clara)
description: views categorized by location


dn: ou=Cupertino,ou=Location Views,dc=example,dc=com
```

```
objectclass: top
objectclass: organizationalUnit
objectclass: nsView
ou: Cupertino
nsViewFilter: (l=Cupertino)
description: views categorized by location
```

ou=Location Views,dc=example,dc=com に基づくサブツリー検索は、フィルター **(l=Sunnyvale)**、または **(l=Cupertino)** に一致する **dc = example,dc=com** の下のすべてのエントリーを返します。反対に、1 レベルの検索では、子ビューエントリー以外のエントリーは返されません。これは、すべての該当するエントリーが 3 つの子孫ビューにあるためです。

ou=Location Views,dc=example,dc=com ビューエントリー自体にはフィルターが含まれていません。この機能は、ビューに含まれるエントリーをさらに制限する必要なしに、階層組織を容易にします。すべてのビューがフィルターを省略できます。例示したフィルターは非常にシンプルですが、使用するフィルターは必要に応じて複雑にすることができます。

ビューに含まれるエントリーのタイプを制限することが望ましい場合があります。たとえば、この階層をユーザーエントリーのみに限定するには、**ou=Location Views,dc=example,dc=com** にフィルター値 **(objectclass=organizationalperson)** を指定して **nsfilter** 属性を追加します。

フィルターを含む各ビューは、すべての子孫のビューのコンテンツを制限し、フィルターが含まれる子孫のビューも先祖の内容を制限します。たとえば、上記の新しいフィルターと共に最上位ビュー **ou=Location Views** を最初に作成すると、**organization** オブジェクトクラスを持つすべてのエントリーを含むビューが作成されます。さらにエントリーを制限する子孫のビューが追加されると、子孫のビューに表示されているエントリーは、先祖のビューから削除されます。これは、仮想 DIT ビューが従来の DIT の動作を模倣する方法を示しています。

仮想 DIT ビューは従来の DIT の動作を模倣しますが、ビューは従来の DIT ができなかったことを実行できます。エントリーを複数の場所に表示できます。たとえば、エントリー **B** を **Mountain View** と **Sunnyvale** の両方に関連付けるには(図4.16「仮想 DIT ビュー階層を含む DIT」 を参照)、location 属性に **Sunnyvale** の値を追加すると、エントリーが両方のビューに表示されます。

4.4.4. 表示およびその他のディレクトリー機能

Directory Server では、**サービスクラス** と **ロール** は、どちらもビューをサポートしています。「ディレクトリーエントリーのグループ化」 を参照してください。ビュー階層の下にサービスクラスまたはロールを追加する場合、ビューに論理的および実際的の両方に含まれるエントリーはスコープ内にあると見なされます。つまり、仮想 DIT ビューを使用してロールおよびサービスクラスを適用できますが、フラット名前空間のクエリーであっても、そのアプリケーションの効果を確認できます。

この機能の使用方法は、『Red Hat Directory Server Administration Guide』 の Advanced Entry Management を参照してください。

ビューを使用するには、アクセス制御に若干異なるアプローチが必要です。現在、ビューでは ACL への明示的なサポートがないため、ビューの親でロールベースの ACL を作成し、ロールをビュー階層の適切な部分に追加します。これにより、階層の組織プロパティーを利用できます。

検索のベースがビューであり、検索範囲がベースではない場合、検索はビューベースの検索になります。それ以外の場合は、従来の検索です。

たとえば、**dc=example,dc=com** のベースで検索を実行しても、仮想探索空間からのエントリーが返されません。実際、virtual-search-space の検索は実行されません。ビューの処理は、検索ベースが **ou=Location Views** の場合にのみ発生します。これにより、ビューは、検索により、両方の場所からのエントリーが表示されないようにします。(従来の DIT の場合、両方の場所からのエントリーが返されます。)

4.4.5. パフォーマンスに対する仮想ビューの影響

ビューベースの階層のパフォーマンスは、階層自体の構造と DIT のエントリー数に依存します。一般的には、ディレクトリーサービスで仮想 DIT ビューが有効になっている場合、パフォーマンスに関して若干の変化が発生する可能性があります (従来の DIT での同等の検索に対して数パーセントポイント以内)。検索でビューを呼び出さない場合は、パフォーマンスへの影響はありません。導入の前に、予想される検索パターンおよび負荷に対して仮想 DIT ビューをテストします。

また、ビューを組織内の汎用のナビゲーションツールとして使用する場合は、ビューフィルターで使用される属性をインデックス化することが推奨されます。さらに、ビューで使用されるサブフィルターが設定済みの仮想リストビューインデックスと一致する場合、そのインデックスがビューの評価で使用されます。

特にビュー用に、ディレクトリーの他の部分をチューニングする必要はありません。

4.4.6. 既存のアプリケーションとの互換性

仮想 DIT ビューは、従来の DIT を高いレベルで模擬するように設計されています。ビューの存在は、ほとんどのアプリケーションに対して透過的にする必要があります。ビューを使用していることを示すものがあってはいけません。いくつかの特殊なケースを除き、Directory Server インスタンスでビューが使用されていることを、ディレクトリーユーザーが知る必要はありません。ビューは従来の DIT のように表示され、動作します。

特定のタイプのアプリケーションでは、ビュー対応ディレクトリーサービスを使用すると問題が発生する可能性があります。以下に例を示します。

- ターゲットエントリーの DN を使用して DIT をナビゲートするアプリケーション。

  このタイプのアプリケーションは、エントリーが見つかったビュー階層ではなく、エントリーが物理的に存在する階層をナビゲートしていることを把握します。その理由は、ビューは、ビューの階層に合わせてエントリーの DN を変更することで、エントリーの本当の位置を隠そうとはしないからです。これは意図的なものです。エントリーの真の位置が偽装されていると、多くのアプリケーションが機能しなくなります。例えば、一意のエントリーを識別するために DN に依存するアプリケーションなどです。このように DN を分解して上方向にナビゲートするのは、クライアントアプリケーションとしては珍しい手法ですが、それにもかかわらず、これを行うクライアントは意図した通りに機能しない可能性があります。

- **numSubordinates** 操作属性を使用して、ノードの下に存在するエントリーの数を決定するアプリケーション。

  ビュー内のノードについては、現在、仮想検索空間を無視して、実際の検索空間に存在するエントリーのみをカウントしています。そのため、アプリケーションでは検索を使用してビューを評価できない場合があります。

4.5. ディレクトリーツリーの設計例

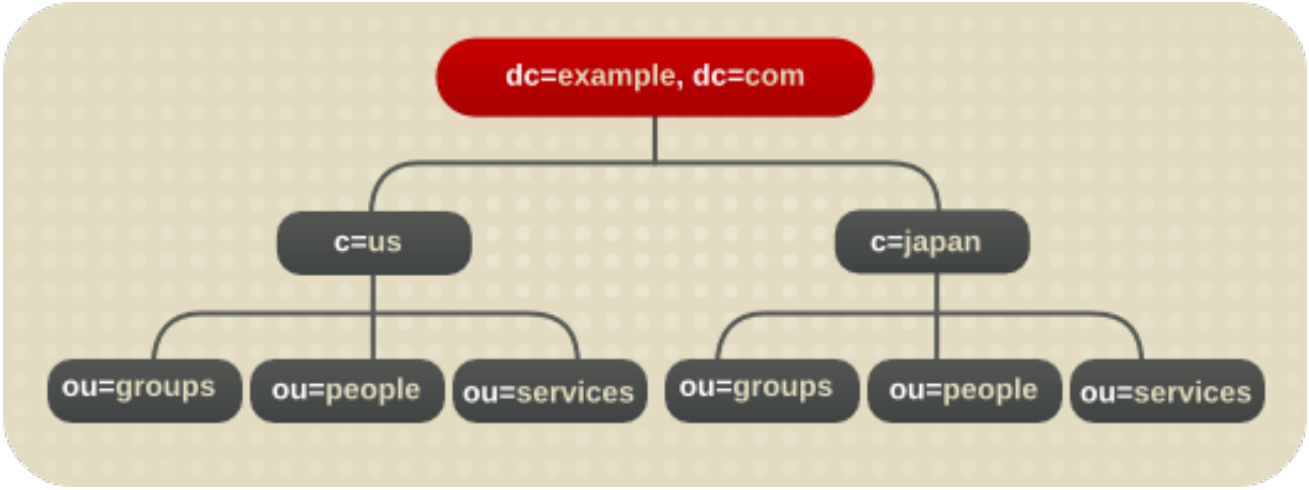
以下のセクションでは、フラット階層をサポートするように設計されたディレクトリーツリーの例と、より複雑な階層のいくつかの例を説明します。

4.5.1. グローバル企業のディレクトリーツリー

グローバル企業をサポートするには、インターネットドメイン名をディレクトリーツリーのルートポイントとして使用し、続いて企業が業務を行う各国用に、そのルートポイントのすぐ下でツリーを分岐します。特に企業がグローバル企業の場合、「接尾辞の命名規則」に説明されているように、ディレクトリーツリーのルートポイントとして国のデジグネーターを使用することは避けてください。

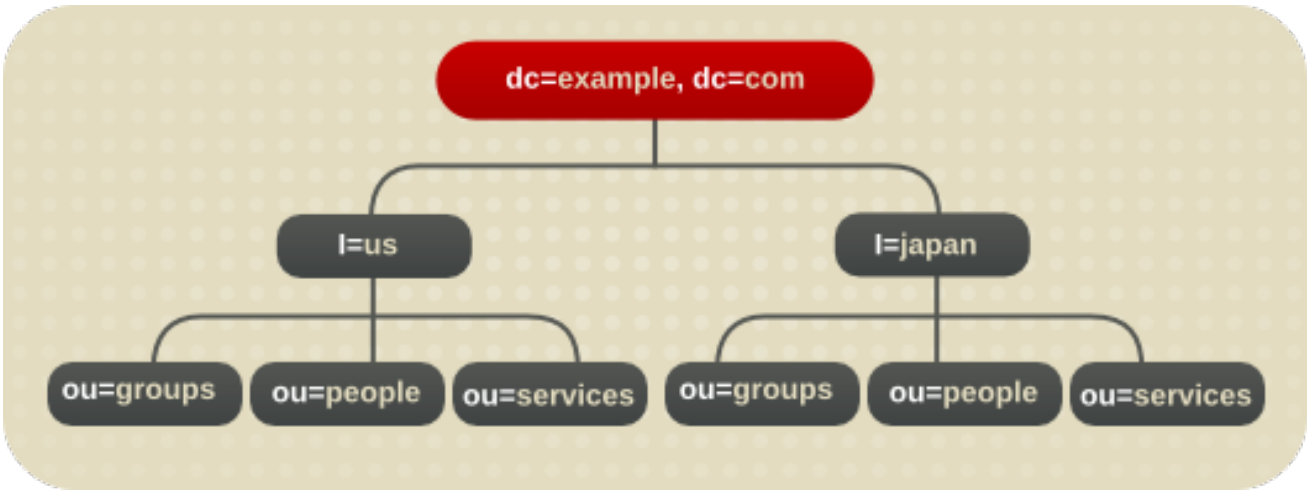
LDAP は DN 内の属性の順序に制限がないため、**c** 属性は各国のブランチを表すことができます。

図4.17 c 属性を使用した様々な国を表現



ただし、管理者によっては、これがまれに不便であると思われるため、代わりに **l** 属性を使用してさまざまな国を表します。

図4.18 l 属性を使用した様々な国を表現



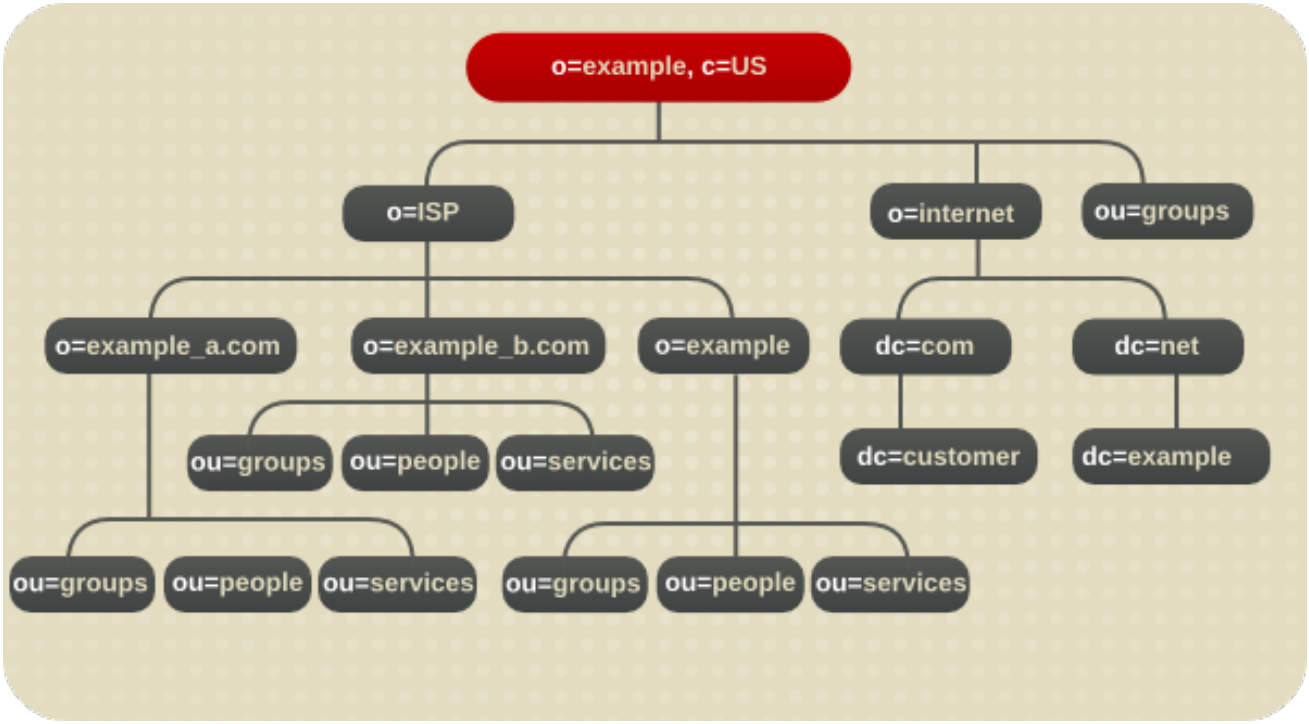
4.5.2. ISP のディレクトリーツリー

インターネットサービスプロバイダー (ISP) は、そのディレクトリーで複数のエンタープライズをサポートする場合があります。ISP では、各顧客を一意の企業として考慮し、それに合わせてディレクトリーツリーを設計する必要があります。セキュリティー上の理由から、各アカウントには一意の接尾辞と独立したセキュリティーポリシーを持つ一意のディレクトリーツリーを提供する必要があります。

ISP は、各顧客に個別のデータベースを割り当て、これらのデータベースを個別のサーバーに保存することを検討する必要があります。それぞれのディレクトリーツリーを独立したデータベースに配置することで、他の顧客に影響を与えることなく、ディレクトリーツリーごとにデータのバックアップやリストアを行うことができます。

また、パーティションを使用すると、ディスクの競合によるパフォーマンスの問題を軽減するのに役立ち、ディスク障害の影響を受ける可能性のあるアカウントの数を削減できます。

図4.19 Example ISP のディレクトリーツリー



4.6. その他のディレクトリーツリーリソース

ディレクトリーツリーの設計に関する詳細は、以下を参照してください。

- *RFC 2247: Using Domains in LDAP/X.500 Distinguished Names*
- *RFC 2253: LDAPv3, UTF-8 String Representation of Distinguished Names*

第5章 動的属性値の定義

「標準属性」 で説明されているように、LDAP エントリーは、エントリーに追加される **属性** *に情報の各部分を保存します。*

Red Hat Directory Server は、ディレクトリーエントリーで一部の属性タイプを動的かつ自動的に維持するためのさまざまなメカニズムいくつか提供します。これらのプラグインおよび設定オプションを使用すると、ディレクトリーデータの管理やエントリー間の関係の表現が容易になります。

5.1. 管理属性の概要

サイト調査を実行する場合の最初のステップの1つは、ディレクトリー内のエントリーの **特性** *を特定することです(「ディレクトリーデータの特徴付け」)。これらの特性は、ディレクトリーエントリーで記録する必要があるエンティティーのさまざまな側面になります。従業員の場合は、その人のマネージャー、肩書き、ビジネスカテゴリー、電子メールアドレス、自宅と会社の電話番号などの情報を意味します。エントリーの各特性は、エントリー属性で維持されます。*

エントリーの特性の一部に、相互の **関係** *があります。マネージャーに従業員がいることは明らかなため、この2つのエントリーには関連性があります。グループはメンバーに関連付けられます。共通の物理的な場所を共有するエントリー間のように、あまり明白でない関係もあります。*

Red Hat Directory Server は、このようなエントリー間の関係をスムーズにかつ一貫して維持する方法を複数提供します。ディレクトリー内のデータの一部として、属性を自動的に適用または生成できるプラグインがいくつかあります。

- **属性の一意性** *は、サブツリーまたはデータベース内の特定の属性のすべてのインスタンスが一意の値を持つことを必要とします。これは、エントリーが作成されるか、属性が変更されるたびに実施されます。*

- **サービスクラス** *では、1つのエントリーをテンプレートとして使用します。その属性値が変更されるたびに、CoS の範囲内の他のすべてのエントリーは、エントリーの同じ属性を自動的に変更します。(CoS の影響を受けるエントリーは、定義エントリーによって特定されます)*

- **管理エントリー** *は、定義された範囲内の別のエントリーが作成されるたびに、定義されたテンプレートに従って1つのエントリーを作成します。特に外部クライアントとの統合では、エントリーペアを自動的に作成し、管理する必要がある場合があります。管理エントリーは、2番目のエントリーのテンプレートを定義し、自動的に更新するメカニズムを提供します。*

- **リンク先属性** *は、あるエントリーの属性の DN 値に従い、参照されるエントリーに (元のエントリーを参照する値を持つ) 所定の属性を自動的に追加します。したがって、エントリーA が直接レポートとしてエントリーB を一覧表示した場合、エントリーB が自動的に更新されて、指定されたマネージャーとしてエントリーA を持つ* ***manager*** *属性を指定できます。*

- **分散数値の割り当て** *では、固有の識別番号をエントリーに自動的に割り当てます。これは、組織全体で一意でなければならない GID または UID 番号の割り当てに役立ちます。*

ディレクトリーデータとディレクトリースキーマの両方の計画の一環として、特定のエントリー属性値に関するいくつかの点を考慮してください。

- *エントリー間の関係はどのようなものですか?エントリー間で共有される共通の属性はありますか?エントリー間のつながりを表す必要がある属性はありますか?*

- *データの元のソースは、どのように、どこで (どのエントリーで) 維持される可能性がありますか?この情報はどのくらいの頻度で更新され、データが変更された場合、どれくらいのエントリーが影響を受けるのでしょうか?*

- *このエントリーではどのようなスキーマ要素が使用されていますか？また、それらの属性の構文は何ですか？*
- *プラグインは、レプリケーションや同期などの分散ディレクトリー設定をどのように処理しますか？*

5.2. 属性の一意性について

Attribute Uniqueness プラグインは、preoperation プラグインです。つまり、サーバーが LDAP 操作を行う **前** *に、プラグインがすべての更新操作をチェックすることを意味します。プラグインは、その操作が、監視するように設定されている属性と接尾辞に適用されるかどうかを判断します。*

更新操作がプラグインによって監視される属性および接尾辞に適用され、2 つのエントリーが同じ属性値を持つと、サーバーは操作を終了し、クライアントに ***LDAP_CONSTRAINT_VIOLATION*** *エラーを返します。*

Attribute Uniqueness プラグインの各インスタンスは、1 つ以上のサブツリーの 1 つの属性に対してチェックを行います。複数の属性の一意性を確認するには、チェックする属性ごとに、プラグインの個別のインスタンスを作成する必要があります。

Attribute Uniqueness プラグインは、ユーザーが定義した特定の方法で操作することができます。

- *指定されたサブツリーのすべてのエントリーを確認できます。*

  たとえば、会社 ***example.com*** *が example_a.com および* ***example_b.com*** *のディレクトリーをホストしている場合、* ***uid=jdoe,ou=people,o=example_a,dc=example,dc=com*** *などのエントリーが追加された場合、* ***o=example_a,dc=example,dc=com*** *サブツリーでのみ一意性を適用する必要があります。これは、Attribute Uniqueness プラグイン設定にサブツリーの DN を明示的にリスト表示することで実施されます。*

- *更新されたエントリーの DN のエントリーに関連するオブジェクトクラスを指定し、その下のすべてのエントリーで一意性チェックを実行します。*

  このオプションはホスト環境で役に立ちます。たとえば、 ***uid=jdoe,ou=people,o=example_a,dc=example,dc=com*** *などのエントリーを追加する場合、* ***o=example_a,dc=example,dc=com*** *サブツリーの下で一意性を強制しますが、このサブツリーを明示的に設定内に一覧表示するのではなく、*marker **オブジェクトクラス** *を示します。マーカーオブジェクトクラスが* ***organization*** *に設定されている場合、一意性チェックアルゴリズムは、このオブジェクトクラス(****o=example_a****)を持つ DN のエントリーを見つけ、その下のすべてのエントリーでチェックを実行します。*

  また、更新されたエントリーに指定のオブジェクトクラスが含まれる場合に限り、一意性を確認することもできます。たとえば、更新されたエントリーに ***objectclass=inetorgperson*** *が含まれている場合にのみ、チェックを実行できます。*

Directory Server は、Directory Server の初回設定時に、 ***uid*** *属性の Attribute Uniqueness プラグインのデフォルトインスタンスを提供します。このプラグインインスタンスは、* ***uid*** *属性に指定された値がルート接尾辞(* ***userRoot*** *データベースに対応する接尾辞)で一意になるようにします。*

このプラグインは、マルチサプライヤーのレプリケーションの操作に影響するため、デフォルトでは無効になっています。

レプリケーション操作の一部として更新が実行されると、Attribute Uniqueness プラグインは属性値のチェックを実行しません。

クライアントアプリケーションによる変更はすべて、サプライヤーサーバーで実行されるため、サプライヤーで Attribute Uniqueness プラグインを有効にする必要があります。コンシューマーサーバーで有効にする必要はないです。

コンシューマーで Attribute Uniqueness プラグインを有効にしても、Directory Server の正常な操作を妨げることはありませんが、パフォーマンスが低下する可能性はあります。

マルチサプライヤーのレプリケーションシナリオでは、サプライヤーは同じレプリカのサプライヤーとコンシューマーの両方として機能します。マルチサプライヤーのレプリケーションでは、大まかに一貫性のあるレプリケーションモデルを使用しているため、サーバーの 1 つで Attribute Uniqueness プラグインを有効化するだけでは、属性値が両方のサプライヤーサーバー間でいつでも一意になるようにするには十分とは言えません。したがって、1 つのサーバーで Attribute Uniqueness プラグインを有効化すると、各レプリカに保持されるデータに不整合が生じる可能性があります。

ただし、以下の両方の条件を満たす場合は、Attribute Uniqueness プラグインを使用することが可能です。

- 一意性のチェックが行われる属性は、ネーミング属性です。
- Attribute Uniqueness プラグインは、両方のサプライヤーサーバーで有効化されます。

これらの条件が満たされると、属性の一意性の競合は、レプリケーション時に命名の競合として報告されます。命名の競合は、手動で解決する必要があります。

5.3. サービスクラスについて

サービスクラス (CoS) は、アプリケーションが認識できない方法でエントリー間で属性を共有します。CoS では、一部の属性値がエントリー自体と一緒に保存されない場合があります。代わりに、エントリーがクライアントアプリケーションに送信される際に、サービスクラスのロジックにより生成されます。

たとえば、ディレクトリーには、すべてが共通の属性 **facsimileTelephoneNumber** を共有する数千ものエントリーが含まれています。従来は、FAX 番号を変更するためには、各エントリーを個別に更新する必要があり、管理者にとっては大きな負担となり、すべてのエントリーが更新されないリスクがありました。CoS を使用すると、属性値を動的に生成できます。**facsimileTelephoneNumber** 属性は 1 つのロケーションに保存され、各エントリーはその場所から fax number 属性を取得します。アプリケーションの場合、これらの属性は、実際にはエントリー自体に保存されていないにもかかわらず、他のすべての属性と同じように表示されます。

各 CoS は、ディレクトリー内の複数のエントリーで設定されています。

- **CoS 定義エントリー** は、CoS のタイプを識別します。これは、影響を与えるブランチの下に LDAP サブエントリーとして保存されます。
- **テンプレートエントリー** には、共有属性値のリストが含まれます。テンプレートエントリー属性値を変更すると、その属性を共有するすべてのエントリーに自動的に適用されます。

CoS 定義エントリーとテンプレートエントリーは相互に作用して、**ターゲットエントリー** (そのスコープ内のエントリー) に属性値を提供します。提供する値は、以下によって異なります。

- エントリーの DN (ディレクトリーツリーの異なる部分に異なる CoS が含まれる場合があります)。
- エントリーで保存されるサービスクラスの属性値。

  サービスクラス属性がない場合、特定のデフォルト CoS を意味することがあります。

- *CoS テンプレートエントリーに保存される属性値。*

  各 CoS テンプレートエントリーは、特定の CoS の属性値を提供します。

- *エントリーのオブジェクトクラス。*

  COS の属性値は、スキーマチェックが有効な場合、エントリーに属性を許可するオブジェクトクラスが含まれている場合にのみ生成され、そうでない場合は、すべての属性値が生成されます。

- *ディレクトリーツリーの特定エントリーの一部に保存される属性。*

重要

*CoS 定義で使用する属性をインデックス化しないでください(**cosAttribute** パラメーター)。*

注記

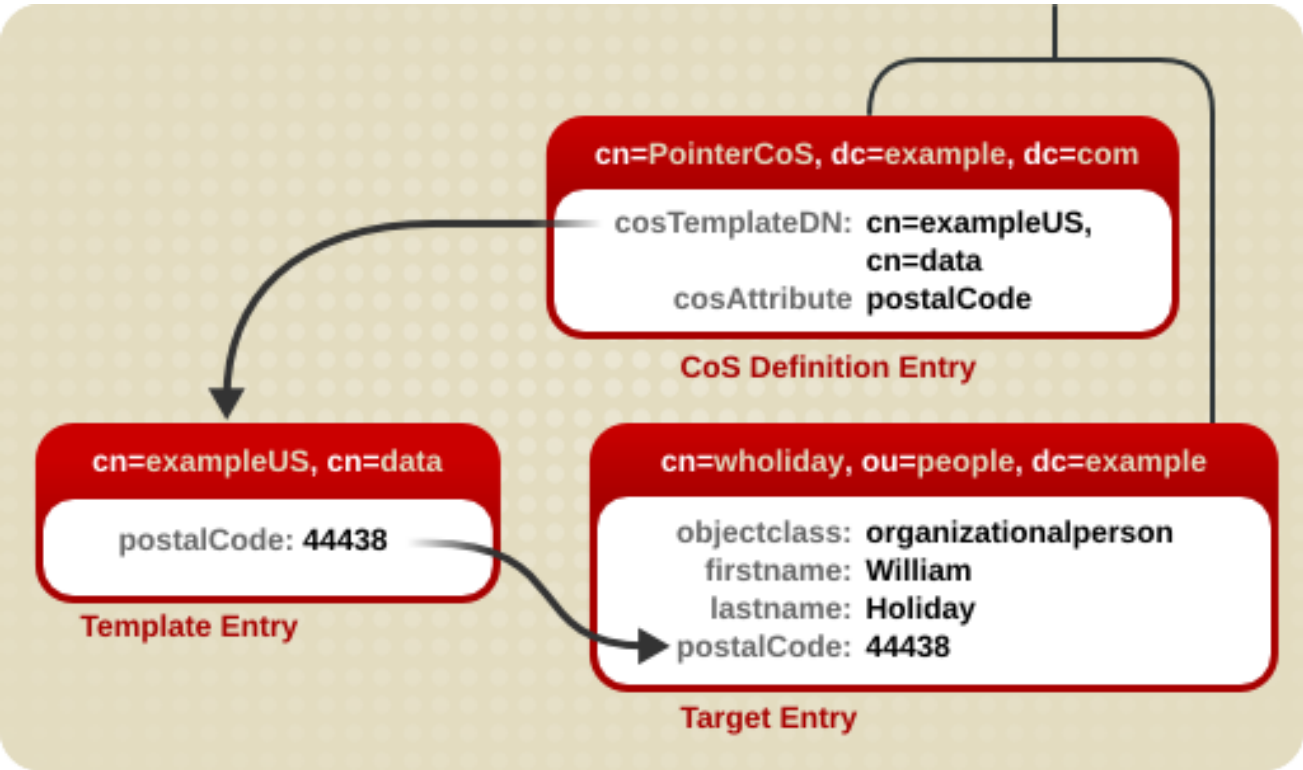
ロールと従来の CoS を併用することで、ロールベースの属性を提供することができます。これらの属性は、エントリーが特定のロールを持ち、関連する CoS テンプレートを持っているために表示されます。たとえば、ロールベースの属性を使用して、サーバーのルックスルー制限をロールごとに設定することができます。

5.3.1. Pointer CoS について

ポインター CoS は、テンプレート DN のみを使用してテンプレートエントリーを特定します。各ポインター CoS ごとに、1 つのテンプレート DN しか存在しない場合があります。ポインター CoS は、テンプレートエントリーの範囲内にあるすべてのエントリーに適用されます。

*たとえば、図5.1「Pointer CoS のサンプル」 のポインター CoS は、**dc=example,dc=com** に保存されているすべてのエントリーと共通の郵便番号を共有します。*

図5.1 Pointer CoS のサンプル

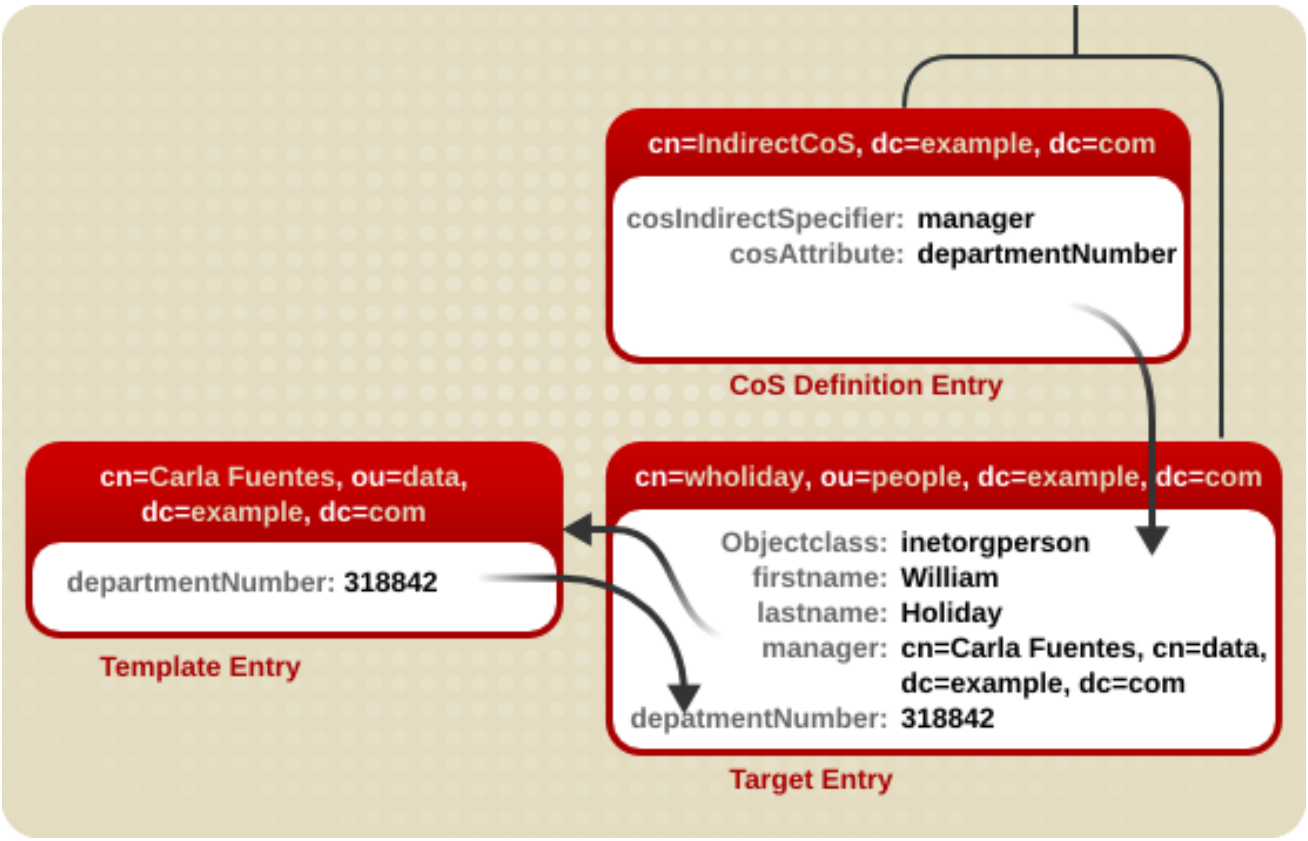


*テンプレートエントリーは、CoS 定義エントリーの DN **cn=exampleUS,cn=data** で識別されます。**postalCode** 属性がエントリー **cn=wholiday,ou=people,dc=example,dc=com** に対してクエリーされるたびに、Directory Server は、テンプレートエントリー **cn=exampleUS,cn=data** で使用可能な値を返します。*

5.3.2. 間接的な CoS について

間接的な CoS は、ターゲットエントリーの属性の 1 つを使用して、テンプレートエントリーを識別します。ターゲットエントリーの属性には、既存のエントリーの DN が含まれる必要があります。

図5.2 間接的な CoS の例

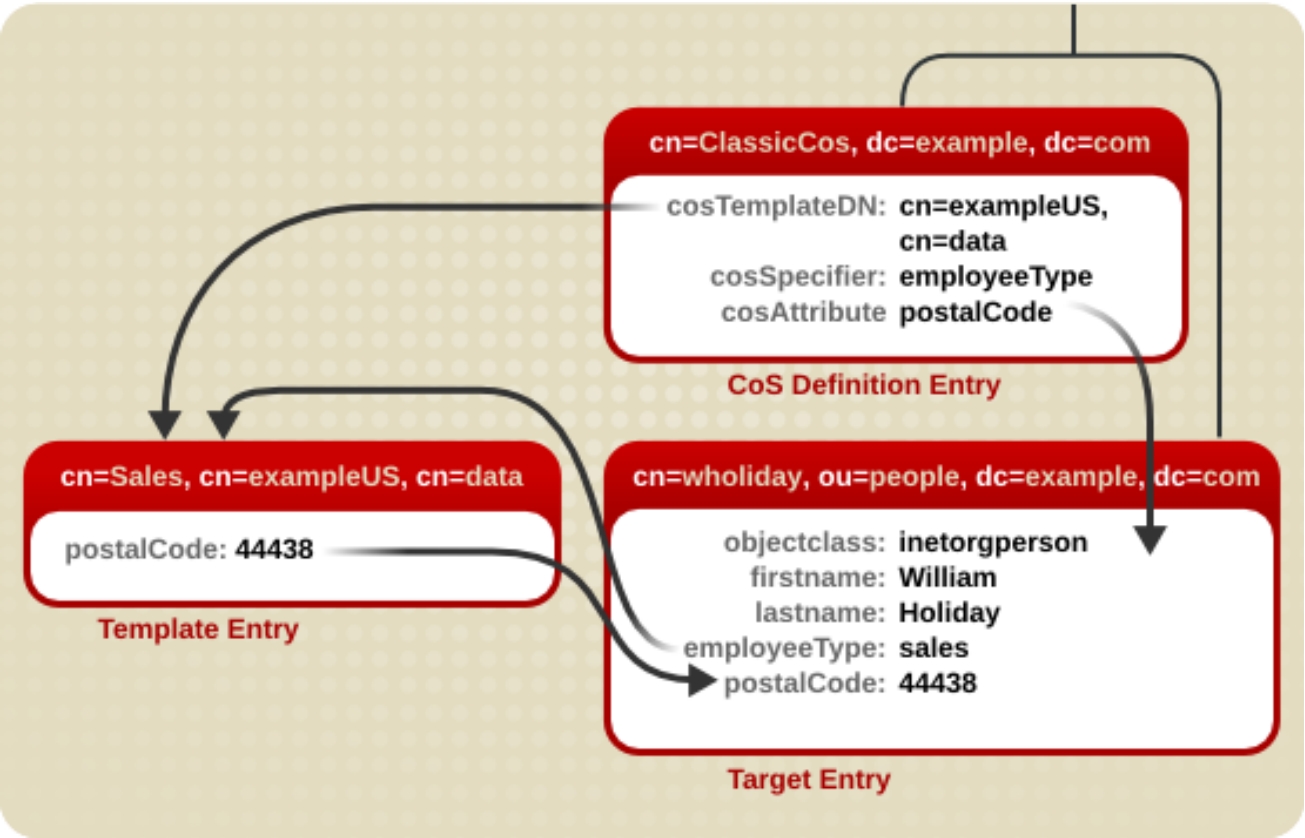


*この例では、Wiiam Holiday のターゲットエントリーには間接指定子 (**manager** 属性) が含まれます。William のマネージャーは Carla Fuentes であるため、**manager** 属性にはテンプレートエントリーの DN へのポインター **cn=Carla Fuentes,ou=people,dc=example,dc=com** が含まれます。テンプレートエントリーは次に、**318842** の **departmentNumber** 属性値を提供します。*

5.3.3. Classic CoS について

*従来の CoS は、DN とターゲットエントリーのいずれかの属性の値の両方で、**テンプレートエントリー**を特定します。Classic CoS には、他の CoS テンプレートに属していないエントリーに適用されるデフォルトの CoS テンプレートなどの複数のテンプレートエントリーを持つことができます。*

図5.3 Classic CoS のサンプル



*この例では、CoS 定義エントリーの **cosSpecifier** 属性は、**employeeType** 属性を指定しています。この属性は、テンプレート DN と組み合わせて、テンプレートエントリーを **cn=sales,cn=exampleUS,cn=data** として特定します。その後、テンプレートエントリーは、**postalCode** 属性値をターゲットエントリーに提供します。*

5.4. 管理エントリーについて

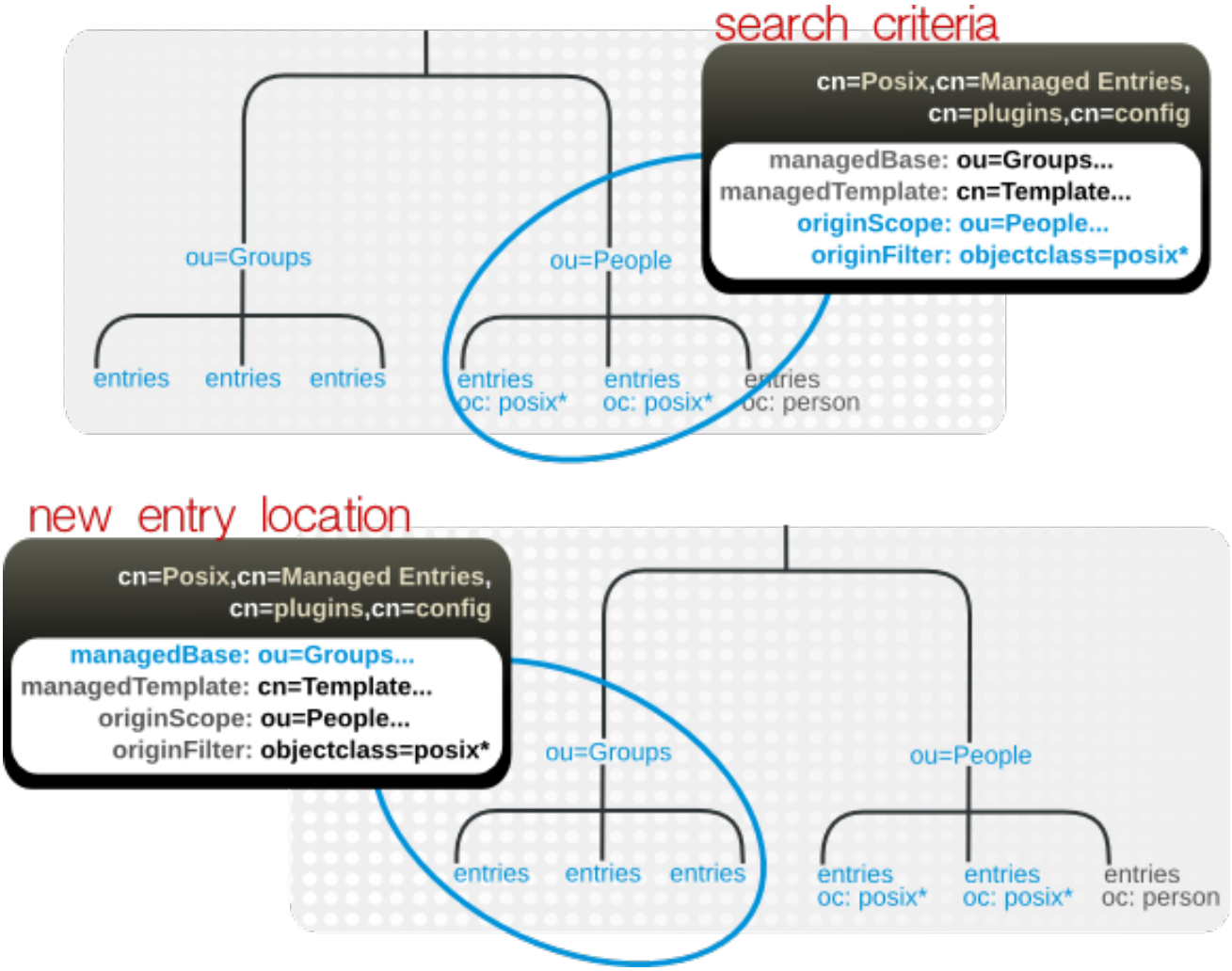
一部のクライアントおよび Red Hat Directory Server との統合には、2 つのエントリーが必要です。たとえば、Possix システムには、通常、各ユーザーにグループがあります。Directory Server の管理エントリープラグインは、適切な作成元のエントリーが作成されるたびに、属性の正確な値と特定の値で新しい管理エントリーを自動的に作成します。

*基本的な概念は、エントリー A が作成され、関連する属性値を持つエントリー B が自動的に作成されるべき状況があるということです。たとえば、Posix ユーザー (**posixAccount** エントリー) が作成されると、対応するグループエントリー (**posixGroup** エントリー) も作成する必要があります。管理エントリープラグインのインスタンスは、プラグインが、新しいエントリー (**管理エントリー**) を自動的に生成するエントリー (**作成元のエントリー**) を識別します。また、管理エントリー設定を定義する個別のテンプレートエントリーも指定します。*

管理エントリープラグインのインスタンスは、以下の 3 つを定義します。

- *(検索範囲と検索フィルターを使用する) 作成元のエントリーを識別する検索基準*
- *管理エントリーを作成するサブツリー (新しいエントリーの場所)*
- *管理エントリーに使用するテンプレートエントリー*

図5.4 管理エントリーの定義



以下に例を示します。

```
dn: cn=Posix User-Group,cn=Managed Entries,cn=plugins,cn=config
objectclass: extensibleObject
cn: Posix User-Group
originScope: ou=people,dc=example,dc=com
originFilter: objectclass=posixAccount
managedBase: ou=groups,dc=example,dc=com
managedTemplate: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
```

作成元のエントリーには、管理エントリーを作成するために特別な設定または設定は必要ありません。プラグインの範囲内に作成し、指定の検索フィルターと一致させる必要があります。

5.4.1. 管理エントリーのテンプレートの定義

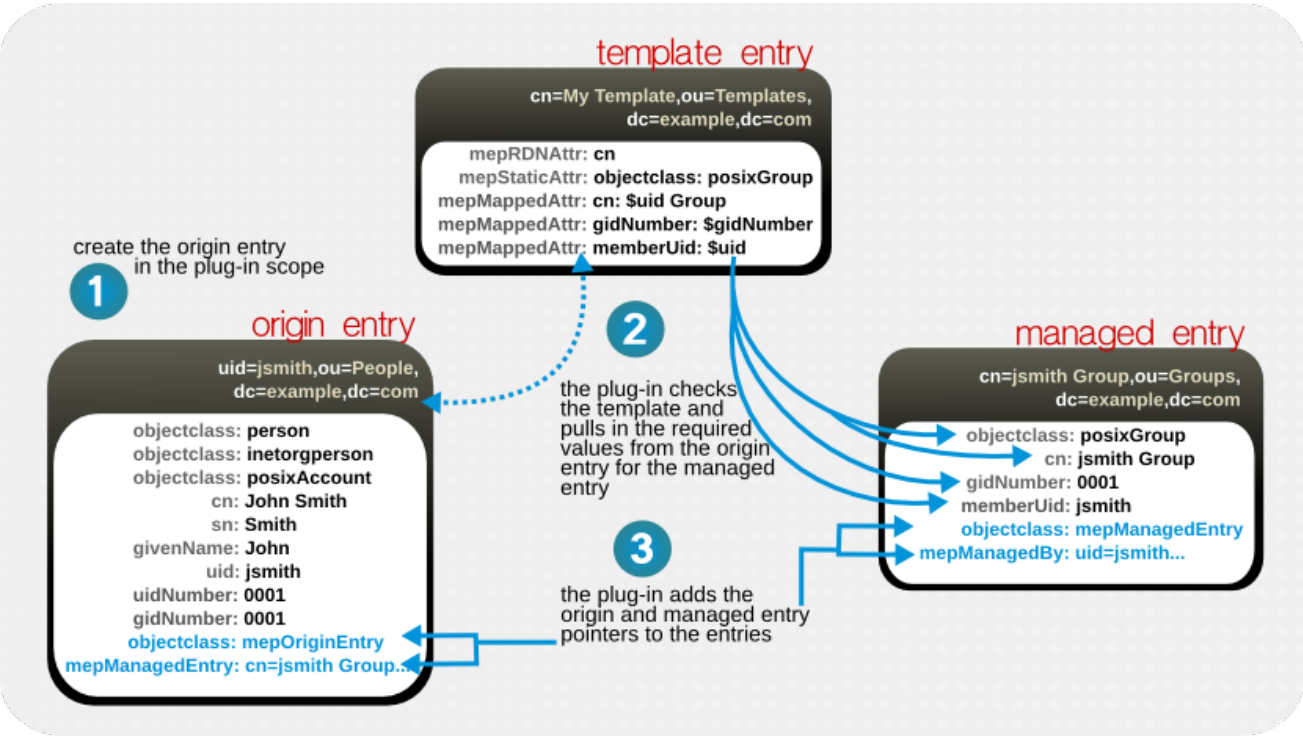
テンプレートエントリーは、静的属性 (事前定義した値を持つ) とマッピングされた属性 (元のエントリーから値をプルするマッピング属性) を使用して、管理エントリーの設定全体をレイアウトします。

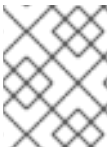
```
dn: cn=Posix User-Group Template,ou=Templates,dc=example,dc=com
objectclass: mepTemplateEntry
cn: Posix User-Group Template
mepRDNAttr: cn
mepStaticAttr: objectclass: posixGroup
```

```
mepMappedAttr: cn: $uid Group
mepMappedAttr: gidNumber: $gidNumber
mepMappedAttr: memberUid: $uid
```

テンプレートのマップされた属性は、先頭にドル記号 ($) を追加して作成元のエントリーから値をプルして管理エントリーで使用するトークンを使用します。

図5.5 管理エントリー、テンプレート、および作成元のエントリー





注記

静的属性およびマップされた属性に与えられた値が、必要な属性の構文に従っていることを確認してください。

5.4.2. 管理エントリープラグインにより書き込まれるエントリー属性

作成元のエントリーと管理エントリーの両方には、管理エントリープラグインのインスタンスによって管理されていることを示す特別な管理エントリー属性があります。作成元のエントリーでは、プラグインは関連付けられた管理エントリーへのリンクを追加します。

```
dn: uid=jsmith,ou=people,dc=example,dc=com
objectclass: mepOriginEntry
objectclass: posixAccount
...
sn: Smith
mail: jsmith@example.com
mepManagedEntry: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
```

管理エントリーでは、プラグインはテンプレートで定義された属性の他に、作成元のエントリーを参照する属性を追加します。

```
dn: cn=jsmith Posix Group,ou=groups,dc=example,dc=com
objectclass: mepManagedEntry
objectclass: posixGroup
```

```
...
mepManagedBy: uid=jsmith,ou=people,dc=example,dc=com
```

特別な属性を使用して、管理および作成元のエントリーを示すことで、関連するエントリーを簡単に特定し、管理エントリープラグインによる変更を評価できます。

5.4.3. 管理エントリープラグインおよびDirectory Server 操作

管理エントリープラグインは、Directory Server が追加や削除などの一般的な操作を実行する方法に多少の影響を与えます。

- **追加**。*すべての追加操作では、サーバーは新しいエントリーが管理エントリープラグインインスタンスの範囲内にあるかどうかを確認します。作成元エントリーの基準を満たすと、管理エントリーが作成され、管理エントリー関連の属性が元のエントリーと管理エントリーの両方に追加されます。*

- **修正**。*作成元のエントリーが変更されると、プラグインをトリガーして管理エントリーを更新します。*

  *ただし、**テンプレート** エントリーを変更しても、自動的に管理エントリーを更新しません。テンプレートエントリーへの変更は、次に作成元エントリーが変更するまで、管理エントリーに反映されません。*

  *管理エントリー**内** でマップされた管理属性は、管理エントリープラグインでのみ手動で変更することができません。マネージドエントリーの他の属性 (管理エントリープラグインによって追加された静的属性を含む) は手動で変更できます。*

- **削除**。*作成元のエントリーが削除されると、管理エントリープラグインもそのエントリーに関連付けられた管理エントリーを削除します。*

  削除できるエントリーにはいくつかの制限があります。

  - *テンプレートエントリーは、プラグインインスタンス定義で現在参照されている場合は削除できません。*
  - *管理エントリーは、管理エントリープラグイン以外では削除できません。*

- **名前の変更**。*元のエントリーの名前を変更した場合は、プラグインは対応する管理エントリーを更新します。エントリーがプラグインスコープの **外** に移動すると、管理エントリーが削除されますが、エントリーがプラグインスコープの **中** に移動した場合は、追加操作のように処理され、新しい管理エントリーが作成されます。*

  削除操作と同様に、名前変更または移動できるエントリーに制限があります。

  - *設定定義エントリーは、コンテナーエントリーの管理エントリープラグインを外に移動できません。エントリーが削除されると、そのプラグインインスタンスが非アクティブになります。*
  - *エントリーが管理エントリープラグインのコンテナーエントリー **内** に移動する場合、これは検証され、アクティブな設定定義として処理されます。*
  - *テンプレートエントリーの名前を変更したり、プラグインインスタンス定義で現在参照している場合は移動したりすることはできません。*
  - *管理エントリープラグイン以外が、管理エントリーの名前を変更したり、移動したりできません。*

- **レプリケーション。***管理エントリープラグイン操作は、***レプリケーションの更新によって開始しません。***プラグインスコープのエントリーの追加または修正操作が別のレプリカに複製される場合、その操作はレプリカで管理エントリープラグインインスタンスを発生させず、エントリーを作成または更新しません。管理エントリーの更新を複製する唯一の方法は、最終管理エントリーをレプリカに複製することです。*

5.5. リンク属性の概要

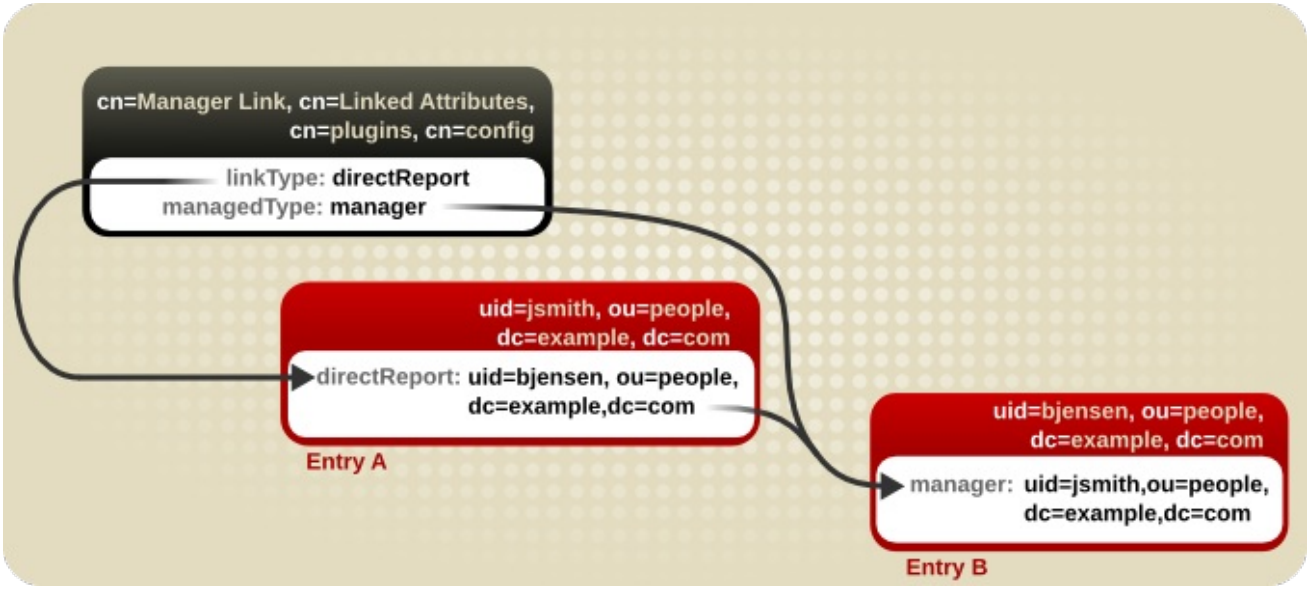
サービスクラスは、住所、郵便番号、主なオフィス番号など、すべてが **同じ値** *の属性を持つエントリーに対して、属性値を動的に提供します。これらは共有属性値であり、単一のテンプレートエントリーで更新されます。*

多くの場合、エントリー間には、それらの間のリンクを表現する方法が必要な関係がありますが、その関係を表現する値 (および場合によっては属性) は異なります。Red Hat Directory Server は、指定された属性を繋ぎ合わせる方法を提供するため、1 つのエントリーの 1 つの属性が変更されると、関連するエントリーの対応する属性が自動的に更新されます。最初の属性には、更新するエントリーを参照する DN 値があります。2 番目のエントリー属性には、1 番目のエントリーへのバックポイントである DN 値もあります。

たとえば、グループエントリーは ***member*** *などの属性のメンバーを一覧表示します。ユーザーがどのグループに所属するかをユーザーエントリーに表示することは自然です。これは* ***memberOf*** *属性で設定されます。****memberOf*** *属性は、MemberOf プラグインによる* **管理** *属性です。プラグインは、それぞれのメンバー属性の変更についてすべてのグループエントリーをポーリングします。グループメンバーがグループから追加または削除されるたびに、対応するユーザーエントリーが* ***memberOf*** *属性を変更して更新されます。これにより、****member*** *（およびその他のメンバー属性）および* ***memberOf*** *属性が* **リンクされ** *ます。*

*MemberOf プラグインは、(単一の) グループメンバー属性専用の単一のインスタンスに限定されています (ネストしたグループの扱いなど、グループに特有の他の動作もあります)。もう 1 つのプラグインであるリンク先属性プラグインは、プラグインの複数のインスタンスを許可します。各インスタンスは、管理者によって手動で維持される 1 つの属性 (****linkType****) およびプラグインによって自動的に維持される 1 つの属性 (****managedType****) を設定します。*

図5.6 リンク先属性の基本設定

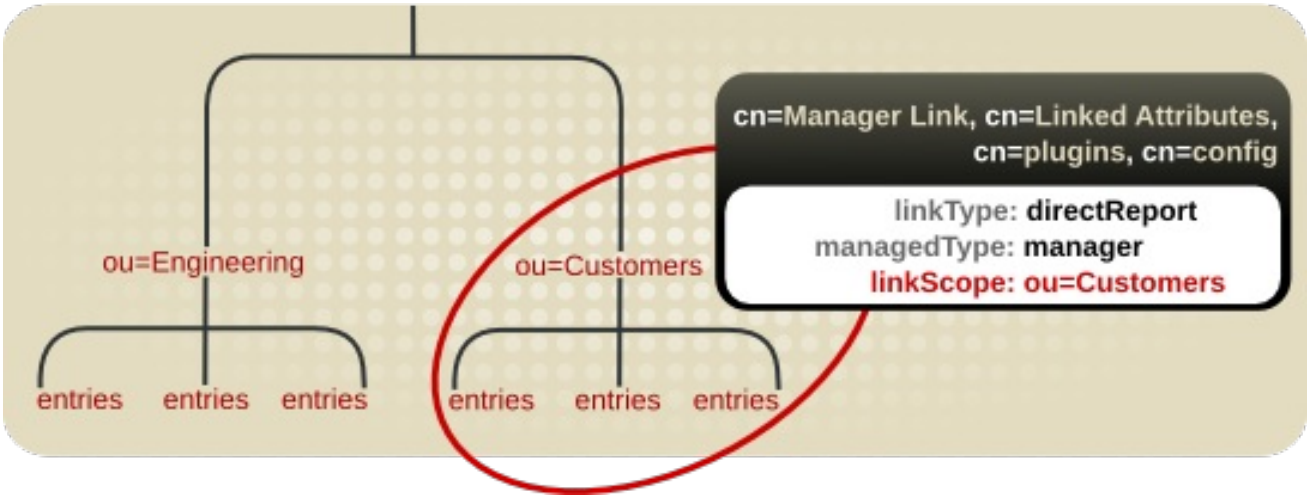


注記

データの一貫性を維持するには、プラグインプロセスのみが管理属性を維持する必要があります。管理属性へのすべての書き込みアクセスを制限するACIの作成を検討してください。

リンク先属性プラグインインスタンスは、ディレクトリー内の単一のサブツリーに制限できます。これにより、属性の組み合わせと影響を受けるエントリーのより柔軟なカスタマイズが可能になります。スコープが設定されていない場合、プラグインはディレクトリー全体で動作します。

図5.7 リンク先属性プラグインを特定のサブツリーに制限

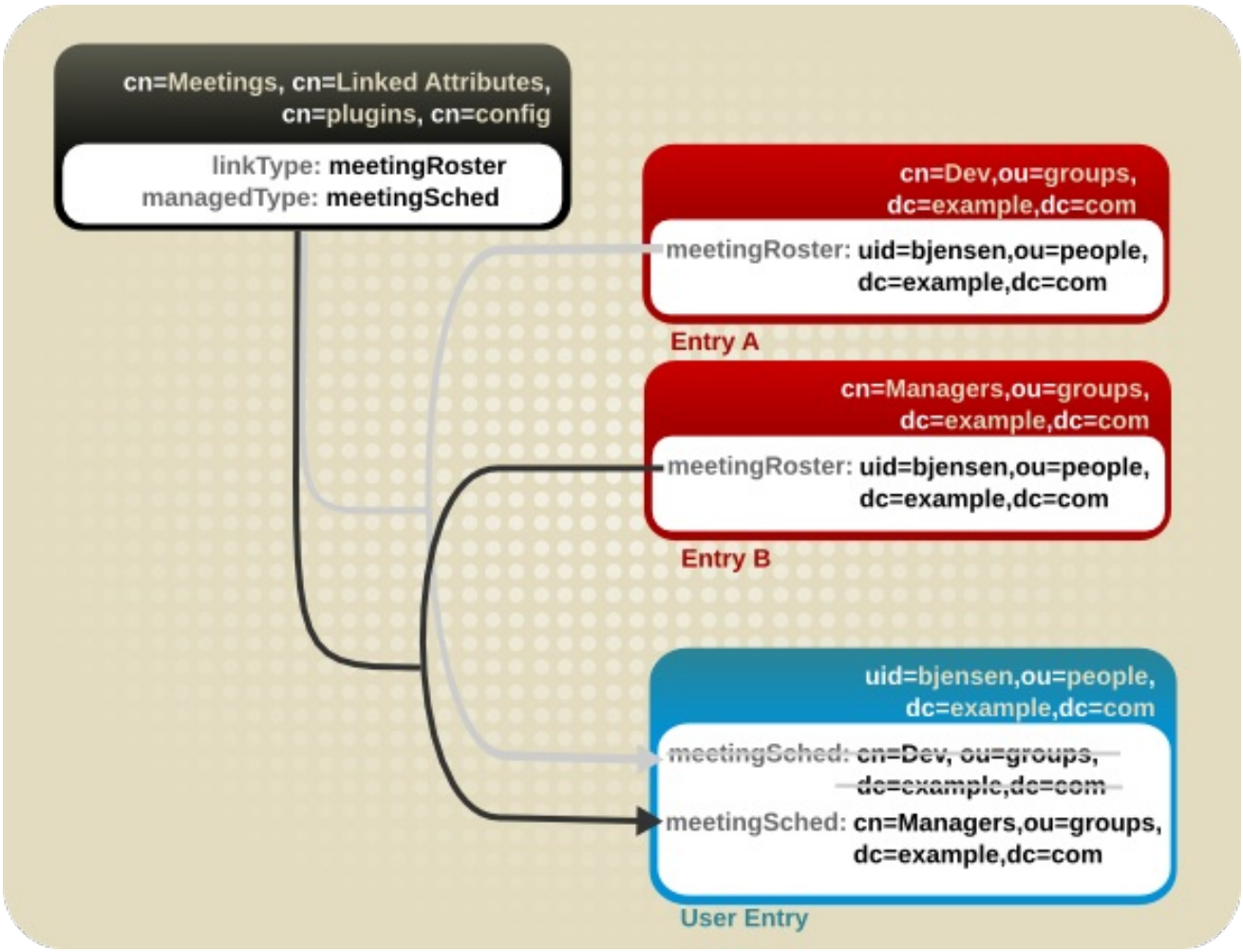


5.5.1. リンク属性のスキーマ要件

管理属性とリンク先属性の両方で、属性定義で識別名の構文が必要です。プラグインは、リンク先属性からDNをプルしてメンテナンスするエントリーを特定し、元のエントリーDNをマネージド属性の値として自動的に割り当てます。つまり、どちらの属性も値としてDNを使用しなければならないことを意味します。

管理属性は多値である必要があります。ユーザーは、複数のグループのメンバーであったり、複数のドキュメントの作成者であったり、複数のsee also参照エントリーを持っていたりします。管理属性が単一値である場合は、値は正しく更新されません。標準的な要素の多くは多値であるため、デフォルトのスキーマではあまり問題になりません。ただし、カスタムスキーマを使用する場合には、検討することが特に重要となります。

図5.8 誤った使用方法: 単一値のリンク先属性の使用

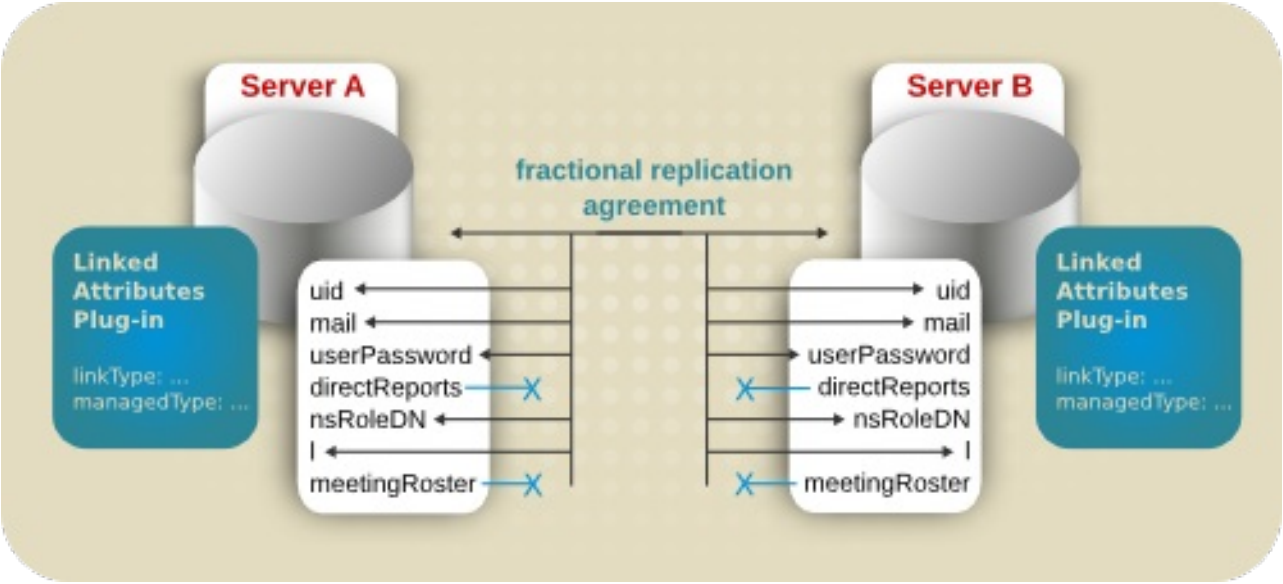


5.5.2. レプリケーションでのリンク先属性の使用

単純なレプリケーションシナリオ (supplier-consumer) では、コンシューマーに書き込みが行われないため、プラグインはサプライヤーにのみ存在する必要があります。

マルチサプライヤーのレプリケーションでは、各サプライヤーには独自のプラグインインスタンスがあり、すべて設定が同一で、管理属性は一部レプリケーションを使用してレプリケーションから除外する必要があります。

図5.9 リンク先属性とレプリケーション

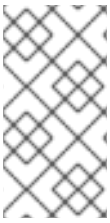


あるサプライヤーに加えられた変更は、プラグインが自動的に発生し、対応するディレクトリーエントリーの値を管理するため、データはサーバー全体で一貫性を維持します。ただし、リンクされたエントリー間でデータの一貫性を保つには、プラグインインスタンスで管理属性を維持する必要があります。つまり、管理属性値は、マルチサプライヤーレプリケーション環境であっても、レプリケーションプロセスではなく、プラグインプロセスによってのみ維持される必要があります。

5.6. 一意の数値を動的に割り当てる方法について

*エントリー属性によっては、**uidNumber** や **gidNumber** などの一意の番号が必要です。Directory Server は、DNA (Distributed Numeric Assignment) プラグインを使用して、指定された属性に一意の番号を自動的に生成して提供できます。*

多くの状況では、UID/GID 番号や PIN 番号などの一意の数値の属性が必要になります。このサーバーは DNA プラグインインスタンスを使用して、数字を生成する属性を指定します。これにより、その属性がエントリーに追加されるたびに、一意の値を割り当てることができます。



注記

属性の一意性 *は、DNA プラグインで維持されるとは限りません。プラグインは、重複しない範囲のみを割り当てますが、管理属性に手動で数字を割り当てることができ、手動で割り当てられた番号が一意であることを検証したり要求したりすることはありません。*

5.6.1. Directory Server の一意の番号の管理方法

一意の番号を割り当てる際に問題となるのは、番号を生成することではなく、エントリーが複製されたときに他の割り当てられた番号と競合しないように、また、すべてのサーバーが割り当てにおいて十分な範囲の番号を持つように、番号を効果的に管理することです。

サーバーの DNA プラグインは、インスタンスが発行することのできる利用可能な番号の範囲を割り当てます。範囲の定義は非常にシンプルで、サーバーの次に利用可能な番号 (範囲の下限) と最大値 (範囲の最後) の 2 つの属性で設定されます。初期の下限範囲は、プラグインインスタンスを設定する際に設定されます。その後、下部の値はプラグインによって更新されます。利用可能な番号を範囲に分割すると、サーバーはすべて、互いに重複せずに、継続的に番号を割り当てることができます。

サーバーは、内部的にソートされた検索を実行し、次に指定された範囲がすでに取得されているかどうかを確認し、管理属性に適切な順序のマッチングルールと同じインデックスを割り当てる必要があります。

マルチサプライヤーのレプリケーションでは、各サプライヤーにしきい値を設定できるため、その範囲内の数字を使い果たしたら、他のサプライヤーから追加の範囲を要求できます。各サプライヤーは、別の設定エントリーで現在の範囲を追跡します。設定エントリーは他のすべてのサプライヤーに複製されるため、各サプライヤーが設定を確認して、新しい範囲で問い合わせるサーバーを見つけることができます。

個々のサーバーに設定した範囲と範囲設定エントリーは、Directory Server がエントリーに対して効率的に番号を **分散** する方法になります。

DNA プラグインは、1 つの属性タイプに、または一意の番号の 1 つの範囲から複数の属性タイプにまたがって、一意の番号を割り当てることができます。

これにより、属性に一意の数字を割り当てるためのオプションが複数提供されます。

- 一意の番号の 1 つの範囲から、1 つの属性タイプに割り当てられた 1 つの番号。
- 1 つのエントリーの 2 つの属性に割り当てられた同じ一意の番号。
- 2 つの異なる属性は、同じ範囲の一意の数字から 2 つの異なる数字を割り当てていました。

多くの場合は、属性タイプごとに一意の番号を割り当てるだけで十分です。新しい従業員エントリーに **employeeID** を割り当てる際には、各従業員エントリーに一意の **employeeID** が割り当てられます。

ただし、同じ範囲の数字から複数の属性に一意の番号を割り当てることが役に立つ場合もあります。たとえば、**uidNumber** と **gidNumber** を **posixAccount** エントリーに割り当てる場合、両方の属性に同じ数を割り当てるように DNA プラグインを設定できます。

DNA プラグインは、常にディレクトリーツリーの特定領域 (**スコープ**) と、そのサブツリー内の特定のエントリータイプ (**フィルター**) に適用されます。

よくあるのは、まったく別のユーザーがディレクトリーツリーの異なるブランチに保存されていることです。たとえば、ホストサービスは、**ou=Example Corp.** ブランチに 1 つのクライアントのユーザーがあり、**ou=Acme Company** ブランチに別のクライアントのユーザーが存在する場合があります。この場合、割り当てられた番号は **サブツリー内** で一意である必要がありますが、ディレクトリー全体で一意である必要はありません。この場合、**ou=Example Corp.** ブランチの Barbara Jensen に、herエントリーに **uidNumber:5** が、**ou=Acme Company** ブランチに **uidNumber:5** が含まれるのは、これらは別々の組織であるためです。特定のサブツリーへの範囲の適用は、**dnaScope: ou=people,dc=example,dc=com** のように DNA スコープに設定されます。

一意の番号は、接頭辞を使用して異なる種類のユーザーエントリーを特定することで、範囲間で区別することもできます。たとえば、DNA 接頭辞が **acme** に設定されている場合、Acme Company ブランチの一意の番号には、**uid: acme 5 のように、番号の前に acme** があります。

5.6.2. DNA を使用した値の属性への割り当て

Directory Server が属性値の生成を処理する方法が複数あります。

最も単純なケースでは、属性のない unique-number 属性を必要とするオブジェクトクラスのディレクトリーにユーザーエントリーが追加されます。値を持たない管理属性を追加する (または必要とする) と、DNA プラグインによる値の割り当てが発生します。エントリーが追加されると、プラグインは、プラグインに設定されたスコープおよびフィルターに基づいて、エントリーが定義された範囲に一致するか

どうかを確認します。エントリーが範囲と一致し、その範囲に対してDNA が管理している属性が、追加されるエントリーにない場合は、DNA プラグインは次の値を割り当てます。このオプションは、一意の値を1 つの属性に割り当てるようにDNA プラグインが設定されている場合に限り機能します。

*たとえば、**posixAccount** オブジェクトクラスには **uidNumber** 属性が必要です。**uidNumber** 属性がDNA プラグインで管理されており、フィルターの範囲内で **uidNumber** 属性なしでユーザーエントリーが追加されると、サーバーは新しいエントリーをチェックし、マネージドの **uidNumber** 属性が必要であることを確認し、自動的に割り当てられた値で属性を追加します。*

```
ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: posixAccount
uid: jsmith
cn: John Smith
....
```

プラグインは不足している属性を処理し、次に利用可能な番号をサーバーに要求し、エントリーの値を提供します

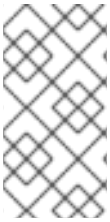
*同様の管理可能なオプションは、**マジック番号** を使用することです。このマジック番号は、マネージド属性のテンプレート値であり、数字または単語などのサーバーの範囲外のもので、プラグインが新しい割り当て値に置き換える必要があると認識しているものです。その番号でエントリーが追加され、エントリーが設定されたDNA プラグインの範囲およびフィルター内にある場合は、マジック番号を使用することで、プラグインが自動的にトリガーされ、新しい値を生成します。*

*DNA プラグインが **uidNumber** と **gidNumber** の両方に同じ一意の番号を **posixAccount** エントリーに割り当てるように設定されていると、DNA プラグインは同じ番号を両方の属性に割り当てます。これを行うには、マジック番号を指定して、変更操作に両方の管理属性を渡します。以下に例を示します。*

```
ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: posixAccount
uid: jsmith
cn: John Smith
uidNumber: magic
gidNumber: magic
....
```

マジック番号は、LDIF からエントリーをインポートする場合や、複数の異なる属性に対して一意の番号を生成するためにDNA プラグインをトリガーする際に、非常に便利です。

DNA プラグインは、新規の一意の値のみを生成します。DNA プラグインが制御する属性に特定の値を使用するためにエントリーを追加または変更した場合には、指定した番号が使用されます。DNA プラグインは、その番号を上書きしません。



注記

属性の一意性 *は、DNA プラグインで維持されるとは限りません。プラグインは、重複しない範囲のみを割り当てますが、管理属性に手動で数字を割り当てることができ、手動で割り当てられた番号が一意であることを検証したり要求したりすることはありません。*

5.6.3. レプリケーションでの DNA プラグインの使用

マルチサプライヤーのレプリケーションでは、サーバーが参照するエントリーが 2 つあります。

- *DNA プラグインのマネージド範囲*
- *サーバーで利用可能な範囲に関する情報を保存する共有設定エントリー*

プラグインインスタンスが作成されると、DNA プラグインによって、サプライヤー設定のある共有設定エントリーの下にエントリーが自動的に作成されます。以下に例を示します。

```
dn: dnaHostname=ldap1.example.com+dnaPortNum=389,cn=Account
UIDs,ou=Ranges,dc=example,dc=com
objectClass: extensibleObject
objectClass: top
dnahostname: ldap1.example.com
dnaPortNum: 389
dnaSecurePortNum: 636
dnaRemainingValues: 1000
```

サーバーに新しい番号の範囲が必要な場合は、コンテナーエントリーの下にある設定エントリーを検索します。利用可能な範囲が最も大きいサーバーを見つけると、その範囲の一部を割り当ててもらうために拡張操作要求を送信します。2 番目のサーバーが同意すると、2 番目のサーバーが要求サーバーへ新しい範囲の割り当てを送信します。

第6章 ディレクトリートポロジーの設計

4章ディレクトリーツリーの設計 では、ディレクトリーサービスがエントリーを保存する方法について説明しています。Red Hat Directory Server は多数のエントリーを保存できるため、ディレクトリーエントリーを複数のサーバーに分散することが可能です。ディレクトリーのトポロジーでは、ディレクトリーツリーを複数の物理 Directory Server に分割する方法と、これらのサーバーが相互にリンクする方法を説明します。

本章では、ディレクトリーサービスのトポロジーのプランニングを説明します。

6.1. トポロジーの概要

Directory Server は、**分散ディレクトリー** をサポートできます。この (4章ディレクトリーツリーの設計 で設計された) ディレクトリーツリーは、複数の物理 Directory Server にまたがって分散されます。これらのサーバーにまたがってディレクトリーを分割する方法は、以下を実現できます。

- ディレクトリー対応アプリケーション向けに最適なパフォーマンスを実現します。
- ディレクトリーサービスの可用性を増やします。
- ディレクトリーサービスの管理を改善します。

データベースは、レプリケーション、バックアップの実行、およびデータの復元などのジョブの基本的な単位です。1 つのディレクトリーは、管理しやすいように分割し、個別のデータベースに割り当てることができます。これらのデータベースは、複数のサーバー間で分散でき、各サーバーのワークロードを減らします。1 台のサーバーに複数のデータベースを配置することができます。たとえば、1 台のサーバーには 3 つの異なるデータベースが含まれる場合があります。

ディレクトリーツリーが複数のデータベースに分割されている場合、各データベースには **接尾辞** と呼ばれるディレクトリーツリーの一部が含まれます。たとえば、1 つのデータベースを使用して、ディレクトリーツリーの **ou=people,dc=example,dc=com** 接尾辞またはブランチにエントリーのみを保存することができます。

ディレクトリーを複数のサーバーに分割した場合、各サーバーはディレクトリーツリーの一部のみを担当します。分散ディレクトリーサービスは、DNS namespace の各部分を特定の DNS サーバーに割り当てる Domain Name Service (DNS) と同様に機能します。同様に、クライアントから見れば 1 つのディレクトリーツリーのように見えるディレクトリーサービスを維持しつつ、ディレクトリーの namespace をサーバー全体で分散することができます。

Directory Server は、さまざまなデータベースに保存されているディレクトリーデータをリンクするためのメカニズムである **ナレッジ参照** も提供します。Directory Server には、**参照** および **チェーン** の 2 種類のナレッジ参照が含まれます。

本章の残りの部分では、データベースとナレッジ参照について、2 種類のナレッジ参照の違いについて、そしてデータベースのパフォーマンスを向上させるためのインデックスの設計方法について説明します。

6.2. ディレクトリーデータの配布

データを分散させることで、企業の各サーバーにディレクトリーエントリーを物理的に格納せずに、複数のサーバーにまたがってディレクトリーサービスをスケーリングすることができます。したがって、分散ディレクトリーでは、1 台のサーバーよりもはるかに多くのエントリーを保持することができます。

また、ディレクトリーサービスを設定することで、ユーザーから配信内容の詳細を隠すことができます。ユーザーとアプリケーションに関する限り、ディレクトリーのクエリーに答えるディレクトリーは 1 つしかありません。

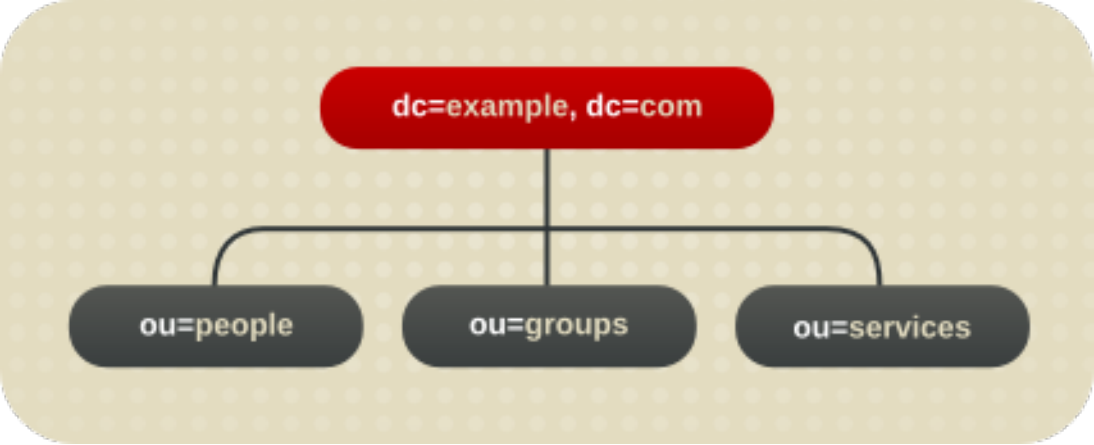
以下のセクションでは、データ配布の仕組みについて詳細に説明します。

- 「複数のデータベースの使用について」
- 「接尾辞について」

6.2.1. 複数のデータベースの使用について

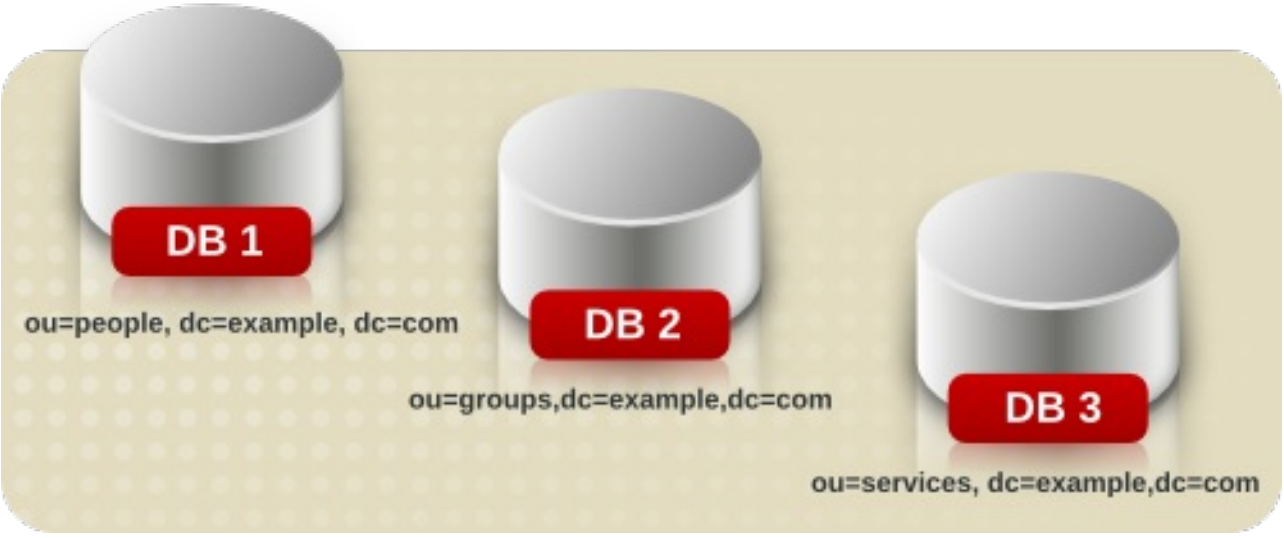
Directory Server は、LDBM データベースにデータを保存します。これは、高性能なディスクベースのデータベースです。各データベースは、割り当てられたすべてのデータが含まれる大規模なファイルのセットで設定されます。

ディレクトリーツリーの異なる部分は、異なるデータベースに格納できます。



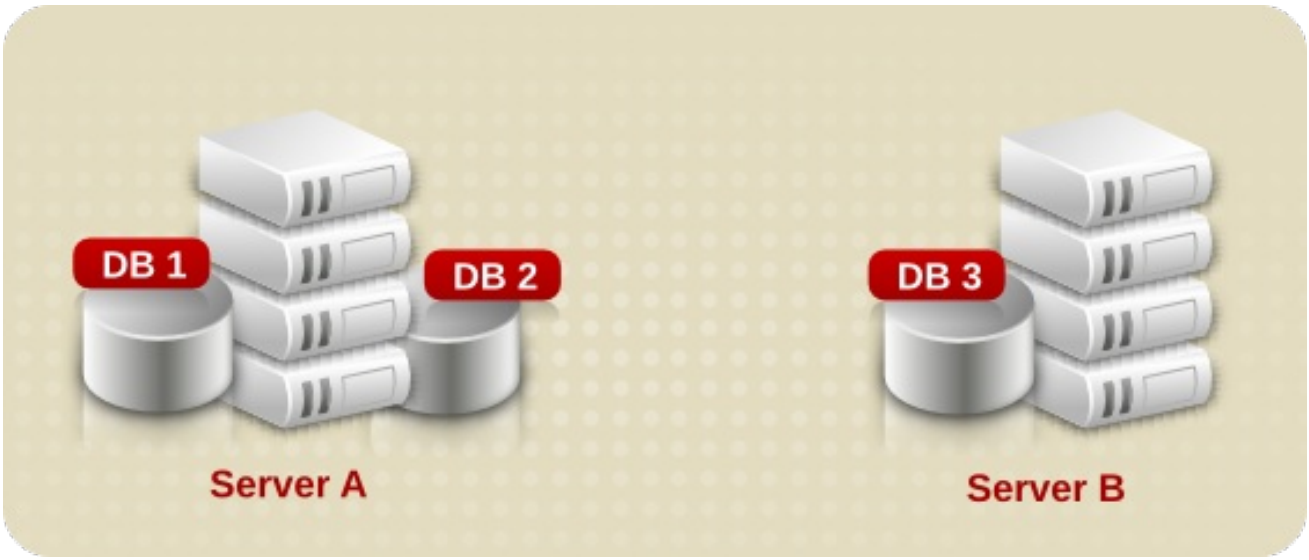
たとえば、図6.1「別のデータベースでの接尾辞データの保存」 は、3 つの別個のデータベースに 3 つの接尾辞が保存されていることを示しています。

図6.1 別のデータベースでの接尾辞データの保存



ディレクトリーツリーを複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させることができます。たとえば、ディレクトリーツリーの 3 つの接尾辞を含むために、DB1、DB2、および DB3 の 3 つのデータベースがある場合、これらはサーバー A およびサーバー B の 2 つのサーバーに保存できます。

図6.2 個別のサーバー間での接尾辞データベースの分割



サーバーA にはDB1 とDB2 が含まれ、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させることで、各サーバーの負荷を軽減することができます。そのため、ディレクトリーサービスは、1 台のサーバーで実現するよりもはるかに多くのエントリー数に対応することができます。

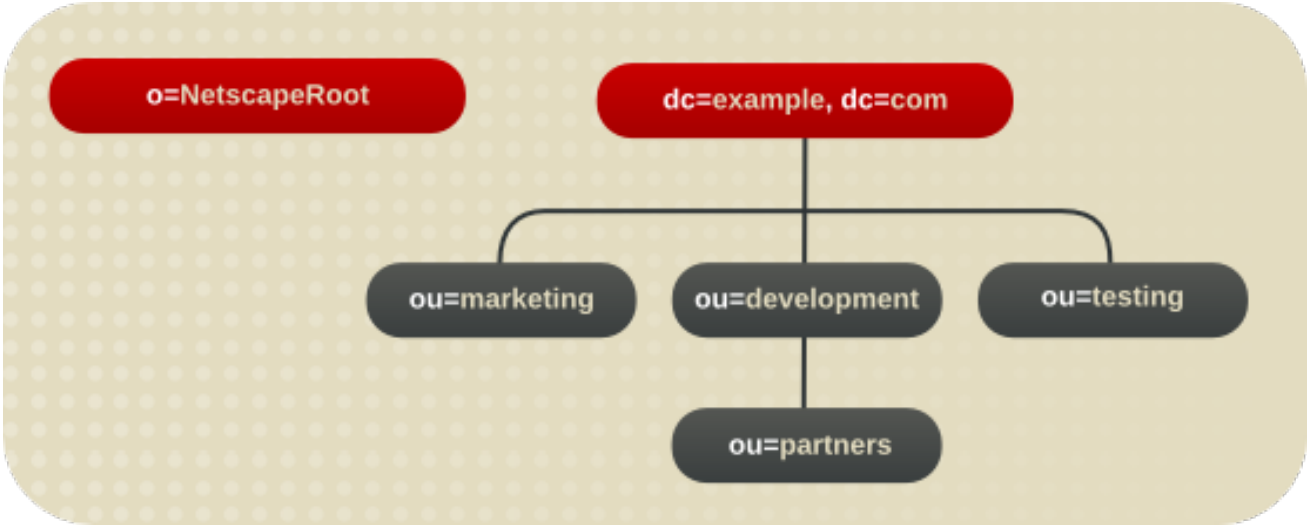
さらに、Directory Server は、データベースを動的に追加することをサポートします。つまり、ディレクトリーサービス全体をオフラインにすることなく、ディレクトリーサービスが必要とするときに新しいデータベースを追加することができます。

6.2.2. 接尾辞について

各データベースには、Directory Server の特定の接尾辞内にデータが含まれます。ディレクトリーツリーのコンテンツを整理するために、root およびサブ接尾辞の両方を作成できます。root 接尾辞は、ツリーの最上部のエントリーです。これは、ディレクトリーツリーのroot だったり、Directory Server 向けに設計された大きなツリーの一部だったりします。サブ接尾辞は、root 接尾辞の下にあるブランチです。root およびサブ接尾辞のデータは、データベースに含まれています。

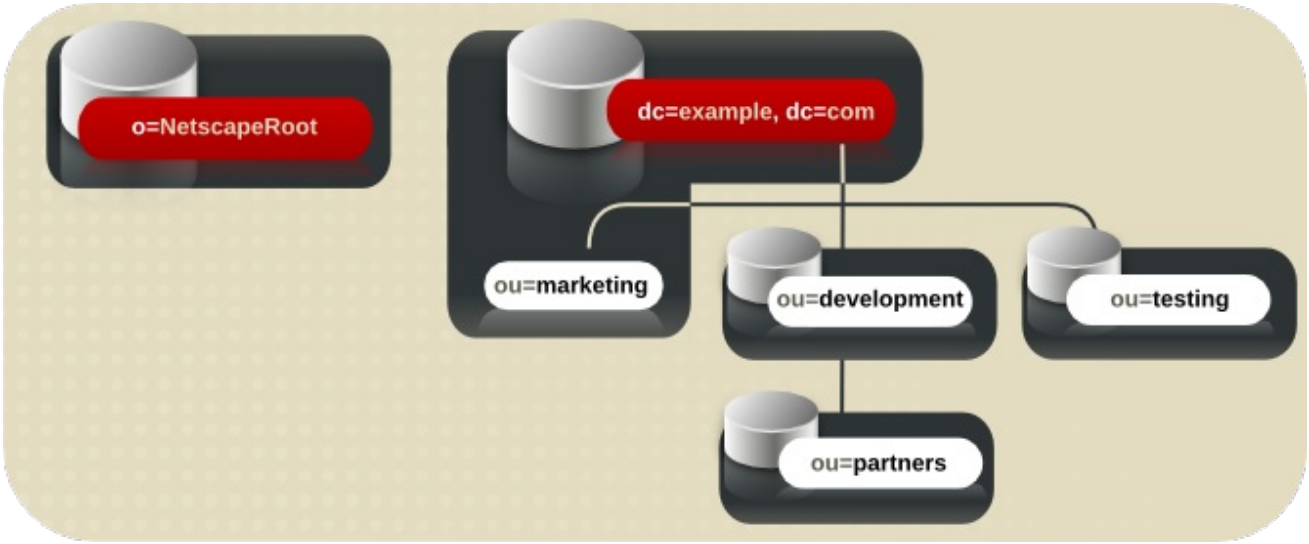
たとえば、Example Corp. は、ディレクトリーデータの配分を表す接尾辞を作成します。

図6.3 Example Corp. のディレクトリーツリー



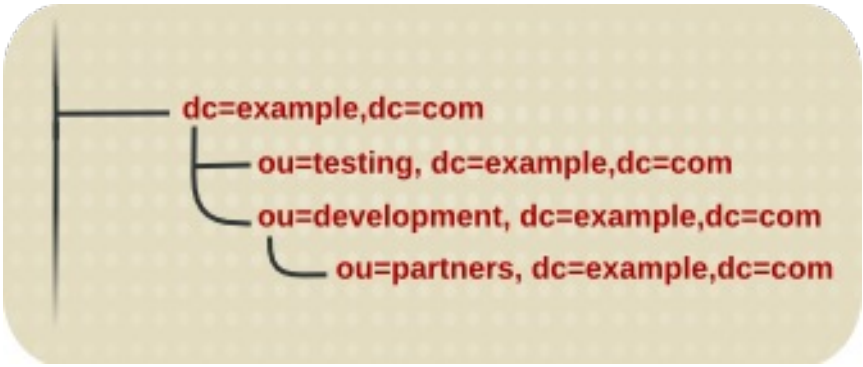
Example Corp. は、図6.4 「複数のデータベースにまたがるディレクトリーツリー」 のように、ディレクトリーツリーを 5 つの異なるデータベースに分散できます。

図6.4 複数のデータベースにまたがるディレクトリーツリー



その結果、接尾辞には以下のエントリーが含まれます。

図6.5 分散ディレクトリーツリーの接尾辞

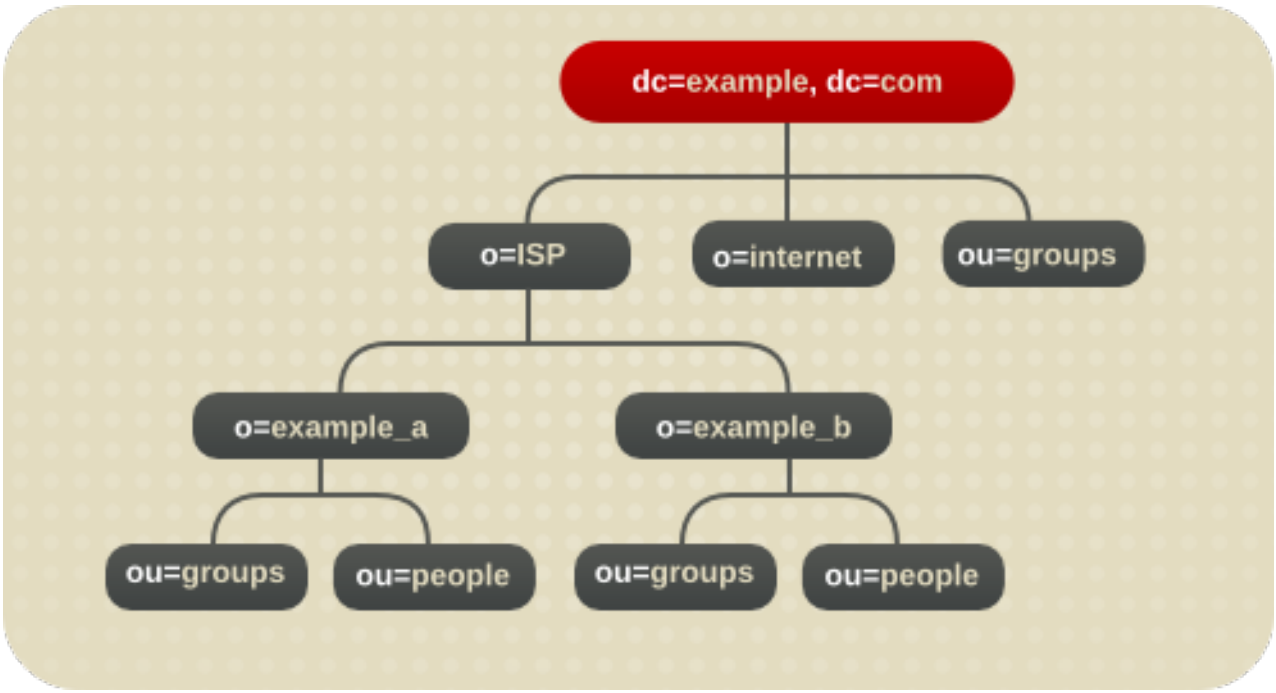


***dc=example,dc=com** 接尾辞は root 接尾辞です。**ou=testing, dc=example,dc=com** 接尾辞、**ou=development,dc=example,dc=com** 接尾辞、および **ou=partners,ou=development,dc=example,dc=com** 接尾辞はすべて、dc=example,dc=com ルート接尾辞のサブ接尾辞です。ルート接尾辞 **dc=example,dc=com** には、元のディレクトリーツリーの **ou=marketing** ブランチのデータが含まれます。*

複数の root 接尾辞の使用

*ディレクトリーサービスには複数の root 接尾辞が含まれる場合があります。たとえば、Example と呼ばれる ISP は、example_a.com 用と example_b.com 用など、複数の Web サイトをホストする可能性があります。ISP は、**o=example_a.com** 命名コンテキストに対応するものと、**o=example_b.com** 命名コンテキストに対応するものの 2 つのルート接尾辞を作成します。*

図6.6 複数の root 接尾辞を含むディレクトリーツリー



dc=example,dc=com *エントリーは root 接尾辞を表します。ホストされる各顧客のエントリーは root 接尾辞でもあります(**o=example_a** および **o=example_b**)。**ou=people** および **ou=groups** ブランチは、各ルート接尾辞の下にサブ接尾辞です。*

6.3. ナレッジ参照について

*複数のデータベースにデータを分散させた後、**ナレッジ参照**、さまざまなデータベースに保持されているディレクトリー情報へのポインターを使用して、分散データ間の関係を定義します。Directory Server には、分散データを 1 つのディレクトリーツリーにリンクできるように、以下のタイプのナレッジ参照を提供します。*

- *参照: サーバーは、クライアントアプリケーションが要求を満たすために別のサーバーに問い合わせ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。*
- *チェーン: サーバーは、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問い合わせ、操作の完了時にクライアントアプリケーションに結果をまとめて返します。*

以下のセクションでは、これら 2 種類のナレッジ参照を詳細に説明し、比較します。

6.3.1. 参照の使用

***参照** とは、サーバーから返される情報のことで、クライアントアプリケーションに対して、操作リクエストを続行するにはどのサーバーに問い合わせればよいかを知らせるものです。このリダイレクトメカニズムは、クライアントアプリケーションがローカルサーバーに存在しないディレクトリーエントリーを要求すると発生します。*

Directory Server は、2 種類の参照をサポートします。

- *デフォルトの参照: サーバーが一致する接尾辞を持たない DN をクライアントのアプリケーションが提示した場合、ディレクトリーはデフォルトの参照を返します。デフォルトの参照は、サーバーの設定ファイルに保存されます。Directory Server にはデフォルトの参照 1 つを設定し、各データベースには個別のデフォルト参照を設定できます。*

各データベースのデフォルトの参照は、接尾辞の設定情報により行われます。データベースの接尾辞が無効になっている場合は、その接尾辞に対して行われたクライアントの要求に対してデフォルトの参照を返すようにディレクトリーサービスを設定します。

接尾辞の詳細は、「接尾辞について」を参照してください。接尾辞の設定に関する詳細は、『Red Hat Directory Server Administration Guide』を参照してください。

- *スマート参照: スマート参照は、ディレクトリーサービス自体のエントリーに保存されます。スマート参照は、スマート参照を含むエントリーの DN に一致する DN を持つサブツリーの知識を有する Directory Server を指します。*

すべての参照は、LDAP URL (LDAP Uniform Resource Locator) の形式で返されます。以下のセクションでは、LDAP 参照の構造の説明後、Directory Server でサポートされる 2 つの参照タイプを説明します。

6.3.1.1. LDAP 参照の構造

LDAP 参照には LDAP URL 形式の情報が含まれます。LDAP URL には以下の情報が含まれています。

- *問い合わせるサーバーのホスト名。*
- *LDAP 要求をリッスンするように設定されたサーバーのポート番号。*
- *ベース DN (検索操作用) またはターゲット DN (操作の追加、削除、および変更用)。*

*たとえば、クライアントアプリケーションは、surname 値が **Jensen** のエントリーを **dc=example,dc=com** で検索します。参照は、以下の 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*

*この参照は、クライアントアプリケーションにポート 389 でホスト **europe.example.com** に問い合わせ、root 接尾辞 **ou=people, l=europe,dc=example,dc=com** を使用して検索を送信するように指示します。*

LDAP クライアントアプリケーションは、参照を処理する方法を決定します。クライアントアプリケーションの中には、参照されたサーバーで、操作を自動的に再試行するものがあります。他のクライアントアプリケーションは、参照情報をユーザーに返します。Red Hat 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 (コマンドラインユーティリティーなど) は、自動的に参照に従います。最初のディレクトリー要求で提供されたのと同じバインド認証情報が、サーバーへのアクセスに使用されます。

ほとんどのクライアントアプリケーションは、限られた数の参照 (hops) に従います。従う参照数を制限することで、クライアントアプリケーションがディレクトリールックアップ要求を完了しようとして費やす時間を短縮し、循環参照パターンによるハングプロセスを排除することができます。

6.3.1.2. デフォルトの参照について

問い合わせたサーバーまたはデータベースにリクエストされたデータが含まれていない場合、デフォルトの参照はクライアントに返されます。

Directory Server は、リクエストされたディレクトリーオブジェクトの DN を、ローカルサーバーがサポートするディレクトリー接尾辞と比較することで、デフォルトの参照を返すかどうかを決定します。DN がサポートされる接尾辞と一致しない場合には、Directory Server はデフォルトの参照を返します。

たとえは、ディレクトリークライアントは以トのディレクトリーエントリーを要求します。 ***uid=bjensen,ou=people,dc=example,dc=com***

ただし、サーバーは ***dc=europe,dc=example,dc=com*** *接尾辞に保存されているエントリーのみを管理します。ディレクトリーは、* ***dc=example,dc=com*** *接尾辞に保存されているエントリーについてどのサーバーに問い合わせるかを示すクライアントに参照を返します。その後、クライアントは該当するサーバーに問い合わせ、元のリクエストを再提出します。*

デフォルトの参照が、ディレクトリーサービスのディストリビューションに関する詳細情報を持つ Directory Server を参照するように設定します。サーバーのデフォルトの参照は、 ***nsslapd-referral*** *属性で設定します。ディレクトリーインストールの各データベースのデフォルトの参照は、設定のデータベースエントリーの* ***nsslapd-referral*** *属性で設定されます。これらの属性値は、* ***dse.ldif*** *ファイルに保存されます。*

デフォルトの参照の設定に関する詳細は、 『Red Hat Directory Server Administration Guide 』*を参照してください。*

6.3.1.3. スマート参照

*Directory Server では、***スマート参照*** *を使用することもできます。スマート参照は、ディレクトリーエントリーまたはディレクトリーツリーを特定の LDAP URL に関連付けます。つまり、リクエストは以下のいずれかに転送されます。*

- *別のサーバーに含まれる同じ namespace。*
- *ローカルサーバー上の異なる namespace。*
- *同じサーバー上の異なる namespace。*

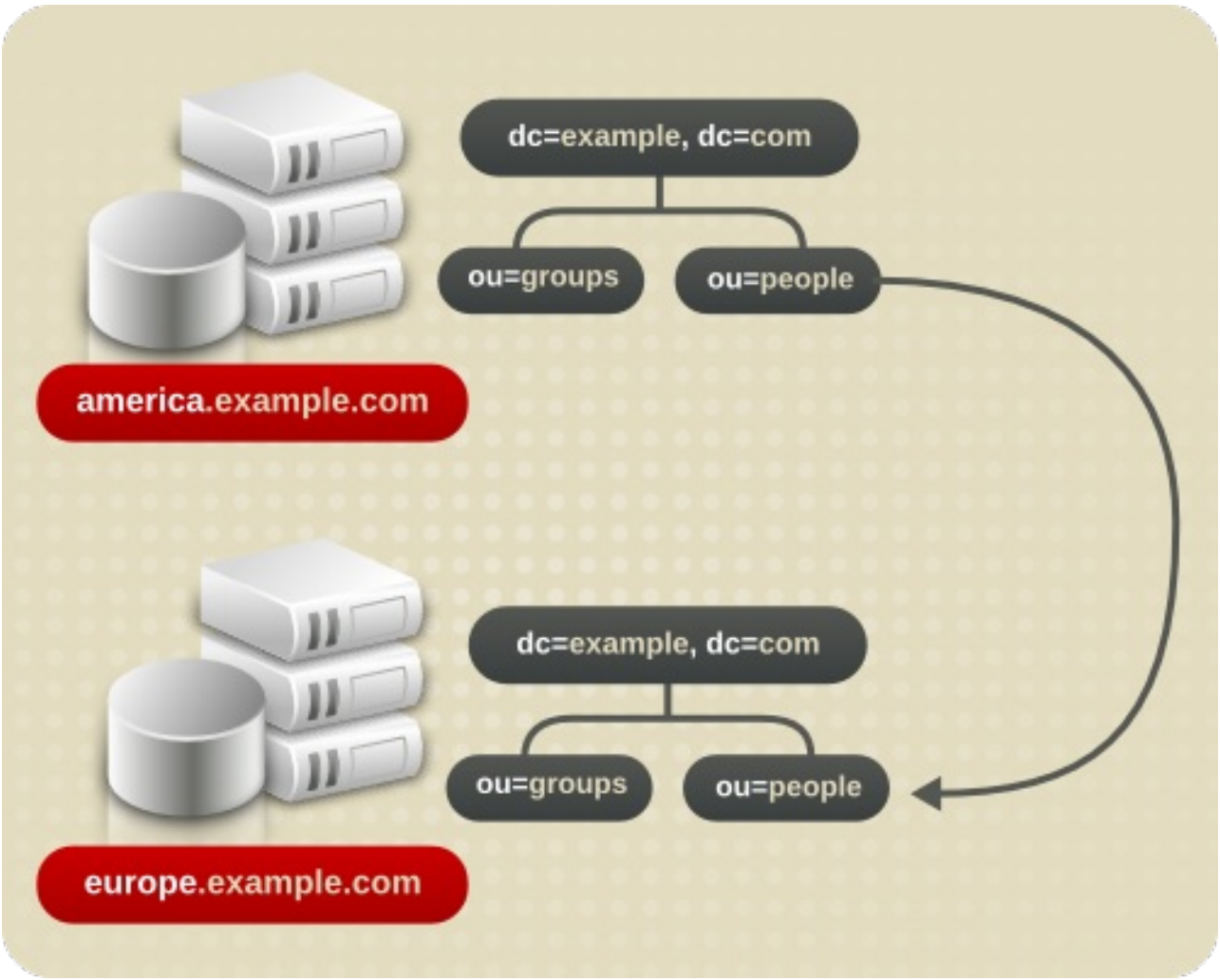
*デフォルトの参照とは異なり、スマート参照はディレクトリーサービス自体に保存されます。スマート参照の設定および管理の詳細は『*Red Hat Directory Server Administration Guide*』を参照してください。*

たとえば、Example Corp. の米国オフィスのディレクトリーサービスには、 ***ou=people,dc=example,dc=com*** *ディレクトリーブランチポイントが含まれます。*

ou=people *エントリー自体にスマートリファーラルを指定して、このブランチのすべての要求を Example Corp. のヨーロッパの* ***ou=people*** *ブランチにリダイレクトします。スマート参照は* ***ldap://europe.example.com:389/ou=people,dc=example,dc=com*** *です。*

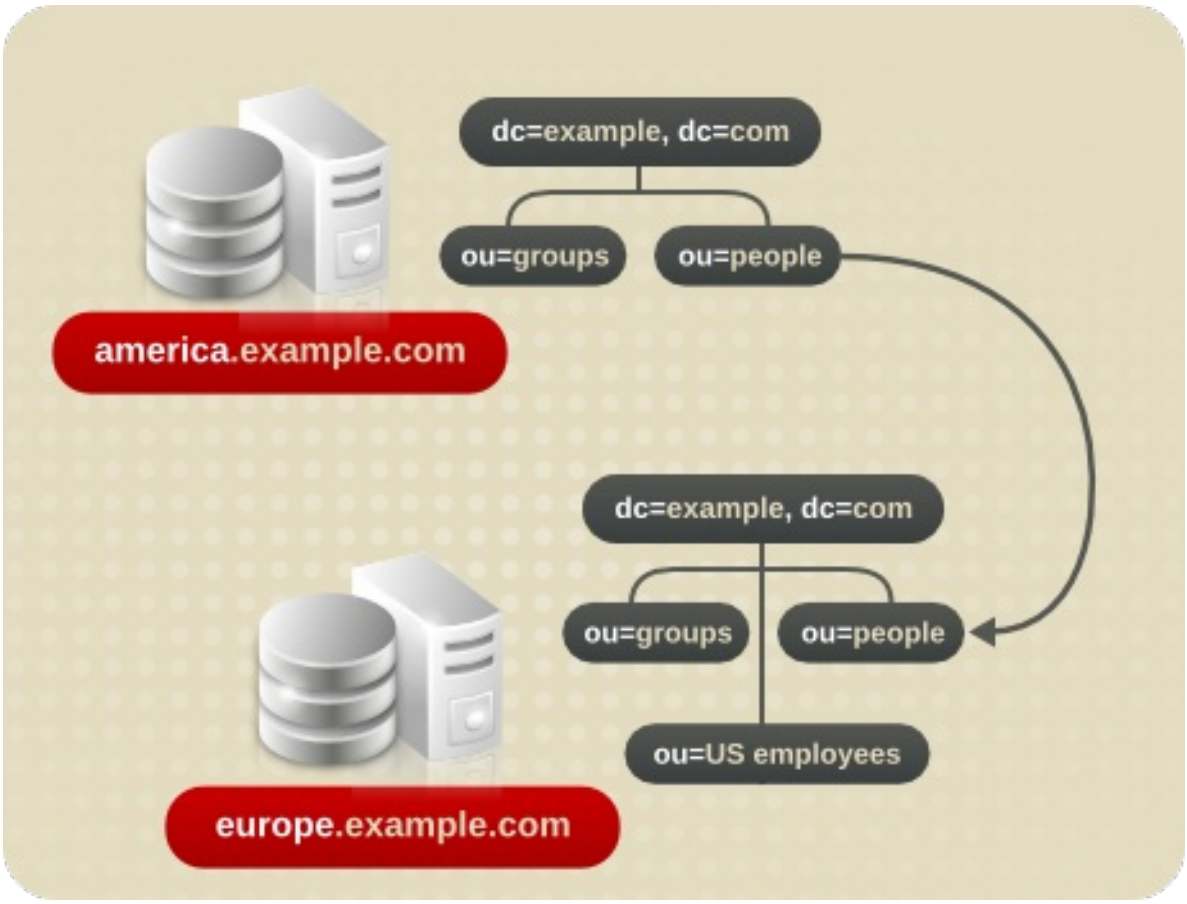
アメリカのディレクトリーサービスの people ブランチへのリクエストは、すべてヨーロッパのディレクトリーにリダイレクトされます。以下で説明します。

図6.7 スマート参照を使用したリクエストのリダイレクト



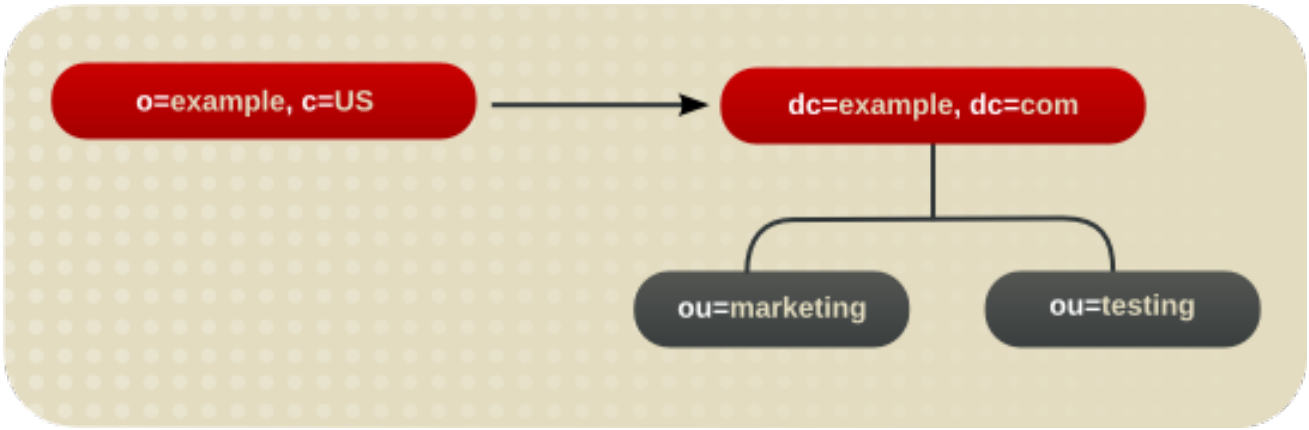
同じメカニズムを使用して、別の namespace を使用する異なるサーバーにクエリーをリダイレクトすることができます。たとえば、Example Corp. のイタリアのオフィスで働く従業員が、Example Corp. のアメリカの従業員の電話番号をヨーロッパのディレクトリーサービスにリクエストするとします。ディレクトリーサービスは参照元 ***ldap://europe.example.com:389/ou=US employees,dc=example,dc=com*** *を返します。*

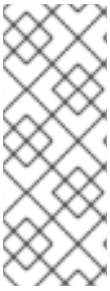
図6.8 異なるサーバーと namespace へのクエリーのリダイレクト



*最後に、複数の接尾辞が同じサーバーで提供される場合、クエリーをある namespace から同じマシンで提供される別の namespace にリダイレクトすることができます。たとえば、**o=example,c=us** のローカルマシンのすべてのクエリーを **dc=example,dc=com** にリダイレクトするには、スマート参照 **ldap:///dc=example,dc=com** を **o=example,c=us** エントリーに配置します。*

図6.9 同一サーバー上のある namespace から別の namespace へのクエリーのリダイレクト





注記

この LDAP URL の 3 番目のスラッシュは、URL が同じ Directory Server を参照していることを示しています。

*ある namespace から他の namespace への参照の作成は、検索がその識別名に基づくクライアントに対してのみ機能します。**ou=people,o=example,c=US** 以下の検索など、その他の種類の操作は正しく実行されません。*

*LDAP URLS と Directory Server エントリーにスマート URL を追加する方法は、『*Red Hat Directory Server Administration Guide*』を参照してください。*

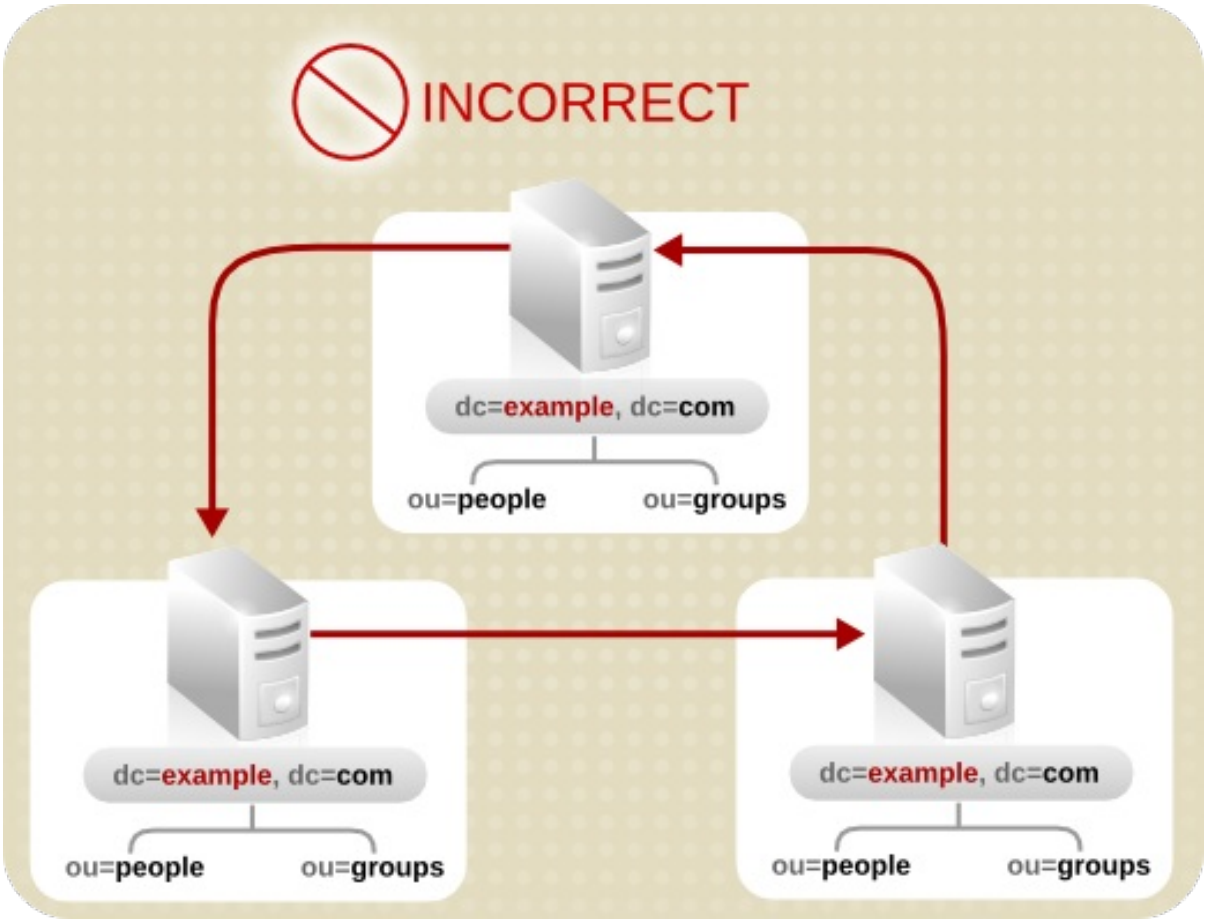
6.3.1.4. スマート参照の設計に関するヒント

スマート参照の実装は簡単ではありますが、使用する前に以下の点を考慮してください。

- *設計はシンプルにします。*

  複雑な参照の Web を使用してディレクトリーサービスをデプロイすると、管理が困難になります。スマート参照を使いすぎると、循環参照パターンが発生する可能性もあります。たとえば、参照が LDAP URL を指し、その URL がまた別の LDAP URL を指し、といった具合に、チェーンのどこかの参照が元のサーバーを指すまで続きます。以下で説明します。

  図6.10 循環参照パターン



- *メジャーなブランチポイントでリダイレクトします。*

  ディレクトリーツリーの接尾辞レベルでリダイレクトを処理する参照の使用を制限します。スマート参照は、リーフ (ブランチ以外) エントリーに対するルックアップ要求を、異なるサーバーおよび DN にリダイレクトします。その結果、スマート参照をエイリアスメカニズムとして使用したくなるため、ディレクトリー構造のセキュア化は複雑で困難な方法になります。ディレクトリーツリーの接尾辞またはメジャーブランチポイントに対する参照を制限すると、管理する必要のある参照数が制限され、その後ディレクトリーの管理オーバーヘッドが削減されます。

- *セキュリティーへの影響を考慮します。*

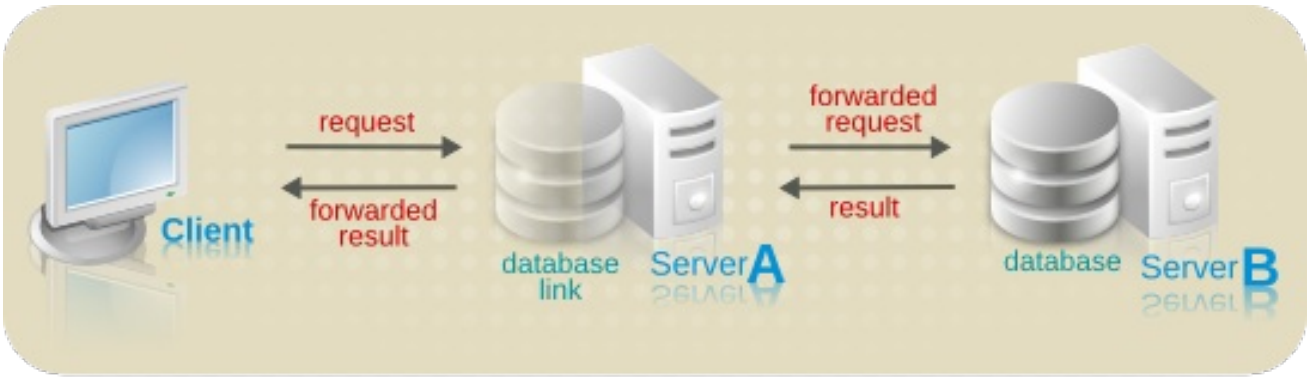
アクセス制御は参照の境界を越えません。リクエストを発信したサーバーがエントリーへのアクセスを許可していても、スマート参照が別のサーバーにクライアントリクエストを送信すると、クライアントアプリケーションのアクセスが許可されない場合があります。

さらに、クライアントの認証が行われるためにクライアントが参照されるサーバー上でクライアントの認証情報が利用可能である必要があります。

6.3.2. チェーンの使用

チェーンは、別のサーバーにリクエストをリレーする方法です。この方法は、データベースリンクを使用して実装されます。「ディレクトリーデータの配布」で説明したように、データベースリンクにはデータは含まれません。代わりに、クライアントアプリケーションのリクエストを、データを含むリモートサーバーにリダイレクトします。

サーバーはチェーン処理中に、クライアントアプリケーションから、サーバーに含まれないデータのリクエストを受信します。データベースリンクを使用して、サーバーは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問い合わせ、クライアントアプリケーションに結果を返します。



各データベースリンクは、データを保持するリモートサーバーに関連付けられます。障害発生時にデータベースリンクが使用するデータのレプリカを含む代替リモートサーバーを設定します。データベースリンクの設定に関する詳細は、『Red Hat Directory Server Administration Guide』を参照してください。

データベースリンクは以下の機能を提供します。

- リモートデータへの非表示のアクセス。

  データベースリンクはクライアントのリクエストを解決するため、データ配布はクライアントから完全に非表示になります。

- 動的管理。

  システム全体をクライアントアプリケーションで利用できる状態で、ディレクトリーサービスの一部を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。データベースリンクは、ディレクトリーサービス全体にエントリーが再分散されるまで、アプリケーションに参照を一時的に返すことができます。

  これは、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、参照を返すことができる接尾辞自体で実施することも可能です。

- アクセス制御。

  データベースリンクは、クライアントアプリケーションになりすまし、適切な認証 ID をリモートサーバーに提供します。アクセス制御の評価が必要ない場合は、リモートサーバーでユーザーのなりすましを無効にすることができます。データベースリンクの設定に関する詳細は、

『Red Hat Directory Server Administration Guide 』を参照してください。

6.3.3. 参照とチェーンのどちらかを選択する

ディレクトリーパーティションをリンクする方法は、いずれも長所と短所があります。使用するメソッドまたはメソッドの組み合わせは、ディレクトリーサービスの特定のニーズによって異なります。

2 つのナレッジ参照の主な違いは、分散された情報の検索方法を知るインテリジェンスの場所にあります。チェーンシステムでは、インテリジェンスはサーバーに実装されています。参照を使用するシステムでは、インテリジェンスはクライアントアプリケーションに実装されています。

チェーンはクライアントの複雑性を軽減しますが、その代償としてサーバーの複雑さが増します。チェーンサーバーはリモートサーバーと連携し、結果をディレクトリークライアントに送信する必要があります。

参照では、クライアントは参照の位置を探したり、検索結果を照合したりする必要があります。ただし、参照はクライアントアプリケーションの作成者にさらなる柔軟性を提供し、開発者が分散ディレクトリー操作の進捗状況についてユーザーに適切にフィードバックできるようにします。

以下のセクションでは、参照とチェーンのより具体的な相違点について詳しく説明します。

6.3.3.1. 使用に関する相違点

クライアントアプリケーションによっては、参照をサポートしないものがあります。チェーンにより、クライアントアプリケーションが単一のサーバーと通信し、多くのサーバーに保存されているデータに引き続きアクセスすることができます。会社のネットワークがプロキシーを使用する場合、参照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は、ファイアウォール内の1 つのサーバーのみと通信する権限を持っている場合があります。そのアプリケーションが別のサーバーを参照する場合、正常にコンタクトすることはできません。

また、参照を使用する際には、クライアントが正しく認証できなければなりません。つまり、クライアントが参照されるサーバーには、クライアントの認証情報が含まれている必要があります。チェーンでは、クライアントの認証は一度しか行われません。クライアントは、リクエストがチェーンされるサーバーで再度認証する必要はありません。

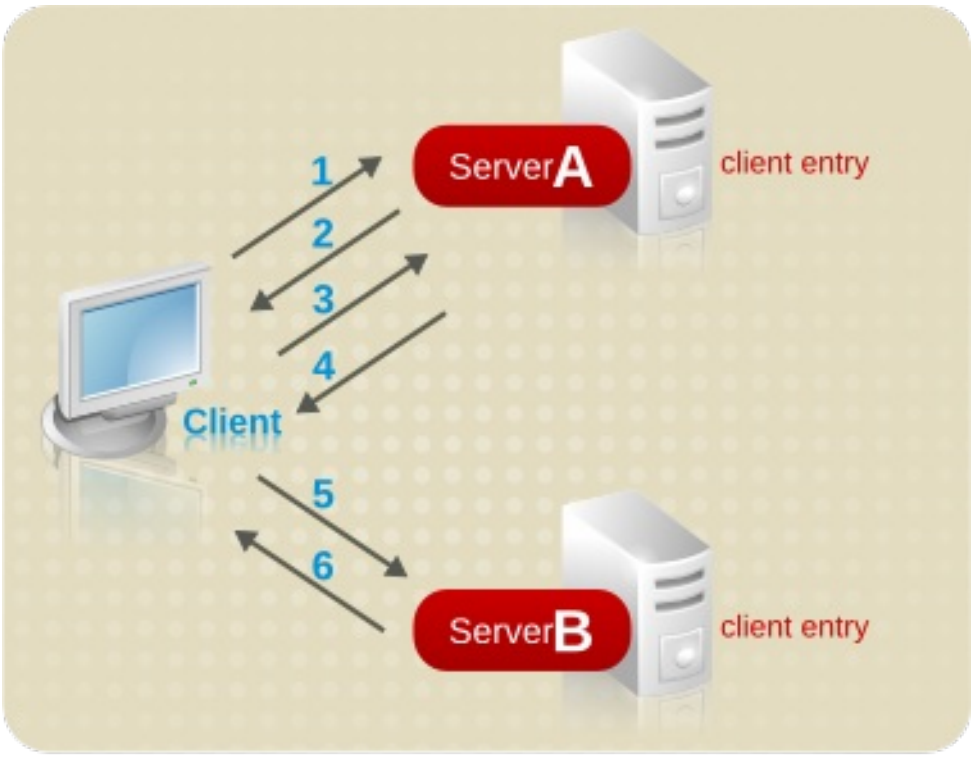
6.3.3.2. アクセス制御の評価

チェーンは、参照とは異なる方法でアクセス制御を評価します。参照では、クライアントのエントリーがすべてのターゲットサーバーに存在している必要があります。チェーンでは、クライアントエントリーはすべてのターゲットサーバー上にある必要はありません。

参照を使用した検索リクエストの実行

以下の図は、参照を使用したクライアントからサーバーへの要求を示しています。

図6.11 参照を使用したクライアント要求のサーバーへの送信



上記の図では、クライアントアプリケーションは以下の手順を実行します。

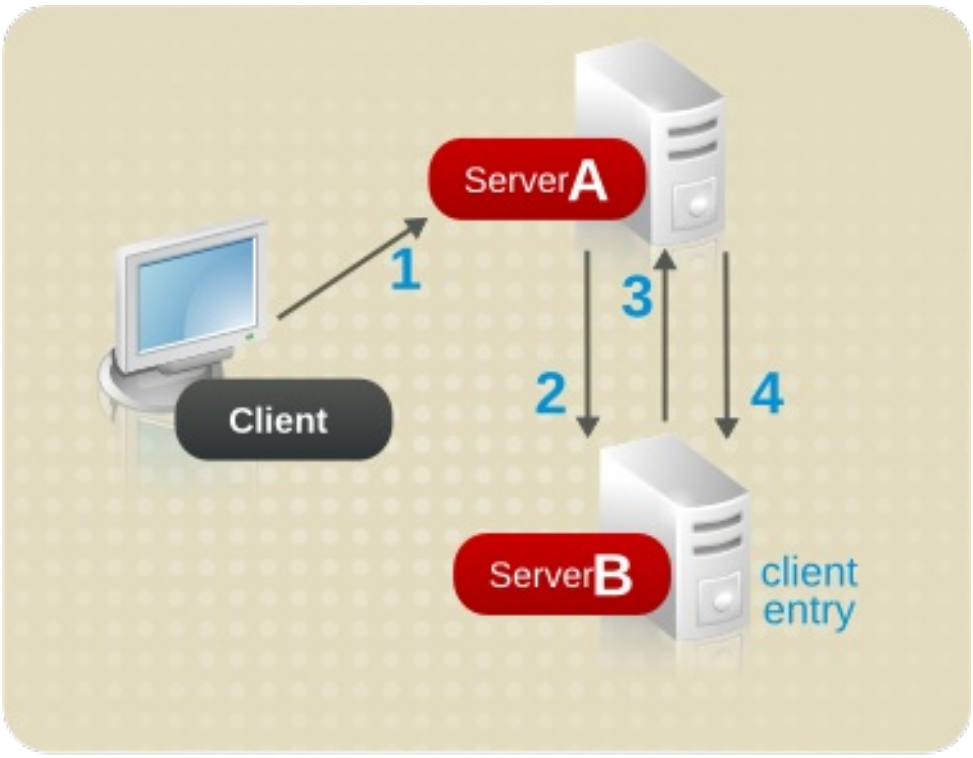
1. *クライアントアプリケーションは、最初にサーバーA にバインドします。*
2. *サーバーA にはユーザー名とパスワードを提供するクライアントのエントリーが含まれるため、バインドの受け入れメッセージを返します。参照を機能させるためには、クライアントエントリーがサーバーA に存在する必要があります。*
3. *クライアントアプリケーションは操作リクエストをサーバーA に送信します。*
4. *ただし、サーバーA には要求された情報が含まれていません。代わりに、サーバーA は、サーバーB に連絡するよう指示するクライアントアプリケーションへの参照を返します。*
5. *その後、クライアントアプリケーションはサーバーB にバインドリクエストを送信します。正常にバインドするには、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ーも含まれている必要があります。*
6. *バインドに成功し、クライアントアプリケーションは検索操作をサーバーB に再送信できるようになりました。*

この方法では、サーバーB が、サーバーA からのクライアントのエントリーの複製コピーを持つ必要があります。

チェーンを使用した検索リクエストの実行

サーバー間でクライアントエントリーを複製する問題は、チェーンを使用して解決されます。チェーンシステムでは、検索リクエストは応答があるまで複数回転送されます。

図6.12 チェーンを使用したクライアント要求のサーバーへの送信



上記の図で、以下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 にバインドすると、サーバーA はユーザー名とパスワードが正しいかどうかを確認しようとします。*

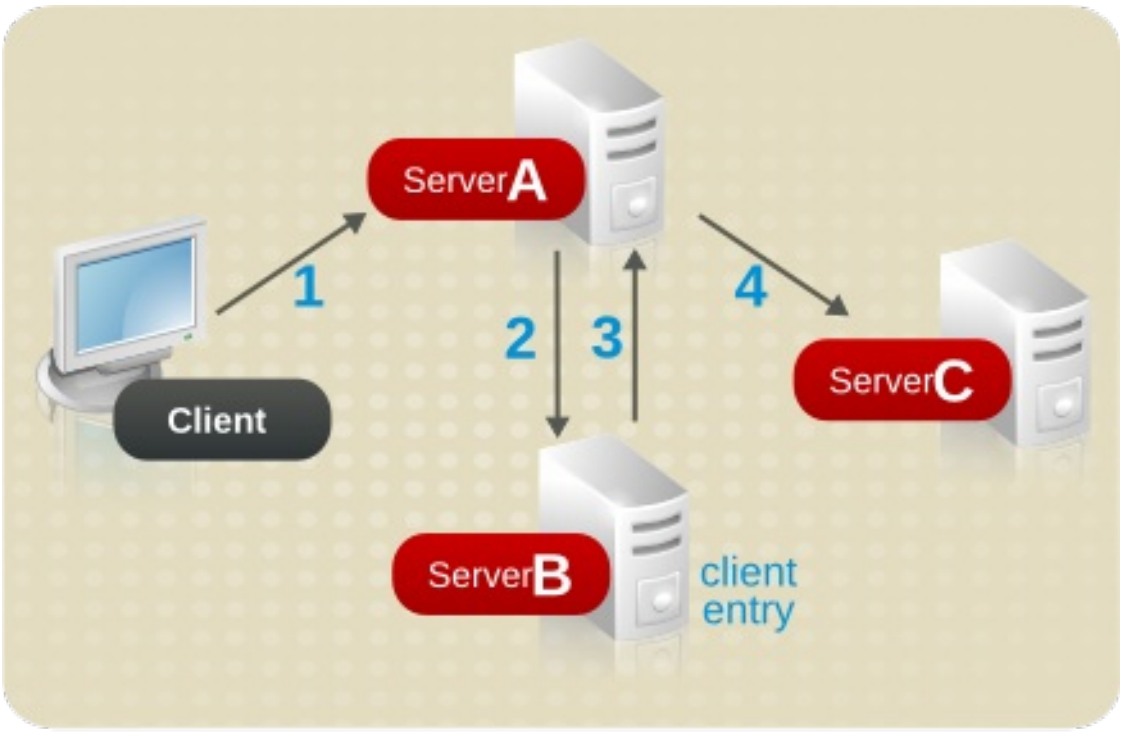
2. *サーバーA にはクライアントアプリケーションに対応するエントリーが含まれません。代わりに、クライアントの実際のエントリーが含まれるサーバーB へのデータベースリンクが含まれます。サーバーA はバインドリクエストをサーバーB に送信します。*

3. *サーバーB はサーバーA に受け入れ応答を送信します。*

4. *次に、サーバーA はクライアントアプリケーションのリクエストをデータベースリンクを使用して処理します。データベースリンクは、サーバーB にあるリモートデータストアに問い合わせ、検索操作を処理します。*

チェーンシステムでは、クライアントアプリケーションに対応するエントリーは、クライアントが要求するデータと同じサーバーに置く必要はありません。

図6.13 異なるサーバーを使用したクライアントの認証およびデータの取得



この図では、以下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 にバインドすると、サーバーA はユーザー名とパスワードが正しいかどうか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ーが含まれません。代わりに、クライアントの実際のエントリーが含まれるサーバーB へのデータベースリンクが含まれます。サーバーA はバインドリクエストをサーバーB に送信します。*

3. *サーバーB はサーバーA に受け入れ応答を送信します。*

4. *サーバーA は次に別のデータベースリンクを使用して、クライアントアプリケーションのリクエストを処理します。データベースリンクは、サーバーC にあるリモートデータストアに問い合わせ、検索操作を処理します。*

サポートされないアクセス制御

データベースリンクは、以下のアクセス制御をサポートしません。

- *ユーザーエントリーが別のサーバーにある場合、ユーザーエントリーのコンテンツにアクセスする必要のあるコントロールはサポートされません。これには、グループ、フィルター、およびロールに基づくアクセス制御が含まれます。*

- *クライアント IP アドレスまたは DNS ドメインに基づいた制御は拒否される可能性があります。これは、データベースリンクがリモートサーバーに問い合わせする際に、クライアントになりすますためです。リモートデータベースに IP ベースのアクセス制御が含まれている場合は、元のクライアントドメインではなく、データベースリンクのドメインを使用して評価されます。*

6.4. データベースパフォーマンスを改善するためのインデックスの使用

クライアントアプリケーションが行う検索は、データベースのサイズによっては、時間とリソースを要することがあります。この問題を軽減するには、インデックスを使用して検索パフォーマンスを向上します。

インデックスは、ディレクトリーデータベースに保存されるファイルです。ディレクトリーサービスのデータベースごとに個別のインデックスファイルが維持されます。各ファイルには、インデックスを付ける属性に従って名前が付けられます。特定の属性のインデックスファイルには複数のインデックスタイプを含めることができるため、属性ごとに複数のインデックスタイプを管理できます。たとえば、**cn.db** というファイルには、共通名属性のすべてのインデックスが含まれます。

ディレクトリーサービスを使用するアプリケーションの種類に応じて、さまざまな種類のインデックスが使用されます。アプリケーションによっては、頻繁に特定の属性を検索するか、別の言語でディレクトリーを検索するか、あるいは特定の形式でデータが必要な場合があります。

6.4.1. ディレクトリーインデックスタイプの概要

Directory Server は、以下のタイプのインデックスをサポートします。

- インデックスの存在： **uid** などの特定の属性を持つエントリーを一覧表示します。
- Equality index: **cn=Babs Jensen** など、特定の属性値を含むエントリーを一覧表示します。
- 概算インデックス: 概算 (または sounds-like) 検索を可能にします。たとえば、エントリーには **cn=Babs L. Jensen** の属性値が含まれる場合があります。概算検索では、**cn~=Babs Jensen**、**cn~=Babs**、および **cn~=Jensen** に対する検索に対してこの値を返します。

  注記

  概算インデックスでは、ASCII 文字を使用して、英語で名前を表記する必要があります。

- 部分文字列インデックス: エントリー内の部分文字列に対する検索を許可します。たとえば、**cn=*derson** の検索は、この文字列を含む共通名と一致します(Bill Anderson、Norma Henderson、Steve Sanderson など)。
- 国際インデックス: 国際ディレクトリーでの情報検索のパフォーマンスが向上します。ロケール (国際化 OID) をインデックス化する属性に関連付けることにより、一致するルールを適用するインデックスを設定します。
- 参照インデックスまたは仮想リストビュー (VLV) インデックス: Web コンソールのエントリーの表示パフォーマンスを向上させます。**参照インデックス** をディレクトリーツリーの任意のブランチに作成して、表示パフォーマンスを向上させることができます。

6.4.2. インデックス化のコストの評価

インデックスはディレクトリーデータベースの検索性能を向上させますが、それにはコストがかかります。

- インデックスを使用すると、エントリーの修正にかかる時間が長くなります。

  維持されるインデックスが増えれば増えるほど、ディレクトリーサービスがデータベースを更新するのに時間がかかります。

- インデックスファイルはディスク領域を使用します。

インデックス化される属性が多ければ多いほど、多くのファイルが作成されます。長い文字列を含む属性に概算インデックスおよび部分文字列インデックスがある場合、これらのファイルは急速に大きくなる可能性があります。

- *インデックスファイルはメモリーを使用します。*

  さらに効率的に実行するには、ディレクトリーサービスは可能な限り多くのインデックスファイルをメモリーに配置します。インデックスファイルは、データベースキャッシュのサイズに応じて利用可能なプールのメモリーを使用します。インデックスファイルの数が多いと、データベースキャッシュも大きくなります。

- *インデックスファイルの作成には時間がかかります。*

  インデックスファイルは検索時の時間を短縮しますが、不要なインデックスを維持することは時間の浪費につながります。ディレクトリーサービスを利用するクライアントアプリケーションが必要とするファイルのみを維持するようにしてください。

第7章 レプリケーションプロセスの設計

ディレクトリーの内容を複製すると、ディレクトリーサービスの可用性やパフォーマンスが向上します。4章ディレクトリーツリーの設計 および 6章ディレクトリートポロジーの設計 で、ディレクトリーツリーとディレクトリートポロジーの設計について説明しています。本章では、データの物理的および地理的な場所について、特に、必要なときに必要な場所でデータを利用できるようにするためのレプリケーションの利用方法について説明します。

本章では、レプリケーションの使用方法を説明し、ディレクトリー環境のレプリケーションストラテジーの設計に関するアドバイスを提供します。

7.1. レプリケーションの概要

レプリケーションは、自動的にディレクトリーデータを Red Hat Directory Server から別の Red Hat Directory Server にコピーするメカニズムです。レプリケーションを使用すると、(独自のデータベースに格納されている) 任意のディレクトリーツリーまたはサブツリーをサーバー間でコピーすることができます。情報のメインコピーを保持している Directory Server は、すべてのレプリカに更新情報を自動的にコピーします。

レプリケーションは高可用性ディレクトリーサービスを提供し、データを地理的に分散できます。現実的には、レプリケーションには以下のようなメリットがあります。

- フォールトトレランスおよびフェイルオーバー - ディレクトリーツリーを複数のサーバーに複製することで、ハードウェア、ソフトウェア、ネットワークの問題により、ディレクトリークライアントアプリケーションが特定のディレクトリーサーバーにアクセスできない場合でも、ディレクトリーサービスを利用できます。クライアントは、読み取りと書き込み操作のために、別の Directory Server を参照します。

  注記

  書き込みフェイルオーバーは、マルチサプライヤーのレプリケーションでのみ可能です。

- 負荷分散: サーバー全体でディレクトリーツリーを複製すると、指定したマシンのアクセス負荷が減り、サーバーの応答時間が改善します。
- パフォーマンスの向上とレスポンスタイムの短縮 - ディレクトリーエントリーをユーザーに近い場所に複製することで、ディレクトリーのレスポンスタイムが大幅に向上します。
- ローカルデータ管理: レプリケーションにより、企業全体で他の Directory Server と情報を共有しながら、ローカルで情報を所有および管理することができます。

7.1.1. レプリケーションの概念

レプリケーションの計画は、常に以下の基本的な決定を行うことから始めます。

- 複製する情報。
- その情報のメインコピー (**読み取り/書き込みレプリカ**) を保持するサーバー。
- その情報の読み取り専用コピー (**読み取り専用レプリカ**) を保持するサーバー。
- 読み取り専用のレプリカが更新要求を受け取ったときに何をすべきか、つまり、どのサーバーにその要求を参照すべきか。

これらの決定は、Directory Server がこれらの概念をどのように処理するかを理解せずに効果的に行うことはできません。たとえば、複製する情報を決定するには、Directory Server が処理できる最小レプリケーションユニットを認識してください。Directory Server が使用するレプリケーションの概念は、必要なグローバルな決定について考えるためのフレームワークを提供します。

7.1.1.1. レプリケーションのユニット

レプリケーションの最小単位はデータベースです。データベース全体はレプリケートできますが、データベース内のサブツリーは複製できません。したがって、ディレクトリーツリーを定義する際には、常にレプリケーションを考慮してください。ディレクトリーツリーの設定方法に関する詳細は、4章ディレクトリーツリーの設計 を参照してください。

レプリケーションメカニズムでは、1 つのデータベースが1 つの接尾辞に対応する必要もあります。2 つ以上のデータベース上で分散される接尾辞 (または namespace) は複製できません。

7.1.1.2. 読み取り/書き込みレプリカおよび読み取り専用レプリカ

レプリケーションに参加するデータベースは、**レプリカ** として定義されます。Directory Server は、2 種類のレプリカ (読み取り/書き込みおよび読み取り専用) をサポートします。読み取り/書き込みレプリカには、ディレクトリー情報のメインコピーが含まれており、更新可能です。読み取り専用レプリカは、すべての更新操作を読み取り/書き込みレプリカに参照します。

7.1.1.3. サプライヤーとコンシューマー

別のサーバーにコピーされたレプリカを保存するサーバーは **サプライヤー** と呼ばれます。別のサーバーからコピーしたレプリカを保存するサーバーは、**コンシューマー** と呼ばれます。通常、サプライヤーサーバーのレプリカは読み取り/書き込みレプリカで、コンシューマーサーバーのレプリカは読み取り専用レプリカになります。ただし、以下の例外が適用されます。

- **カスケードレプリケーション** の場合、**ハブサプライヤー** はコンシューマーに提供する読み取り専用レプリカを保持します。詳細は、「カスケードレプリケーション」 を参照してください。
- **マルチサプライヤーのレプリケーション** の場合、サプライヤーは、同じ読み取り/書き込みレプリカのサプライヤーとコンシューマーの両方として機能します。詳細は、「マルチサプライヤーのレプリケーション」 を参照してください。

注記

Red Hat Directory Server の現行バージョンでは、レプリケーションは常にサプライヤーサーバーによって開始され、コンシューマーによって開始されることはありません。これは、コンシューマーで開始されるレプリケーション (コンシューマーサーバーがサプライヤーサーバーからデータを取得することができる) を許可した Directory Server の以前のバージョンとは異なります。

サプライヤー

特定のレプリカの場合は、サプライヤーサーバーでは以下を行う必要があります。

- ディレクトリークライアントからの読み取りリクエストや更新リクエストへの対応。
- レプリカの状態情報と changelog を維持します。
- コンシューマーサーバーへのレプリケーションを開始します。

サプライヤーサーバーは管理する読み取り/書き込みレプリカに加えられた変更を常に記録するため、サプライヤーサーバーは、変更が確実にコンシューマーサーバーに複製されるようにします。

コンシューマー

コンシューマーサーバーは、以下を行う必要があります。

- 読み取りのリクエストに応える。
- レプリカのサプライヤーサーバーへの更新リクエストを参照する。

コンシューマーサーバーがエントリーの追加、削除、または変更の要求を受信するたびに、要求はレプリカのサプライヤーに参照されます。サプライヤーサーバーはリクエストを実行し、変更を複製します。

ハブサプライヤー

レプリケーションをカスケードする特殊なケースでは、ハブサプライヤーは以下を行う必要があります。

- 読み取りのリクエストに応える。
- レプリカのサプライヤーサーバーへの更新リクエストを参照する。
- コンシューマーサーバーへのレプリケーションを開始します。

カスケードレプリケーションに関する詳細は、「カスケードレプリケーション」 を参照してください。

7.1.1.4. レプリケーションとチェンジログ

すべてのサプライヤーサーバーは **changelog** を維持します。changelog とは、レプリカで発生した変更の記録のことです。次に、サプライヤーサーバーは、マルチサプライヤーレプリケーションの場合には、コンシューマーサーバーに保存されているレプリカまたは他のサプライヤーにこの変更を再生します。

エントリーが変更されると、実行された LDAP 操作を記述する変更レコードが changelog に記録されます。

changelog サイズは、**nsslapd-changelogmaxage** または **nsslapd-changelogmaxentries** の 2 つの属性で維持されます。これらの属性は古い changelog をトリミングして、changelog サイズを妥当な状態に維持します。

7.1.1.5. レプリカ合意

Directory Server はレプリカ合意を使用してレプリケーションを定義します。レプリカ合意は、1 つのサプライヤーと 1 つのコンシューマーとの間のレプリケーションのみを説明します。この合意は、サプライヤーサーバーで設定します。以下を識別します。

- 複製するデータベース。
- データがプッシュされるコンシューマーサーバー。
- レプリケーションが実行される時間。
- サプライヤーサーバーがバインドする際に使用しなければならない DN (**サプライヤーバインド DN** と呼びます)。

- 接続のセキュリティーを確保する方法 (TLS、Start TLS、クライアント認証、SASL、または簡易認証)。
- レプリケートされない属性 (「一部レプリケーションで選択された属性を複製する」 を参照)

7.1.2. データの整合性

一貫性とは、複製されたデータベースの内容が、ある時点でどれだけ一致しているかを意味します。サーバー間のレプリケーションの設定の一部として、更新のスケジューリングがあります。サプライヤーサーバーは、コンシューマーサーバーをいつ更新する必要があるかを常に判断し、レプリケーションを開始します。

Directory Server には、レプリカを常に同期させるオプション、または特定の時刻または曜日の更新をスケジュールするオプションがあります。

レプリカを常に同期させておくことで、データの一貫性が保たれるというメリットがあります。ただし、更新作業が頻繁に行われることにより、ネットワークトラフィックが犠牲となります。このソリューションは、以下の場合に最適なオプションとなります。

- サーバー間に、信頼できる高速接続があります。
- ディレクトリーサービスが提供するクライアント要求は主に検索、読み取り、および比較操作でであり、更新操作は比較的少ないです。

データの一貫性が低くても問題ない場合は、ネットワークの使用パターンに合わせて、あるいはネットワークトラフィックへの影響を少なくするような更新頻度を選択してください。常に更新を行うのではなく、定期的に更新を行うことが最良の解決策である場合がいくつかあります。

- 信頼できな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があります。
- ディレクトリーサービスが提供するクライアントリクエストは主に更新操作です。
- 通信コストを低くする必要があります。

マルチサプライヤーレプリケーションの場合、各サプライヤーに保存されているデータにはいつでも違いが生じる可能性があるため、各サプライヤーのレプリカは **大まかに一貫性がある** と言われています。以下の 2 つの理由から、レプリカが完全に同期されている場合でも、これは当てはまります。

- サプライヤー間の更新操作の伝播にはレイテンシーがあります。
- 更新操作を処理したサプライヤーは、2 番目のサプライヤーが更新操作を検証するのを待たずに、operation successful というメッセージをクライアントに返します。

7.2. 一般的なレプリケーションシナリオ

更新がサーバーからサーバーにどのように移動するか、および更新を伝達するときにサーバーがどのように対話するかを決定します。環境に適した方法を決定するための 4 つの基本的なシナリオといくつかの戦略があります。この基本的なシナリオを組み合わせて、ネットワーク環境に最適なレプリケーショントポロジーを構築できます。

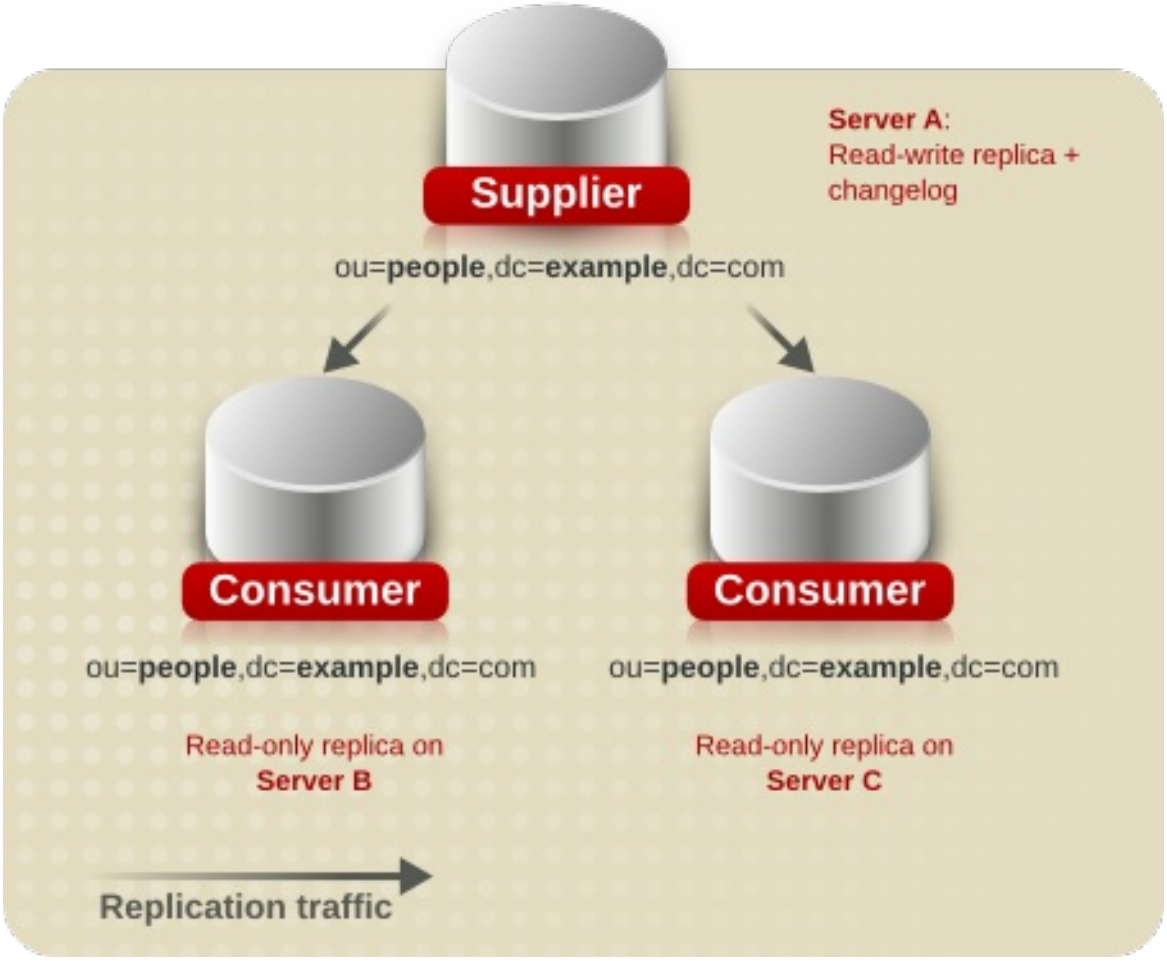
- 「単一サプライヤーレプリケーション」
- 「マルチサプライヤーのレプリケーション」
- 「カスケードレプリケーション」
- 「混合環境」

7.2.1. 単一サプライヤーレプリケーション

最も基本的なレプリケーション設定では、サプライヤーサーバーはレプリカを 1 つ以上のコンシューマーサーバーに直接コピーします。この設定では、すべてのディレクトリーの変更が、サプライヤーサーバーの読み取り/書き込みレプリカで行われ、コンシューマーサーバーにはデータの読み取り専用レプリカが含まれます。

サプライヤーサーバーは、コンシューマーサーバーに保存されている読み取り/書き込みレプリカに対して変更をすべて実行する必要があります。以下で説明します。

図7.1 単一サプライヤーレプリケーション



サプライヤーサーバーは、読み取り/書き込みレプリカを複数のコンシューマーサーバーに複製できます。1 つのサプライヤーサーバーが管理できるコンシューマーサーバーの合計数は、ネットワークの速度と毎日変更されるエントリーの合計数によって異なります。ただし、サプライヤーサーバーは複数のコンシューマーサーバーを維持することができます。

7.2.2. マルチサプライヤーのレプリケーション

マルチサプライヤーのレプリケーション 環境では、同じ情報のメインコピーが複数のサーバーに存在することがあります。つまり、異なる場所で同時にデータを更新できることを意味します。各サーバーで発生した変更は、他のサーバーに複製されます。つまり、各サーバーはサプライヤーとコンシューマーの両方として機能します。

注記

Red Hat Directory Server は、任意のレプリケーション環境で最大 20 個のサプライヤーサーバーと、数量無制限のハブサプライヤーをサポートします。読み取り専用レプリカを保持するコンシューマーサーバーの数は無制限です。

同じデータが複数のサーバーで変更された場合、どの変更が保持されるかを決定するための競合解決手順があります。Directory Server は、有効な変更を最新の変更とみなします。

複数のサーバーが同じデータの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の範囲内に存在するのは、1 つのサプライヤーサーバーと 1 つのコンシューマーのみです。したがって、同じデータに対して責任を共有する 2 つのサプライヤーサーバー間にマルチサプライヤー環境を作成するには、複数のレプリケーションアグリーメントを作成します。

図7.2 簡素化されたマルチサプライヤーレプリケーション設定

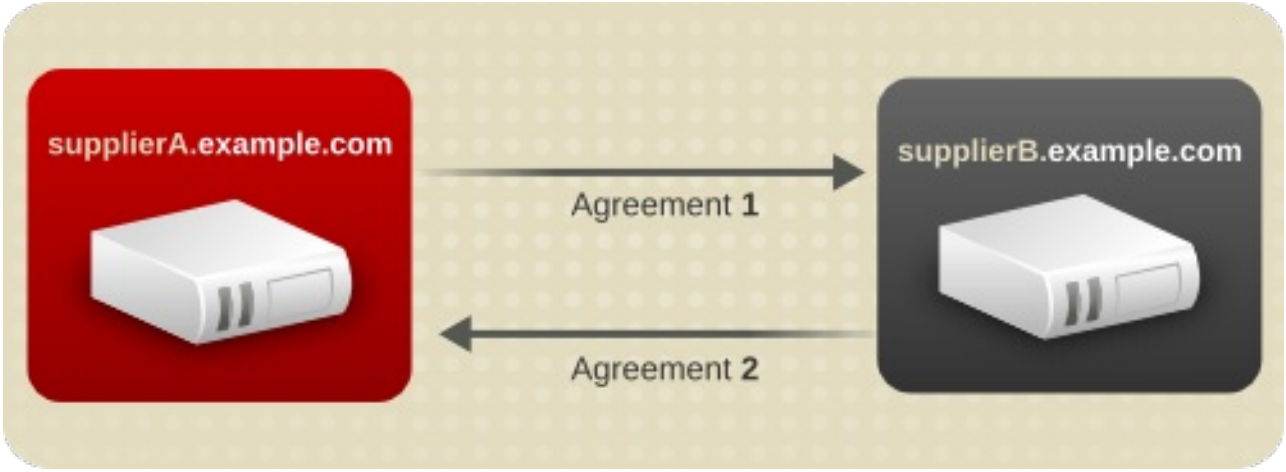
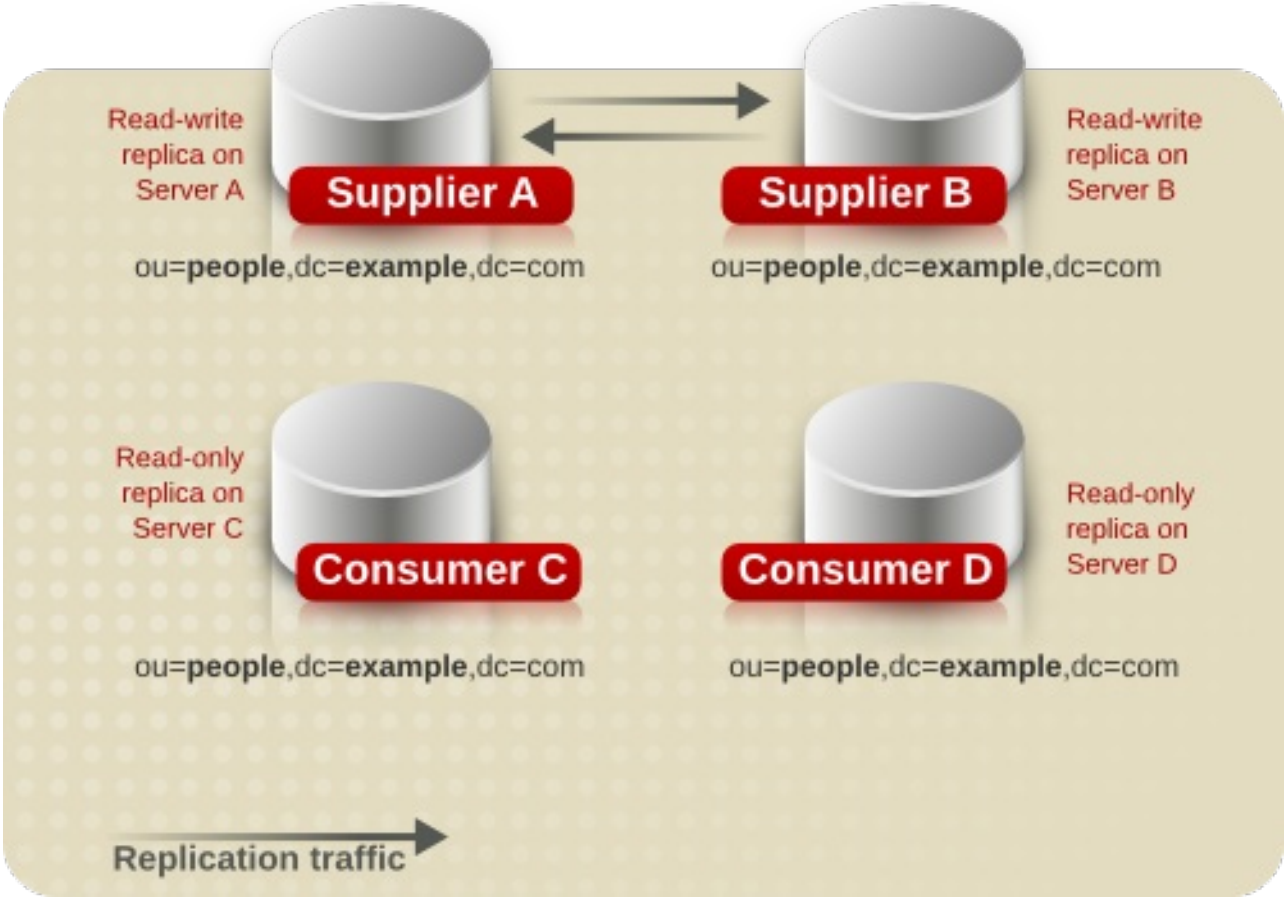


図7.2「簡素化されたマルチサプライヤーレプリケーション設定」では、サプライヤー A とサプライヤー B はそれぞれ、同じデータの読み取り/書き込みレプリカを保持します。

図7.3「簡素なマルチサプライヤー環境でのレプリケーショントラフィック」は、2 つのサプライヤー (図では読み取り/書き込みレプリカ) と 2 つのコンシューマー (図では読み取り専用レプリカ) のレプリケーショントラフィックを説明しています。コンシューマーは、両方のサプライヤーによって更新できます。サプライヤーサーバーは、変更が競合しないようにします。

図7.3 簡素なマルチサプライヤー環境でのレプリケーショントラフィック



Directory Server でのレプリケーションは、最大 20 のサプライヤーをサポートできます。これらのサプライヤーはすべて、同じデータに対する責任を共有します。このような多くのサプライヤーを使用するには、さまざまなレプリカアグリーメントを作成する必要があります。(また、マルチサプライヤーレプリケーションでは、各サプライヤーを異なるトポロジーで設定できることも覚えておいてください。つまり、20 の異なるディレクトリーツリーや、スキーマの違いさえある可能性があります。トポロジーの選択に直接影響する変数は多数あります)。

マルチサプライヤーのレプリケーションでは、サプライヤーは他のすべてのサプライヤーに更新を送信することや、他のサプライヤーのサブセットに更新を送信することができます。他のすべてのサプライヤーへの更新を送信すると、変更がより迅速に伝播され、全体的なシナリオは障害耐性が大幅に向上します。ただし、サプライヤーの設定が複雑になり、ネットワークの需要とサーバーの需要が高くなります。サプライヤーのサブセットへの更新の送信は、設定がはるかに簡単になり、ネットワークとサーバーの負荷が軽減されますが、複数のサーバーに障害が発生した場合、データが失われるリスクがあります。

図7.4「マルチサプライヤーのレプリケーション設定 A」 は、4 つのサプライヤーサーバーが、他の 3 つのサプライヤーサーバー (コンシューマーとしても機能する) にデータをフィードする、完全に接続されたメッシュトポロジーを示しています。4 つのサプライヤーサーバー間には、合計 12 のレプリケーションアグリーメントが存在します。

図7.4 マルチサプライヤーのレプリケーション設定A

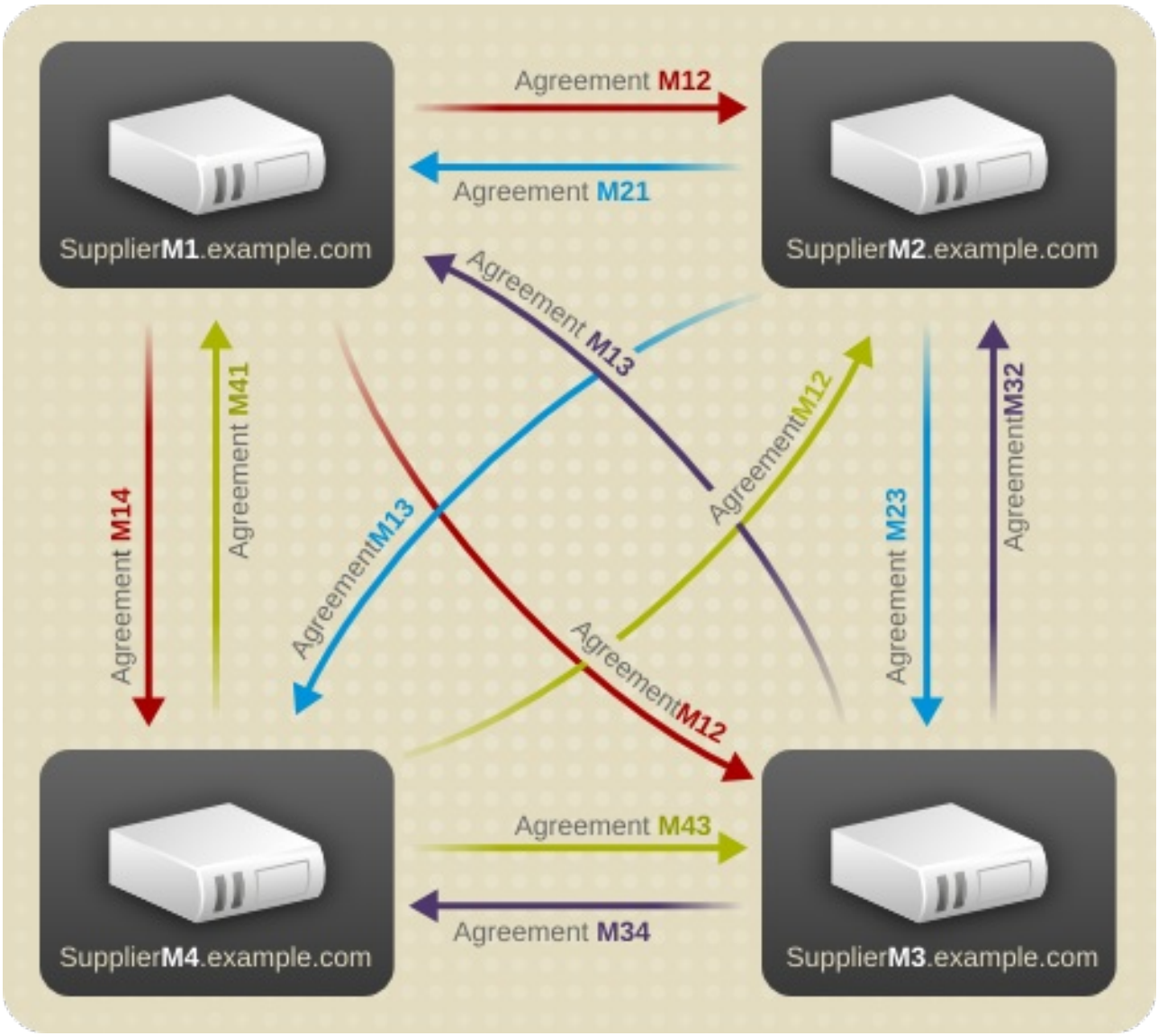
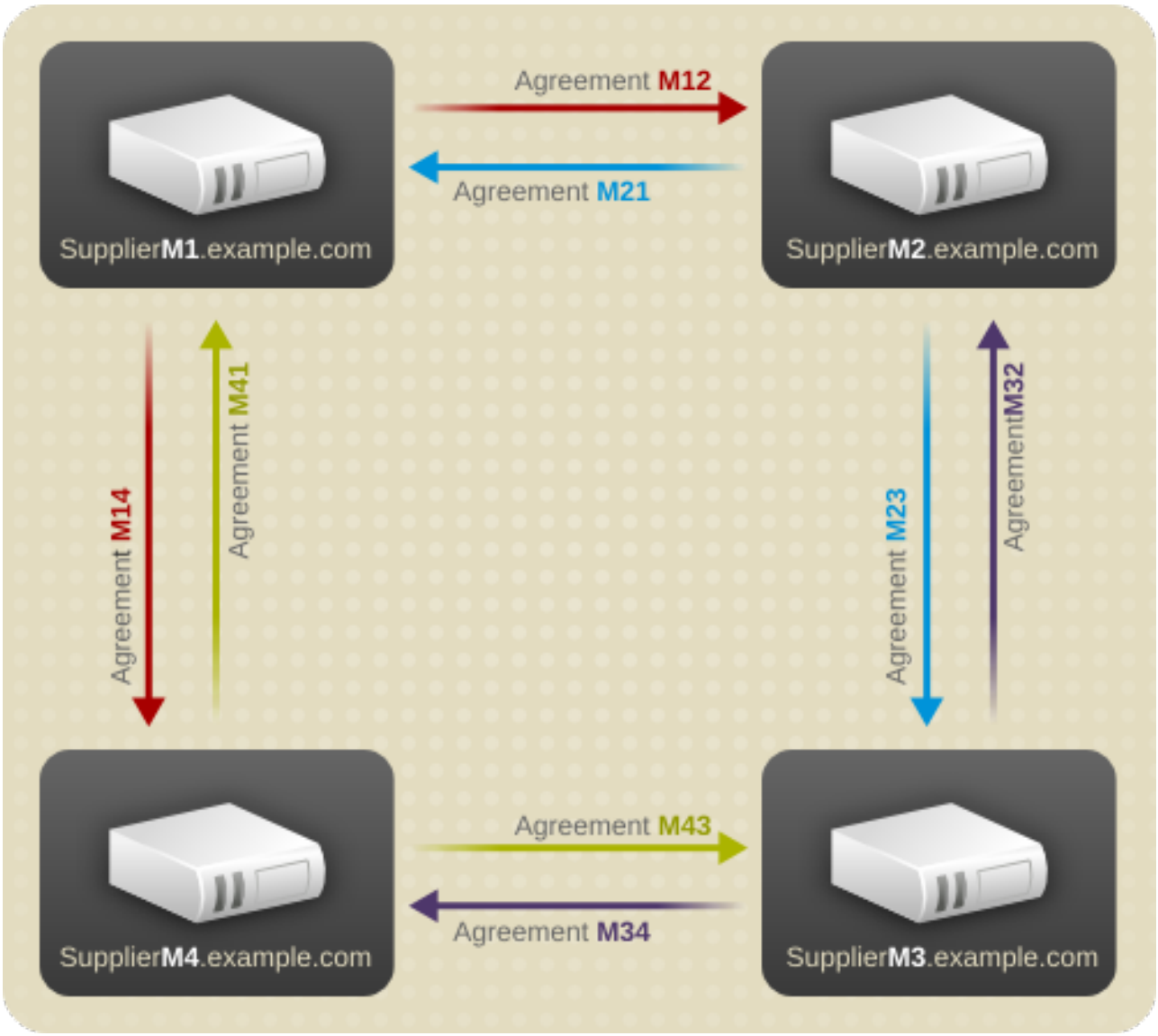


図7.5「マルチサイトのレプリケーション設定B」 は、各サプライヤーサーバーが他の2 つのサプライヤーサーバー(コンシューマーとしても機能する) にデータをフィードするトポロジーを示しています。*図7.4「マルチサプライヤーのレプリケーション設定A」* のトポロジーでは12 の合意があるのに対し、4 つのサプライヤーサーバー間には、8 つのレプリカ合意しかありません。このトポロジーは、2 つ以上のサーバーに同時に障害が発生する可能性を無視できる場合に役立ちます。

図7.5 マルチサイトのレプリケーション設定 B



これら 2 つの例は、簡素化したマルチサプライヤーのシナリオです。Red Hat Directory Server では、1 つのマルチサプライヤー環境で 20 ものサプライヤーと無制限のハブサプライヤーを持つことができるため、レプリケーションのトポロジーはより複雑になります。たとえば、図7.4「マルチサプライヤーのレプリケーション設定 A」 は、12 のレプリカ合意があります (4 つのサプライヤーにそれぞれ 3 つの合意があります)。サプライヤー数が 20 の場合、レプリカ合意は 380 になります (20 台のサーバーにそれぞれ 19 の合意があります)。

マルチサプライヤーレプリケーションを計画する際には、以下を考慮してください。

- *サプライヤーの数*
- *地理的な場所*
- *サプライヤーが他の場所にあるサーバーを更新するために使用するパス*
- *異なるサプライヤーのトポロジー、ディレクトリーツリー、およびスキーマ*
- *ネットワークの品質*
- *サーバーの負荷およびパフォーマンス*
- *ディレクトリーデータに必要な更新間隔*

7.2.3. カスケードレプリケーション

*カスケードレプリケーションのシナリオでは、**ハブサプライヤー** は、サプライヤーサーバーから更新を受け取り、コンシューマーサーバーでそれらの更新を再生します。ハブサプライヤーはハイブリッドで、通常のコンシューマーサーバーなどの読み取り専用レプリカを保持し、一般的なサプライヤーサーバーなどの changelog を維持します。*

ハブサプライヤーは、元のサプライヤーからサプライヤーデータを受信すると、これを転送します。同様に、ハブサプライヤーがディレクトリークライアントから更新リクエストを受け取ると、クライアントをサプライヤーサーバーに照会します。

カスケードレプリケーションは、組織内の様々な場所のネットワーク接続の一部が他よりも優れている場合に有効です。たとえば、Example Corp. では、ディレクトリーデータのメインコピーをミネアポリスに、コンシューマーサーバーをニューヨークとシカゴに置いています。ミネアポリスとニューヨークの間のネットワーク接続は非常に良いですが、ミネアポリスとシカゴの間の接続は悪いです。ニューヨークとシカゴの間のネットワークは正常であるため、Example の管理者はカスケードレプリケーションを使用して、ディレクトリーデータをミネアポリスからニューヨーク、そしてシカゴへと移動させます。

図7.6 カスケードレプリケーションのシナリオ

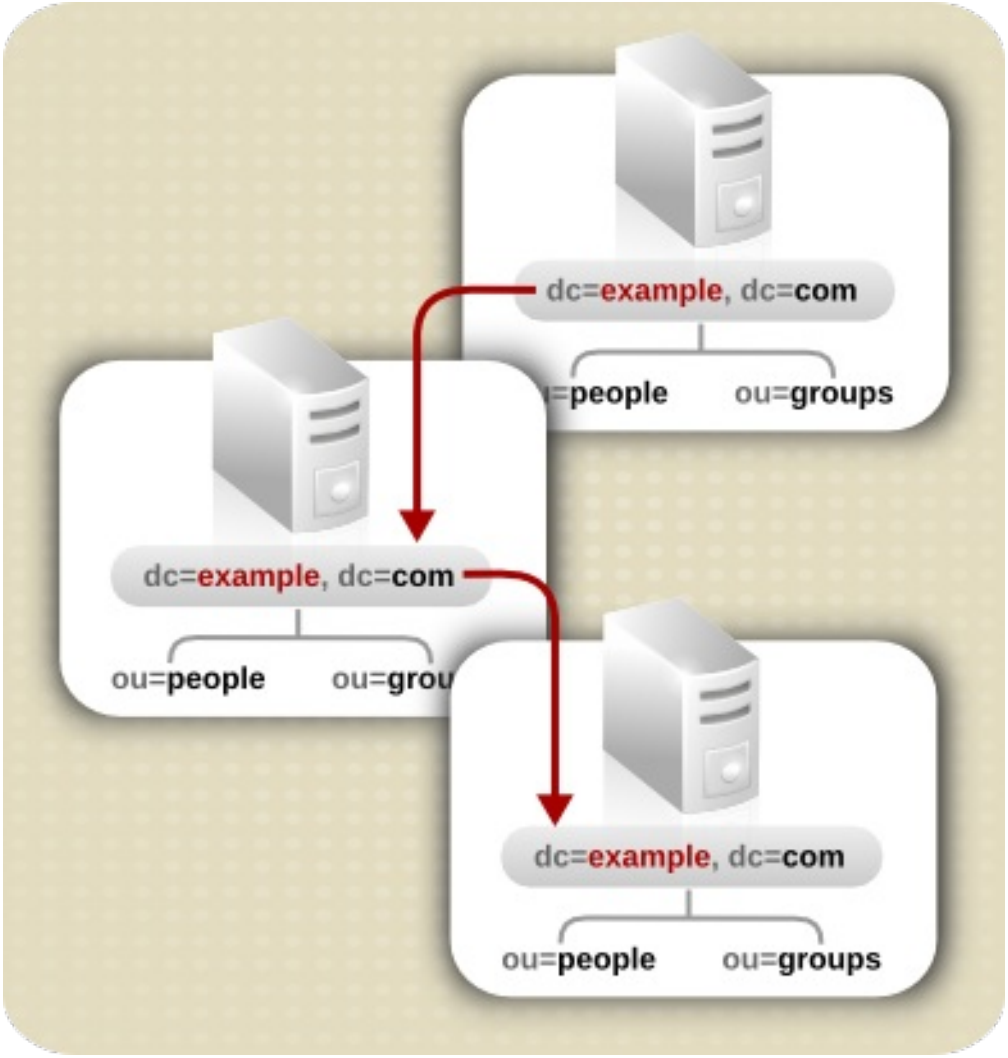
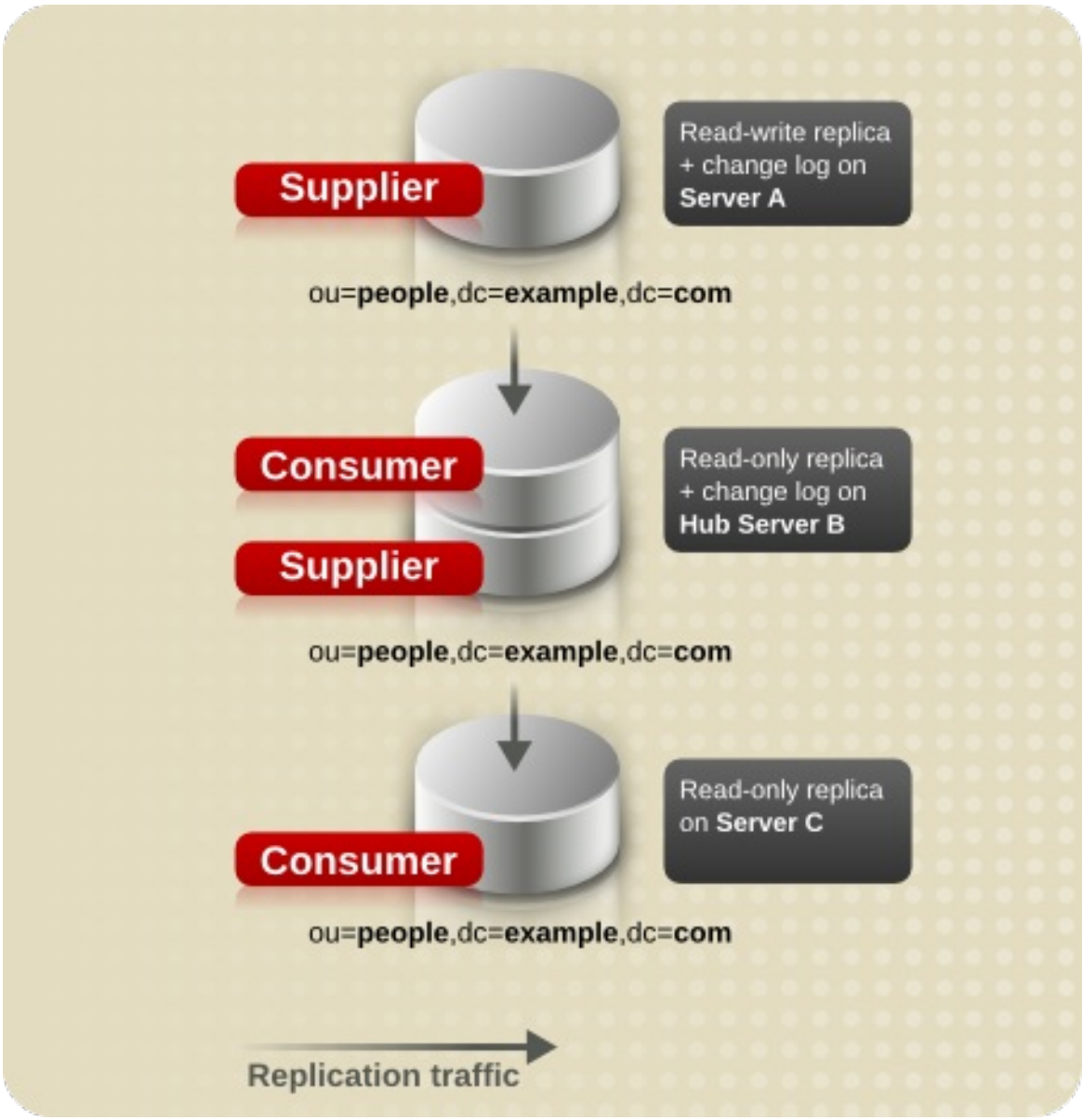


図7.7「カスケードレプリケーションにおけるレプリケーショントラフィックと changelog」 は、同じシナリオを別の視点から説明したもので、各サーバーでレプリカがどのように設定されているか (読み取り/書き込みか、または読み取り専用か)、どのサーバーが changelog を維持しているかを示しています。

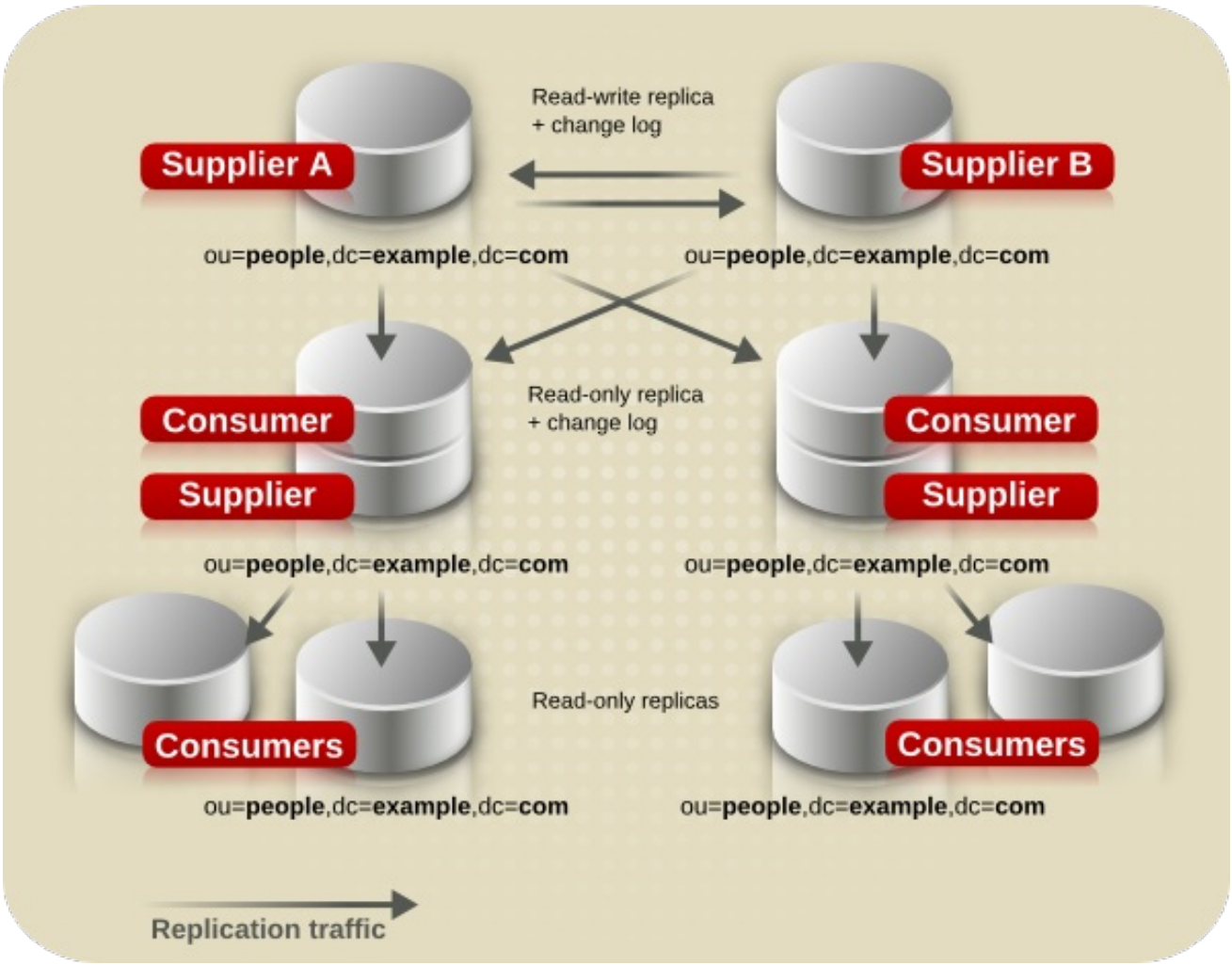
図7.7 カスケードレプリケーションにおけるレプリケーショントラフィックと changelog



7.2.4. 混合環境

ネットワークやディレクトリー環境のニーズに合わせて、いずれのレプリケーションシナリオも組み合わせることができます。一般的な組み合わせの1つとして、マルチサプライヤー設定をカスケード設定と使用する組み合わせがあります。

図7.8 マルチサプライヤーとカスケードレプリケーションの組み合わせ



7.3. レプリケーションストラテジーの定義

レプリケーションストラテジーは、提供する必要のあるサービスで決定されます。レプリケーションストラテジーを確認するには、ネットワーク、ユーザー、アプリケーション、ディレクトリーサービスの使用方法などの調査から始めます。

- *ネットワーク内のリソース、トラフィックの負荷、ディレクトリーサービスのリソース要件を評価します。*

  「レプリケーションサーベイの実施」、「レプリケーションリソース要件」、および「マルチサプライヤーレプリケーションに必要なディスク領域の管理」 を参照してください。

- *会社の場所やセクションごとに複数のコンシューマーがいる場合や、一部のサーバーがセキュアでない場合は、**一部レプリケーション** を使用して、機密情報やめったに変更されない情報を除外することで、機密情報を損なうことなくデータの整合性を保つことができます。*

  詳細は、「一部レプリケーションで選択された属性を複製する」 を参照してください。

- *ネットワークが地理的に広い地域に広がっている場合、複数のサイトに複数の Directory Server があり、ローカルのデータサプライヤーはマルチサプライヤーレプリケーションで接続されています。*

  詳細は、「ワイドエリアネットワーク全体でのレプリケーション」 を参照してください。

- 高可用性が主な懸念である場合は、1 つのサイトに複数の Directory Server を持つデータセンターを作成します。単一サプライヤーレプリケーションは、read-failover を提供しますが、マルチサプライヤーレプリケーションは write-failover を提供します。

  詳細は、「高可用性でのレプリケーションの使用」を参照してください。

- ローカル可用性が主要な懸念事項である場合は、レプリケーションを使用して、世界中の現地事務所の Directory Server にデータを地理的に分散します。すべての情報のメインコピーは、本社などの 1 つの場所に保持することも、現地事務所 で DIT の関連する部分を管理することもできます。

  詳細は、「ローカル可用性でのレプリケーションの使用」を参照してください。

- いずれの場合も、Directory Server によって処理される要求の負荷を分散し、ネットワークの輻輳を回避します。

  詳細は、「ロードバランシングでのレプリケーションの使用」を参照してください。

レプリケーションストラテジーを計画したら、ディレクトリーサービスをデプロイすることができます。管理者は、ディレクトリーサービスにかける負荷に応じてディレクトリーサービスを調整できるため、ディレクトリーサービスを段階的にデプロイすることが推奨されます。負荷分析がすでに動作しているディレクトリーに基づいている場合を除き、ディレクトリーに対する実際の需要が明確になったときにディレクトリーサービスを変更できるようにしてください。

7.3.1. レプリケーションサーベイの実施

サイト調査でネットワークの品質と使用状況に関する情報を収集して、レプリケーションストラテジーの定義に役立てます。

- さまざまな建物やリモートサイトを接続する LAN と WAN の品質、および使用可能な帯域幅の量。

- ユーザーの物理的な場所、各サイトのユーザー数、使用状況パターン。これがディレクトリーサービスの使用目的になります。

- ディレクトリーサービスにアクセスするアプリケーションの数。読み取り、検索、および比較操作と書き込み操作の相対的な割合。

- メッセージングサーバーがディレクトリーを使用する場合は、処理する電子メールメッセージごと実行する操作の数を調べます。ディレクトリーサービスに依存するその他の製品は、通常、認証アプリケーションやメタディレクトリーアプリケーションなどの製品です。それぞれについて、ディレクトリーサービスで実行される操作の種類および頻度を決定します。

- ディレクトリーサービスに保存されているエントリーの数およびサイズ。

人事データベースまたは財務情報管理するサイトは、電話帳の目的でのみディレクトリーを使用する技術者が含まれるサイトよりも、ディレクトリーサービスの負荷を大きくする可能性があります。

7.3.2. 一部レプリケーションで選択された属性を複製する

フラクショナルレプリケーションにより、管理者はサプライヤーからコンシューマー (または別のサプライヤー) に送信されない属性セットを選択できます。したがって、管理者は、含まれるすべての情報を複製せずにデータベースを複製できます。

一部レプリケーションは、レプリケーション合意ごとに有効になり、設定されます。属性の除外は、すべてのエントリーに等しく適用されます。コンシューマーサーバーに関する限り、除外された属性には常に値がありません。そのため、コンシューマーサーバーに対する検索を実行するクライアントは、除

外された属性を見ることはありません。同様に、フィルターにそれらの属性を指定して検索を行うと、一致するエントリーはありません。

一部レプリケーションは、以下のような状況で特に便利です。

- コンシューマーサーバーが低速なネットワークを使用して接続されている場合、頻繁に変更されていない属性や **jpegPhoto** などの大きな属性により、ネットワークトラフィックが少なくなります。
- コンシューマーサーバーがパブリックインターネットなどの信頼できないネットワーク上に配置されている場合、電話番号などの機密属性を除外すると、サーバーのアクセス制御手段が無効になったり、マシンが攻撃者によって悪用されても、これらの属性にアクセスしないことを保証する追加レベルの保護が提供されます。

一部レプリケーションの設定は、『Administration Guide』の第 8 章 Managing Replication のレプリケーション契約およびサプライヤーの設定のセクションで説明しています。

7.3.3. レプリケーションリソース要件

レプリケーションを使用すると、より多くのリソースが必要になります。レプリケーションストラテジーを定義する際に、以下のリソース要件を検討してください。

- ディスク使用量 - サプライヤーサーバーでは、変更ログは各更新操作の後に書き込まれます。多数の更新操作を受信するサプライヤーサーバーでは、ディスク使用量が増える可能性があります。

注記

各サプライヤーサーバーは 1 つの changelog を使用します。サプライヤーに複数の複製されたデータベースが含まれる場合、changelog はより頻繁に使用され、ディスク使用量がさらに高くなります。

- 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 1 つのサーバースレッドを消費します。そのため、クライアントアプリケーションが使用できるスレッドの数が減少し、クライアントアプリケーションのサーバーパフォーマンスに影響を与える可能性があります。
- ファイル記述子 - サーバーで利用可能なファイル記述子の数は、変更ログ (1 つのファイル記述子) および各レプリケーションアグリーメント (アグリーメントごとに 1 つのファイル記述子) によって削減されます。

7.3.4. マルチサプライヤーレプリケーションに必要なディスク領域の管理

マルチサプライヤーレプリカは、ディレクトリー編集の changelog、更新エントリーの状態情報、削除されたエントリーの tombstone エントリーなど、追加のログを保持します。この情報は、マルチサプライヤーレプリケーションを実行するために必要です。これらのログファイルは非常に大きくなる可能性があるため、ディスク領域を残すためにこれらのファイルを定期的にクリーンアップする必要があります。

マルチサプライヤーレプリカに対して changelog メンテナンスを設定できる属性は 4 つあります。2 つは **cn=changelog5** の下にあり、変更ログのトリミングに直接関連します。

- **nsslapd-changelogmaxage** changelog のエントリーの最大期間を設定します。エントリーがその制限より古い場合は、削除されます。これにより、変更ログが無期限に大きくなるのを防ぎます。

- ***nsslapd-changelogmaxentries*** *changelog で許可されるエントリーの最大数を設定します。**nsslapd-changelogmaxage** と同様に、changelog もトリミングされますが、設定に注意してください。これは、ディレクトリー情報の完全なセットを許可するのに十分な大きさである必要があります。そうしないと、マルチサプライヤーのレプリケーションが正しく機能しない可能性があります。*

*他の 2 つの属性は、**cn=replica, cn=**suffixDN, **cn=mapping tree, cn=config** のレプリケーションアグリーメントエントリーの下にあります。これらの 2 つの属性は、ディレクトリーの編集情報ではなく、changelog に保持されるメンテナンス情報である tombstone および状態情報に関連します。*

- ***nsDS5ReplicaPurgeDelay*** *tombstone （削除済み）エントリーおよび状態情報が changelog に設定可能な最大期間を設定します。tombstone または状態情報エントリーがその時間よりも古くなると、削除されます。**nsslapd-changelogmaxage** の値は、tombstone および状態情報エントリーにのみ適用される点で **nsDS5ReplicaPurgeDelay** 属性とは異なります。**nsslapd-changelogmaxage** は、ディレクトリーの変更など、変更ログ内のすべてのエントリーに適用されます。*

- ***nsDS5ReplicaTombstonePurgeInterval*** *サーバーがパージ操作を実行する頻度を設定します。この間隔で、Directory Server は内部操作を実行して、tombstone および状態のエントリーを削除します。最大経過時間が最長のレプリケーション更新スケジュールよりも長いことを確認してください。そうしないと、マルチサプライヤーレプリケーションがレプリカを適切に更新できない場合があります。*

レプリケーションおよび変更を管理するパラメーターの説明は、
『Configuration, Command, and File Reference』の第 2 章 Core Configuration Attributes にあります。

7.3.5. ワイドエリアネットワーク全体でのレプリケーション

ワイドエリアネットワークは通常、ローカルエリアネットワークよりも遅延や帯域幅遅延積が大きく、速度が遅くなります。Directory Server は、サプライヤーとコンシューマーがワイドエリアネットワークを使用して接続されている場合に効率的なレプリケーションをサポートします。

以前のバージョンの Directory Server では、サプライヤーとコンシューマー間のエントリーおよび更新の送信に使用されたレプリケーションプロトコルは、サプライヤーは 1 つの更新操作のみを送信し、コンシューマーからの応答を待つため、レイテンシーの影響を大きく受けていました。これにより、スループットが低下し、レイテンシーが長くなりました。

サプライヤーは、応答を待たずに、多くの更新とエントリーをコンシューマーに送信します。したがって、レイテンシーが高いネットワークでは、多くのレプリケーション操作がネットワーク上で転送されている可能性があり、レプリケーションスループットはローカルエリアネットワークで達成できるスループットと同様です。

注記

サプライヤーが 7.1 よりも前のバージョンの Red Hat Directory Server を実行する別のサプライヤーに接続されている場合は、互換性のために以前のレプリケーションメカニズムにフォールバックします。したがって、遅延の影響を受けないレプリケーションを実現するには、サプライヤーサーバーとコンシューマーサーバーの両方で少なくともバージョン 7.1 を実行する必要があります。

Directory Server とネットワーク接続の効率の両方について、パフォーマンスとセキュリティーの両方の問題を考慮する必要があります。

- インターネットなどのパブリックネットワークを介してレプリケーションを実行する場合は、TLS を使用することが強く推奨されます。これにより、レプリケーショントラフィックの盗聴が防止されます。

- ネットワークには T-1 以上のインターネット接続を使用してください。

- ワイドエリアネットワークを介したレプリケーションアグリーメントを作成するときは、サーバー間を持続的に同期しないでください。レプリケーショントラフィックは帯域幅を大量に消費し、ネットワークとインターネット接続全体の速度を低下させる可能性があります。

- コンシューマーを初期化するときは、コンシューマーをすぐに初期化しないでください。代わりに、ファイルシステムレプリカの初期化を利用します。これは、オンライン初期化やファイルからの初期化よりもはるかに高速です。ファイルシステムレプリカの初期化の使用方法については、『Red Hat Directory Server Administration Guide 』を参照してください。

7.3.6. 高可用性で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、単一のサーバーが失われることでディレクトリーサービスが使用できなくならないようにします。少なくとも、ローカルディレクトリーツリーを 1 つ以上のバックアップサーバーにレプリケートします。

一部のディレクトリーアーキテクトは、データの信頼性を最大限にするには、情報を物理的な場所ごとに 3 回複製する必要があると主張しています。フォールトトレランスにレプリケーションを使用する範囲は、環境と個人の好みによって異なりますが、ディレクトリーサービスで使用されるハードウェアとネットワークの品質に基づいてこれを決定します。信頼性の低いハードウェアには、より多くのバックアップサーバーが必要です。

注記

通常のデータバックアップポリシーの代わりにレプリケーション を使用**しないでください**。ディレクトリーデータのバックアップに関する詳細は『Red Hat Directory Server Administration Guide』を参照してください。

すべてのディレクトリークライアントの書き込みフェイルオーバーを保証するには、マルチサプライヤーレプリケーションシナリオを使用します。read-failover で十分な場合は、単一サプライヤーレプリケーションを使用します。

LDAP クライアントアプリケーションは通常、1 つの LDAP サーバーのみを検索するように設定できます。異なる DNS ホスト名にある LDAP サーバーを介してローテーションするカスタムクライアントアプリケーションがなければ、LDAP クライアントアプリケーションは、Directory Server の単一の DNS ホスト名のみを検索するように設定できます。したがって、DNS のラウンドロビンまたはネットワークソートを使用して、バックアップ Directory Server へのフェイルオーバーを提供する必要があるでしょう。DNS のラウンドロビンまたはネットワークソートの設定および使用に関する詳細は、DNS のドキュメントを参照してください。

7.3.7. ローカル可用性でのレプリケーションの使用

ローカル可用性をレプリケートする必要性は、ネットワークの質とサイトのアクティビティーによって決まります。さらに、ディレクトリーサービスに含まれるデータの性質と、そのデータが一時的に利用できなくなった場合の企業への影響を慎重に検討してください。データがミッションクリティカルであるほど、ネットワーク接続の不良によるシステムの停止に対する耐性が低くなります。

以下の理由により、ローカル可用性のレプリケーションを使用します。

- データのローカルメインコピーを保持する。

これは、特定の国の社員のみに関連するディレクトリー情報を維持する必要がある大規模な多国籍企業にとって重要なストラテジーです。データのローカルメインコピーを保持することは、データを部門または組織レベルで管理する企業においても重要です。

- 信頼できな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への対策。

  国際ネットワークで頻繁に発生しますが、信頼性の低い WAN がある場合は、ネットワーク接続が断続的になる可能性があります。

- ディレクトリーサービスのパフォーマンスを大幅に低下させる可能性のある、定期的で非常に重いネットワーク負荷を補正。

  ネットワークが古い企業でもパフォーマンスに影響が出る可能性があり、通常の営業時間中にこのような状態が発生する可能性があります。

7.3.8. ロードバランシングでのレプリケーションの使用

レプリケーションは、複数の方法で Directory Server の負荷のバランスを取ることができます。

- 複数のサーバーにユーザーの検索アクティビティーを分散させます。
- サーバーを読み取り専用にします (書き込みはサプライヤーサーバーでのみ行われます)。
- メールサーバーのアクティビティーのサポートなど、特定のタスク専用の特別なサーバーを提供します。

ネットワークのワークロードの分散は、ディレクトリーデータのレプリケーションにより実行される重要な機能です。可能な限り、適度に高速で信頼性の高いネットワーク接続を使用してアクセスできるサーバーにデータを移動します。最も重要な考慮事項は、サーバーとディレクトリーユーザー間のネットワーク接続の速度と信頼性です。

通常、ディレクトリーエントリーのサイズは平均で約 1 キロバイト (KB) です。したがって、ディレクトリールックアップを行うたびにネットワークの負荷に約 1KB が追加されます。ディレクトリーユーザーが 1 日に 10 回のディレクトリールックアップを実行すると、ディレクトリーユーザーごとに 1 日あたり約 10KB のネットワーク負荷が追加されます。サイトの WAN の速度が遅い、負荷が高い、または信頼性が低い場合は、ディレクトリーツリーをローカルサーバーにレプリケートすることを検討してください。

また、ローカルで利用可能なデータの利点が、レプリケーションによって増加するネットワーク負荷のコストに見合うかどうかも検討してください。たとえば、ディレクトリーツリー全体がリモートサイトに複製されると、ユーザーのディレクトリールックアップによって発生するトラフィックよりも大きな負担がネットワークにかかる可能性があります。これは、ディレクトリーツリーが頻繁に変更されている場合に特に該当しますが、リモートサイトで 1 日に数回のディレクトリールックアップを実行するユーザーはごくわずかです。

表7.1「ネットワークにおけるレプリケーションおよびリモートルックアップの影響」 は 100 万エントリーのディレクトリーを複製するおおよそのコストを比較しています。これらのエントリーの 10% は毎日変更され、社員が 100 人の小さなリモートサイトで 1 日あたり 10 回の検索を実行するコストと比較します。いずれの場合も、ディレクトリーエントリーの平均サイズは 1KB であると想定されています。

表7.1 ネットワークにおけるレプリケーションおよびリモートルックアップの影響

負荷タイプ	オブジェクト[a]	アクセス/日[b]	平均エントリーサイズ	負荷

負荷タイプ	オブジェクト[a]	アクセス/日[b]	平均エントリーサイズ	負荷
レプリケーション	100 万	100,000	1KB	100Mb/日
リモートルックアップ	100	1,000	1KB	1Mb/日

[a] レプリケーションの場合、**オブジェクト** はデータベースのエントリー数を参照します。リモートルックアップの場合、データベースにアクセスするユーザーの数を参照します。

[b] レプリケーションの場合、**Accesses/Day** は、レプリケートする必要があるデータベースの変更率を 10% としています。リモートルックアップの場合、各リモートユーザーの 1 日あたり 10 ルックアップに基づきます。

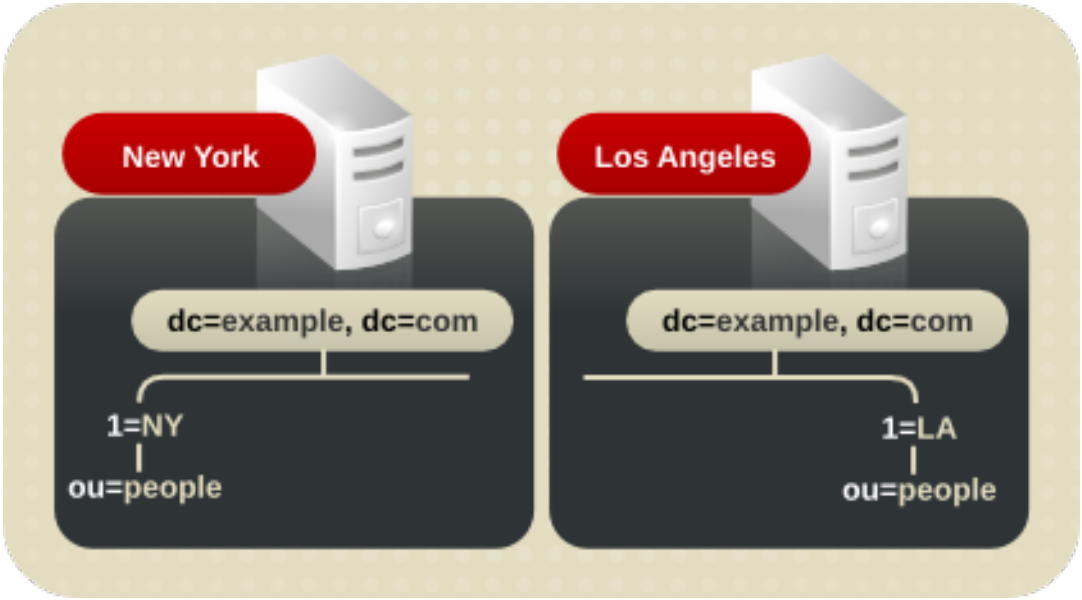
レプリケーションによって引き起こされる負荷と通常のディレクトリーの使用によって引き起こされる負荷の違いを考えると、ネットワークの負荷分散の目的でレプリケーションを使用することは適切でない場合があります。一方、ローカルで利用可能なディレクトリーデータの利点は、ネットワークの負荷に関する考慮事項をはるかに上回ります。

ローカルサイトでデータを利用できるようにすることと、ネットワークを過負荷にすることの適切な妥協点は、スケジュールされたレプリケーションを使用することです。データの一貫性とレプリケーションスケジュールの詳細は、「データの整合性」 を参照してください。

7.3.8.1. ネットワーク負荷分散の例

この例では、ニューヨークとロサンゼルスに事務所を持ち、各事務所には管理する特定のサブツリーがあります。

図7.9 リモートオフィスでのエンタープライズサブツリーの管理

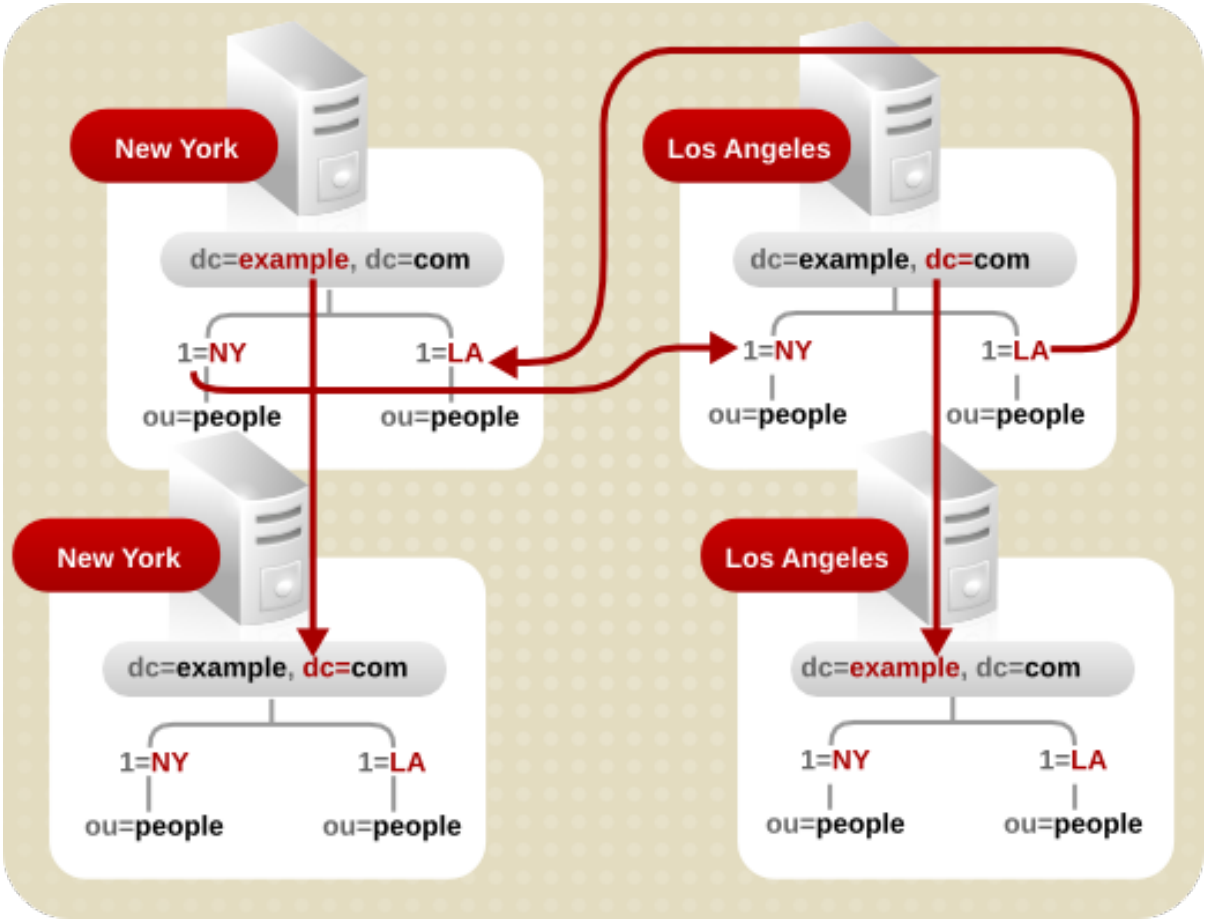


各オフィスには高速ネットワークがありますが、2 つの都市間の接続は不安定です。ネットワークの負荷のバランスを取るには、以下を行います。

1. ローカルで管理されているデータのサプライヤーサーバーとして、各オフィスで 1 台ずつサーバーを選択します。

2. *ローカルに管理されているデータを、そのサーバーからリモートオフィスの対応するサプライヤーサーバーに複製します。*

3. *各サプライヤーサーバーのディレクトリーツリー (リモートオフィスから提供されるデータを含む) を少なくとも 1 つのローカルの Directory Server に複製して、ディレクトリーデータの可用性を確保します。ローカルで管理される接尾辞にマルチサプライヤーレプリケーションを使用し、リモートサーバーからデータのメインコピーを受け取る接尾辞にはカスケードレプリケーションを使用します。*



7.3.8.2. パフォーマンス向上のための負荷分散の例

企業には以下の特性があるとします。

- *100 万人のユーザーをサポートする 150 万エントリーの Directory Server を使用しています。*
- *各ユーザーは、1 日あたり 10 個のディレクトリールックアップを実行します。*
- *1 日あたり 2,500 万通のメールを処理するメッセージングサーバーを使用します。*
- *メッセージングサーバーは、処理するメールごとに 5 つのディレクトリールックアップを行います。*

これは、ユーザーの 1 日あたり 1,000 万回のディレクトリールックアップと、メールの 1 日あたり 1.25 億回のディレクトリールックアップ (合計 1 日あたり 1.35 億 ディレクトリールックアップ) に相当します。

*たとえば、1 日の営業時間が 8 時間で、ユーザーが 4 つのタイムゾーンに分散している場合、4 つのタイムゾーンでの営業日 (またはピーク時の使用時間) は 12 時間になります。したがって、サービスは 1 日 12 時間で 1.35 億回のディレクトリールックアップに対応する必要があります。これは、1 秒間に 3,125 回のルックアップ (135,000,000 / (60*60*12)) に相当します。*

表7.2 Directory Server の読み込みの計算

アクセスタイプ	タイプ数	1 日あたりのアクセス数	合計アクセス数
ユーザー検索	100 万	10	1,000 万
メールルックアップ	2,500 万	5	1.25 億
アクセスの合計			1.35 億
合計		1.35 億 (3,125/秒)	

Directory Server を実行するハードウェアが1 秒あたり 50 0 回の読み取りをサポートする場合、この負荷をサポートするには、少なくとも 6 つまたは 7 つの Directory Server を使用する必要があります。ディレクトリーユーザーが100 万人いる企業の場合は、ローカルでの可用性を確保するために Directory Server を追加します。

レプリケーションにはいくつかの方法があります。

- すべての書き込みトラフィックを処理するために、1 つの都市のマルチサプライヤー設定に 2 つの Directory Server を配置します。

  この設定は、すべてのディレクトリーデータに単一の制御ポイントがあることを前提としています。

- これらのサプライヤーサーバーを使用して、1 つ以上のハブサプライヤーを複製します。

  ディレクトリーサービスによって処理される読み取り、検索、および比較の要求は、コンシューマーサーバーを対象にする必要があります。これにより、サプライヤーサーバーは書き込み要求を処理できるようになります。

- ハブサプライヤーを使用して、企業全体のローカルサイトに複製します。

  ローカルサイトに複製することで、サーバーおよびWAN のワークロードのバランスを取ることや、ディレクトリーデータを高可用性を確保するのに役立ちます。

- 各サイトで、少なくとも読み取り操作のために、最低1 回複製して高可用性を確保します。

- DNS ソートを使用して、ローカルユーザーがディレクトリールックアップに使用できるローカルディレクトリーサーバーを常に見つけられるようにします。

7.3.8.3. 小規模サイトのレプリケーションストラテジーの例

Example Corp. には以下の特徴があります。

- 企業全体が1 つのビルに入っています。

- ビルには、非常に高速 (毎秒 100 Mb) で、使用量の少ないネットワークがあります。

- ネットワークは非常に安定しており、サーバーハードウェアと OS プラットフォームは信頼できます。

- 単一のサーバーでサイトの負荷を簡単に処理できます。

この場合、Example Corp. は、プライマリーサーバーがメンテナンスまたはハードウェアのアップグレードでシャットダウンした場合に、少なくとも 1 回複製することを決定します。また、Directory Server の 1 つが利用できなくなった場合に、LDAP 接続のパフォーマンスを向上するために DNS ラウンドロビンを設定します。

7.3.8.4. 大規模サイトのレプリケーションストラテジーの例

Example Corp. が成長するにつれ、いくつかの変更を加えて以前の特性 (「小規模サイトのレプリケーションストラテジーの例」) を保持します。

- 会社は 2 つビルに分かれています。
- ビル間の接続は遅く、これらの接続は通常の営業時間中は非常に混雑しています。

ネットワークの変更が必要になると、Example Corp. の管理者はレプリケーションストラテジーを調整します。

- 2 つのビルのいずれかで、ディレクトリーデータのメインコピーを格納する単一のサーバーを選択します。

  このサーバーは、ディレクトリーデータのメインコピーを担当するユーザーが最も多いビルに配置する必要があります。このビルを Buidling A とします。

- ディレクトリーデータの高可用性のために、Buidling A 内で最低 1 回複製を行います。

  マルチサプライヤーレプリケーション設定を使用して、書き込みフェイルオーバーを確実に実行します。

- 別のビル (Building B) に 2 つのレプリカを作成します。
- サプライヤーサーバーとコンシューマーサーバーの間で厳密な一貫性を保つ必要がない場合は、オフピーク時にのみ実行されるようにレプリケーションをスケジュールします。

7.4. 他の DIRECTORY SERVER 機能でのレプリケーションの使用

レプリケーションは、その他の Directory Server 機能に対応して、高度なレプリケーション機能を提供します。次のセクションでは、レプリケーションストラテジーをより適切に設計するための機能の連携について説明します。

7.4.1. レプリケーションおよびアクセス制御

ディレクトリーサービスは ACI をエントリーの属性として保存します。つまり、ACI は他のディレクトリー内容とともに複製されます。Directory Server は ACI をローカルで評価するため、これは重要になります。

ディレクトリーのアクセス制御の設計に関する詳細は、9章セキュアなディレクトリーの設計 を参照してください。

7.4.2. レプリケーションおよび Directory Server プラグイン

レプリケーションは、Directory Server に同梱されるほとんどのプラグインと連携します。次のプラグインを使用したマルチサプライヤーレプリケーションの場合、いくつかの例外と制限があります。

- Attribute Uniqueness プラグイン

*Attribute Uniqueness プラグインは、ローカルエントリーに追加された属性値を検証し、すべての値が一意であることを確認します。ただし、このチェックはサーバー上で直接行われ、他のサプライヤーから複製されることはありません。たとえば、Example Corp. では、**mail** 属性が一意でなければなりませんが、同じ **mail** 属性を持つ 2 人のユーザーが 2 つの異なるサプライヤーサーバーに同時に追加されます。命名の競合がない限り、レプリケーションの競合はありませんが、**mail** 属性は一意ではありません。*

- *参照整合性プラグイン*

  参照整合性は、このプラグインがマルチサプライヤーセットの 1 つのサプライヤーでのみ有効になっている場合に、マルチサプライヤーレプリケーションで機能します。これにより、参照整合性の更新が 1 つのサプライヤーサーバーのみで行われ、他のサーバーに伝播されます。

注記

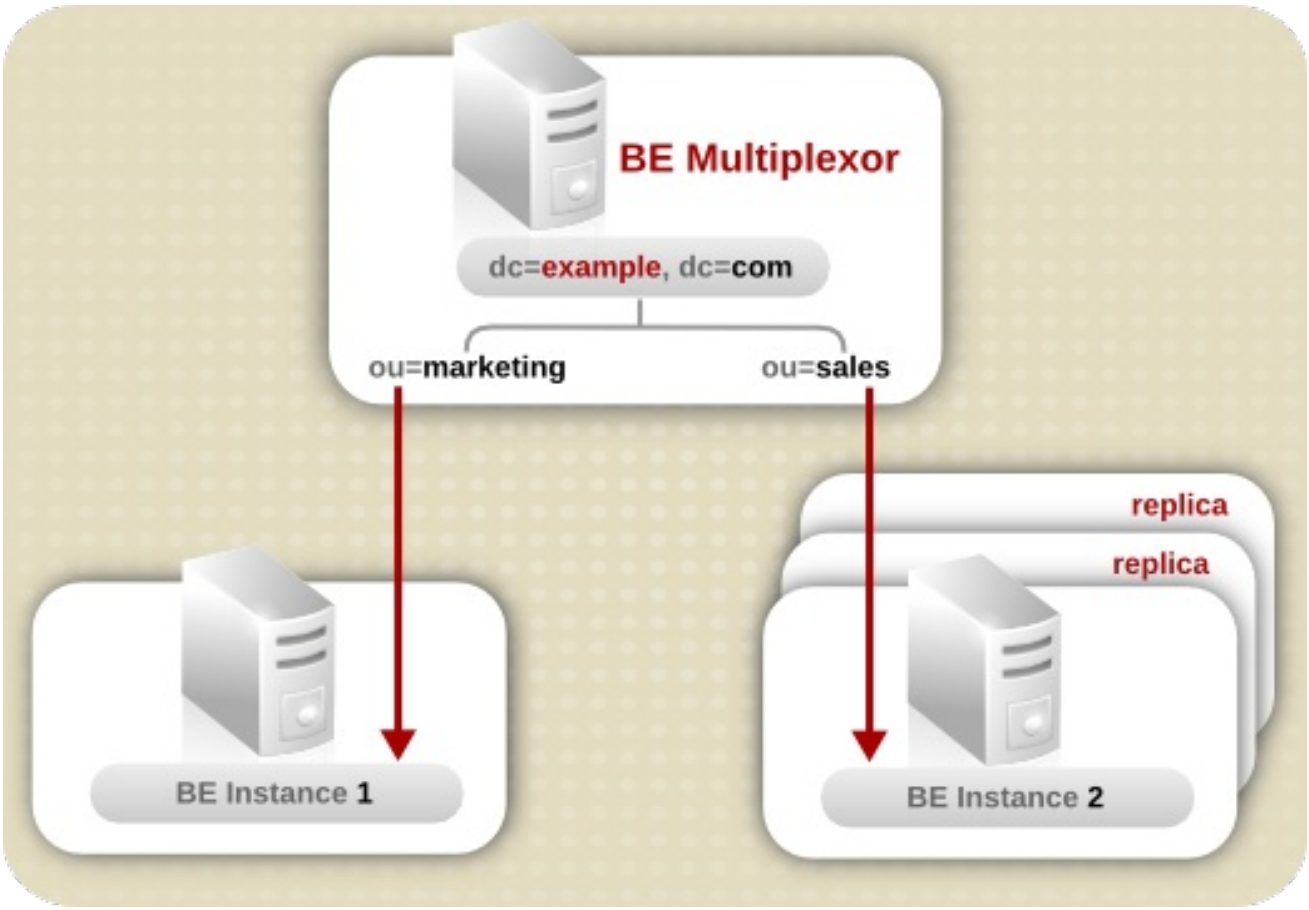
デフォルトでは、これらのプラグインは無効になっており、手動で有効にする必要があります。

7.4.3. レプリケーションおよびデータベースのリンク

ディレクトリーエントリーを配布するためにチェーンを使用すると、データベースリンクが含まれるサーバーは、実際のデータが含まれるリモートサーバーを参照します。この環境では、データベースリンク自体を複製できません。ただし、リモートサーバーの実際のデータが含まれるデータベースを複製できます。

レプリケーションプロセスは、データベースリンクのバックアップとして使用しないでください。データベースリンクは手動でバックアップする必要があります。チェーンとエントリーの配布の詳細は、6 章ディレクトリートポロジーの設計 を参照してください。

図7.10 チェーンされたデータベースの複製



7.4.4. スキーマレプリケーション

標準スキーマでは、コンシューマーサーバーにデータを複製する前に、サプライヤーサーバーは、独自のバージョンのスキーマがコンシューマーサーバーに保存されているスキーマのバージョンと同期されているかどうかを確認します。以下の条件が適用されます。

- サプライヤーとコンシューマーの両方のスキーマエントリーが同じである場合、レプリケーション操作が続行されます。
- サプライヤーサーバー上のスキーマのバージョンがコンシューマーに保存されているバージョンより新しい場合、サプライヤーサーバーはそのスキーマをコンシューマーに複製してからデータの複製を続行します。
- サプライヤーサーバーのスキーマがコンシューマーに保存されているバージョンよりも古い場合、コンシューマーのスキーマが新しいデータをサポートできないため、サーバーはレプリケーション中に多くのエラーを返すことがあります。

注記

サプライヤーとレプリカ間のスキーマが一致しない場合でも、スキーマレプリケーションは実行されます。

複製可能な変更には、Web コンソールを介して行われたスキーマへの変更、ldapmodify を介して行われた変更、および 99user.ldif ファイルに直接行われた変更が含まれます。カスタムスキーマファイル、およびカスタムスキーマファイルに追加された変更は複製されません。

コンシューマーには、スキーマが異なる 2 つのサプライヤーからの複製データが含まれる場合があります。どのサプライヤーが更新されても最後が優先され、そのスキーマがコンシューマーに伝播されます。

警告

サプライヤーサーバーは競合を解決できず、レプリケーションは失敗するため、コンシューマーサーバーでスキーマを更新しないでください。スキーマは、複製されたトポロジーのサプライヤーサーバーで維持する必要があります。

同じ Directory Server は、サプライヤーとしてのロールを果たす読み取り/書き込みレプリカと、コンシューマーとしてのロールを果たす読み取り専用レプリカを保持することができます。そのため、スキーマのサプライヤーとして機能するサーバーを常に特定し、このサプライヤーと、スキーマ情報のコンシューマーとして機能するレプリケーション環境の他のサーバーとの間にレプリケーションアグリーメントを設定します。

スキーマを複製するために特別なレプリケーションアグリーメントは必要ありません。サプライヤーとコンシューマーの間でレプリケーションが設定されていると、スキーマレプリケーションがデフォルトで実行されます。

スキーマ設計の詳細は、3章ディレクトリースキーマの設計 を参照してください。

カスタムスキーマ

*標準の **99user.ldif** ファイルがカスタムスキーマに使用される場合、これらの変更はすべてのコンシューマーに複製されます。*

*すべてのサーバーで同じスキーマファイルの情報を維持するには、カスタムスキーマファイルを各サーバーにコピーする必要があります。カスタムスキーマファイル、およびこれらのファイルへの変更は、Web コンソールまたは **ldapmodify** を介して加えられても複製されません。*

カスタムスキーマファイルがある場合は、サプライヤーでの変更後にこれらのファイルがすべてのサーバーにコピーされていることを確認します。すべてのファイルがコピーされたら、サーバーを再起動します。

カスタムスキーマファイルの詳細は、「カスタムスキーマファイルの作成」 を参照してください。

7.4.5. レプリケーションおよび同期

Directory Server 全体で、同期された Windows エントリーを伝播するには、マルチサプライヤー環境内で同期を使用します。同期アグリーメントは、可能な限り最小限に抑える必要があり、できればデプロイメントごとに 1 つにします。マルチサプライヤーレプリケーションにより、データアクセスポイントを単一のディレクトリーサーバーに制限しながら、Windows 情報をネットワーク全体で利用できるようになります。

第8章 同期の設計

既存のサイト(「サイト調査の実行」)のサイト調査を実施する際に考慮すべき重要な要素は、Active Directory ディレクトリーサービスの構造とデータ型を含めることです。Windows Sync を使用すると、既存の Windows ディレクトリーサービスを Directory Server と同期および統合できます。これには、Directory Server での Windows アカウントの作成、変更、および削除が含まれ、逆に Windows での Directory Server アカウントの作成、変更、および削除も含まれます。これにより、ディレクトリーサービス全体でディレクトリー情報の整合性を維持するための効率的かつ効果的な方法が提供されます。

8.1. WINDOWS 同期の概要

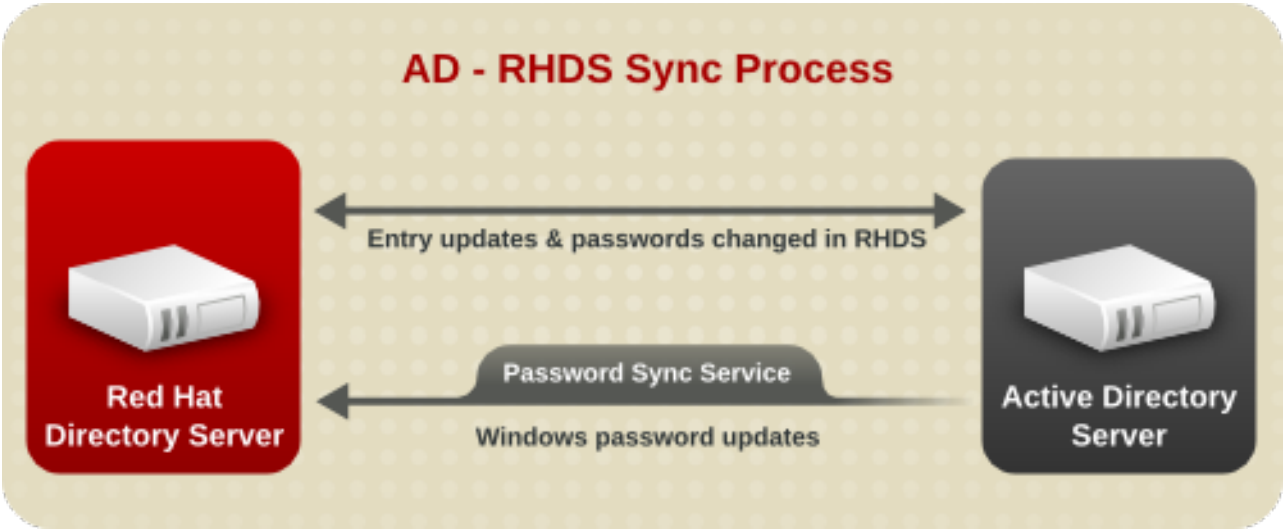
同期プロセスはレプリケーションプロセスと似ています。これは、プラグインによって有効にされ、同期アグリーメントを通じて設定および開始されます。ディレクトリー変更の記録が維持され、そのログに従って更新が送信されます。

Windows 同期プロセスを完了するには、次の 2 つの部分があります。

- **ユーザーおよびグループの同期** マルチサプライヤーレプリケーションと同様に、ユーザーとグループのエントリーは、デフォルトで有効になっているプラグインを介して同期されます。マルチサプライヤーレプリケーションに使用されるものと同じ changelog は、LDAP 操作として Directory Server から Windows 同期ピアサーバーに更新を送信するためにも使用されます。サーバーは、Windows サーバーに対して LDAP 検索操作を実行し、Windows エントリーに加えた変更を対応する Directory Server エントリーと同期します。

- **Password Sync.** このアプリケーションは、Windows ユーザーのパスワードの変更をキャプチャーし、これらの変更を LDAPS 経由で Directory Server に中継します。Active Directory マシンにインストールする必要があります。

図8.1 同期プロセス



8.1.1. 同期合意

同期は、1 つ以上の**同期アグリーメント**によって設定および制御されます。これらは、レプリケーションアグリーメントと目的が似ており、Windows サーバーのホスト名とポート番号、同期されているサブツリーなどの、同様の情報セットが含まれています。Directory Server は、LDAP または LDAP over TLS を使用してピア Windows サーバーに接続し、更新を送受信します。

単一の Windows サブツリーが単一の Directory Server サブツリーと同期され、その逆も同様です。**データベース** に接続するレプリケーションとは異なり、同期はディレクトリーツリー構造の **接尾辞** 間で行

われます。したがって、ディレクトリーツリーを設計する場合は、Directory Server と同期する必要のある Windows サブツリーを考慮し、対応する Directory Server サブツリーを設計または追加します。同期された Windows および Directory Server の接尾辞は、両方同期アグリーメントで指定されます。指定された接尾辞の直接の子ではないエントリーなど、それぞれのサブツリー内のすべてのエントリーを同期に使用できます。

注記

子孫コンテナーエントリーは、管理者が Windows サーバーで個別に作成する必要があります。Windows Sync は、コンテナーエントリーを作成しません。

8.1.2. changelog

Directory Server は、発生した変更を記録するデータベースである **changelog** を維持します。changelog は、Windows 同期ピアサーバーに追加された変更を調整および送信するために、Windows 同期により使用されます。Windows サーバーのエントリーへの変更は、Active Directory の Dirsync 検索機能を使用して確認できます。Active Directory 側には changelog がないため、Dirsync 検索はデフォルトでは 5 分ごとに定期的に発行されます。Dirsync を使用すると、以前の検索以降に変更されたエントリーのみが取得されます。

8.2. サポート対象の ACTIVE DIRECTORY のバージョン

Windows 同期およびパスワード同期サービスは、32 ビットおよ 64 ビットプラットフォームの Windows 2008 R2 および Windows 2012 R2 でサポートされます。

8.3. WINDOWS 同期の計画

データの整理やレプリケーションの計画のためのサイト調査と同様に、同期を設定する前に、情報の種類、Windows サーバー、およびその他の考慮事項を評価しておくと便利な場合があります。

8.3.1. リソース要件

同期はサーバーリソースを使用します。レプリケーションストラテジーを定義する際に、以下のリソース要件を検討してください。

- ディスク使用量 - changelog は各更新操作の後に書き込まれます。多数の更新操作を受信するサーバーでは、ディスク使用量が増える可能性があります。さらに、すべてのレプリケーションデータベースと同期データベースに対して単一の changelog が維持されます。サプライヤーに複数の複製および同期されたデータベースが含まれる場合、changelog はより頻繁に使用され、ディスク使用量がさらに高くなります。

- サーバースレッド - 同期アグリーメントは 1 つのサーバースレッドを使用します。

- ファイル記述子 - サーバーで利用可能なファイル記述子の数は、変更ログ (1 つのファイル記述子) および各レプリケーションおよび同期アグリーメント (アグリーメントごとに 1 つのファイル記述子) によって削減されます。

- さまざまな建物やリモートサイトを接続する LAN と WAN の品質、および使用可能な帯域幅の量。

- ディレクトリーに保存されたエントリーの数およびサイズ。

人事データベースまたは財務情報管理するサイトは、簡単な電話帳の目的でディレクトリーを使用する技術者が含まれるサイトよりも、ディレクトリーの負荷を大きくする可能性があります。

8.3.2. changelog のディスク領域の管理

マルチサプライヤーレプリケーションと同様に、同期には、ディレクトリーの編集を追跡するための changelog、更新エントリーの状態情報のログエントリー、および削除されたエントリーのトゥームストーンエントリーが必要です。この情報は同期に必要です。これらのログファイルは非常に大きくなる可能性があるため、ディスク領域を残すためにこれらのファイルを定期的にクリーンアップする必要があります。

changelog を維持できる属性は 4 つあります。2 つは **cn=changelog5** の下にあり、変更ログのトリミングに直接関連します。

- **nsslapd-changelogmaxage** changelog のエントリーの最大期間を設定します。エントリーがその制限より古い場合は、削除されます。これにより、変更ログが無期限に大きくなるのを防ぎます。

- **nsslapd-changelogmaxentries** changelog で許可されるエントリーの最大数を設定します。**nsslapd-changelogmaxage** と同様に、changelog もトリミングされますが、設定に注意してください。これは、ディレクトリー情報の完全なセットを許可するのに十分な大きさである必要があります。そうしないと、同期が正しく機能しない可能性があります。

他の 2 つの属性は、**cn=**sync_agreement**,cn=WindowsReplica,cn=**suffixDN**,cn=mapping tree,cn=config** の同期合意エントリーの下にあります。これらの 2 つの属性は、ディレクトリーの編集情報ではなく、changelog に保持されるメンテナンス情報である tombstone および状態情報に関連します。

- **nsDS5ReplicaPurgeDelay** tombstone （削除済み）エントリーおよび状態情報が changelog に設定可能な最大期間を設定します。tombstone または状態情報エントリーがその時間よりも古くなると、削除されます。**nsslapd-changelogmaxage** の値は、tombstone および状態情報エントリーにのみ適用される点で **nsDS5ReplicaPurgeDelay** 属性とは異なります。**nsslapd-changelogmaxage** は、ディレクトリーの変更など、変更ログ内のすべてのエントリーに適用されます。

- **nsDS5ReplicaTombstonePurgeInterval** サーバーがパージ操作を実行する頻度を設定します。この間隔で、Directory Server は内部操作を実行して、tombstone および状態のエントリーを削除します。最大経過時間が最長のレプリケーション更新スケジュールよりも長いことを確認してください。そうしないと、マルチサプライヤーレプリケーションがレプリカを適切に更新できない場合があります。

レプリケーションおよび変更を管理するパラメーターの説明は、
『Configuration, Command, and File Reference』の第 2 章 Core Configuration Attributes にあります。

8.3.3. 接続型の定義

同期は、標準ポートを介した単純な認証、TLS / TLS、または Start TLS (標準ポートを介した安全な接続) を使用して行うことができます。

これは必須ではありませんが、TLS や他のセキュアな接続を同期に使用することが強く推奨されます。パスワードが Windows サーバーから同期される場合は、同期が安全なポートを介して続行されるように、両方のサーバーで TLS を有効にする必要があります。

8.3.4. データサプライヤーの検討

データサプライヤー は、データのサプライヤーソースであるサーバーです。これは、データのプライマリーソースまたは権威ソースです。

Windows および Directory Server サービスは、同期アグリーメントにより継続的に同期されるため、2

つのサービス間で競合の可能性が最小限に抑えられます。ただし、Directory Server がレプリケーションデプロイメントの一部である場合は、レプリケーションスケジュールによっては、Directory Server レプリケーションシナリオ内と Windows ドメインの間で行われた変更で競合が発生する可能性があります。

データが2 つの異なるディレクトリーサービスに存在する場合、どのサーバーがデータサプライヤーになるかを検討し、その情報を共有するかを決定します。最適なのは、データを管理する1 つのディレクトリーサービスを選択し、同期プロセスで他のサービスのエントリーを追加、更新、または削除できるようにすることです。

データの管理には、1 つの領域 (Windows ドメインまたは Directory Server) を選択します。または、データサプライヤーとして Directory Server を1 つ選択し、各 Windows ドメインと同期します。Directory Server がレプリケーションに関与する場合は、競合、データの損失、またはデータの上書きを回避するようにレプリケーション構造を設計します。

データのメインコピーの維持方法は、特定のデプロイメントのニーズによって異なります。データサプライヤーがどのように維持されているかに関係なく、単純性と一貫性を維持します。たとえば、複数のサイトでデータの管理を試みないでください。その場合、競合するアプリケーション間でデータを自動的に交換します。これを行うと、last change wins のシナリオが発生し、管理のオーバーヘッドが増加します。

8.3.5. 同期するサブツリーの決定

単一の Directory Server サブツリーのみを単一の Windows サブツリーと同期できます。また、ディレクトリーサービス間で同期アグリーメントを1 つのみ使用することが推奨されます。同期するディレクトリーツリーの一部を選択または設計します。同期されたエントリー専用の接尾辞を設計することを検討してください。

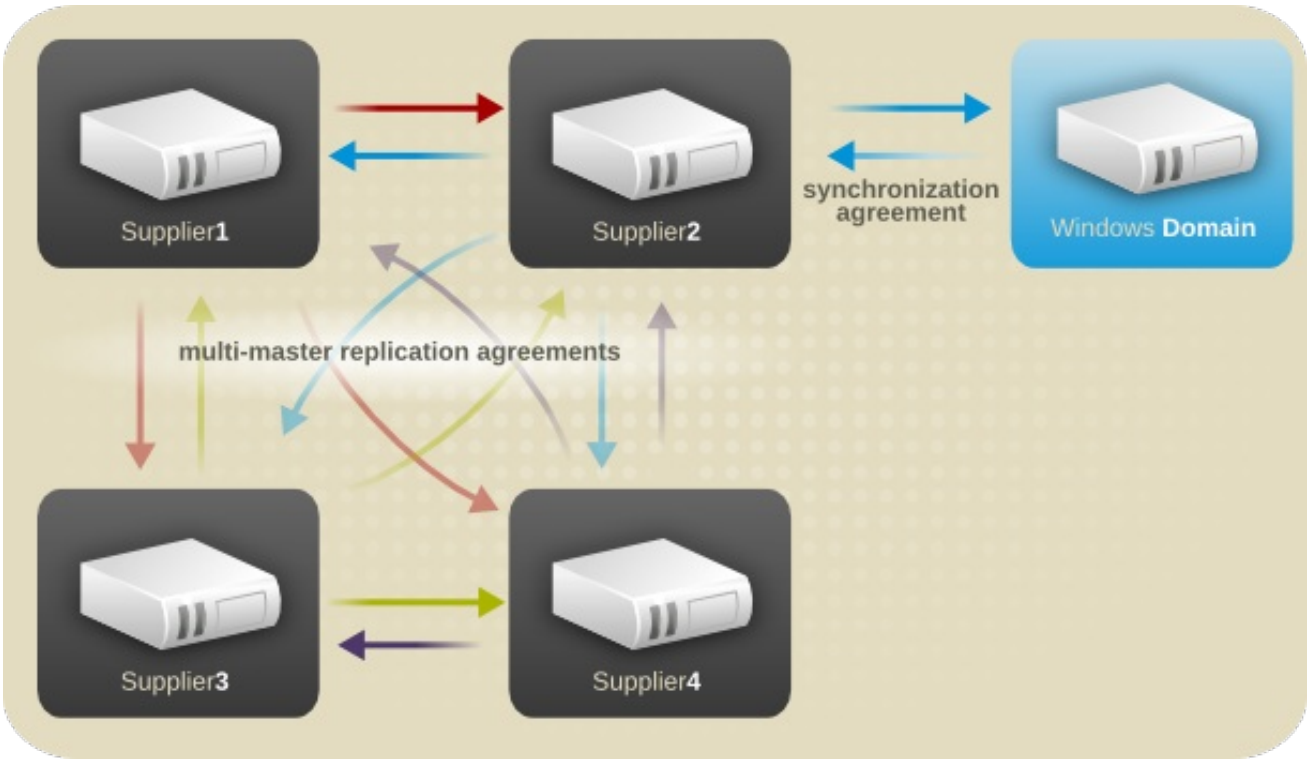
サブツリー計画では、Active Directory ディレクトリーと Directory Server ディレクトリーの間で対応する可能性があり、同期されたサブツリーの範囲外のエントリーも考慮する必要があります。同期プロセスは実際にルート DN で開始し、同期のエントリーの評価を開始します。エントリーは、Active Directory の **samAccount** と Directory Server の **uid** 属性に基づいて相関します。Synchronization プラグインは、(**samAccount**/**uid** 関係に基づいて)エントリーが削除または移動されたために同期されたサブツリーから削除された場合、その旨を通知します。これは、そのエントリーがもう同期されないことを同期プラグインに示すためのものです。この問題は、同期プロセスで、移動したエントリーの処理方法を決定するための設定が必要となることです。同期アグリーメントでは、3 つのオプションを設定できます (適切な Directory Server エントリーの削除、変更の無視 (デフォルト)、またはエントリーを同期せずに現状を維持)。

8.3.6. 複製された環境との連携

同期は、Directory Server の接尾辞とサブツリー（例： **ou=People,dc=example,dc=com**）を対応する Windows ドメインとサブツリー(**cn=Users,dc=test,dc=com**)にリンクします。各サブツリーは、命名の競合や変更の競合を回避するために、他の1 つのサブツリーにのみ同期させることができます。

Windows Sync を利用するには、Windows ドメインのメンバーに同期したマルチサプライヤーレプリケーションで Directory Server サプライヤーと共に使用します。これにより、情報が一元化され、維持が簡単になり、両方のディレクトリーシステムより変更が伝播されます。また、データの管理も容易になります。

図8.2 マルチサプライヤーディレクトリーサーバー - Windows ドメインの同期



指定の Windows ドメインに対する同期アグリーメントを 1 つだけ作成します。Windows サーバーから同期された変更と情報を Directory Server 全体に伝搬するには、マルチサプライヤーサプライヤー (できればレプリケーションデプロイメントのデータサプライヤー) との同期アグリーメントを作成します。

8.3.7. 同期方向の制御

図8.1「同期プロセス」に示すように、同期はデフォルトで **双方向** *となります。つまり、Active Directory の変更が Directory Server に送信され、Directory Server の変更が Active Directory に送信されることを意味します。*

oneWaySync** パラメーターを同期合意に追加することで、一方向*** 同期 を作成できます。この属性は、変更を送信する方向を定義します。*

Active Directory サーバー **から** *Directory Server に変更を送信する場合、値は* ***fromWindows*** *になります。この場合、定期的な同期の更新間隔で、Directory Server が Active Directory Server に接続し、DirSync コントロールを送信して更新を要求します。ただし、Directory Server は、その側から変更やエントリーは送信されません。つまり、同期更新は、Active Directory の変更内容が Directory Server のエントリーに送信され、更新されることで設定されています。*

Directory Server **から** *Active Directory サーバーに変更を同期するには、値は* ***toWindows*** *になります。Directory Server は通常の更新で Active Directory サーバーにエントリー変更を送信しますが、Active Directory 側から更新を要求しないように DirSync 制御は含まれません。*

一方向同期を有効にすると、同期されていないサーバーで自動的に変更ができなくなる **わけではないため**、*同期更新間の同期ピア間で不整合が生じる可能性があります。たとえば、一方向性同期は、Active Directory から Directory Server に行くように設定されているため、Active Directory が (実質的に) データサプライヤーとなります。Directory Server でエントリーを変更または削除すると、Directory Server 情報はその情報とは異なるため、これらの変更は Active Directory に引き継がれません。次の同期更新時に、編集内容は Directory Server で上書きされ、削除済みのエントリーが再追加されます。*

データの不整合が発生するのを防ぐには、アクセス制御ルールを使用して、同期 **されていない** *サーバーの同期サブツリー内のエントリーを編集または削除しないようにします。Directory Server のアク*

セス制御については、「アクセス制御の設計」で説明しています。Active Directory の場合は、適切な Windows ドキュメントを参照してください。

8.3.8. 同期されるエントリーの制御

Windows Sync は、さまざまなデプロイメントシナリオをサポートするのに十分な柔軟性を提供するために、同期されるエントリーをある程度制御します。このコントロールは、Directory Server に設定された異なる設定属性を使用して設定されます。

- Windows サブツリー内では、ユーザーおよびグループのエントリーのみを Directory Server に同期できます。同期アグリーメントを作成するとき、新しい Windows ユーザーおよびグループエントリーを作成時に同期するオプションがあります。これらの属性が **on** に設定されている場合は、既存の Windows エントリーが Directory Server に同期され、Windows サーバーで作成されるエントリーが Directory Server と同期されます。

- Active Directory エントリーと同様に、Directory Server のユーザーおよびグループエントリーのみを同期できます。同期されるエントリーには、**ntUser** または **ntGroup** オブジェクトクラスと必須の属性が必要で、その他のエントリーはすべて無視されます。

Directory Server の changelog で平文のパスワードが保持されるため、Directory Server のパスワードは他のエントリー属性と同期されます。Windows サーバー上で加えられた変更を取得するには、Password Sync Service が必要です。Password Sync Service がないと、パスワードは Windows サーバーでハッシュ化され、Windows ハッシュ関数は Directory Server で使用されているものと互換性がないため、Windows パスワードを同期できません。

8.3.9. 同期するディレクトリーデータの特定

Windows Sync は、ディレクトリーサービス間でユーザーおよびグループエントリーを同期します。同期するサブツリーを決定したら、以下のようにそれらのサブツリーに保存する情報を計画します。

- 電話番号、自宅住所、会社の住所、電子メールアドレスなどのディレクトリーユーザーおよび従業員の連絡先情報。

- 取引先、取引先、および顧客の連絡先情報。

- ユーザーのソフトウェア設定またはソフトウェア設定情報。

- グループ情報とグループメンバーシップ。

  グループメンバーは、同期済みの接尾辞内である場合のみ同期されます。アグリーメントの範囲外のグループメンバーは、両側で変更されません。つまり、それらは適切なディレクトリーサービスのグループのメンバーとして一覧表示されますが、グループエントリーの **member** 属性は同期ピアと同期されません。

同期されるエントリーは、同期アグリーメントで設定されます。ユーザーエントリーは、グループエントリーとは別に同期されます。さらに、エントリーの削除は個別に設定されます。削除は明確に同期する必要があります。

Directory Server では、**ntGroup** または **ntUser** オブジェクトクラスと必須の属性が含まれるエントリーのみが同期されます。Windows サーバーと同期する既存および将来のエントリーを決定します。

ディレクトリーに存在する必要があるエントリーを決定した後、これらのオブジェクトのどの属性をディレクトリーに保持する必要があるかを決定します。Directory Server または Active Directory の可能な属性のサブセットのみが同期されます。さらに、この属性のサブセットは、同期アグリーメント (部分同期) で特定の属性を除外することにより、さらに制限できます。

利用可能な同期属性に応じて、エントリーとこれらのエントリーに含まれるデータの両方を計画します。同期された属性と、Directory Server と Active Directory スキーマの相違点は、「Active Directory と Directory Server 間で同期されるスキーマ要素」 で説明します。

8.3.10. ユーザーとグループの POSIX 属性の同期

すべての可能なユーザーと属性のサブセットが、Active Directory と Red Hat Directory Server の間で同期されます。一部の属性はマッピングされ、Active Directory と Directory Server スキーマには違いがあり、一部の属性が直接照合されます。デフォルトでは、これらの属性のみが同期されます。

その同期リストにない属性のタイプの 1 つは、POSIX 関連の属性です。Linux システムでは、システムユーザーおよびグループは POSIX エントリーとして識別され、LDAP POSIX 属性に必要な情報が含まれています。ただし、Windows ユーザーが同期すると、Windows アカウントであることを示す **ntUser** 属性および **ntGroup** 属性が自動的に追加されますが、POSIX 属性は同期されず(Active Directory エントリーに存在していても)、Directory Server 側では POSIX 属性は追加されません。

POSIX Winsync API プラグインは、Active Directory エントリーと Directory Server エントリーとの間で POSIX 属性を同期します。このプラグインはデフォルトで無効になっていますが、有効にすると、データサプライヤーで POSIX 属性の識別を設定でき、ピアサーバー側で同期できます。POSIX 属性を Active Directory エントリーに直接設定し、Directory Server ディレクトリーで同期および保存できるため、Active Directory がユーザーアカウントストアとして使用される場合に役立ちます。

注記

プラグインが有効な場合は、すべての POSIX 属性(**uidNumber**、**gidNumber**、および **homeDirectory**)は、Active Directory エントリーと Directory Server エントリー間で同期されます。ただし、新しい POSIX エントリーまたは POSIX 属性が Directory Server の既存のエントリーに追加されると、**POSIX 属性のみが Active Directory に対応するエントリーと同期します**。POSIX オブジェクトクラス (ユーザーの場合は **posixAccount**、グループの場合は **posixGroup**) は Active Directory エントリーに追加されません。

8.3.11. パスワードの同期およびパスワードサービスのインストール

DirSync プラグインは Directory Server と共にインストールされ、デフォルトで有効になりますが、パスワードを同期するには、Windows マシンに追加の Windows サービスである Password Sync をインストールする必要があります。このサービスは、Windows サーバーで行われたパスワードの変更を Directory Server に転送するために必要です。

Password Sync サービスがインストールされていないと、パスワードの同期(**userPassword** 属性の同期)は有効になりません。つまり、Directory Server ユーザーエントリーが Windows サーバーに同期されていても、ユーザーエントリーは Windows ドメインでアクティブではありません (つまり、同期されたユーザーはパスワードがないため、ドメインにログインできません)。

注記

ユーザーパスワードの最終更新に別のタイムスタンプを記録する **passwordTrackUpdateTime** というパスワードポリシー属性があります。これにより、Active Directory と Directory Server 間または他のクライアントとのパスワード変更を同期することが簡単になります。

8.3.12. 更新ストラテジーの定義

必要な同期属性が含まれるように変更された既存の Directory Server エントリーは、次の更新まで同期されません。すでに同期されている Windows エントリーおよび Directory Server エントリーへの変更は、次の増分更新で実行されます。このストラテジーの一部として、データを一元管理し、データの変

更が可能なアプリケーションを制限し、必要な更新をスケジュールするようにしてください。更新は既存情報を上書きまたは削除せず、新しいエントリーを追加して変更を送信します。

デフォルトでは、Windows および Directory Server インスタンスは継続的に同期され、変更が 5 分ごとに公開されています。このスケジュールを変更するには、同期アグリーメント属性を手動で設定して更新間隔(**winSyncInterval**)を変更するか、別の更新スケジュール(**nsDS5ReplicaUpdateSchedule**)を設定して変更できます。

8.3.13. 同期アグリーメントの編集

Web コンソールで設定された基本的な同期アグリーメントは、ホストおよびポート情報、同期済みのサブツリー、接続タイプなどの同期に関する非常に簡単な情報を設定します。

ただし、一部レプリケーションや同期スケジュールなどのマルチサプライヤーレプリケーションで利用できる多くの設定は、Windows-Directory Server の同期で利用できます。これらの設定は、同期アグリーメントに手動で追加する必要があります。

デフォルトの同期アグリーメントの変更は、『Administration Guide』で説明しています。利用可能な同期アグリーメント属性は、『Configuration, Command, and File Reference』に記載されています。

8.4. ACTIVE DIRECTORY と DIRECTORY SERVER 間で同期されるスキーマ要素

Directory Server または Windows サーバーで開始されるかどうかに関わらず、Directory Server のすべての同期済みエントリーには、以下のような特別な同期属性があります。

- **ntUniqueId** 対応する Windows エントリーの **objectGUID** 属性の値が含まれます。この属性は同期プロセスで設定され、手動で設定または変更しないでください。
- **ntUserDeleteAccount** Windows エントリーが同期されても、Directory Server エントリーに対して手動で設定する必要がある場合に自動的に設定されます。**ntUserDeleteAccount** の値が **true** であれば、Directory Server エントリーが削除されると、対応する Windows エントリーが削除されます。
- **ntDomainUser** Active Directory エントリーの **samAccountName** 属性に対応します。**ユーザーエントリーのみ。**
- **ntGroupType** 同期される Windows グループには自動的に設定されますが、同期する前に Directory Server エントリーに手動で設定する必要があります。**グループエントリーのみ。**

事前定義済みの属性のリストは、Directory Server と Active Directory エントリーの間で同期されます。Directory Server の **givenName** 属性と Active Directory の **givenName** 属性が一致するように、これらの属性の一部は同じです。Active Directory と Red Hat Directory Server で定義されたスキーマは若干異なるため、Active Directory と Red Hat Directory Server との間で他の属性はマッピングされます。これらのほとんどは、Directory Server の Windows 固有の属性です。

8.4.1. Directory Server と Active Directory 間で同期されるユーザー属性

Directory Server 属性および Active Directory 属性のサブセットのみが同期されます。これらの属性はハードコーディングされ、エントリーが同期される方法に関係なく定義されます。Directory Server または Active Directory のいずれかにあるエントリーにあるその他の属性は、同期の影響を受けないままになります。

Directory Server および Active Directory で使用される属性の一部は同一です。これは通常、すべての LDAP サービスに共通する LDAP 標準で定義された属性です。これらの属性は、相互に正確に同期され

ます。表8.2「Directory Server および Windows サーバーで同一のユーザースキーマ」は、Directory Server と Windows サーバーとの間で同じ属性を示しています。

同じ情報を定義する属性もありますが、属性やスキーマ定義の名前が異なります。これらの属性は Active Directory と Directory Server の間でマッピングされるため、1 つのサーバーの属性 A がもう 1 つのサーバーの属性 B として扱われます。同期の場合、これらの属性の多くは Windows 固有の情報に関連します。表8.1「Directory Server と Active Directory との間でマッピングされるユーザースキーマ」は、Directory Server と Windows サーバーとの間で同じ属性を示しています。

Directory Server および Active Directory が一部のスキーマ要素を処理する方法の違いについての詳細は、「Red Hat Directory Server と Active Directory との間のユーザースキーマの相違点」を参照してください。

表8.1 Directory Server と Active Directory との間でマッピングされるユーザースキーマ

Directory Server	Active Directory
cn	name
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

表8.2 Directory Server および Windows サーバーで同一のユーザースキーマ

cn	physicalDeliveryOfficeName
description	postOfficeBox

destinationIndicator	postalAddress
facsimileTelephoneNumber	postalCode
givenName	registeredAddress
homePhone	sn
homePostalAddress	st
initials	street
l	telephoneNumber
mail	teletexTerminalIdentifier
manager	telexNumber
mobile	title
o	userCertificate
ou	x121Address
pager	

8.4.2. Red Hat Directory Server と Active Directory との間のユーザースキーマの相違点

Active Directory は Directory Server と同じ基本的な X.500 オブジェクトクラスをサポートしますが、管理者が認識すべき非互換性がいくつかあります。

8.4.2.1. cn 属性の値

*Directory Server では、**cn** 属性に複数の値を指定できますが、Active Directory ではこの属性に単一の値しか持たせません。Directory Server の **cn** 属性が同期されると、単一の値のみが Active Directory ピアに送信されます。*

*これは、同期の意味としては、**cn** の値が Active Directory エントリーに追加され、その値が Directory Server の **cn** の値のいずれでもない場合、Directory Server の **cn** 値がすべて単一の Active Directory 値で上書きされます。*

*もう 1 つの重要な相違点として、Active Directory は **cn** 属性をその命名属性として使用するのに対し、Directory Server では **uid** を使用する点があります。つまり、Directory Server で **cn** 属性が編集されると、エントリーの名前が完全に変更になる可能性があります。この **cn** の変更が Active Directory エントリーに書き込まれると、エントリーの名前が変更になり、新しい名前付きエントリーが Directory Server に書き込まれます。ただし、**cn** 属性が同期されている場合のみ発生します。変更が同期されていない場合、エントリーの名前は変更されません。*

8.4.2.2. パスワードポリシー

Active Directory と Directory Server の両方は、パスワードの最小長や最大期間などのパスワードポリシーを強制できます。Windows Sync では、ポリシーの一貫性、強制、または同期が保証されません。パスワードポリシーが DirectoryServer と Active Directory の両方で一貫していない場合、一方のシステムで行われたパスワードの変更は、もう一方のシステムに同期されたときに失敗する可能性があります。Directory Server におけるデフォルトのパスワード構文設定は、Active Directory が実施するデフォルトのパスワードの複雑さルールに準拠します。

8.4.2.3. street および streetAddress の値

Active Directory は、ユーザーまたはグループの住所に **streetAddress** 属性を使用します。これは、Directory Server が **street** 属性を使用する方法です。Active Directory および Directory Server が **streetAddress** 属性および **street** 属性を使用する方法には 2 つの重要な相違点があります。

- Directory Server では、**streetAddress** は **street** のエイリアスです。Active Directory にも **street** 属性がありますが、**streetAddress** のエイリアスではなく、別の属性で個別の値を保持することができます。

- Active Directory は **streetAddress** と **street** を単一値の属性として定義しますが、Directory Server は RFC 4519 で指定されるように **street** を多値属性として定義します。

Directory Server および Active Directory が **streetAddress** および **street** 属性を処理する方法が異なるため、Active Directory および Directory Server で address 属性を設定する際に従う 2 つのルールがあります。

- Windows Sync は、Windows エントリーの **streetAddress** を Directory Server の **street** にマッピングします。競合を避けるために、**street** 属性は Active Directory では使用しないようにしてください。

- Directory Server の **street** 属性値は 1 つだけ Active Directory に同期されます。**streetAddress** 属性が Active Directory で変更され、新しい値が Directory Server に存在しない場合は、Directory Server のすべての **street** 属性値が新しい Active Directory 値に置き換えられます。

8.4.2.4. initials 属性の制約

initials 属性では、Active Directory は最大長 6 文字の制限を課しますが、Directory Server には長さ制限がありません。6 文字を超える **initials** 属性が Directory Server に追加されると、その値は Active Directory エントリーと同期したときにトリミングされます。

8.4.3. Directory Server と Active Directory 間で同期されたグループ属性

Directory Server 属性および Active Directory 属性のサブセットのみが同期されます。これらの属性はハードコーディングされ、エントリーが同期される方法に関係なく定義されます。Directory Server または Active Directory のいずれかにあるエントリーにあるその他の属性は、同期の影響を受けないままになります。

Directory Server エントリーおよび Active Directory グループエントリーで使用される属性の一部は同一です。これは通常、すべての LDAP サービスに共通する LDAP 標準で定義された属性です。これらの属性は、相互に正確に同期されます。表8.4「Directory Server と Active Directory との間のグループエントリー属性」は、Directory Server と Windows サーバーとの間で同じ属性を示しています。

同じ情報を定義する属性もありますが、属性やスキーマ定義の名前が異なります。これらの属性は Active Directory と Directory Server の間でマッピングされるため、1 つのサーバーの属性 A がもう 1 つのサーバーの属性 B として扱われます。同期の場合、これらの属性の多くは Windows 固有の情報に関

連します。表8.3「Directory Server と Active Directory との間のグループエントリー属性のマッピング」は、Directory Server と Windows サーバーとの間で同じ属性を示しています。

Directory Server および Active Directory が一部のスキーマ要素を処理する方法の違いについての詳細は、「Red Hat Directory Server と Active Directory のグループスキーマの相違点」を参照してください。

表8.3 Directory Server と Active Directory との間のグループエントリー属性のマッピング

Directory Server	Active Directory
cn	name
ntGroupAttributes	groupAttributes
ntGroupId	cn name sAMAccountName
ntGroupType	groupType

表8.4 Directory Server と Active Directory との間のグループエントリー属性

cn	member
description	ou
l	seeAlso

8.4.4. Red Hat Directory Server と Active Directory のグループスキーマの相違点

Active Directory は Directory Server と同じ基本的な X.500 オブジェクトクラスをサポートしますが、管理者が認識すべき非互換性がいくつかあります。

ネストされたグループ (グループにメンバーとして別のグループが含まれる) がサポートされ、WinSync であれば同期されます。ただし、Active Directory では、ネストされたグループの設定として特定の制約が適用されます。たとえば、グローバルグループには、ドメインローカルグループがメンバーとして含まれています。Directory Server にはローカルグループとグローバルグループの概念がないため、同期時に Active Directory の制約に違反する Directory Server 側でエントリーを作成できます。

第9章 セキュアなディレクトリーの設計

Red Hat Directory Server のデータがどのように保護されるかは、これまでのすべての設計領域に影響します。セキュリティー設計では、ディレクトリーに含まれるデータを保護し、ユーザーとアプリケーションのセキュリティーとプライバシーのニーズを満たす必要があります。

この章では、セキュリティーのニーズを分析する方法と、これらのニーズを満たすようにディレクトリーを設計する方法について説明します。

9.1. セキュリティーの脅威

ディレクトリーのセキュリティーには多くの潜在的な脅威が存在します。最も一般的な脅威を理解すると、セキュリティー設計全体の概要を把握するのに役立ちます。ディレクトリーのセキュリティーの脅威は、以下の 3 つの主要カテゴリーに分類されます。

- 不正アクセス
- 不正な改ざん
- サービス拒否

9.1.1. 不正アクセス

不正アクセスからディレクトリーを保護することは簡単に見えるかもしれませんが、安全なソリューションの実装は、最初の印象よりも複雑になる可能性があります。未承認のクライアントがデータにアクセスできる可能性のあるディレクトリー情報配信パスには、潜在的なアクセスポイントが複数存在します。

たとえば、未承認のクライアントは別のクライアントのクレデンシャルを使用してデータにアクセスする可能性があります。これは、ディレクトリーが保護されていないパスワードを使用している場合に特に発生する可能性があります。未承認のクライアントは、承認されたクライアントと Directory Server の間で交換される情報を盗聴する可能性もあります。

不正アクセスは会社内部から行われる可能性があります。また、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会社外部から行われる可能性があります。

以下のシナリオは、承認されていないクライアントがディレクトリーデータにアクセスする方法の例をいくつか説明します。

Directory Server が提供する認証方法、パスワードポリシー、およびアクセス制御メカニズムは、不正アクセスを阻止する効率的な方法を提供します。詳細は、以下のセクションを参照してください。

- 「適切な認証方法の選択」
- 「パスワードポリシーの設計」
- 「アクセス制御の設計」

9.1.2. 不正な改ざん

侵入者がディレクトリーにアクセスしたり、Directory Server とクライアントアプリケーション間の通信を傍受したりすると、ディレクトリーデータを変更 (または改ざん) する可能性があります。クライアントがデータを信頼できなくなった場合、またはディレクトリー自体がクライアントから受け取った変更やクエリーを信頼できない場合、ディレクトリーサービスを使用する意味がありません。

たとえば、ディレクトリーで改ざんを検出できない場合、攻撃者はクライアントのサーバーへの要求を変更し (または転送しない)、サーバーのクライアントへの応答を変更できる可能性があります。TLS および同様のテクノロジーは、接続の両側で情報に署名することで、この問題を解決できます。Directory Server で TLS を使用する方法の詳細は、「サーバー接続の保護」 を参照してください。

9.1.3. サービス拒否

サービス拒否攻撃の目的は、ディレクトリーがクライアントにサービスを提供できなくすることです。たとえば、攻撃者はシステムのすべてのリソースを使用する可能性があり、これらのリソースが他のユーザーに使用されないようにします。

Directory Server では、特定のバインド DN に割り当てられるリソースの制限を設定すると、サービス拒否攻撃を防ぐことができます。ユーザーのバインド DN に基づいてリソース制限を設定する方法の詳細は、『Red Hat Directory Server Administration Guide 』の User Account Management の章を参照してください。

9.2. セキュリティーニーズの分析

環境およびユーザーを分析し、セキュリティーニーズを特定します。3章ディレクトリースキーマの設計 のサイト調査は、ディレクトリーの個々のデータの読み書き権限を持つユーザーに関する基本的な決定を明確化しています。この情報は、セキュリティー設計の基盤となります。

セキュリティーの実装方法は、ディレクトリーサービスを使用してビジネスをサポートする方法にも左右されます。イントラネットを提供するディレクトリーは、インターネットに公開されているエクストラネットまたは e コマースアプリケーションをサポートするディレクトリーと同じセキュリティー対策を必要としません。

ディレクトリーがイントラネットのみを提供する場合は、情報に必要なアクセスレベルを検討してください。

- ユーザーおよびアプリケーションに、ジョブの実行に必要な情報へのアクセスを提供する方法。
- 社員またはビジネスに関する機密データを一般アクセスから保護する方法。

ディレクトリーがエクストラネットを提供するか、インターネットを介した e コマースアプリケーションをサポートする場合は、追加の考慮事項があります。

- 顧客にプライバシーの保証を提供する方法。
- 情報の整合性を保証する方法。

以下のセクションでは、セキュリティーニーズの分析について説明します。

9.2.1. アクセス権限の決定

データ分析により、ユーザー、グループ、パートナー、顧客、およびアプリケーションがディレクトリーサービスにアクセスするのに必要な情報が特定されます。

アクセス権は、以下の 2 つの方法のいずれかで付与できます。

- 機密データを保護しながら、すべてのカテゴリーのユーザーにできるだけ多くの権限を付与します。

  オープンな方法では、どのデータがビジネスにとって機密または重要であるかを正確に判断する必要があります。

- 各カテゴリーのユーザーに、業務の遂行に必要な最小限のアクセス権を付与します。

  制限的な方法では、組織の内部、場合によっては外部のユーザーの各カテゴリーの情報ニーズを詳細に理解する必要があります。

アクセス権を決定するために使用される方法に関係なく、組織内のユーザーのカテゴリーとそれぞれに付与されたアクセス権をリスト表示する簡単な表を作成します。ディレクトリーに保持される機密データと、データごとにそれを保護するために実行する手順が記載された表の作成を検討してください。

ユーザーの ID を確認する方法は、「適切な認証方法の選択」 を参照してください。ディレクトリー情報へのアクセスを制限する方法は、「アクセス制御の設計」 を参照してください。

9.2.2. データのプライバシーおよび整合性の確保

ディレクトリーを使用して、エクストラネットを介したビジネスパートナーとの交流をサポートしたり、インターネット上の顧客との e コマースアプリケーションをサポートしたりする場合は、交換されるデータのプライバシーと整合性を確保してください。

これにはいくつかの方法があります。

- データ転送を暗号化する。
- 証明書を使用してデータ転送に署名する。

Directory Server で提供される暗号化の方法の詳細は、「パスワードストレージスキーム」 を参照してください。

データの署名に関する詳細は、「サーバー接続の保護」 を参照してください。

Directory Server データベースに格納される機密情報の暗号化については、「データベースの暗号化」を参照してください。

9.2.3. 定期的な監査の実施

追加のセキュリティー対策として、定期的な監査を実施して、SNMP エージェントによって記録されたログファイルと情報を調べ、セキュリティーポリシー全体の効率性を検証します。

SNMP の詳細は、『Red Hat Directory Server Administration Guide 』を参照してください。ログファイルおよびSNMP の詳細は、『Red Hat Directory Server Administration Guide 』を参照してください。

9.2.4. セキュリティーニーズ分析の例

このセクションで使用する例は、架空の ISP 企業 example.com がセキュリティーニーズを分析する方法を示しています。

example.com のビジネスは、Web ホスティングとインターネットアクセスを提供することです。example.com は活動の一部として取引先企業のディレクトリーをホストしています。また、多くの個人加入者へのインターネットアクセスも提供します。

そのため、example.com ディレクトリーには、3 つの主要な情報カテゴリーがあります。

- example.com の内部情報
- 法人のお客様の情報
- 個人加入者に関する情報

example.com には、以下のアクセス制御が必要です。

- ホストしている会社 (example_a および example_b) のディレクトリー管理者に独自のディレクトリー情報へのアクセスを提供します。
- ホストしている会社のディレクトリー情報にアクセス制御ポリシーを実装します。
- 自宅からのインターネットアクセスに example.com を使用するすべての個人顧客に標準のアクセス制御ポリシーを実装します。
- example.com の企業ディレクトリーへのアクセスをすべての部外者に拒否します。
- example.com の加入者名簿への読み取りアクセスを全員に付与します。

9.3. セキュリティーメソッドの概要

Directory Server には、特定のニーズに適合する全体的なセキュリティーポリシーを設計する方法が複数あります。セキュリティーポリシーは、機密情報が権限のないユーザーによって変更および取得されないように厳密でなければなりませんが、同時に簡単に管理できる必要があります。複雑なセキュリティーポリシーは、アクセスが必要な情報へのアクセスを妨げるミスや、アクセスを許可してはならないディレクトリー情報の変更や取得を許可するミスにつながる可能性があります。

表9.1 Directory Server で利用可能なセキュリティーメソッド

セキュリティーメソッド	説明
認証	ある当事者が別の当事者のアイデンティティーを検証する手段です。たとえば、クライアントは LDAP バインド操作時に Directory Server にパスワードを提供します。
パスワードポリシー	パスワードが有効であると見なされるために満たさなければならない基準を定義します (期限、長さ、構文など)。
暗号化	情報のプライバシーを保護します。データを暗号化する際には、受信者だけが理解できるようにスクランブルをかけます。
アクセス制御	異なるディレクトリーユーザーに付与されたアクセス権を調整し、必要な認証情報またはバインド属性を指定する方法を提供します。
アカウントの非アクティブ化	すべての認証の試行が自動的に拒否されるよう、ユーザーアカウント、アカウントグループ、またはドメイン全体を無効にします。
セキュアな接続	TLS、Start TLS、または SASL での接続を暗号化することで、情報の整合性を維持します。送信中に情報が暗号化されていれば、受信者は送信中に情報が変更されていないと判断できます。最小限のセキュリティー強度係数を設定することにより、セキュアな接続が必要になる場合があります。
監査	ディレクトリーのセキュリティーが侵害されたかどうかを判断します。簡単な監査方法の 1 つは、ディレクトリーが維持するログファイルを確認することです。
SELinux	Red Hat Enterprise Linux マシンでセキュリティーポリシーを使用して、Directory Server ファイルとプロセスへのアクセスを制限して制御します。

セキュリティー設計においてセキュリティーを維持するこれらのツールをいくつか組み合わせ、レプリケーションやデータ配布などのディレクトリーサービスのその他の機能を組み込んで、セキュリティー設計をサポートします。

9.4. 適切な認証方法の選択

セキュリティーポリシーに関する基本的な決定は、ユーザーがディレクトリーにアクセスする方法です。匿名ユーザーはディレクトリーにアクセスできるのですか？それとも、すべてのユーザーがユーザー名とパスワードを使用してディレクトリーにログイン (認証) する必要があるのですか？

Directory Server は、認証に以下の方法を提供します。

- 「匿名および認証されていないアクセス」
- 「シンプルバインドとセキュアなバインド」
- 「証明書ベースの認証」
- 「プロキシー認証」
- 「パスワードなしの認証」

このディレクトリーは、ユーザーまたは LDAP 対応のアプリケーションに関係なく、すべてのユーザーに同じ認証メカニズムを使用します。

クライアントまたはクライアントのグループによる認証を防止する方法の詳細は、「アカウントロックアウトポリシーの設計」 を参照してください。

9.4.1. 匿名および認証されていないアクセス

匿名でのアクセスは、ディレクトリーへの最も簡単なアクセス方法です。これは、認証の有無に関係なく、ディレクトリーのすべてのユーザーがデータを利用できるようになります。

しかし、匿名アクセスでは、管理者は誰がどのような検索を行っているかを追跡することはできず、誰かが検索を行っていることだけがわかります。匿名アクセスでは、ディレクトリーに接続するユーザーは誰でもデータにアクセスできます。

したがって、管理者は特定のユーザーまたはユーザーのグループが特定の種類のディレクトリーデータにアクセスするのをブロックしようとする場合がありますが、そのデータへの匿名アクセスが許可されている場合は、それらのユーザーはディレクトリーに匿名でバインドするだけでデータにアクセスできます。

匿名アクセスは制限することができます。通常、ディレクトリー管理者は、読み取り、検索、および権限の比較に対してのみ匿名アクセスを許可します (書き込み、追加、削除、または自己書き込みは許可しません)。多くの場合、管理者は名前、電話番号、電子メールアドレスなどの一般的な情報を含む属性のサブセットへのアクセスを制限します。政府が管理する ID 番号 (アメリカの Social Security Number など)、自宅の電話番号や住所、給料明細などのより機密性の高いデータへの匿名アクセスはは絶対に許可しないでください。

ディレクトリーデータにアクセスするユーザーにより厳密なルールが必要な場合は、匿名アクセスを完全に無効にすることもできます。

認証されていないバインド は、ユーザーがユーザー名でバインドしようとするが、ユーザーのパスワード属性を持たない場合です。以下に例を示します。

```
ldapsearch -x -D "cn=jsmith,ou=people,dc=example,dc=com" -b "dc=example,dc=com" "(cn=joe)"
```

Directory Server は、ユーザーがパスワードを提供しようとしない場合に、匿名アクセスを付与します。認証されていないバインドでは、バインド DN が既存のエントリーである必要はありません。

匿名バインドと同様に、データベースへのアクセスを制限し、未認証のバインドを無効にして、セキュリティーを強化できます。非認証バインドの無効化には別の利点があります。クライアントのサイレントバインドの失敗を防ぐために使用できます。不適切に作成されたアプリケーションの場合、実際にはパスワードの受け渡しに失敗し、未認証のバインドに接続しただけでも、バインド成功メッセージを受信すると、ディレクトリーの認証に成功したと判断する可能性があります。

9.4.2. シンプルバインドとセキュアなバインド

匿名アクセスが許可されない場合、ユーザーはディレクトリーの内容にアクセスする前にディレクトリーに対して認証する必要があります。シンプルパスワード認証では、クライアントは再利用可能なパスワードを送信してサーバーに対して認証します。

たとえば、クライアントは、識別名と認証情報のセットを提供するバインド操作を使用して、ディレクトリーに対して認証を行います。サーバーは、クライアント DN に対応するディレクトリーのエントリーを特定し、クライアントによって指定されるパスワードがエントリーに保存されている値と一致するかどうかを確認します。一致する場合、サーバーはクライアントを認証します。そうでない場合は、認証操作は失敗し、クライアントはエラーメッセージを受信します。

*バインド DN は多くの場合、ユーザーのエントリーに対応しています。ただし、一部のディレクトリー管理者は、個人ではなく組織エントリーとしてバインドすると便利であると感じています。このディレクトリーでは、バインドに使用されるエントリーが、**userPassword** 属性を許可するオブジェクトクラスである必要があります。これにより、ディレクトリーはバインド DN およびパスワードを確実に認識します。*

ほとんどの LDAP クライアントはバインド DN をユーザーに対して非表示にします。これは、長い文字列の DN を覚えるのは難しい可能性があるからです。クライアントがユーザーに対してバインド DN を非表示にする場合、以下のようなバインドアルゴリズムが使用されます。

1. *ユーザーは、ユーザー ID (**fchen**など) などの一意の識別子を入力します。*
2. *LDAP クライアントアプリケーションは、ディレクトリーでその識別子を検索し、関連付けられた識別名（例： **uid=fchen,ou=people,dc=example,dc=com**）を返します。*
3. *LDAP クライアントアプリケーションは、取得した識別名と、ユーザーが指定したパスワードを使用してディレクトリーにバインドします。*

シンプルなパスワード認証では、ユーザーの認証が簡単になりますが、安全に使用するためには、追加のセキュリティーが必要です。組織のイントラネットでその使用を制限することを検討してください。エクストラネットを介したビジネスパートナー間の接続、またはインターネット上の顧客との送信に使用するには、安全な (暗号化された) 接続を要求するのが最良である場合があります。

注記

シンプルパスワード認証の欠点は、パスワードがプレーンテキストで送信されることです。不正ユーザーがリッスンしている場合、そのユーザーが許可されたユーザーになりすます可能性があるため、ディレクトリーのセキュリティーが悪用される可能性があります。

***nsslapd-require-secure-binds** 設定属性では、TLS または Start TLS を使用してセキュアな接続で簡単なパスワード認証を行う必要があります。これにより、プレーンテキストのパスワードが実質的に暗号化されるため、ハッカーが盗み見ることができなくなります。*

TLS または Start TLS 操作を使用して Directory Server とクライアントアプリケーションの間でセキュアな接続が確立されると、クライアントはパスワードをプレーンテキストで送信しないという保護レベルを追加したシンプルバインドを実行します。**nsslapd-require-secure-binds** 設定属性には、TLS または Start TLS などのセキュアな接続でシンプルなパスワード認証が必要です。この設定により、SASL 認証や証明書ベースの認証などの他のセキュアな接続も使用できます。

セキュアな接続の詳細は、「サーバー接続の保護」 を参照してください。

9.4.3. 証明書ベースの認証

ディレクトリー認証の代替形式は、デジタル証明書を使用してディレクトリーにバインドします。ディレクトリーは、ユーザーの初回アクセス時にパスワードを要求します。ただし、ディレクトリーに保存されているパスワードと照合するのではなく、パスワードによってユーザーの証明書データベースが解放されます。

ユーザーが正しいパスワードを入力すると、ディレクトリークライアントアプリケーションは証明書データベースから認証情報を取得します。次に、クライアントアプリケーションとディレクトリーは、この情報を使用して、ユーザーの証明書をディレクトリー DN にマッピングし、ユーザーを識別します。ディレクトリーはこの認証プロセス中に特定されたディレクトリー DN に基づいて、アクセスを許可または拒否します。

証明書および TLS の詳細は、『Administration Guide』を参照してください。

9.4.4. プロキシー認証

ディレクトリーへのアクセスを要求するユーザーは、自身の DN ではなく**プロキシー DN** にバインドするため、プロキシー認証は特殊な形式の認証です。

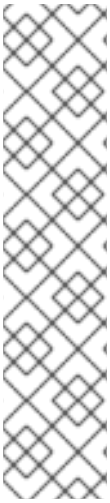
プロキシー DN は、ユーザーが要求する操作を実行するための適切な権限を持つエンティティーです。プロキシー権限が個人またはアプリケーションに付与されると、Directory Manager DN を除き、プロキシー DN として DN を指定する権限が付与されます。

プロキシー権限の主な利点の 1 つは、LDAP アプリケーションを有効にして、単一のバインドで単一のスレッドを使用し、Directory Server に対して要求を行う複数のユーザーに対応できることです。クライアントアプリケーションは、ユーザーごとにバインドおよび認証する代わりに、プロキシー DN を使用して Directory Server にバインドします。

プロキシー DN は、クライアントアプリケーションによって送信される LDAP 操作で指定されます。以下に例を示します。

```
ldapmodify -D "cn=Directory Manager" -W -x -D "cn=directory manager" -W -p 389 -h
server.example.com -x -Y "cn=joe,dc=example,dc=com" -f mods.ldif
```

この **ldapmodify** コマンドは、manager エントリー(**cn=Directory Manager**)に Joe という名前のユーザー(**cn=joe**)のパーミッションを付与し、**mods.ldif** ファイルに変更を適用します。マネージャーは、この変更を加えるためにジョーのパスワードを提供する必要はありません。



注記

プロキシーメカニズムは非常に強力であり、慎重に使用する必要があります。プロキシー権限は ACL の範囲内で付与され、プロキシー権限を持つエントリーによって偽装できるユーザーを制限する方法はありません。つまり、ユーザーにプロキシー権限が付与されると、そのユーザーは対象の任意ユーザーをプロキシーすることができます。特定のユーザーのみにプロキシー権限を制限する方法はありません。

*たとえば、エンティティーに **dc=example,dc=com** ツリーへのプロキシー権限がある場合、そのエンティティーは何でも実行できます。したがって、プロキシー ACI が DIT の可能な限り低いレベルに設定されていることを確認してください。*

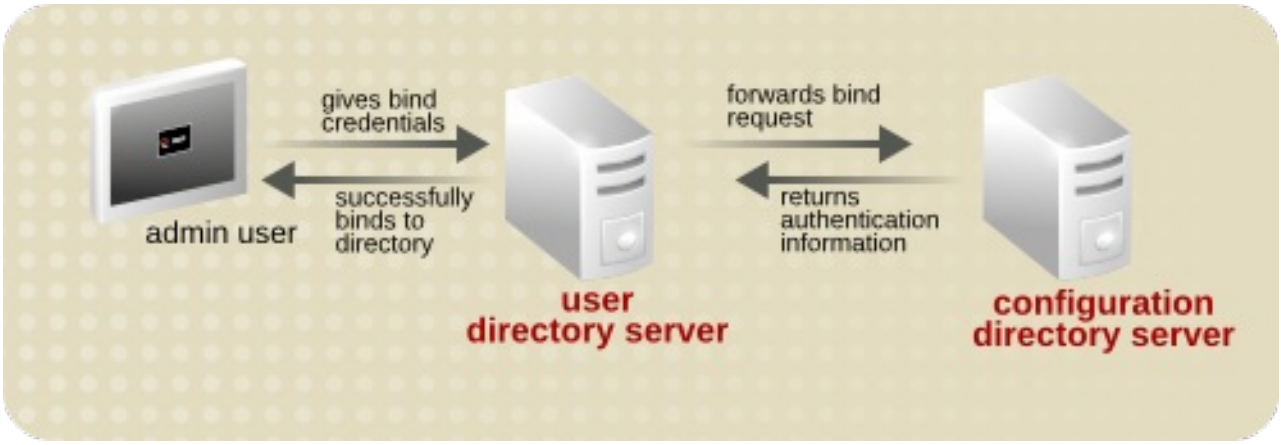
このトピックに関する詳細情報は、『Administration Guide』の Managing Access Control の章の Proxied Authorization ACI Example のセクションを参照してください。

9.4.5. パススルー認証

パススルー認証とは、*認証要求が1 つのサーバーから別のサービスに転送されることです。*

たとえば、インスタンスのすべての設定情報が別のディレクトリーインスタンスに保存されている場合、Directory Server は User Directory Server にパススルー認証を使用して Configuration Directory Server に接続します。Directory Server から Directory Server へのパススルー認証は、PTA プラグインで処理されます。

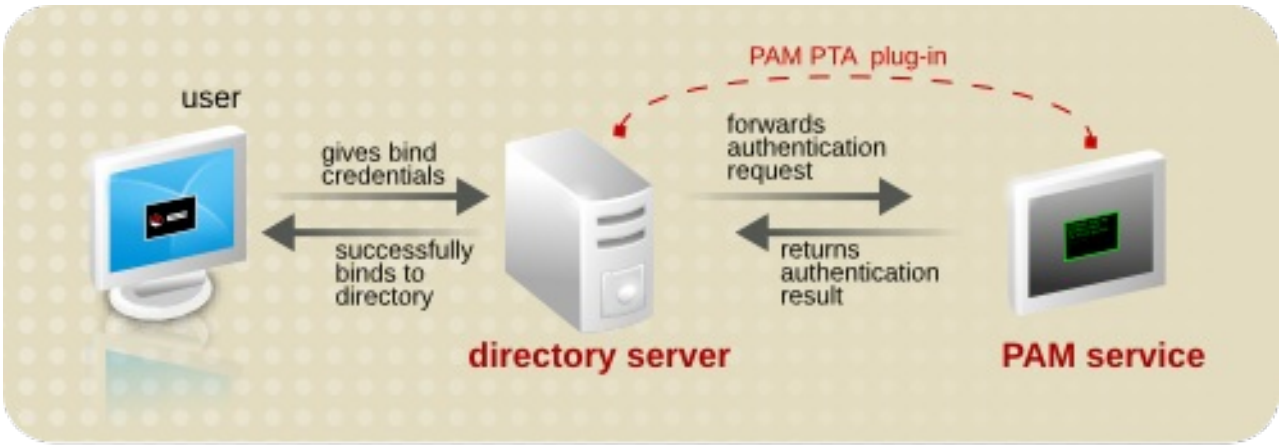
図9.1 簡単なパススルー認証プロセス



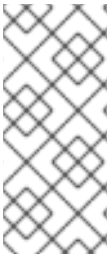
*多くのシステムでは、Unix ユーザーおよび Linux ユーザー用の認証メカニズムがすでに含まれています。最も一般的な認証フレームワークの1 つは、**プラグ可能な認証モジュール** (PAM) です。多くのネットワークではすでに既存の認証サービスを利用できるため、管理者はこれらのサービスを引き続き使用したいことがあります。PAM モジュールは、Directory Server に対して LDAP クライアントに既存の認証ストアを使用するように指示するように設定できます。*

Red Hat Directory Server における PAM パススルー認証は、PAM パススルー認証プラグインを使用します。これにより、Directory Server が PAM サービスと通信して LDAP クライアントを認証できます。

図9.2 PAM パススルー認証プロセス



PAM パススルー認証では、ユーザーがDirectory Server にバインドしようとすると、クレデンシャルが PAM サービスに転送されます。クレデンシャルがPAM サービスの情報と一致すると、ユーザーは Directory Server アクセス制御の制限とアカウント設定がある状態で、Directory Server に正常にバインドできます。



注記

Directory Server は、PAM を使用するように設定できますが、認証に Directory Server を使用するように PAM を **設定** *することはできません。PAM が 認証に Directory Server インスタンスを使用するには、* ***pam_ldap*** *モジュールを適切に設定する必要があります。* ***pam_ldap*** *に関する一般的な情報は、man ページ(http://linux.die.net/man/5/pam_ldap など) を参照してください。*

*PAM サービスは、***System Security Services Daemon (SSSD)** *などのシステムツールを使用して設定できます。SSSD は、Active Directory、Red Hat Directory Server、OpenLDAP などのその他のディレクトリー、またはローカルシステム設定などのさまざまなアイデンティティープロバイダーを使用できます。SSSD を使用するには、PAM パススルー認証プラグインが、SSSD が使用する PAM ファイル（デフォルトでは* ***/etc/pam.d/system-auth*** *）を指すようにします。*

9.4.6. パスワードなしの認証

認証の試行では、最初に、ユーザーアカウントに認証機能があるかどうかが評価されます。アカウントはアクティブである必要があり、ロックされてはならず、該当するパスワードポリシーに従って有効なパスワードを持っている必要があります(つまり、期限切れにならず、リセットする必要があります)。

ユーザーに認証を許可する必要があるかどうかの **評価** *を実行する必要がある場合がありますが、ユーザーは実際には Directory Server にバインドするべきではありません (またはバインドできません)。たとえば、システムは PAM を使用してシステムアカウントを管理することができ、PAM は LDAP ディレクトリーをアイデンティティーストアとして使用するように設定されます。ただし、システムは SSH キーや RSA トークンなどのパスワードなしのクレデンシャルを使用しており、これらのクレデンシャルを渡して Directory Serve r に認証することはできません。*

Red Hat Directory Server は、 ***ldapsearch*** *の* **Account Usability Extension Control** *をサポートしています。このコントロールは、アカウントのステータスと有効なパスワードポリシー(リセットの要求、パスワード期限切れの警告、パスワード期限切れ後の猶予ログインの回数など) に関する情報を返します。これは、バインドの試行で返されるすべての情報ですが、ユーザーとして Directory Server には認証およびバインドされません。これにより、クライアントは Directory Server の設定と情報に基づいてユーザーに認証を許可するかどうかを決定できますが、実際の認証プロセスは Directory Server の外部で実行されます。*

この制御を PAM などのシステムレベルのサービスで使用すると、パスワードなしのログインが可能になります。このログインでは、引き続き Directory Server を使用して ID を保存し、アカウントのステータスを制御します。

注記

Account Usability Extension Control は、デフォルトでは Directory Manager のみが使用できます。他のユーザーが制御を使用できるようにするには、サポートされるコントロールエントリーに適切な ACI を設定します(**oid=1.3.6.1.4.1.42.2.27.9.5.8,cn=features,cn=config**)。

9.5. アカウントロックアウトポリシーの設計

アカウントロックアウトポリシーは、ディレクトリーへの不正なアクセスや危険なアクセスを防ぐことで、ディレクトリーデータとユーザーパスワードの両方を保護することができます。アカウントがロックされたり、**非アクティブ化** されたりすると、そのユーザーはディレクトリーにバインドできなくなり、認証操作は失敗します。

アカウントの非アクティブ化は、操作属性 **nsAccountLock** で実装されます。値が **true** の **nsAccountLock** 属性がエントリーに含まれる場合、サーバーはそのアカウントによるバインドの試行を拒否します。

アカウントロックアウトポリシーは、特定の自動基準に基づいて定義できます。

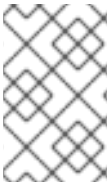
- アカウントロックアウトポリシーをパスワードポリシー(「パスワードポリシーの設計」) に関連付けることができます。ユーザーが一定回数以上、適切な認証情報でログインできない場合、管理者が手動でロックを解除するまでアカウントがロックされます。

  これにより、ユーザーのパスワードを繰り返し推測してディレクトリーに侵入しようとするクラッカーから守ることができます。

- 一定の時間が経過すると、アカウントをロックすることができます。これは、アカウントが作成された時間に基づいて、時間制限のあるアクセス権を持つ、インターン、学生、季節労働者などの一時的なユーザーのアクセスを制御するために使用できます。または、アカウントが最後のログイン時刻から一定期間非アクティブの場合に、ユーザーアカウントを非アクティブにするアカウントポリシーを作成することもできます。

  タイムベースのアカウントロックアウトポリシーは、ディレクトリーのグローバル設定を行う Account Policy プラグインによって定義されます。複数のアカウントポリシーサブエントリーを作成して、異なる有効期限やタイプを設定し、サービスクラスを通じてエントリーに適用することができます。

さらに、1 つのユーザーアカウントまたは (ロールを介した) 一連のアカウントを手動で非アクティブ化することができます。



注記

ロールを非アクティブ化すると、そのロールのすべてのメンバーが非アクティブ化され、ロールエントリー自体は非アクティブ化されません。ロールの詳細は、 「ロールの概要」 を参照してください。

9.6. パスワードポリシーの設計

パスワードポリシーとは、特定のシステムでパスワードがどのように使用されるかを管理する一連のルールのことです。Directory Server のパスワードポリシーは、期間、長さ、ユーザーがパスワードを

再利用できるかどうかなど、パスワードが有効であると見なされるために満たす必要のある基準を指定します。

以下のセクションでは、健全なパスワードポリシーを設計するための詳細情報を提供します。

- 「パスワードポリシーの仕組み」
- 「パスワードポリシーの属性」
- 「レプリケートされた環境でのパスワードポリシーの設計」

9.6.1. パスワードポリシーの仕組み

Directory Server は、きめ細かいパスワードポリシーをサポートします。つまり、パスワードポリシーをサブツリーおよびユーザーレベルで定義することができます。これにより、ディレクトリーツリー内の任意の時点でパスワードポリシーを定義する柔軟性が可能になります。

- ディレクトリー全体。

  このようなポリシーは **グローバル** パスワードポリシーと呼ばれます。ポリシーは設定および有効化されると、ディレクトリー内のすべてのユーザーに適用されます。ただし、Directory Manager エントリーと、ローカルのパスワードポリシーが有効になっているユーザーエントリーは適用から除外されます。

  これにより、すべてのディレクトリーユーザーに共通の単一のパスワードポリシーを定義することができます。

- ディレクトリーの特定のサブツリー。

  このようなポリシーは **サブツリーレベル** または **ローカル** パスワードポリシーと呼ばれます。設定および有効化されると、ポリシーは指定されたサブツリー配下のすべてのユーザーに適用されます。

  これは、すべてのホストされた会社に単一のポリシーを適用するのではなく、ホストされた会社ごとに異なるパスワードポリシーをサポートするホスティング環境に適しています。

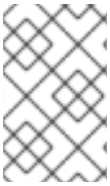
- ディレクトリーの特定のユーザー。

  このようなポリシーは **ユーザーレベル** または **ローカル** パスワードポリシーと呼ばれます。設定および有効化されると、ポリシーは指定されたユーザーのみに適用されます。

  これにより、ディレクトリーユーザーごとに異なるパスワードポリシーを定義できます。たとえば、一部のユーザーがパスワードを毎日変更し、一部のユーザーが毎月パスワードを変更し、他のすべてのユーザーが 6 か月ごとにパスワードを変更するように指定します。

デフォルトでは、Directory Server にはグローバルパスワードポリシーに関連するエントリーおよび属性が含まれます。つまり、同じポリシーがすべてのユーザーに適用されます。サブツリーまたはユーザーにパスワードポリシーを設定するには、サブツリーまたはユーザーレベルでエントリーを追加し、**cn=config** エントリーの **nsslapd-pwpolicy-local** 属性を有効にします。この属性はスイッチとして機能し、粒度の細かいパスワードポリシーをオンおよびオフに切り替えます。

コマンドラインまたは Web コンソールを使用してパスワードポリシーを変更できます。**dsconf pwpolicy** コマンドを使用してグローバルポリシーを変更し、**dsconf localpwp** コマンドを使用してローカルポリシーを変更します。パスワードポリシーの設定に関する詳細は、『管理ガイド を参照してください』。



注記

*以前にローカルパスワードポリシーを管理していた **ns-newpwpolicy.pl** スクリプトが非推奨になりました。ただし、このスクリプトは **389-ds-base-legacy-tools** パッケージで使用できます。*

パスワードポリシーエントリーがディレクトリーに追加されると、Directory Server が強制するパスワードポリシーのタイプ (グローバルまたはローカル) が決定されます。

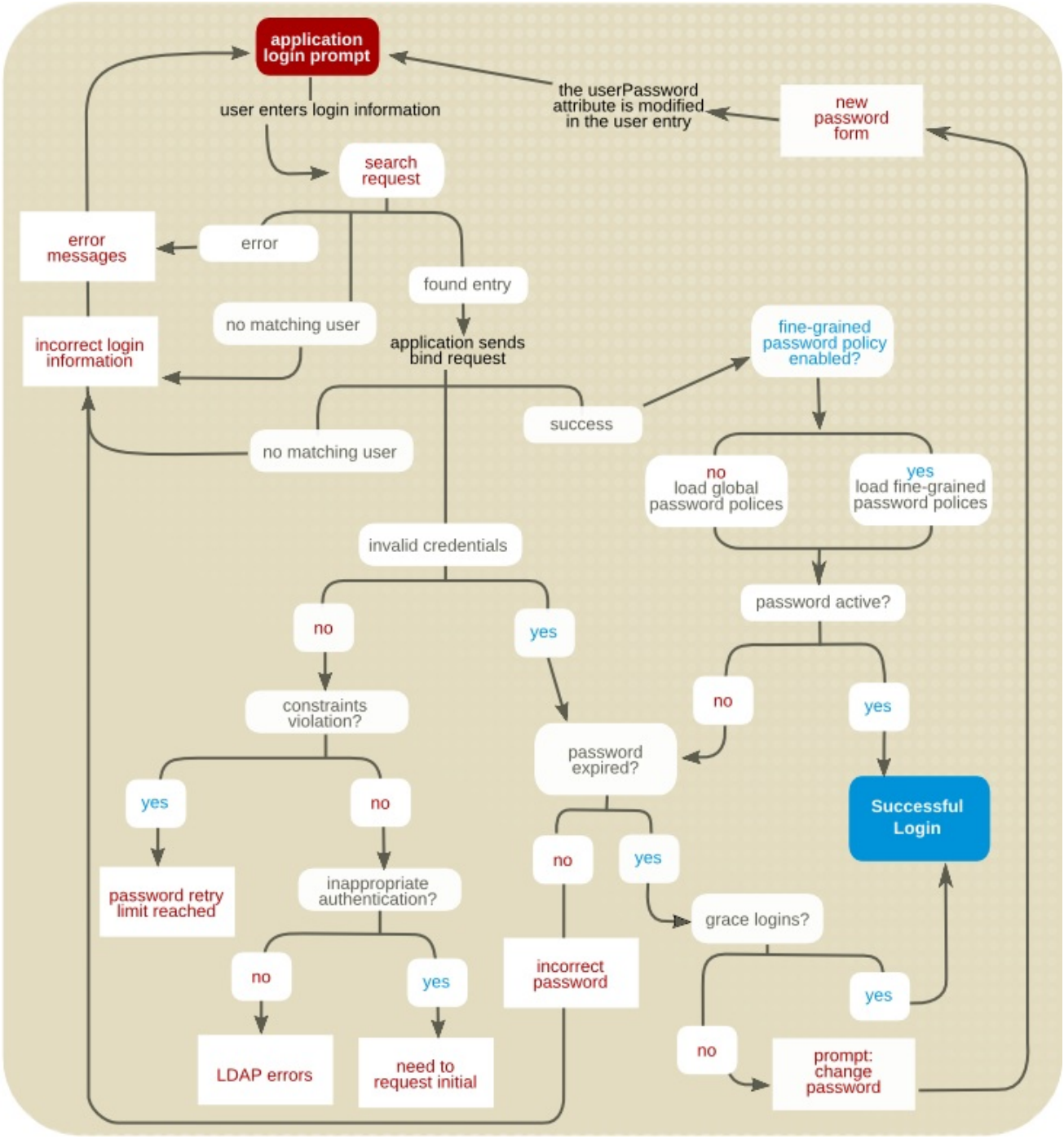
ユーザーがディレクトリーにバインドしようとすると、Directory Server は、ローカルポリシーが定義され、ユーザーのエントリーに対して有効になっているかどうかを判断します。

- *詳細なパスワードポリシーが有効になっているかどうかを判断するために、サーバーは **cn=config** エントリーの **nsslapd-pwpolicy-local** 属性に割り当てられた値(**on** または **off**) をチェックします。値が **off** の場合、サーバーはサブツリーおよびユーザーレベルで定義されたポリシーを無視し、グローバルパスワードポリシーを適用します。*

- *ローカルポリシーがサブツリーまたはユーザーに定義されているかどうかを判断するために、サーバーは対応するユーザーエントリーの **pwdPolicysubentry** 属性をチェックします。属性が存在する場合は、サーバーは、ユーザーに設定されたローカルパスワードポリシーを適用します。属性が存在しない場合、サーバーはエラーメッセージをログに記録し、グローバルパスワードポリシーを強制します。*

その後、サーバーはユーザーが指定したパスワードを、ユーザーのディレクトリーエントリーで指定された値と比較して、それらが一致することを確認します。サーバーは、パスワードポリシーで定義されたルールも使用して、ユーザーがディレクトリーにバインドできるようになる前にパスワードが有効であることを確認します。

図9.3 パスワードポリシーの確認プロセス



*バインド要求の他に、**userPassword** 属性（以下のセクションを参照）が要求に存在する場合、パスワードポリシーチェックは追加および変更操作中にも行われます。*

***userPassword** の値を変更すると、以下の 2 つのパスワードポリシー設定が確認されます。*

- *パスワードの最低期間ポリシーがアクティブになります。最低期間要件が満たされないと、サーバーは constraintViolation エラーを返します。パスワード更新操作は失敗します。*
- *パスワード履歴ポリシーがアクティブになります。**userPassword** の新しい値がパスワード履歴にある場合、または現在のパスワードと同じ場合は、サーバーは constraintViolation エラーを返します。パスワード更新操作は失敗します。*

***userPassword** の値を追加および変更すると、パスワードポリシーでパスワード構文がチェックされます。*

- パスワードの最小長ポリシーがアクティブになります。**userPassword** の新しい値が必要な最小長未満の場合、サーバーは constraintViolation エラーを返します。パスワード更新操作は失敗します。
- パスワード構文の確認ポリシーがアクティブになります。**userPassword** の新しい値がエントリーの別の属性と同じ場合、サーバーは constraintViolation エラーを返します。パスワード更新操作は失敗します。

9.6.2. パスワードポリシーの属性

以下のセクションでは、サーバーのパスワードポリシーを作成する属性を説明します。

- 「失敗の最大回数」
- 「リセット後のパスワード変更」
- 「ユーザー定義のパスワード」
- 「パスワードの有効期限」
- 「有効期限の警告」
- 「猶予ログイン制限」
- 「パスワードの構文チェック」
- 「パスワードの長さ」
- 「パスワードの最小期間」
- 「パスワード履歴」
- 「パスワードストレージスキーム」
- 「パスワードの最終変更時刻」

これらの属性の設定方法は『Red Hat Directory Server Administration Guide』を参照してください。

9.6.2.1. 失敗の最大回数

これは、パスワードベースのアカウントのロックアウトを有効にするパスワードポリシーの設定です。ユーザーが一定の回数ログインを試みて失敗すると、そのアカウントは管理者がアンロックするか、オプションで特定の時間が経過するまでロックされます。これは **passwordMaxFailure** パラメーターで設定されます。

試行失敗の最大数に達したときを評価する場合、ログイン試行をカウントする方法は 2 つあります。数に達したときにアカウントをロックする (n) か、カウントを超えたときにのみアカウントをロックする (n + 1) ハード制限を適用できます。たとえば、失敗の制限が試行 3 回である場合、アカウントは 3 回目の失敗 (n) または 4 回目の失敗 (n + 1) でロックできます。n+1 の動作は LDAP サーバーの過去の動作であるため、レガシー動作とみなされます。新しい LDAP クライアントでは、より厳密なハード制限が想定されます。デフォルトでは、Directory Server は厳密な制限(n) を使用しますが、**passwordLegacyPolicy** パラメーターでレガシー動作を有効にできます。

9.6.2.2. リセット後のパスワード変更

Directory Server 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後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指定できます。

管理者が設定するデフォルトのパスワードは、通常、ユーザーのイニシャル、ユーザーID、会社名などの会社の規則に従います。この規則が検出された場合、通常、侵入者がシステムの侵入に使用するのは最初の値です。したがって、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ことが推奨されます。パスワードポリシーにこのオプションが設定されている場合、ユーザー定義のパスワードが無効であっても、パスワードを変更する必要があります。

ユーザーが自分のパスワードを変更する必要がないまたは許可されていない場合、管理者が割り当てたパスワードは、明白な規則に従わないようにし、発見を困難にします。

デフォルト設定では、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。

詳細は、「ユーザー定義のパスワード」 を参照してください。

9.6.2.3. ユーザー定義のパスワード

パスワードポリシーは、ユーザーが自分のパスワードを変更できるようにするかどうかを設定できます。強固なパスワードポリシーには、適切なパスワードが必要です。適切なパスワードには普通の言葉が含まれません。辞書にある単語、ペットや子供の名前、誕生日、ユーザーID、または簡単に見つけられる (またはディレクトリー自体に保存されている) ユーザーに関するその他の情報は、パスワードとしては適切ではありません。

適切なパスワードには、文字、数字、および特殊文字の組み合わせが含まれる必要があります。ただし、便宜上、ユーザーは覚えやすいパスワードを使用することがよくあります。そのため、企業によっては、強固なパスワード基準を満たすユーザーにパスワードを設定し、ユーザーがパスワードを変更できないようにすることがあります。

管理者にユーザーのパスワードを設定させることには、2 つの欠点があります。

- *管理者の時間がかなり必要です。*
- *管理者が指定したパスワードは通常、覚えるのが難しいため、ユーザーはパスワードを書き留める可能性が高く、発見されるリスクが高くなります。*

デフォルトでは、ユーザー定義のパスワードが許可されます。

9.6.2.4. パスワードの有効期限

パスワードポリシーを使用すると、同じパスワードを無期限に使用することを許可したり、一定期間後にパスワードの有効期限が切れるように設定することが可能です。一般的に、パスワードの使用期間が長いほど、パスワードが検出される可能性が高くなります。ただし、パスワードの期限切れが頻繁に発生すると、ユーザーはパスワードを覚えるのに苦労し、パスワードを書き留めてしまう可能性があります。一般的なポリシーでは、パスワードは 30 - 90 日ごとに失効します。

サーバーは、パスワードの有効期限が無効であっても、パスワード失効の仕様を記憶します。パスワードの有効期限が再度有効になると、パスワードは最後に無効になった前に設定された期間のみ有効になります。

たとえば、パスワードポリシーが 90 日ごとに期限切れになるように設定されてから、パスワードの有効期限が無効になり、再度有効にすると、デフォルトのパスワード有効期限は 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。

9.6.2.5. 有効期限の警告

パスワードの有効期限が設定されている場合、パスワードの期限が切れる前に警告をユーザーに送信することが推奨されます。

ユーザーがサーバーにバインドすると、Directory Server が警告を表示します。パスワードの失効が有効になっている場合、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、ユーザーのパスワードの有効期限が切れる 1 日前に警告が (LDAP メッセージを使用して) ユーザーに送信されます。

パスワード失効の警告が送信される有効な範囲は、1 - 24,855 日です。

注記

パスワードは、有効期限の警告が送信されるまで期限切れに**なることはありません。**

9.6.2.6. 猶予ログイン制限

期限切れパスワードの **猶予期間** とは、パスワードが期限切れであっても、ユーザーがシステムにログインできることを意味します。一部のユーザーが期限切れのパスワードを使用してログインできるようにするには、パスワードの有効期限が切れた後にユーザーに許可される猶予ログインの試行回数を指定します。

デフォルトでは、猶予ログインは許可されていません。

9.6.2.7. パスワードの構文チェック

パスワード構文チェック は、パスワード文字列のルールを強制するため、パスワードは特定の基準満たす必要があります。すべてのパスワード構文チェックは、サブツリーごとまたはユーザーごとにグローバルに適用できます。パスワードの構文チェックは **passwordCheckSyntax** 属性で設定されます。

デフォルトのパスワード構文には、最低 8 文字が必要で、パスワードに普通の言葉を使用できません。単純な単語は、ユーザーのエントリーの **uid**、**cn**、**sn**、**givenName**、**ou**、または **mail**属性に保存されている値です。

さらに、他の形式のパスワード構文を強制でき、パスワード構文にさまざまなオプションのカテゴリーを提供します。

- パスワードに必要な最小文字数(**passwordMinLength**)
- 桁の最小数。ゼロから 9 の間の数字を意味します(**passwordMinDigits**)
- ASCII アルファベットの最小数（大文字と小文字の両方）(**passwordMinAlphas**)
- 大文字の ASCII アルファベットの最小数(**passwordMinUppers**)
- 小文字の ASCII アルファベットの最小数(**passwordMinLowers**)
- **!@#$** などの特殊 ASCII 文字の最小数(**passwordMinSpecials**)
- 8 ビット文字の最小数(**passwordMin8bit**)
- **aaabbb** (**passwordMaxRepeats**) など、同じ文字をすぐに繰り返すことができる最大回数()
- パスワードごとに必要な文字カテゴリーの最小数。カテゴリーは大文字、小文字、特殊文字、数字、または 8 ビット文字(**passwordMinCategories**)

- *Directory Server は **CrackLib** ディクショナリーに対してパスワードをチェックします (**passwordDictCheck**)。*
- *Directory Server はパスワードに回文が含まれているかどうかを確認します (**passwordPalindrome**)*
- *Directory Server は、同じカテゴリーの文字を連続して持つパスワードを設定できなくなります (**passwordMaxClassChars**)*
- *Directory Server が特定の文字列を含むパスワードが設定されないようにする (**passwordBadWords**)*
- *Directory Server は管理者定義属性に設定された文字列を含むパスワードが設定されないようにする(**passwordUserAttributes**)*

必要な構文のカテゴリーが多いほど、パスワードは強力になります。

デフォルトでは、パスワード構文のチェックは無効になっています。

9.6.2.8. パスワードの長さ

パスワードポリシーでは、ユーザーパスワードの最小の長さを要求できます。一般的に、パスワードが短いほど解読されやすくなります。パスワードの適切な長さは 8 文字です。これは、解読が難しく、ユーザーがパスワードを書き留めなくても覚えられる長さです。この属性に有効な値の範囲は、2 - 512 文字です。

デフォルトでは、パスワードの最小長は設定されません。

9.6.2.9. パスワードの最小期間

*パスワードポリシーにより、ユーザーが指定された期間はパスワードを変更できないようにすることができます。**passwordHistory** 属性とともに使用すると、古いパスワードの再使用は推奨されません。*

*たとえば、パスワードの最小期間(**passwordMinAge**)属性が 2 日である場合、ユーザーは 1 つのセッション中にパスワードを繰り返し変更できません。これにより、パスワード履歴を循環して、古いパスワードを再利用できるようになります。*

この属性に有効な値の範囲は、0 - 24,855 日です。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ています。

9.6.2.10. パスワード履歴

*Directory Server は、**パスワード履歴** に 2 - 24 個のパスワードを保存できます。パスワードが履歴にある場合、ユーザーは自分のパスワードをその古いパスワードに再設定することはできません。これにより、ユーザーは覚えやすいいくつかのパスワードを再利用できなくなります。または、パスワード履歴を無効にして、ユーザーがパスワードを再利用できるようにすることもできます。*

パスワード履歴がオフの場合でもパスワードは履歴に残るため、パスワード履歴をオンに戻しても、パスワード履歴をオフにする前に履歴にあったパスワードは再利用できません。

サーバーはデフォルトではパスワード履歴を維持しません。

9.6.2.11. パスワードストレージスキーム

パスワードストレージスキームは、ディレクトリー内に Directory Server パスワードを保存するために使用される暗号化の種類を指定します。Directory Server は、さまざまなパスワードストレージスキームをサポートします。

- **Salted Secure Hash Algorithm**(SSHA、SSHA-256、SSHA-384、および SSHA-512)。これは最も安全なパスワードストレージスキームで、これがデフォルトになります。 推奨される SSHA スキームは SSHA-256 以上です。

- **CLEAR**、暗号化がないことを意味します。これは SASL Digest-MD5 と併用できる唯一のオプションであるため、SASL を使用するには CLEAR パスワードストレージスキームが必要です。

  ディレクトリーに保存されているパスワードは、アクセス制御情報 (ACI) 命令を使用して保護できますが、プレーンテキストのパスワードをディレクトリーに保存することは適切とはいえません。

- **Secure Hash Algorithm** (SHA、SHA-256、SHA-384、および SHA-512).これは SSHA よりも安全性が低いです。

- **UNIX CRYPT アルゴリズム**。このアルゴリズムは、UNIX パスワードとの互換性を提供します。

- **MD5**。このストレージスキームは SSHA よりも安全性が低くなりますが、MD5 を必要とするレガシーアプリケーション用に含まれています。

- **Salted MD5**。このストレージスキームは、プレーンな MD5 ハッシュよりも安全ですが、SSHA よりも安全性が低くなります。このストレージスキームは、新しいパスワードと併用するために含まれていませんが、salted MD5 に対応するディレクトリーからのユーザーアカウントを移行するのに役立ちます。

9.6.2.12. パスワードの最終変更時刻

passwordTrackUpdateTime 属性は、エントリーのパスワードが最後に更新されたときのタイムスタンプを記録するようにサーバーに指示します。パスワードの変更時間はユーザーエントリーで ***pwdUpdateTime*** 操作属性 (***modifyTimestamp*** または ***lastModified*** 操作属性とは別) として保存されます。

デフォルトでは、パスワードの変更時刻は記録**されません**。

9.6.3. レプリケートされた環境でのパスワードポリシーの設計

パスワードとアカウントのロックアウトポリシーは、以下のようにレプリケートされた環境で次のように適用されます。

- パスワードポリシーはデータサプライヤーで実施されます。

- アカウントのロックアウトは、レプリケーション設定のすべてのサーバーに適用されます。

ディレクトリーのパスワードポリシー情報。パスワードの期間、アカウントのロックアウトカウンター、および期限切れ警告カウンターなどがすべてレプリケートされます。ただし、設定情報はローカルに保存され、複製されません。この情報には、パスワード構文とパスワード変更の履歴が含まれます。

レプリケートされた環境でパスワードポリシーを設定する場合は、以下の点を考慮してください。

- すべてのレプリカは、パスワードの期限切れが近いことを警告します。この情報は各サーバーでローカルに保存されるため、ユーザーが複数のレプリカに順番にバインドすると、ユーザーは同じ警告を複数回受け取ります。さらに、ユーザーがパスワードを変更すると、この情報が

レプリカにフィルターされるまで時間がかかる場合があります。ユーザーがパスワードを変更してからすぐに再バインドすると、レプリカが変更を登録するまでバインドに失敗する可能性があります。

- サプライヤーやレプリカなど、すべてのサーバーで同じバインド動作が発生する必要があります。各サーバーに常に同じパスワードポリシー設定情報を作成します。
- アカウントロックアウトカウンターは、マルチサプライヤー環境で想定どおりに機能しない場合があります。

9.7. アクセス制御の設計

ディレクトリークライアントのアイデンティティーを確立するために使用する認証スキームを決定した後、そのスキームを使用してディレクトリーに含まれる情報を保護する方法を決定します。アクセス制御では、特定のクライアントが特定の情報にアクセスでき、その他のクライアントはアクセスできないように指定できます。

アクセス制御は、1 つ以上の**アクセス制御リスト** (ACL) を使用して定義されます。ディレクトリーの ACL は、指定されたエントリーとその属性へのパーミッション (読み取り、書き込み、検索、比較など) を許可または拒否する一連の 1 つ以上の**アクセス制御情報** (ACI) ステートメントで設定されます。

ACL を使用すると、ディレクトリーツリーの任意のレベルでパーミッションを設定できます。

- ディレクトリー全体。
- ディレクトリーの特定のサブツリー。
- ディレクトリーの特定のエントリー。
- エントリー属性の特定セット。
- 指定の LDAP 検索フィルターと一致するエントリー。

さらに、特定のユーザー、特定のグループに所属するすべてのユーザー、またはディレクトリーのすべてのユーザーにパーミッションを設定できます。最後に、アクセスは IP アドレス (IPv4、IPv6) や DNS 名などのネットワークの場所に定義できます。

9.7.1. ACI 形式

セキュリティーポリシーを設計する場合は、ACI がディレクトリーでどのように表されているかを理解しておくと便利です。また、ディレクトリーに設定できる権限を理解しておくと便利です。このセクションでは、ACI メカニズムの概要を簡単に説明します。ACI 形式の完全な説明は『Red Hat Directory Server Administration Guide』を参照してください。

ディレクトリー ACI は、一般的な形式 target permission bind_rule を使用します。

ACI 変数を以下に定義します。

- **target**ACI が対象とするエントリー (通常はサブツリー)、対象とする属性、またはその両方を指定します。ターゲットは ACI が適用されるディレクトリー要素を識別します。ACI はエントリーを 1 つだけターゲットにできますが、複数の属性をターゲットにできます。さらに、ターゲットには LDAP 検索フィルターを含めることができます。パーミッションは、共通の属性値を含む、広範囲に散在するエントリーにパーミッションを設定することができます。

- **パーミッション**.このACI によって設定される実際のパーミッションを特定します。permission 変数は、ACI が指定されたターゲットへの特定タイプのディレクトリーアクセス (読み取りや検索など) を許可または拒否していることを示します。

- **bind rule**パーミッションが適用されるバインドDN またはネットワークの場所を特定します。バインドルールはLDAP フィルターを指定する場合もあり、そのフィルターがバインドクライアントアプリケーションに対して true であると評価された場合、ACI はクライアントアプリケーションに適用されます。

そのため、ACI は For the directory object target, allow or deny permission if bind_rule is true と表すことができます。

permission および **bind_rule** はペアとして設定され、ターゲットごとに複数の **permission-bind_rule** ペアを設定できます。複数のアクセス制御を任意のターゲットに効果的に設定できます。以下に例を示します。

```
target (permission bind_rule)(permission bind_rule)...
```

Babs Jensen としてバインドするユーザーが誰でも Babs Jensen の電話番号へ書き込みできるパーミッションを設定できます。このパーミッションのバインドルールは、**if you bind as Babs Jensen**の部分です。ターゲットは Babs Jensen の電話番号で、パーミッションが**書き込み** アクセスです。

9.7.1.1. ターゲット

ディレクトリーに作成されたすべてのACI のターゲットとなるエントリーを決定します。ディレクトリー分岐点エントリーをターゲットにすると、その分岐点とそのすべての子エントリーがパーミッションの範囲に含まれます。ターゲットエントリーがACI に明示的に定義されていない場合、ACI はACI ステートメントが含まれるディレクトリーエントリーに対してターゲットとなります。***targetattr*** パラメーターを設定して、1 つ以上の属性をターゲットにします。***targetattr*** パラメーターが設定されていない場合、ターゲットになる属性はありません。詳細は、『Red Hat Directory Server Administration Guide』の該当セクションを参照してください。

各 ACI に対し、1 つのエントリーのみ、または1 つの LDAP 検索フィルターに一致するエントリーのみをターゲットにすることができます。

エントリーをターゲットにすることに加えて、エントリーの属性をターゲットにすることができます。これは属性値のサブセットにのみパーミッションが適用されます。ターゲットとなる属性に明示的に命名するか、ACI によってターゲットにされていない属性に明示的に命名することによる属性のターゲットセット。ターゲットで属性を除外すると、オブジェクトクラス構造によって許可されるいくつかの属性を除くすべての属性にパーミッションが設定されます。

詳細は、『Red Hat Directory Server Administration Guide』の該当セクションを参照してください。

9.7.1.2. パーミッション

パーミッションは、アクセスを許可または拒否できます。一般的には、パーミッションを拒否しないようにします (「アクセスの許可または不許可」 に記載の理由により)。パーミッションは、ディレクトリーサービスに対して実行される操作のいずれかです。

パーミッション	説明
Read	ディレクトリーデータを読み取ることができるかどうかを示します。

パーミッション	説明
Write	ディレクトリーデータを変更または作成できるかどうかを示します。このパーミッションは、ディレクトリーデータの削除を許可しますが、エントリー自体は削除できません。エントリー全体を削除するには、ユーザーに delete パーミッションが必要です。
Search	ディレクトリーデータを検索できるかどうかを示します。これは、検索操作の一部として返される場合にディレクトリーデータを表示できる点で、read パーミッションとは異なります。 たとえば、共通名の検索と、人の部屋番号の読み取りが許可された場合は、共通名の検索の一部として部屋番号を返すことができますが、部屋番号自体は検索の対象として使用できません。この組み合わせを使用して、ディレクトリーを検索してどの部屋に誰がいるかを検索できないようにします。
Compare	比較操作でデータを使用できるかどうかを示します。compare パーミッションは検索可能であることを意味しますが、検索の結果として実際のディレクトリー情報は返されません。代わりに、比較した値と一致するかどうかを示す簡単なブール値が返されます。これは、ディレクトリー認証中に **userPassword** 属性値を照合するために使用されます。
Self-write	グループ管理にのみ使用されます。このパーミッションにより、ユーザーはグループに自身を追加したり、グループから削除したりできます。
追加	子エントリーを作成できるかどうかを示します。このパーミッションにより、ユーザーはターゲットとなるエントリーの下に子エントリーを作成できます。
削除	エントリーを削除できるかどうかを示します。このパーミッションにより、ユーザーはターゲットとなるエントリーを削除できます。
Proxy	ユーザーがディレクトリーマネージャー以外の他の DN を使用して、この DN の権限でディレクトリーにアクセスできることを示します。

9.7.1.3. バインドルール

バインドルールは通常、パーミッションの対象となるバインド 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。

バインドルールは、ACI がユーザー自身のエントリーにのみ適用されることを簡単に表現します。これにより、ユーザーは別のユーザーのエントリーを更新するリスクを負わずに、独自のエントリーを更新できます。

バインドルールは、ACI が特定の状況に適用可能なことを示します。

- バインド操作が特定の IP アドレス (IPv4 または I Pv6) または DNS ホスト名から送信される場合のみ。これは、特定のマシンまたはネットワークドメインからすべてのディレクトリー更新を強制的に実行するためによく使用されます。
- ユーザーが匿名でバインドする場合。匿名バインドのパーミッションを設定すると、ディレクトリーにバインドするすべてのユーザーにもパーミッションが適用されます。
- ディレクトリーに正常にバインドされるユーザーの場合。これにより、匿名アクセスを阻止する一方で、一般的なアクセスが可能になります。

- クライアントがエントリーの直接の親としてバインドされている場合のみ。
- ユーザーがバインドしたエントリーが特定の LDAP 検索条件を満たしている場合のみ。

Directory Server は、これらの種類のアクセスをより簡単に表現するためにキーワードをいくつか提供します。

- **Parent**バインド DN が直接の親エントリーである場合、バインドルールは true になります。これは、ディレクトリー分岐点がその直接の子エントリーを管理できるようにする特定のパーミッションを付与できることを意味します。
- **Self**バインド DN がアクセスを要求するエントリーと同じである場合、バインドルールは true になります。個人が独自のエントリーを更新できるように特定のパーミッションを付与できます。
- **All**ディレクトリーに正常にバインドされたユーザーは、バインドルールは true になります。
- **Anyone***bind* ルールは、すべてのユーザーで true になります。このキーワードは、匿名アクセスを許可または拒否するために使用されます。

9.7.2. パーミッションの設定

デフォルトでは、Directory Manager を除くすべてのユーザーには、いかなる種類のアクセス権も与えられていません。そのため、ユーザーがディレクトリーにアクセスできるように、一部の ACI をディレクトリーに設定する必要があります。

ディレクトリーに ACI を設定する方法は『Red Hat Directory Server Administration Guide 』を参照してください。

9.7.2.1. 優先度ルール

ユーザーがディレクトリーエントリーへの何らかのアクセスを試みると、Directory Server はディレクトリーに設定されているアクセス制御を調べます。アクセスを決定するため、Directory Server は**優先度ルール** を適用します。このルールは、競合する 2 つのパーミッションが存在する場合に、アクセスを拒否するパーミッションが、アクセスを許可するパーミッションよりも常に優先されることを示しています。

たとえば、書き込みパーミッションがディレクトリーの root レベルで拒否され、そのパーミッションがディレクトリーにアクセスするすべてのユーザーに適用される場合、書き込みアクセスを許可する他のパーミッションに関係なく、ユーザーにはそのディレクトリーへ書き込みできません。特定ユーザーのディレクトリーへの書き込みパーミッションを許可するには、元の書き込み拒否の範囲を設定して、そのユーザーが含まれないようにする必要があります。次に、対象のユーザーに書き込み許可パーミッションが必要です。

9.7.2.2. アクセスの許可または不許可

ディレクトリーツリーへのアクセスは明示的に許可または拒否することができますが、ディレクトリーへのアクセスを明示的に拒否する場合は注意してください。優先度ルールにより、ディレクトリーがアクセスを明示的に禁止するルールを見つけると、アクセスを付与する可能性のある競合するパーミッションに関係なく、ディレクトリーへのアクセスが禁止されます。

ユーザーまたはクライアントアプリケーションの可能な限り小さなサブセットのみが含まれるように、アクセスルールのスコープを制限します。たとえば、パーミッションを設定して、ユーザーがディレクトリーエントリーの任意の属性に書き込むことができますが、Directory Administrators グループのメンバー以外のすべてのユーザーが **uid** 属性に書き込む権限を拒否します。または、以下の方法で書き込みアクセスを許可する 2 つのアクセス制御ルールを作成します。

- ***uid*** *属性以外のすべての属性への書き込み権限を許可するルールを 1 つ作成します。このルールはすべてのユーザーに適用される必要があります。*

- ***uid*** *属性への書き込み権限を許可するルールを 1 つ作成します。このルールは、Directory Administrators グループのメンバーにのみ適用する必要があります。*

許可特権のみを提供すると、明示的な拒否特権を設定する必要がなくなります。

9.7.2.3. アクセスを拒否する場合

明示的な拒否権限を設定する必要はほとんどありませんが、便利な状況がいくつかあります。

- *複雑な ACL が分散している大きなディレクトリーツリーがある。*

  セキュリティー上の理由から、特定のユーザー、グループ、または物理的な場所へのアクセスを突然拒否しなければならない場合があります。時間をかけて既存の ACL を注意深く調べて、許可パーミッションを適切に制限する方法を理解するのではなく、分析を行う時間ができるまで、明示的な拒否特権を一時的に設定します。ACL がこのように複雑になった場合、長期的には、拒否 ACI は管理オーバーヘッドを増やすだけです。できるだけ早く ACL を作り直して、明示的な拒否特権を回避し、アクセス制御スキーム全体を簡素化します。

- *アクセス制御は、曜日または 1 日の時間を基にする必要があります。*

  たとえば、すべての書き込みアクティビティーは日曜日の午後 11 時から拒否できます。(2300) to Monday at 1:00 a.m.(0100).管理の観点からは、すべての allow-for-write ACI のディレクトリーを検索し、この期間でスコープを制限するよりも、このタイプの時間ベースのアクセスを明示的に制限する ACI を管理する方が簡単かもしれません。

- *ディレクトリー管理者権限を複数のユーザーに委譲する場合は、特権を制限する必要があります。*

  個人またはユーザーのグループに、ツリーの一部を変更を許可せずに、ディレクトリーツリーの一部の管理を許可するには、明示的な拒否特権を使用します。

  たとえば、メール管理者が共通名属性への書き込みアクセスを許可しないようにするには、共通名属性への書き込みアクセスを明示的に拒否する ACI を設定します。

9.7.2.4. アクセス制御ルールの配置場所

アクセス制御ルールは、ディレクトリー内の任意のエントリーに配置できます。多くの場合、管理者は、オブジェクトクラス ***domainComponent***、***country***、***organization***、***organizationalUnit***、***inetOrgPerson***、*または* ***group*** *を使用して、エントリーにアクセス制御ルールを配置します。*

ACL 管理を簡素化するために、ルールを可能な限りグループに編成します。ルールは通常、ターゲットエントリーとそのエントリーのすべての子に適用されます。したがって、アクセス制御ルールは、個々のリーフ (個人など) のエントリーに分散させるのではなく、ディレクトリーのルートポイントまたはディレクトリーブランチポイントに配置するのが最適です。

9.7.2.5. フィルターされたアクセス制御ルールの使用

Directory Server ACI モデルのより強力な機能の 1 つは、LDAP 検索フィルターを使用してアクセス制御を設定する機能です。LDAP 検索フィルターを使用して、定義された基準セットに一致するディレクトリーエントリーへのアクセスを設定します。

*たとえば、Marketing に設定された **organizationalUnit** 属性が含まれるエントリーの読み取りアクセスを許可します。*

フィルターされたアクセス制御ルールにより、事前定義のアクセスレベルが許可されます。ディレクトリーに自宅の住所と電話番号情報が含まれるとします。この情報を公開したい人もいれば、非公開にしたい人もいます。これに対処する方法はいくつかあります。

- ***publishHomeContactInfo** という属性をすべてのユーザーのディレクトリーエントリーに作成します。*
- ***homePhone** 属性が **true** （有効）に設定されているエントリーに対してのみ、**homePostalAddress** 属性および **publishHomeContactInfo** 属性への読み取りアクセスを許可するアクセス制御ルールを設定します。LDAP 検索フィルターを使用して、このルールのターゲットを表します。*
- *ディレクトリーユーザーが、独自の **publishHomeContactInfo** 属性の値を **true** または **false** に変更できるようにします。これにより、ディレクトリーユーザーは、この情報が公開されているかどうかを判断できます。*

LDAP 検索フィルターの使用および ACI での LDAP 検索フィルターの使用に関する詳細は、『Red Hat Directory Server Administration Guide』を参照してください。

9.7.3. ACI の表示: Get Effective Rights

*きめ細かいアクセス制御を付与するため、または効率的なエントリー管理のために、エントリーに設定されたアクセス制御を表示する必要がある場合があります。Get effective rights は拡張された **ldapsearch** で、エントリー内の各属性に設定されたアクセス制御パーミッションを返し、LDAP クライアントがサーバーのアクセス制御設定によってユーザーが実行できる操作を決定できるようにします。*

アクセス制御情報は、エントリーに対する権限と属性に対する権限の 2 つのアクセスグループに分けられます。エントリーに対する権限とは、その特定のエントリーに限定された、変更または削除などの権限を意味します。属性に対する権限とは、ディレクトリー全体のその属性のすべてのインスタンスへのアクセス権を意味します。

この種の詳細なアクセス制御は、次のような状況で必要になる場合があります。

- *管理者は、get effective rights コマンドを使用して、特定のグループまたはユーザーのエントリーへのアクセスを許可し、他のユーザーを制限するなど、細かなアクセス制御を行うことができます。たとえば、QA Managers グループのメンバーには、**title** や **salary** などの属性を検索および読み取る権限がありますが、HR Group メンバーのみが変更または削除する権限を持ちます。*
- *ユーザーは、有効な権限を取得するオプションを使用して、個人エントリーで表示または変更できる属性を決定できます。たとえば、ユーザーは **homePostalAddress** および **cn** などの属性にアクセスできますが、**title** および **salary** への読み取りアクセスのみが許可されます。*

***-E** スイッチを使用して実行される **ldapsearch** は、通常の検索結果の一部として、特定のエントリーのアクセス制御を返します。次の検索は、ユーザー Ted Morris が個人エントリーに対して持っている権限を示しています。*

```
ldapsearch -x -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -W -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
!1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=tmorris,ou=people,dc=example,dc=com "(objectClass=*)"

version: 1
```

```
dn: uid=tmorris,ou=People,dc=example,dc=com
givenName: Ted
sn: Morris
ou: Accounting
ou: People
l: Santa Clara
manager: uid=dmiller,ou=People,dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: vadn
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rscow, manager:rsc, roomNumber:rscwo,
mail:rscwo, facsimileTelephoneNumber:rscwo, objectClass:rsc, uid:rsc, cn:rsc, userPassword:wo
```

*この例では、**entryLevelRights** の結果が示すように、Ted Morris には自分のエントリーの DN を追加、表示、削除、または名前変更する権限があります。**l**の結果に示されるように、場所の読み取り、検索、比較、自己変更、または自己削除を行えますが、パスワードに対する自己書き込みおよび自己削除の権限のみになります。**attributeLevelRights***

*デフォルトでは、値を持たない、またはエントリー内に存在しないエントリー内の属性に対して有効な権限情報は返されません。たとえば、**userPassword** の値が削除されると、自己書き込みおよび自己削除の権限が許可されていても、将来的に上記のエントリーで有効な権限を検索しても、**userPassword** の有効な権限は返されません。同様に、**street** 属性に読み取り、比較、および検索の権限が追加されると、**street: rsc** は **attributeLevelRights** の結果に表示されます。*

存在しない属性や操作属性など、通常は検索結果に含まれない属性の権限を返すことができます。アスタリスク(*)を使用すると、存在しない属性を含む、エントリーの可能なすべての属性の権限が返されます。

```
ldapsearch -x -E !1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=scarter,ou=people,dc=example,dc=com "
(objectclass=*)" "*"
```

*プラス記号(**+**)を使用すると、エントリーの操作属性が返されます。これは通常 **ldapsearch** アスタリスク(*)では返されません。以下に例を示します。*

```
ldapsearch -x -E !1.3.6.1.4.1.42.2.27.9.5.2:dn:uid=scarter,ou=people,dc=example,dc=com "
(objectclass=*)" "+"
```

*アスタリスク(*)とプラス記号(**+**)を一緒に使用して、エントリーのすべての属性を返すことができます。*

9.7.4. ACI の使用: いくつかのヒントとコツ

セキュリティーポリシーを実装する際には、このヒントに留意してください。これらは、ディレクトリーセキュリティーモデルを管理する管理上の負担を軽減し、ディレクトリーのパフォーマンス特性を改善するのに役立ちます。

- *ディレクトリー内の ACI の数を最小限に抑えます。*

Directory Server は 50,000 を超える ACI を評価できますが、多数の ACI ステートメントを管理するのは困難です。ACI の数が多いと、人間の管理者が特定のクライアントが使用できるディレクトリーオブジェクトをすぐに判断することが難しくなります。

Directory Server は、マクロを使用してディレクトリー内の ACI の数を最小限に抑えます。マクロは、ACI の DN または DN の一部を表すために使用されるプレースホルダーです。マクロを使用して、ACI のターゲット部分、バインドルール部分、またはその両方で DN を表現することができます。マクロ ACI の詳細は、『Red Hat Directory Server Administration Guide 』の Managing Access Control の章を参照してください。

- *アクセス許可の許可と拒否のバランスを取ります。*

  デフォルトのルールでは、特にアクセスを許可されていないユーザーへのアクセスは拒否されますが、1 つの ACI を使用してツリーのルートに近いアクセスを許可し、少数の拒否を許可することで、ACI の数を減らす方がよい場合があります。リーフエントリーに近い ACI。このシナリオでは、リーフエントリーの近くで複数の許可 ACI を使用することを回避できます。

- *特定の ACI で最小の属性セットを特定します。*

  オブジェクトの属性のサブセットへのアクセスを許可または拒否する場合、最小のリストが、許可される属性のセットであるか、拒否される属性のセットであるかを決定します。次に、最小のリストのみを管理する必要があるように ACI を表現します。

  *たとえば、**person** オブジェクトクラスには多数の属性が含まれます。ユーザーがこれらの属性の 1 つまたは 2 つだけを更新できるようにするには、それらのいくつかの属性のみに書き込みアクセスを許可するように ACI を記述します。ただし、1 つまたは 2 つの属性を除くすべての属性をユーザーが更新できるようにするには、いくつかの名前付き属性を除くすべての属性に対して書き込みアクセスを許可するように ACI を作成します。*

- *LDAP 検索フィルターの使用には注意が必要です。*

  *検索フィルターは、アクセスを管理しているオブジェクトに直接名前を付けません。したがって、それらを使用すると、予期しない結果が生じる可能性があります。これは、ディレクトリーがより複雑になるにつれて特に当てはまります。ACI で検索フィルターを使用する前に、同じフィルターを使用して **ldapsearch** 操作を実行し、変更の結果がディレクトリーにとって何を意味するかを明確にします。*

- *ディレクトリーツリーの異なる部分で ACI を複製しないでください。*

  *ACI の重複を防ぎます。たとえば、**commonName** 属性および **givenName** 属性へのグループの書き込みアクセスを許可する ACI と、同じグループの書き込みアクセスを許可する別の ACI がディレクトリールートポイントにある場合は、1 つの制御のみがグループに書き込みアクセスを付与するように ACI を再起動することを検討してください。**commonName***

  ディレクトリーが複雑になるにつれて、誤って ACI が重複するリスクが急速に高まります。ACI の重複を避けることで、セキュリティー管理が容易になり、ディレクトリーに含まれる ACI の総数が減る可能性があります。

- *ACI に名前を付けます。*

  ACI の命名は任意ですが、各 ACI に短くて意味のある名前を付けることは、セキュリティーモデルの管理に役立ちます。

- *ディレクトリー内で ACI をできるだけ密接にグループ化します。*

ACI の配置をディレクトリールートポイントと主要なディレクトリーブランチポイントに制限するようにしてください。ACI をグループ化すると、ディレクトリー内の ACI の総数を最小限に抑えるだけでなく、ACI の合計リストを管理するのにも役立ちます。

- **バインド DN が cn=Joe と等しくない場合に書き込みを拒否する** *など、二重否定を使用しないでください。*

  この構文はサーバーには完全に受け入れられますが、人間の管理者にとっては混乱を招きます。

9.7.5. ルート DN への ACI の適用 (Directory Manager)

通常のアクセス制御ルールは、Directory Manager ユーザーには適用されません。Directory Manager は、通常のユーザーデータベースではなく ***dse.ldif*** *ファイルで定義されているため、ACI ターゲットにはそのユーザーが含まれません。*

また、メンテナンスの観点からも理にかなっています。Directory Manager では、メンテナンスタスクを実行し、インシデントへの対応に高いレベルのアクセスが必要です。

それでも、Directory Manager ユーザーの権限により、ある程度のアクセス制御は、root ユーザーとして、承認されていないアクセスや攻撃を阻止することを推奨します。

RootDN アクセス制御プラグインは、Directory Manager ユーザーに固有の特定のアクセス制御ルールを設定します。

- *8 a.m. から 5 p.m. までのような時間的な範囲での時間ベースのアクセス制御(0800 - 1700)。*
- *曜日のアクセス制御により、アクセスは明示的に定義された日数でのみ許可されます。*
- *指定した IP アドレス、ドメイン、またはサブネットのみが明示的に許可または拒否される IP アドレスルールです。*
- *ホストアクセスルール。指定されたホスト名、ドメイン名、またはサブドメインのみが明示的に許可または拒否されます。*

他のアクセス制御ルールと同様、deny ルールは allow ルールよりも優先されます。

重要

Directory Manager が常に適切なレベルのアクセスを許可されていることを確認してください。Directory Manager は、オフタイム (ユーザーの負荷が少ない時) や障害対応のためにメンテナンス作業を行う必要があります。その場合、時間や曜日ベースのアクセス制御ルールを厳しく設定すると、Directory Manager がディレクトリーを適切に管理できなくなる可能性があります。

ルート DN アクセス制御ルールはデフォルトで無効になっています。RootDN アクセスコントロールプラグインを有効にする必要があります。その後、適切なアクセスコントロールルールを設定できます。

注記

Directory Manager には 1 つのアクセス制御ルールがあり、プラグインエントリーには、ディレクトリー全体のすべてのアクセスに適用されます。

9.8. データベースの暗号化

情報はデータベースでプレーンテキストで保存されます。その結果、政府発行の ID 番号やパスワードなどの非常に機密性の高い情報は、アクセス制御手段によって十分に保護されない場合があります。ファイルシステムを介して直接、または廃棄されたディスクドライブまたはアーカイブメディアにアクセスすることにより、サーバーの永続ストレージファイルにアクセスできる場合があります。

データベースの暗号化により、個々の属性をデータベースに格納する際に暗号化できます。設定すると、特定の属性のすべてのインスタンス (インデックスデータも含む) が暗号化され、TLS などの安全なチャネルを使用してのみアクセスできます。

データベース暗号化の使用については、『Red Hat Directory Server Administration Guide』の *Configuring Directory Databases* の章を参照してください。

9.9. サーバー接続の保護

識別されたユーザーの認証スキームと、ディレクトリー内の情報を保護するためのアクセス制御スキームを設計したら、次のステップは、サーバーとクライアントアプリケーションの間を通過する情報の整合性を保護する方法を設計することです。

サーバーからクライアントへの接続とサーバーからサーバーへの接続の両方について、Directory Server はさまざまなセキュアな接続タイプをサポートしています。

- **Transport Layer Security (TLS)**

  ネットワーク上で安全な通信を提供するために、Directory Server は Transport Layer Security (TLS) で LDAP を使用できます。

  TLS は、RSA の暗号化アルゴリズムと組み合わせて使用できます。特定の接続に対して選択される暗号化方式は、クライアントアプリケーションと Directory Server 間のネゴシエーションの結果です。

- **Start TLS**

  Directory Server は、通常の暗号化されていない LDAP ポートを介して Transport Layer Security (TLS) 接続を開始する方法である Start TLS もサポートしています。

- **Simple Authentication and Security Layer (SASL)**

  SASL はセキュリティーフレームワークです。つまり、クライアントアプリケーションとサーバーアプリケーションの両方で有効になっているメカニズムに応じて、サーバーに対してユーザーを認証するさまざまなメカニズムを許可するシステムをセットアップします。また、クライアントとサーバー間で暗号化されたセッションを確立することもできます。Directory Server では、SASL を GSS-API とともに使用して Kerberos ログインを有効にし、レプリケーション、連鎖、パススルー認証など、ほぼすべてのサーバー間接続に使用できます。(SASL は Windows Sync では使用できません。)

レプリケーションなどの機密情報を処理する操作には安全な接続が推奨され、Windows パスワード Synchronization などの一部の操作では必要になります。Directory Server は、TLS 接続、SASL、および非セキュア接続を同時にサポートできます。

SASL 認証と TLS 接続の両方を同時に設定できます。たとえば、Directory Server インスタンスは、サーバーへの TLS 接続を要求し、レプリケーション接続の SASL 認証もサポートするように設定できます。これは、ネットワーク環境で TLS **または** SASL のどちらを使用するかを選択する必要がないことを意味します。両方を使用できます。

サーバーへの接続の最小レベルのセキュリティーを設定することもできます。**セキュリティー強度係数**は、キーの強度で、安全な接続の強度を測定します。特定の操作 (パスワードの変更など) を必要とする

ACI は、接続が特定の強度以上の場合にのみ発生するように設定できます。最小限の SSF を設定することもできます。これにより、基本的に標準接続が無効になり、接続ごとに TLS、Start TLS、または SASL が必要になります。Directory Server は TLS と SASL を同時にサポートし、サーバーは利用可能なすべての接続タイプの SSF を計算し、最も強力なものを選択します。

TLS、Start TLS、および SASL の使用に関する詳細は、『Administration Guide』を参照してください。

9.10. SELINUX ポリシーの使用

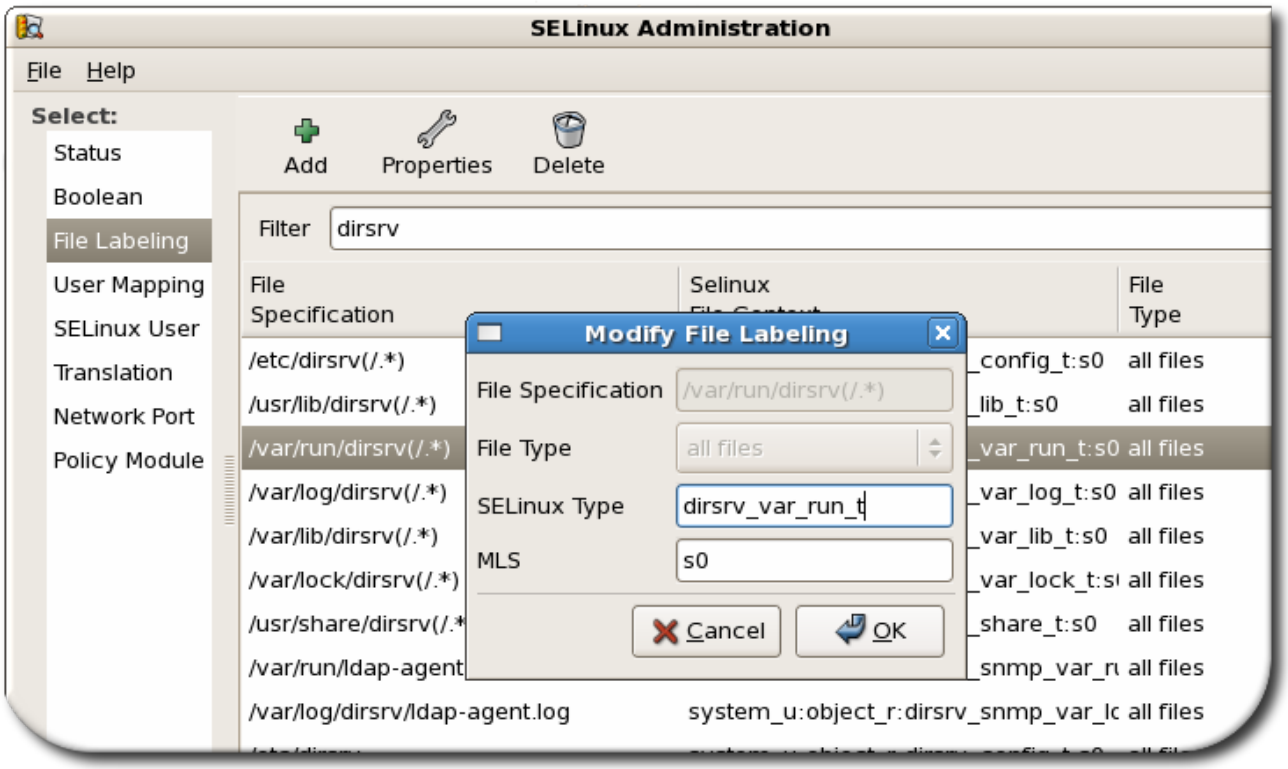
SELinux は、承認されていないアクセスと改ざんを制限するためにシステム全体で実施される、必須のアクセス制御ルールのコレクションです。SELinux は、サーバー上のファイル、ディレクトリー、ポート、プロセス、ユーザー、およびその他のオブジェクトを分類します。各オブジェクトは適切なセキュリティーコンテキストに配置され、オブジェクトがそのロール、ユーザー、およびセキュリティーレベルによってサーバー上でどのように動作できるかを定義します。これらのオブジェクトのロールはドメインにグループ化され、SELinux ルールは、あるドメイン内のオブジェクトが別のドメイン内のオブジェクトと対話できる方法を定義します。

Directory Server には次のドメインがあります。

- *Directory Server の dirsrv_t*
- *SNMP の場合は dirsrv_snmp_t*

Directory Server は、LDAP ポート用に 1 つの追加のデフォルトドメインも使用します: ldap_port_t

図9.4 Directory Server のファイルラベリングの編集



これらのドメインは、Directory Server のすべてのプロセス、ファイル、ディレクトリー、ポート、ソケット、およびユーザーのセキュリティーコンテキストを提供します。

- *各インスタンスのファイルとディレクトリーは、特定の SELinux コンテキストでラベル付けされています。(Directory Server が使用するメインディレクトリーのほとんどには、ローカルイ*

ンスタンスの数に関係なく、すべてのローカルインスタンス用のサブディレクトリーがあるため、1 つのポリシーを新しいインスタンスに簡単に適用できます。)

- 各インスタンスのポートは、特定の SELinux コンテキストでラベル付けされています。
- すべての Directory Server プロセスは、適切なドメイン内に制限されます。
- 各ドメインには、ドメインに承認されたアクションを定義する特定のルールがあります。
- SELinux ポリシーで指定されていないアクセスは、インスタンスに対して拒否されます。

SELinux には、無効 (SELinux なし)、許容 (ルールは処理されるが適用されない)、強制 (すべてのルールが厳密に適用される) の 3 つの異なるレベルの適用があります。Red Hat Directory Server は、厳密な SELinux 強制モードで通常どおり実行できるようにする SELinux ポリシーを定義していますが、注意が必要です。Directory Server はさまざまなモードで実行できます。1 つは通常の操作用で、もう 1 つはインポートなどのデータベース操作用 (ldif2db モード) です。Directory Server の SELinux ポリシーは、通常モードにのみ適用されます。

デフォルトでは、Directory Server は SELinux ポリシーによって制限されて実行されます。

9.11. その他のセキュリティーリソース

安全なディレクトリーの設計に関する詳細は、以下を参照してください。

- 『Understanding and Deploying LDAP Directory Services. 』T. Howes, M. Smith, G. Good, Macmillan Technical Publishing, 1999.
- SecurityFocus.com http://www.securityfocus.com
- Computer Emergency Response Team (CERT) Coordination Center http://www.cert.org

第10章 ディレクトリー設計の例

ディレクトリーサービスの設計は、企業の規模と性質によって異なります。この章では、さまざまな設定でディレクトリーを適用する方法の例をいくつか示します。これらの例は、実際のディレクトリーサービスのデプロイメント計画を作成するための出発点です。

10.1. 設計例: ローカル企業

Example Corp. は自動車部品メーカーで、従業員 500 人の小さな会社です。Example Corp. は、自社で使用するディレクトリー対応アプリケーションをサポートするために、Red Hat Directory Server のデプロイを決定しました。

10.1.1. ローカルエンタープライズデータの設計

Example Corp. はまず、ディレクトリーに格納するデータの種類を決定します。これを行うために、Example Corp. は、サイト調査を実行してディレクトリーの使用方法を決定するデプロイメントチームを作成します。デプロイメントチームは以下を決定します。

- Example Corp. のディレクトリーは、メッセージングサーバー、Web サーバー、カレンダーサーバー、人事アプリケーション、およびホワイトページアプリケーションによって使用されます。
- メッセージングサーバーは、**uid**、**mailServerName**、**mailAddress** などの属性に対して正確な検索を実行します。データベースのパフォーマンスを向上させるために、Example Corp. はこれらの属性のインデックスを維持して、メッセージングサーバーによる検索をサポートします。

  インデックスの使用に関する詳細は、「データベースパフォーマンスを改善するためのインデックスの使用」 を参照してください。
- ホワイトページアプリケーションは、ユーザー名と電話番号を頻繁に検索します。したがって、ディレクトリーは、大量の結果を返す部分文字列、ワイルドカード、およびあいまい検索を頻繁に実行できる必要があります。Example Corp. は、**cn** 属性、**sn** 属性、および **givenName** 属性の存在、等価、近似、および部分文字列インデックスと、**telephoneNumber** 属性の存在、等価、および部分文字列インデックスを維持することを決定します。
- Example Corp. のディレクトリーは、組織全体にデプロイメントされた LDAP サーバーベースのイントラネットをサポートするために、ユーザーとグループの情報を保持しています。Example Corp. のユーザーおよびグループ情報のほとんどは、ディレクトリー管理者のグループによって集中管理されます。ただし、Example Corp. は、メール情報を別のメール管理者グループで管理することも望んでいます。
- Example Corp. は、S/MIME メールなどの公開鍵基盤 (PKI) アプリケーションを将来的にサポートする予定であるため、ユーザーの公開鍵証明書をディレクトリーに格納する準備を整える必要があります。

10.1.2. ローカルエンタープライズスキーマの設計

Example Corp. のデプロイメントチームは、ディレクトリー内のエントリーを表すために **inetOrgPerson** オブジェクトクラスを使用することにしました。このオブジェクトクラスは、**userCertificate** と **uid** (userID) 属性を許可します。これらは、Example Corp. のディレクトリーがサポートするアプリケーションに必要です。

Example Corp. は、デフォルトのディレクトリースキーマをカスタマイズしたいとも考えています。Example Corp. は、Example Corp の従業員を表す **examplePerson** オブジェクトクラスを作成します。このオブジェクトクラスは、**inetOrgPerson** オブジェクトクラスから派生します。

examplePerson オブジェクトクラスは、1 つの属性(**exampleID** 属性) を許可します。この属性には、Example Corp. の各従業員に割り当てられた特別な従業員番号が含まれています。

今後、Example Corp. は必要に応じて **examplePerson** オブジェクトクラスに新しい属性を追加できます。

10.1.3. ローカルエンタープライズのディレクトリーツリー設計

前述のセクションで説明されているデータおよびスキーマ設計に基づいて、Example Corp. は以下のディレクトリーツリーを作成します。

- ディレクトリーツリーのルートは、Example Corp. のインターネットドメイン名である **dc=example,dc=com** です。
- ディレクトリーツリーには、**ou=people**、**ou=groups ou=roles**、および, のブランチポイントが 4 つあります。**ou=resources**
- Example Corp. の人のエントリーはすべて **ou=people** ブランチの下に作成されます。

  人のエントリーはすべて **person**、**organizationalPerson**、、**inetOrgPerson**、および **examplePerson** オブジェクトクラスのメンバーになります。**uid** 属性は、各エントリーの DN を一意に識別します。たとえば、Example Corp. には、Babs Jensen (**uid=bjensen**) および Emily Stanton (**uid=estanton**) のエントリーが含まれます。
- これらは、Example Corp. の営業、マーケティング、会計の各部門に 1 つずつ、3 つのロールを作成します。

  人のエントリーにはそれぞれ、その人が所属する部門を識別するロール属性が含まれます。Example Corp. は、これらのロールに基づいて、ACI を作成できるようになりました。

  ロールの詳細は、「ロールの概要」 を参照してください。
- **ou=groups** ブランチに 2 つのグループブランチを作成します。

  最初のグループ **cn=administrators** には、ディレクトリーの内容を管理するディレクトリー管理者のエントリーが含まれます。

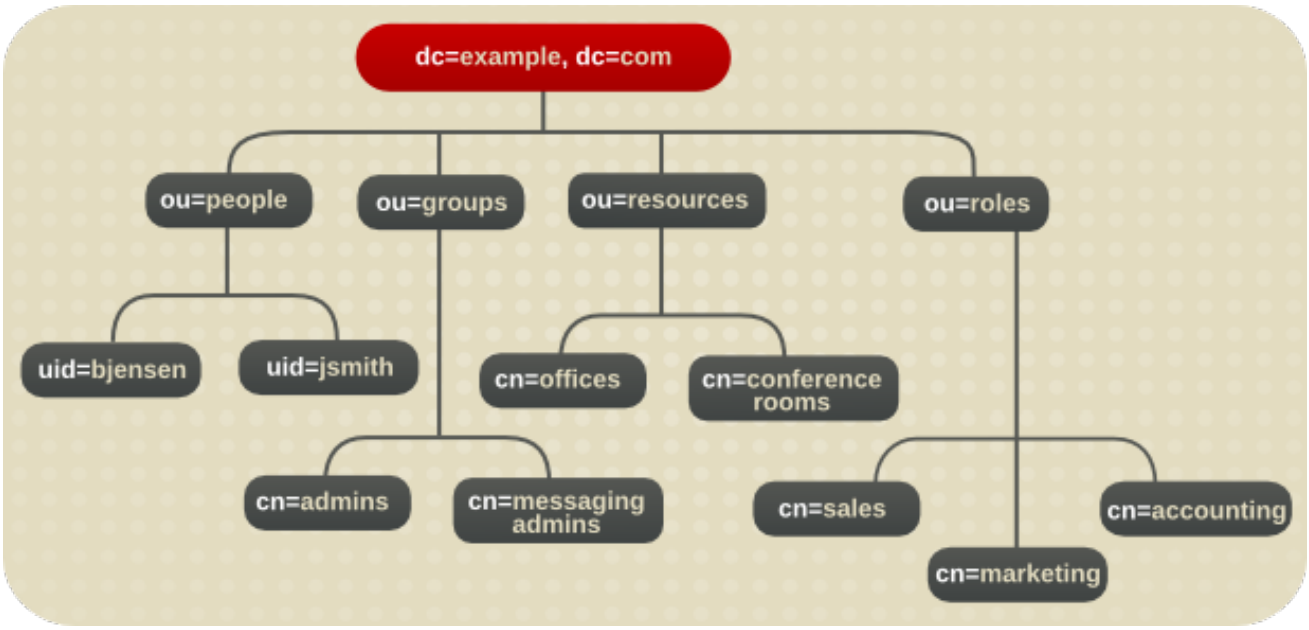
  2 つ目のグループ **cn=messaging admin** には、メールアカウントを管理するメール管理者のエントリーが含まれています。このグループは、メッセージングサーバーによって使用される管理者グループに対応します。Example Corp. は、メッセージングサーバーに設定するグループが、Directory Server 用に作成するグループとは異なるようにします。
- これらは、**ou=resources** ブランチの下に、2 つのブランチを作成します。1 つは会議部屋 (**ou=conference rooms**) 用で、もう 1 つはオフィスが 1 つです。**ou=offices**
- これらは、エントリーが管理グループに属するかどうかに応じて **mailquota** 属性 の値を提供するサービスクラス(CoS) を作成します。

  この CoS では、管理者には 100GB のメールクォータが与えられ、一般の Example Corp. の従業員には 5GB のメールクォータが与えられます。

  サービスクラスの詳細は、「サービスクラスについて」 を参照してください。

以下の図は、上記の設計手順から得られるディレクトリーツリーを示しています。

図10.1 Example Corp. のディレクトリーツリー



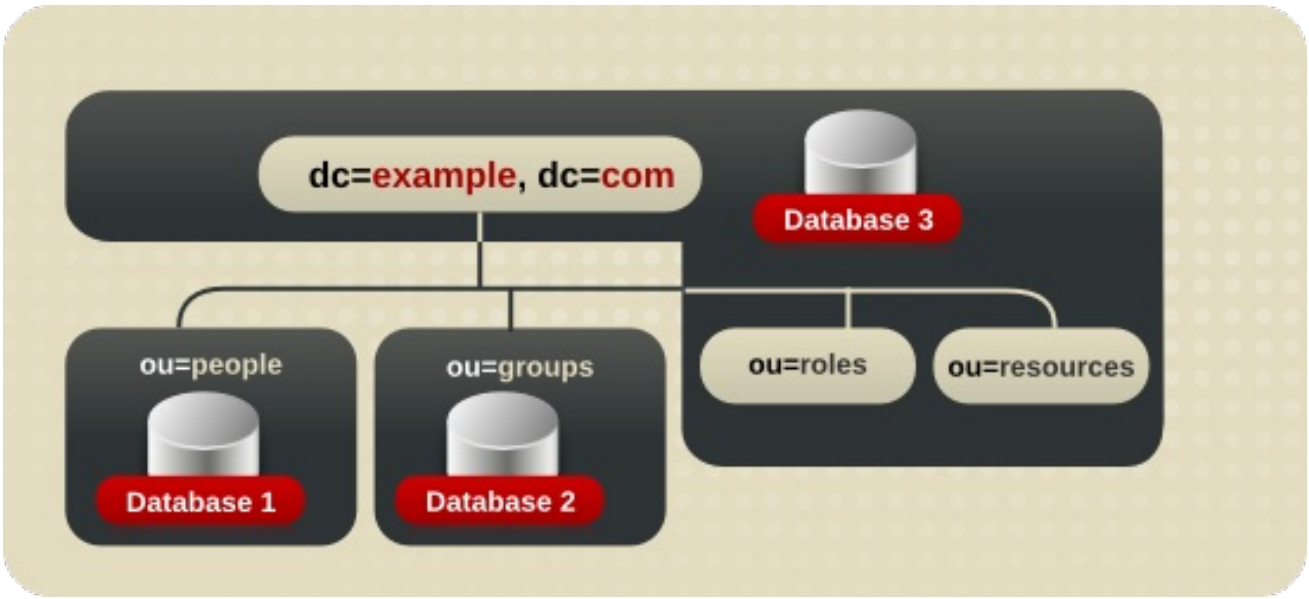
10.1.4. ローカルエンタープライズのトポロジー設計

この時点で、Example Corp. は、自社のデータベースおよびサーバートポロジーを設計する必要があります。以下のセクションでは、各トポロジーの詳細を説明します。

10.1.4.1. データベーストポロジー

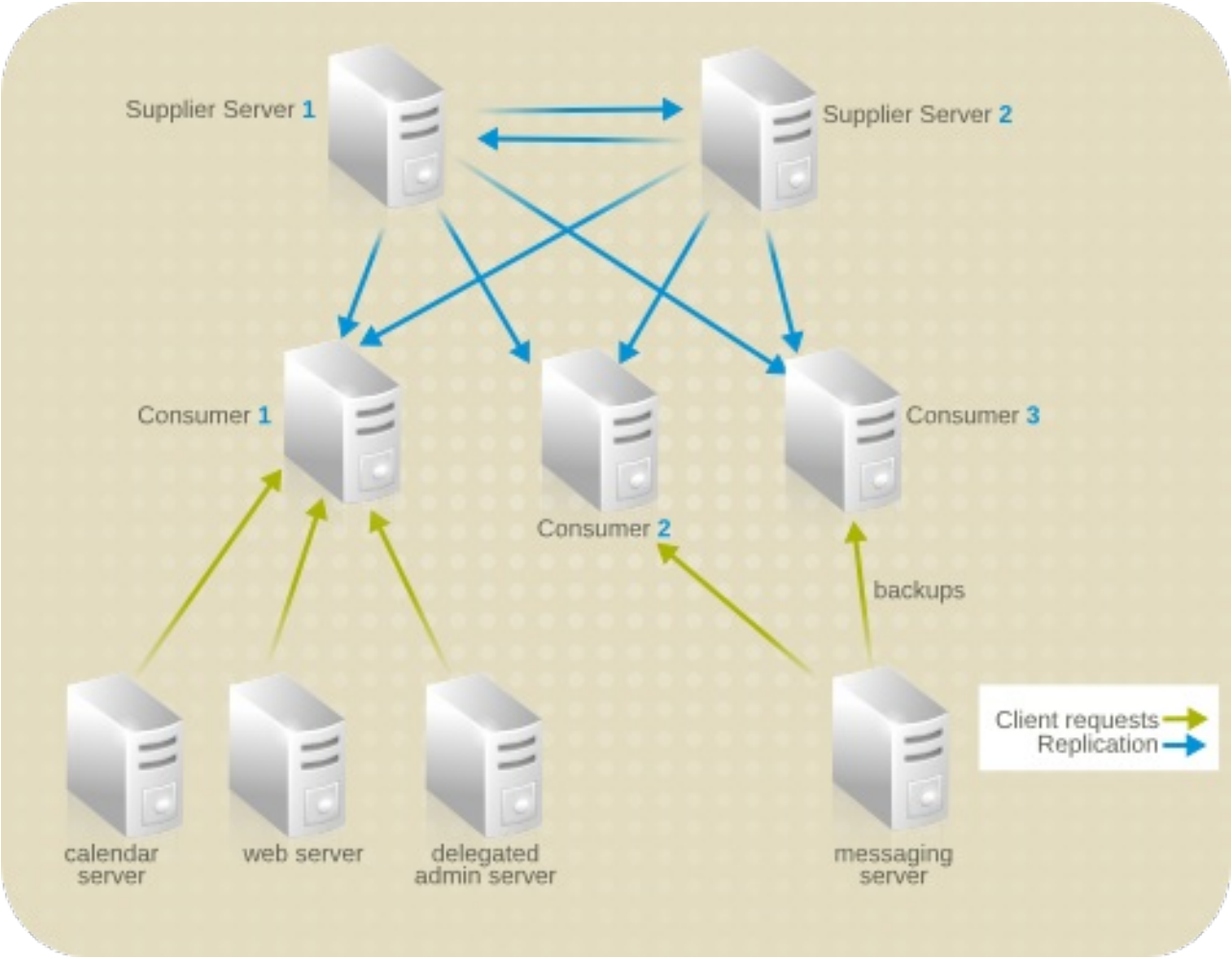
*企業は、人のブランチが1 つのデータベース(DB1)に保存され、グループブランチは別のデータベース(DB2)に保存され、リソースブランチ、ロールブランチ、**root suffix** の情報は 3 番目のデータベース(DB3)に格納されます。これは 図10.2「Example Corp. のデータベーストポロジー」 で説明されています。*

図10.2 Example Corp. のデータベーストポロジー



2 つのサプライヤーサーバーは、それぞれ Directory Server の Example Corp. の 3 つのコンシューマーサーバーをすべて更新します。これらのコンシューマーは、1 つのメッセージングサーバーおよび他の統一されたユーザー管理製品にデータを提供します。

図10.3 Example Corp. のサーバートポロジー



*互換性のあるサーバーからの **Modify** リクエストは、適切なコンシューマーサーバーにルーティングされます。コンシューマーサーバーは、スマート参照を使用して、変更されるデータのメインコピーを担当するサプライヤーサーバーへのリクエストをルーティングします。*

10.1.5. ローカルエンタープライズレプリケーション設計

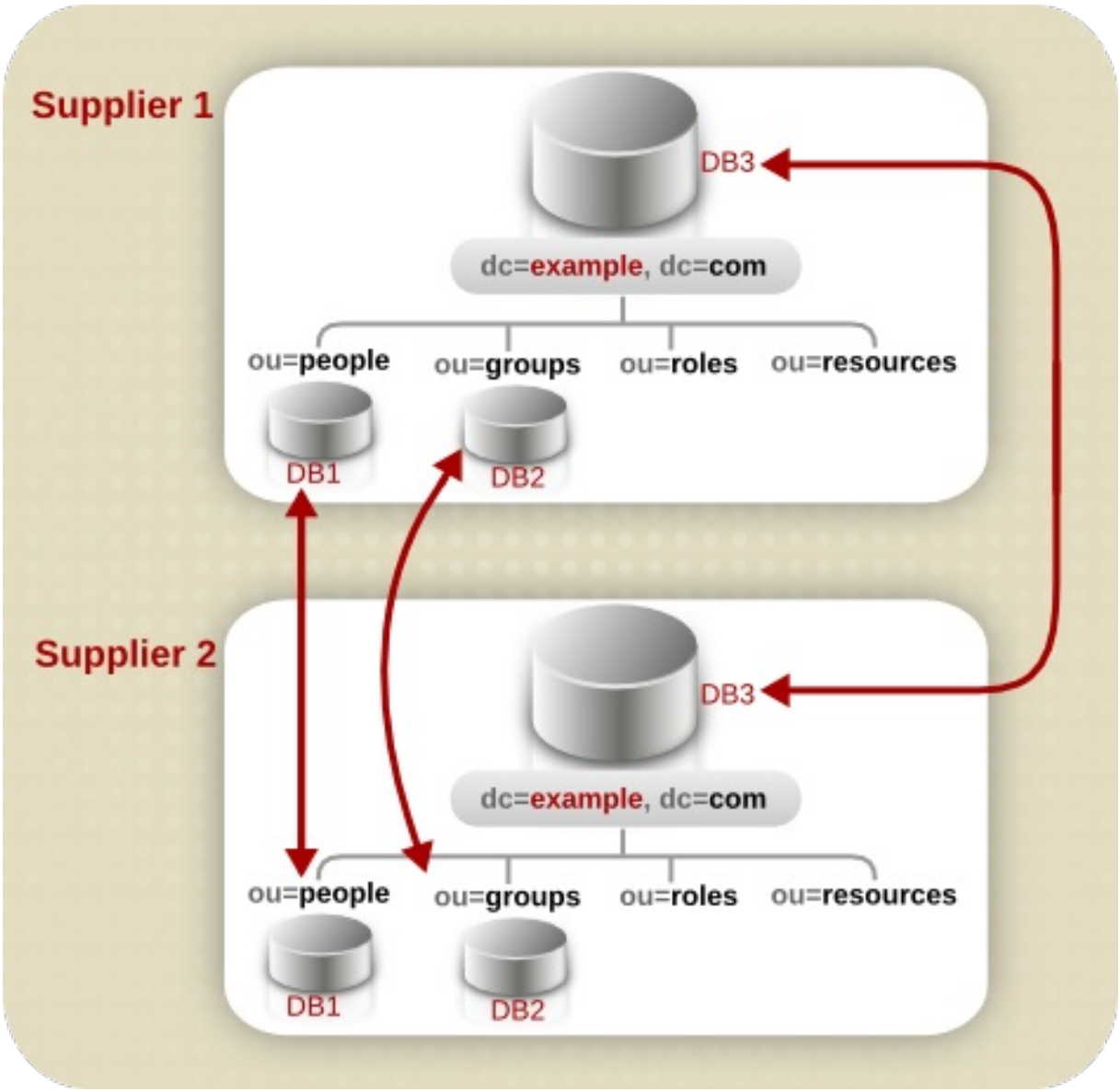
Example Corp. は、ディレクトリーデータの高可用性を確保するために、マルチサプライヤーのレプリケーション設計を使用することを決定します。マルチサプライヤーレプリケーションの詳細は、「マルチサプライヤーのレプリケーション」 を参照してください。

次のセクションでは、サプライヤーサーバーアーキテクチャーおよびサプライヤーコンシューマーサーバートポロジーの詳細を説明します。

10.1.5.1. サプライヤーアーキテクチャー

Example Corp. は、マルチサプライヤーのレプリケーションアーキテクチャーで 2 つのサプライヤーサーバーを使用します。サプライヤーが相互に更新することで、ディレクトリーデータの整合性が保たれます。Example Corp. のサプライヤーアーキテクチャーを以下に説明しています。

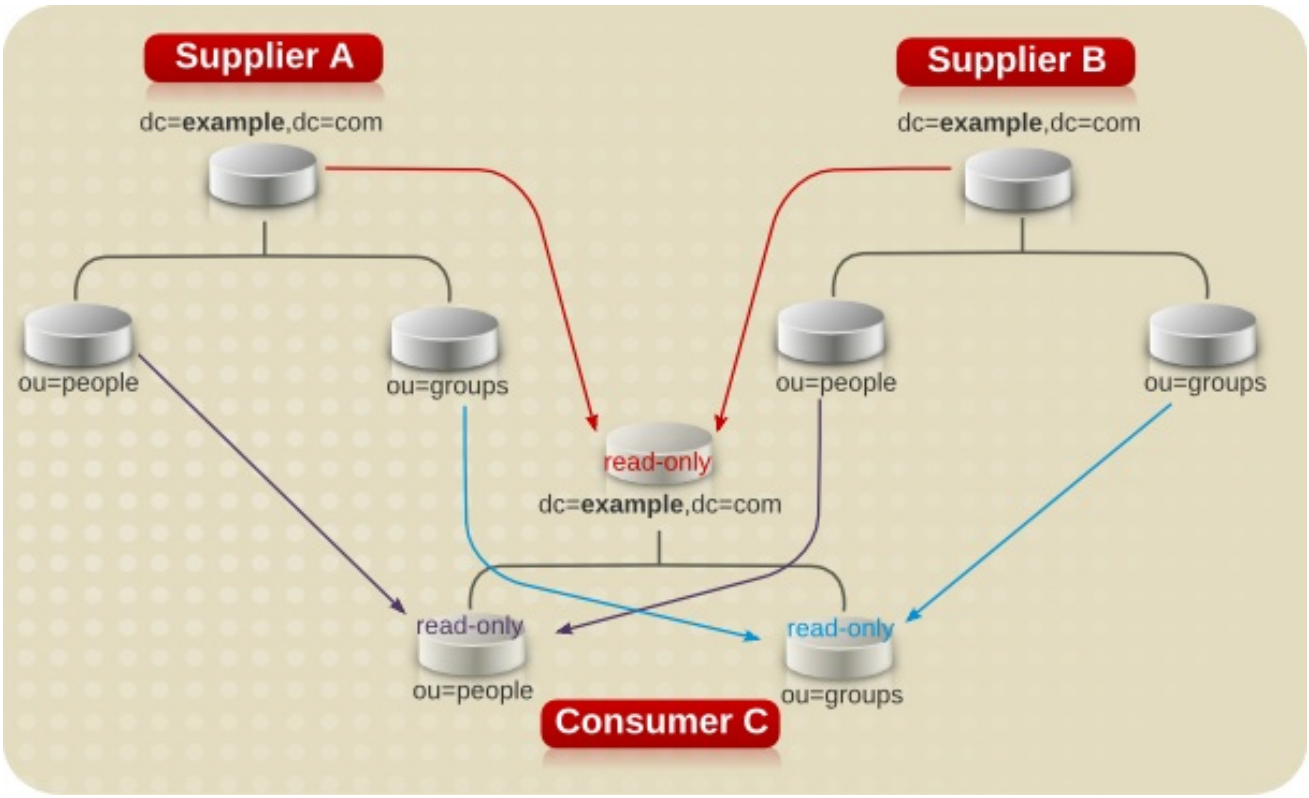
図10.4 Example Corp. のサプライヤーアーキテクチャー



10.1.5.2. サプライヤーコンシューマーアーキテクチャー

以下の図は、Example Corp. のディレクトリーのデプロイメントで、サプライヤーサーバーが各コンシューマーに複製する方法を示しています。3 つのコンシューマーサーバーが、それぞれ 2 つのサプライヤーサーバーで更新されます。これにより、サプライヤーサーバーの 1 つに障害が発生しても、コンシューマーは影響を受けなくなります。

図10.5 Example Corp. のサプライヤーとコンシューマーアーキテクチャー



10.1.6. ローカルの企業セキュリティー設計

Example Corp. は以下のセキュリティー設計を決定して、ディレクトリーデータを保護します。

- *従業員が独自のエントリーを変更できるようにする ACI を作成します。*

  *ユーザーは、**uid** 属性、**manager** 属性、および **department** 属性以外のすべての属性を変更できます。*

- *従業員データのプライバシーを保護するために、従業員とそのマネージャーのみが従業員の自宅住所と電話番号を閲覧できる ACI を作成します。*

- *2 つの管理者グループに適切なディレクトリー権限を与える ACI をディレクトリーツリーの root に作成します。*

  *ディレクトリー管理者グループは、ディレクトリーへのフルアクセスが必要です。メッセージング管理者グループは、**mailRecipient** および **mailGroup** オブジェクトクラス、およびそれらのオブジェクトクラスに含まれる属性、および **mail** 属性への書き込みおよび削除アクセスが必要です。Example Corp. は、メッセージング管理者グループ **write**、**delete**、および **add** に、メールグループを作成するためのグループサブディレクトリーへのパーミッションも付与します。*

- *ディレクトリーツリーのルートに一般的な ACI を作成し、読み取り、検索、および比較アクセスの匿名アクセスを許可します。*

  この ACI は、パスワード情報への匿名書き込みアクセスを拒否します。

- *サービス拒否攻撃や不適切な使用からサーバーを保護するために、ディレクトリークライアントがバインドするために使用する **DN** に基づいてリソース制限を設定しました。*

Example Corp. では、匿名ユーザーが検索要求に応じて一度に 100 エントリーを受信できるようにし、管理ユーザーにメッセージを送信して 1,000 エントリーを受信し、ディレクトリー管理者が無制限の数のエントリーを受信できるようにしています。

バインド DN に基づいてリソース制限を設定する方法の詳細は、『Red Hat Directory Server Administrator's Guide』の User Account Management の章を参照してください。

- *彼らは、パスワードの長さが 8 文字以上で、90 日後に有効期限が切れる必要があることを指定するパスワードポリシーを作成します。*

  パスワードポリシーの詳細は、「パスワードポリシーの設計」 を参照してください。

- *彼らは、会計ロールのメンバーがすべての給与情報にアクセスできるようにする ACI を作成します。*

10.1.7. ローカルエンタープライズ操作の決定

会社は、ディレクトリーの日常業務に関して次の決定を下します。

- *毎晩、データベースをバックアップします。*
- *SNMP を使用してサーバーの状態を監視します。*

  SNMP の詳細は、『Red Hat Directory Server Administrator's Guide 』を参照してください。

- *アクセスログとエラーログを自動でローテーションします。*
- *エラーログを監視し、サーバーが想定どおりに実行されていることを確認します。*
- *アクセスログを監視して、インデックスを作成すべき検索を選別します。*

アクセス、エラー、および監査ログの詳細は、『Red Hat Directory Server Administrator's Guide 』の Monitoring Server and Database Activity の章を参照してください。

10.2. 設計の例: 多国籍企業とそのエクストラネット

この例では、Example Corp. 用のディレクトリーインフラストラクチャーを構築します。国際。前述の例の Example Corp. は、大規模な多国籍企業に成長しました。この例は、Example Corp. の最後の例で作成されたディレクトリー構造に基づいて構築されており、新しいニーズに合わせてディレクトリー設計を拡張しています。

Example Corp. は、アメリカ、ヨーロッパ、アジアの 3 つの主要地域に広がる組織に成長しました。Example Corp. には現在 20,000 人を超える従業員がおり、その全員が Example Corp. のオフィスがある国に住んで働いています。Example Corp. は、社内コミュニケーションを改善し、Web アプリケーションの開発とデプロイメントを容易にし、セキュリティーとプライバシーを強化するために、全社的な LDAP ディレクトリーを開始することを決定しました。

国際企業のディレクトリーツリーを設計するには、ディレクトリーエントリーを論理的に収集する方法、データ管理をサポートする方法、およびグローバルスケールでのレプリケーションをサポートする方法を決定する必要があります。

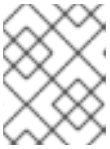
さらに、Example Corp. は、部品サプライヤーと取引先が使用するエクストラネットを作成したいと考えています。エクストラネット は、企業のイントラネットを外部クライアントに拡張したものです。

以下のセクションでは、Example Corp. 向けに多国籍ディレクトリーサービスとエクストラネットをデプロイするプロセスの手順について説明します。国際。

10.2.1. 多国籍企業のデータ設計

Example Corp.International は、サイト調査を行うためのデプロイメントチームを作成します。デプロイメントチームは、サイトサーベイから以下を決定します。

- メッセージングサーバーは、ほとんどの Example Corp. のサイト向けに、電子メールルーティング、配信、および読み取りサービスを提供するために使用されます。企業サーバーは、ドキュメント公開サービスを提供します。すべてのサーバーは Red Hat Enterprise Linux 7 で実行されます。
- Example Corp. は、データをローカルで管理できるようにする必要があります。たとえば、ヨーロッパのサイトは、ディレクトリーのヨーロッパのブランチを管理します。これは、ヨーロッパが自身のデータのメインコピーに責任を持つということでもあります。
- Example Corp. のオフィスは地理的に分散しているため、ユーザーやアプリケーションが 24 時間ディレクトリーを利用できるようにする必要があります。
- データ要素の多くは、複数の異なる言語のデータ値に対応している必要があります。



注記

すべてのデータは UTF-8 文字セットを使用しています。他の文字セットは LDAP 標準に違反しています。

デプロイメントチームは、エクストラネットのデータ設計に関する以下についても決定します。

- パーツのサプライヤーは、Example Corp. のディレクトリーにログインして、Example Corp との契約を管理する必要があります。パーツのサプライヤーは、名前やユーザーパスワードなどの、認証に使用するデータ要素に依存します。
- Example Corp. のパートナーは、ディレクトリーを使用して、パートナーネットワーク内の人々の連絡先情報 (メールアドレスや電話番号など) を検索します。

10.2.2. 多国籍企業のスキーマ設計

Example Corp. は、エクストラネットをサポートするスキーマ要素を追加することにより、元のスキーマ設計に基づいて構築されています。Example Corp. は、**exampleSupplier** オブジェクトクラスと **examplePartner** オブジェクトクラスの 2 つの新しいオブジェクトを追加します。

exampleSupplier オブジェクトクラスでは、1 つの属性(**exampleSupplierID** 属性)が許可されます。この属性には、Example Corp. によって割り当てられた一意の ID が含まれます。国際から、提携している各自動車部品サプライヤーへ。

examplePartner オブジェクトクラスでは、1 つの属性(**examplePartnerID** 属性)が許可されます。この属性には、Example Corp. によって割り当てられた一意の ID が含まれます。各トレードパートナーへの International。

デフォルトのディレクトリースキーマのカスタマイズに関する詳細は、「スキーマのカスタマイズ」 を参照してください。

10.2.3. 多国籍企業のディレクトリーツリー設計

拡張された要件に基づいて、Example Corp. は以下のディレクトリーツリーを作成します。

- ディレクトリーツリーの root は、**dc=com** 接尾辞です。この接尾辞の下に、Example Corp. は

2 つのブランチを作成します。1 つのブランチ **dc=exampleCorp,dc=com** には、Example Corp の内部のデータが含まれます。国際。他のブランチ **dc=exampleNet,dc=com** には、エクストラネットのデータが含まれます。

- イントラネットのディレクトリーツリー(**dc=exampleCorp,dc=com)** 下)には、Example Corp. がオフィスを持つリージョンの1 つに対応する 3 つの主要なブランチがあります。これらのブランチは、**l**（ローカリティー）属性を使用して識別されます。

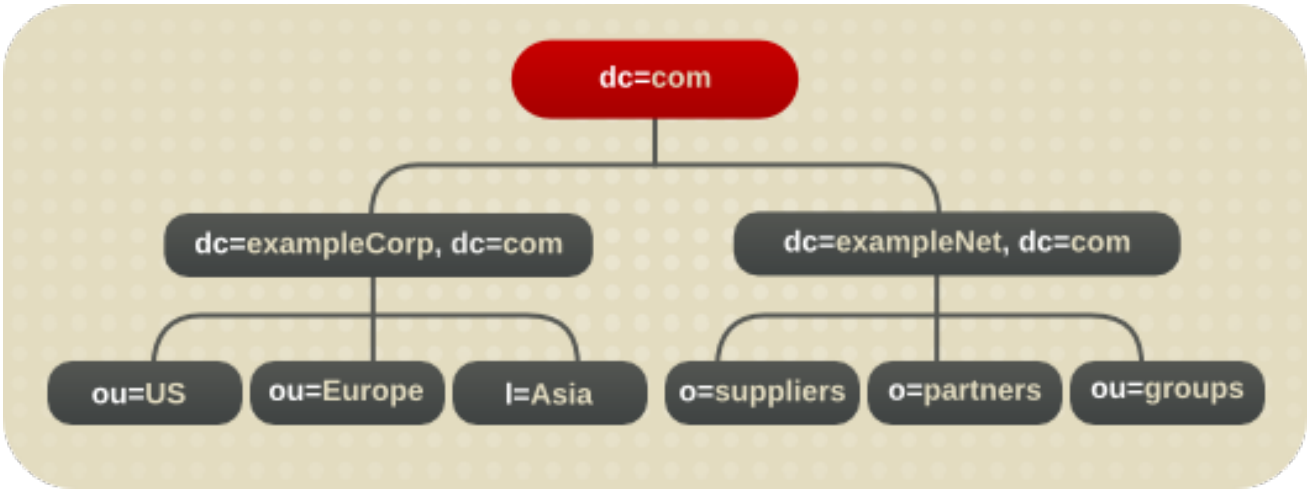
- **dc=exampleCorp,dc=com** 下の各メインブランチは、Example Corp の元のディレクトリーツリー設計に似ています。各ローカリティーの下に、Example Corp. は **ou=people**、**ou=groups**、**ou=roles**、および **ou=resources** ブランチを作成します。このディレクトリーツリーの設計の詳細は、図10.1「Example Corp. のディレクトリーツリー」 を参照してください。

- **dc=exampleNet,dc=com** ブランチの下に、Example Corp. は 3 つのブランチを作成します。サプライヤー用に1 つのブランチ(**o=suppliers**)、Partner (Partner) (**o=partners**)用ブランチを1 つ、および groups]-[用にブランチを1 つ、および1 つのブランチです。**ou=groups**

- エクストラネットの **ou=groups** ブランチには、エクストラネットの管理者のエントリーと、パートナーが自動検出したパーツの製造に関する最新情報をサブスクライブするメーリングリストのエントリーが含まれています。

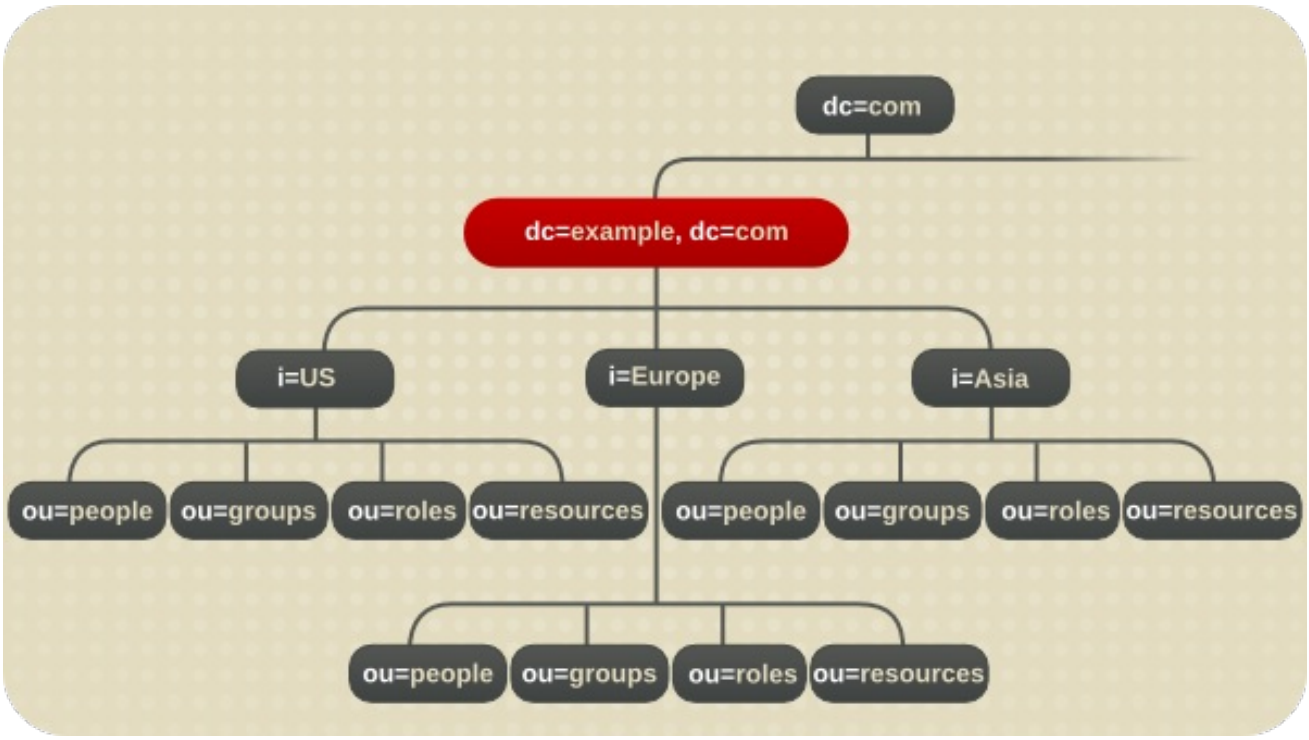
以下の図は、上記の設計手順から得られる基本的なディレクトリーツリーを示しています。

図10.6 Example Corp. の基本的なディレクトリーツリーインド在外のお客様:



以下の図は、Example Corp. イントラネットのディレクトリーツリーを示しています。

図10.7 Example Corp. のディレクトリーツリーInternational のイントラネット

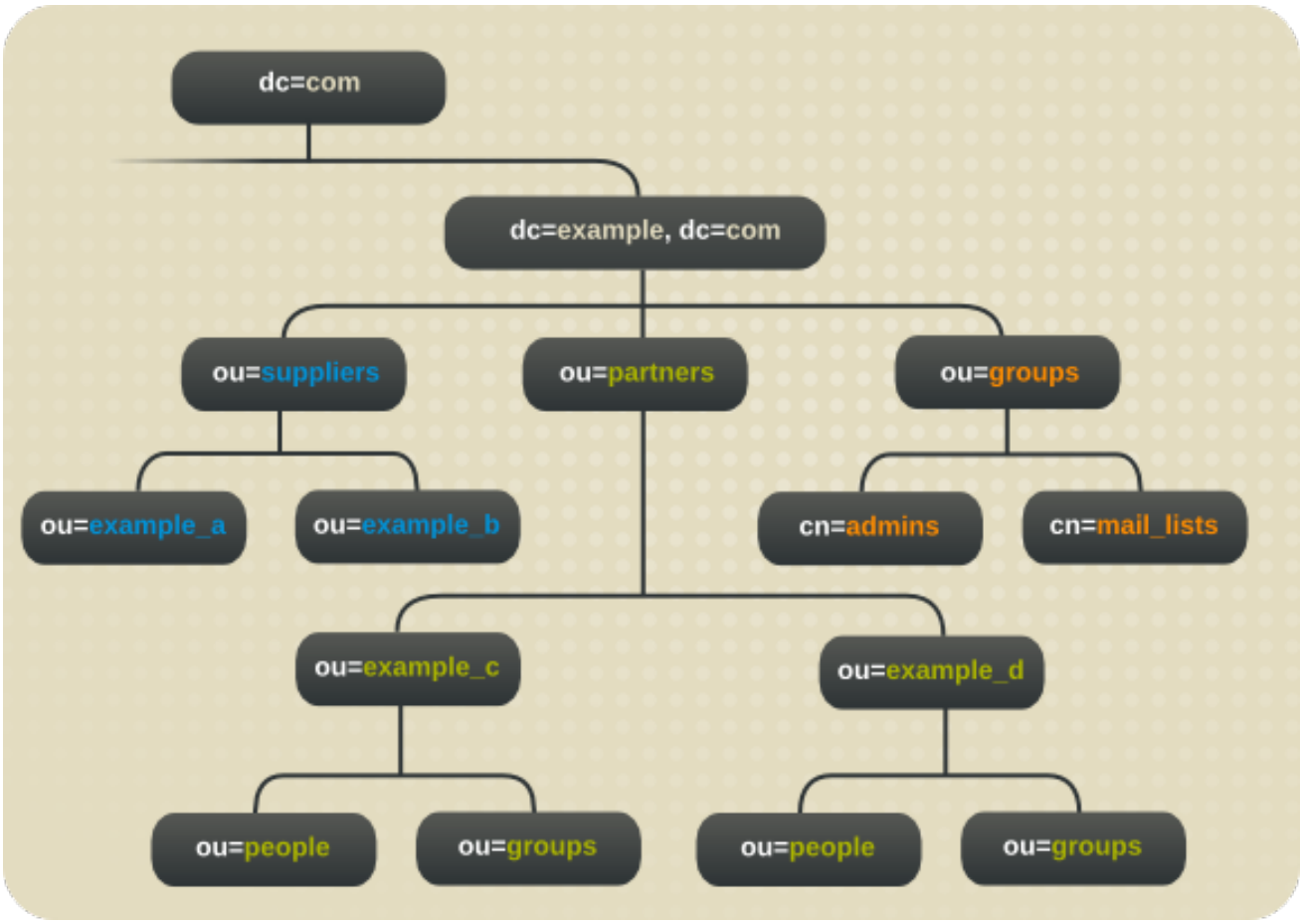


***l=Asia** エントリーのエントリーは、以下のように LDIF に表示されます。*

```
dn: l=Asia,dc=exampleCorp,dc=com

objectclass: top

objectclass: locality

l: Asia

description: includes all sites in Asia
```

以下の図は、Example Corp. のエクストラネットのディレクトリーツリーを示しています。

図10.8 Example Corp. のディレクトリーツリーInternational のエクストラネット



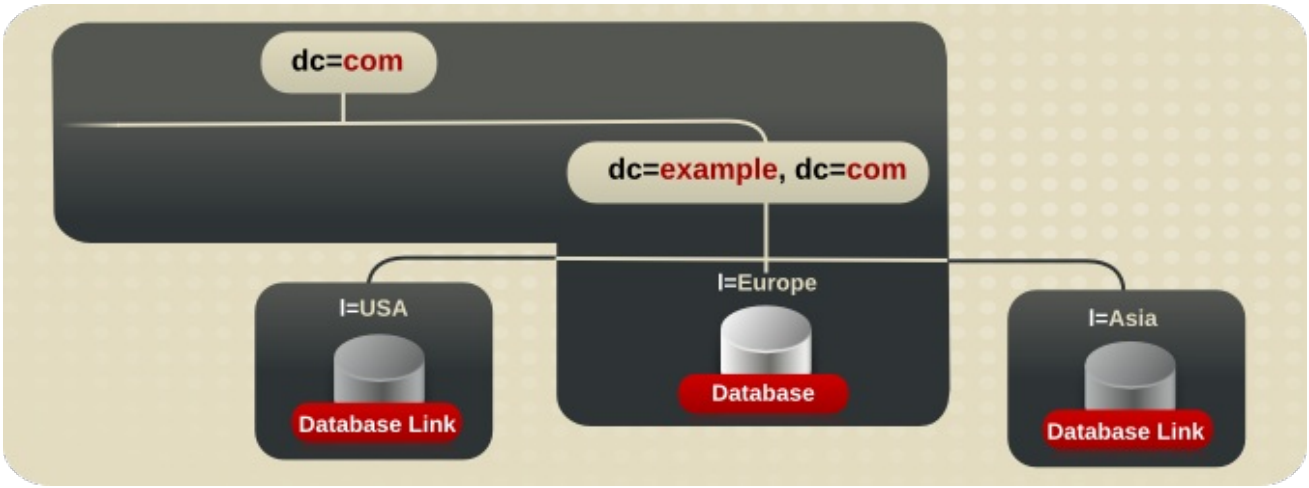
10.2.4. 多国籍企業のトポロジー設計

この時点で、Example Corp. はデータベースおよびサーバートポロジーを設計します。以下のセクションでは、各トポロジーの詳細を説明します。

10.2.4.1. データベーストポロジー

以下の図は、Example Corp. の主要地域の1 つであるヨーロッパのデータベーストポロジーを示しています。

図10.9 Example Corp. のデータベーストポロジーヨーロッパ



データベースリンクは、各国にローカルに保存されたデータベースを参照します。たとえば、Example

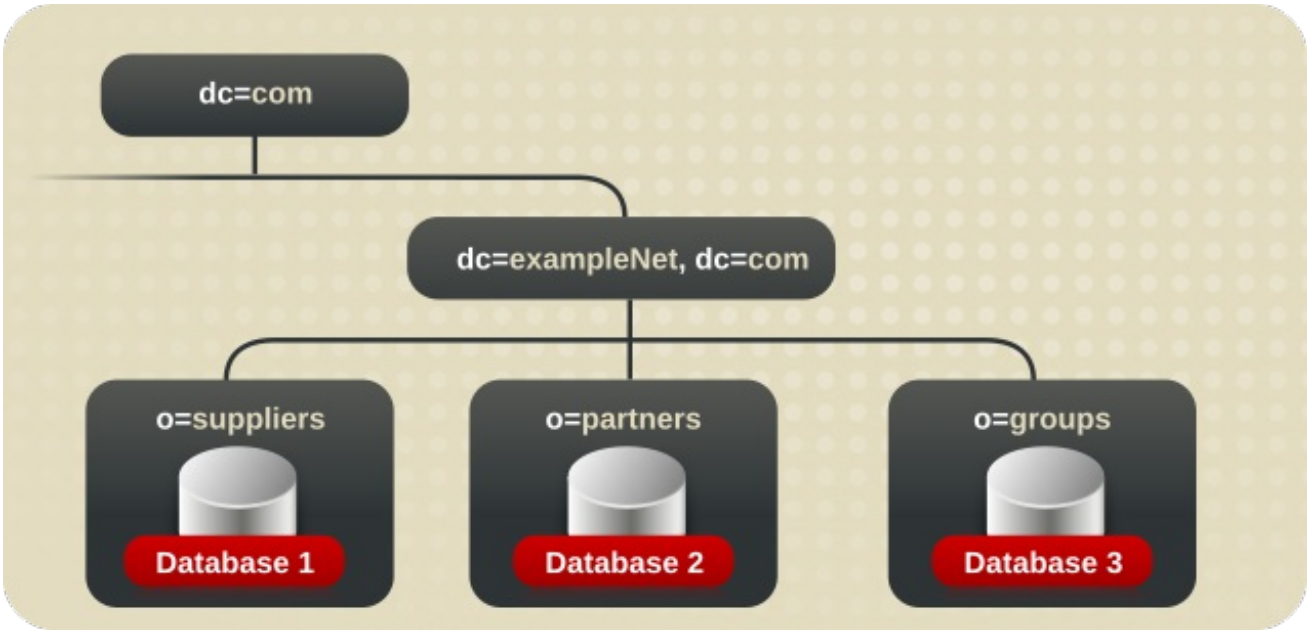
Corp. から受信した操作リクエストなどです。**l=US** ブランチ下のデータのヨーロッパサーバーは、オースティン Texas 内のサーバー上のデータベースへのデータベースリンクによってチェーンされます。データベースリンクおよびチェーンの詳細は、「チェーンの使用」 を参照してください。

dc=exampleCorp,dc=com およびルートエントリーである **dc=com** のデータのメインコピーは、**l=Europe** データベースに保存されます。

ヨーロッパのデータセンターには、エクストラネットのデータのメインコピーが含まれます。エクストラネットデータは、メインのブランチごとに 1 つずつ、3 つのデータベースに保存されます。**o=suppliers** のデータのメインコピーは 1 つ(DB1)に保存され、**o=partners** 用の **ou=groups** はデータベース 2 (DB2)に保存され、の場合はデータベース 3 (DB3)に保存されます。

エクストラネットのデータベースのトポロジーは以下のようになっています。

図10.10 Example Corp. のデータベーストポロジー—International のエクストラネット



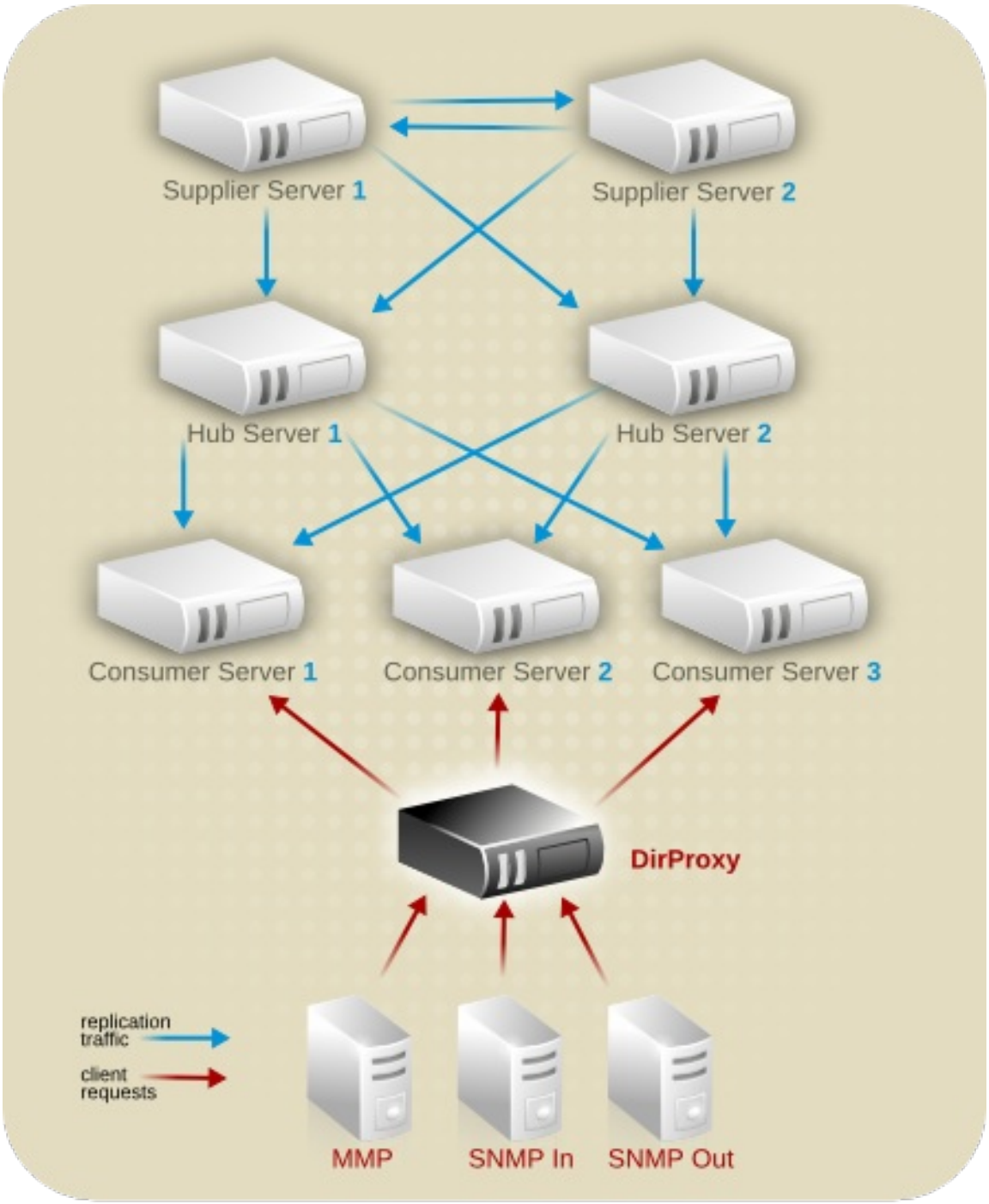
10.2.4.2. サーバートポロジー

Example Corp. は、企業イントラネット用とパートナーエクストラネット用の 2 つのサーバートポロジーを開発しています。

イントラネットの場合、Example Corp. は主要な地域ごとにサプライヤーデータベースを用意することにしました。つまり、3 つのデータセンターがあり、それぞれサプライヤーサーバー 2 台、ハブサーバー 2 台、およびコンシューマーサーバー 3 台が含まれます。

次の図は、Example Corp のアーキテクチャーを示しています。ヨーロッパのデータセンター:

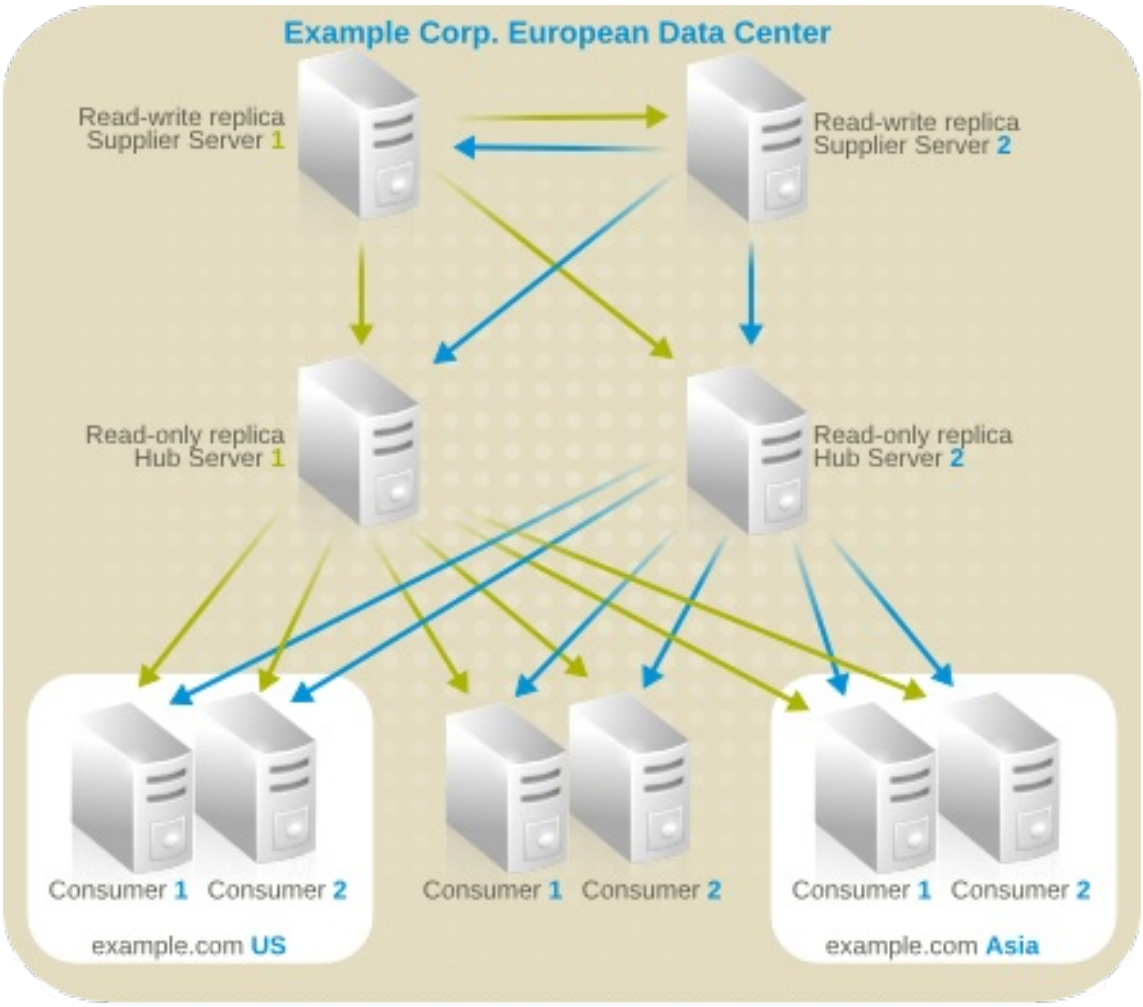
図10.11 Example Corp. のサーバートポロジーヨーロッパ



Example Corp.のエクストラネットのデータサプライヤーはヨーロッパにあります。このデータは、米国のデータセンターにある 2 つのコンシューマーサーバーと、アジアのデータセンターにある 2 つのコンシューマーサーバーにレプリケートされます。全体として、Example Corp. はエクストラネットをサポートするために 10 台のサーバーを必要とします。

次の図は、ヨーロッパのデータセンターにある Example Corp. のエクストラネットのサーバーアーキテクチャーを示しています。

図10.12 Example Corp. のサーバートポロジーInternational のエクストラネット



ハブサーバーは、ヨーロッパ、米国、アジアの各データセンターにある 2 つのコンシューマーサーバーにデータを複製します。

10.2.5. 多国籍企業のレプリケーション設計

Example Corp. では、ディレクトリーのレプリケーションを設計する際に、以下の点を考慮しています。

- *データはローカルで管理されます。*
- *ネットワーク接続の品質は、サイトごとに異なります。*
- *データベースリンクは、リモートサーバー上のデータを接続するために使用されます。*
- *データの読み取り専用コピーを含むハブサーバーは、コンシューマーサーバーにデータを複製するために使用されます。*

ハブサーバーは、メールサーバーや Web サーバーなどの重要なディレクトリー対応アプリケーションの近くに配置されます。

ハブサーバーは、サプライヤーサーバーからレプリケーションの負担を取り除くため、サプライヤーサーバーは書き込み操作に集中できます。将来、Example Corp. が拡大し、さらに多くのコンシューマーサーバーを追加する必要がある場合、追加のコンシューマーがサプライヤーサーバーのパフォーマンスに影響を与えることはありません。

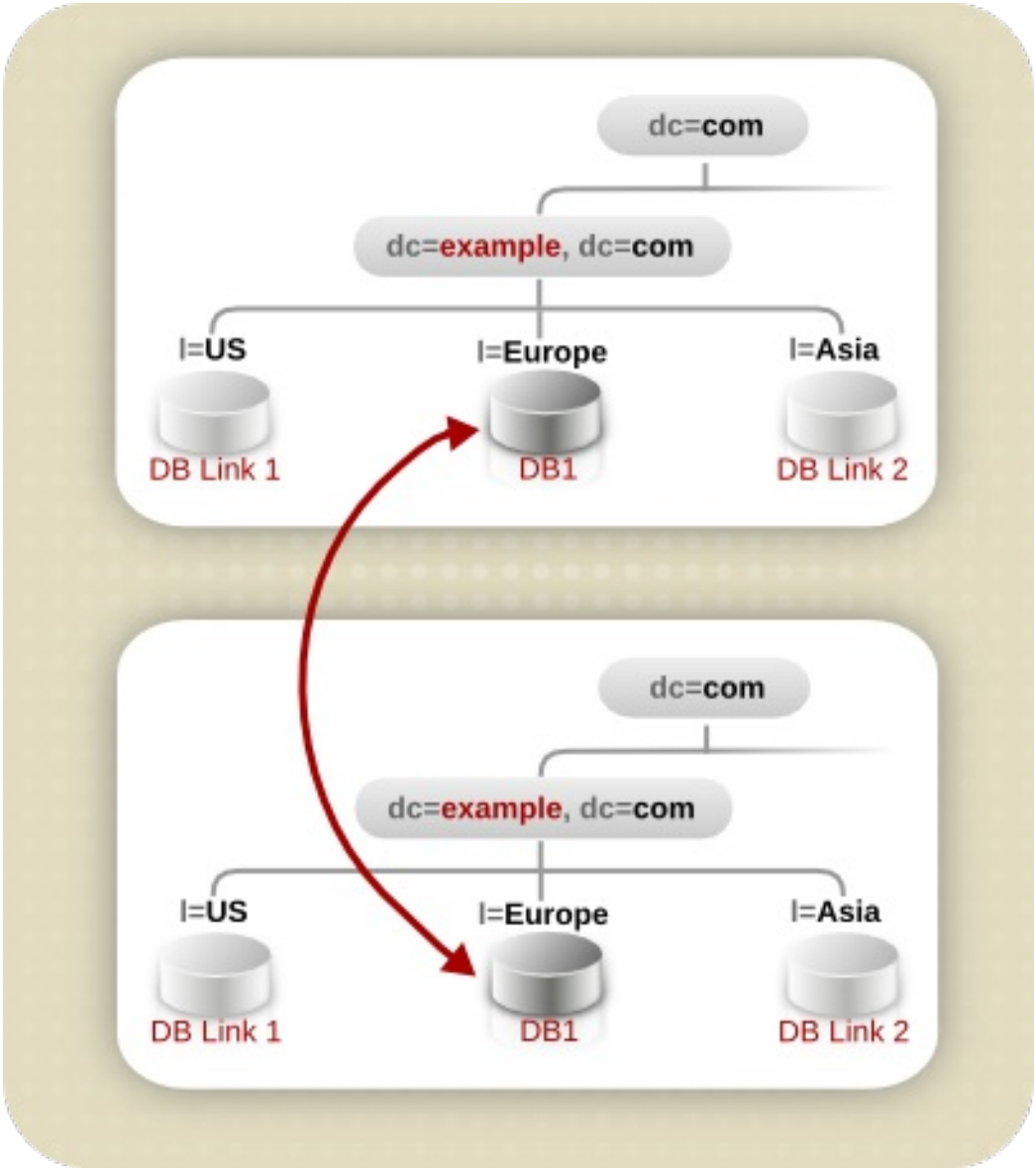
ハブサーバーの詳細は、「カスケードレプリケーション」 を参照してください。

10.2.5.1. サプライヤーアーキテクチャー

Example Corp. のイントラネットの場合、各地域はそのデータのメインコピーを格納し、データベースリンクを使用して他の地域のデータにチェーンします。データのメインコピーについては、各地域でマルチサプライヤーレプリケーションアーキテクチャーが使用されています。

*次の図は、ヨーロッパのサプライヤーアーキテクチャーを示しています。これには、**dc=exampleCorp,dc=com** および **dc=com** の情報が含まれます。*

図10.13 Example Corp. のサプライヤーアーキテクチャーヨーロッパ

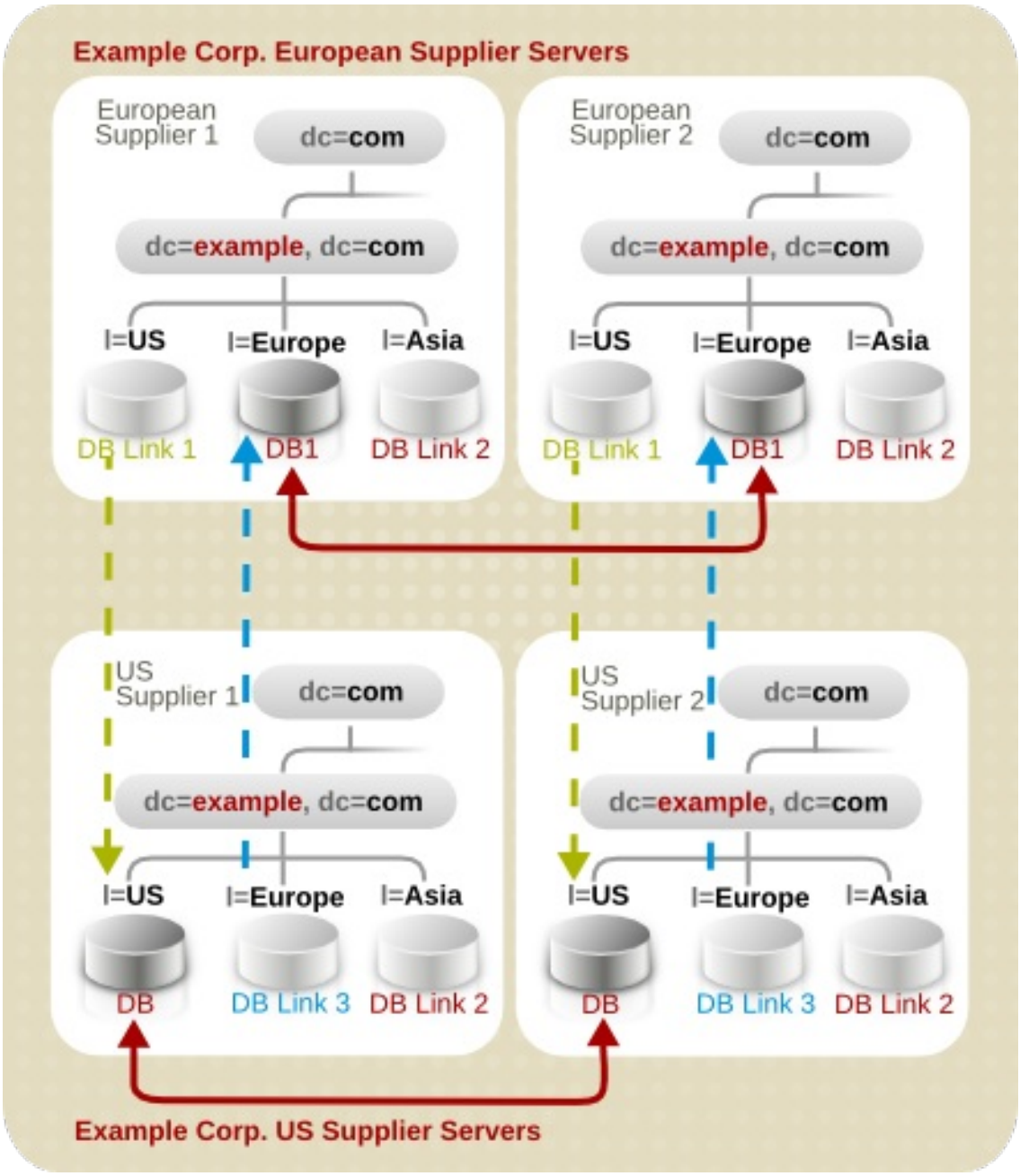


各地域には 2 つのサプライヤーが含まれており、そのサイトのデータのメインコピーを共有しています。したがって、各地域は、独自のデータのメインコピーを担当します。マルチサプライヤーアーキテクチャーを使用すると、データの可用性が確保され、各サプライヤーサーバーによって管理されるワークロードのバランスを取ることができます。

全体的な障害のリスクを軽減するために、Example Corp. では、各サイトで複数の読み書き可能なサプライヤーディレクトリーサーバーを使用しています。

以下の図は、ヨーロッパの 2 つのサプライヤーサーバーと、米国内の 2 つのサプライヤーサーバー間の対話を示しています。

図10.14 Example Corp. のマルチサプライヤーレプリケーション設計ヨーロッパと Example Corp.US



Example Corp. の間にも同じ関係が存在します。米国および Example Corp. アジア、および Example Corp. 間ヨーロッパと Example Corp.Asia.

10.2.6. 多国籍企業のセキュリティー設計

Example Corp.International は、以前のセキュリティー設計を基に、新しい多国籍イントラネットをサポートするために以下のアクセス制御を追加しました。

- *Example Corp. は、一般的な ACI をイントラネットのルートに追加し、各国および各国の下の支社により制限の厳しい ACI を作成します。*
- *Example Corp. は、マクロ ACI を使用して、ディレクトリー内の ACI の数を最小限に抑えることにしました。*

  Example Corp. はマクロを使用して、ACI のターゲットまたはバインドルール部分で DN を表します。ディレクトリーが着信 LDAP 操作を取得すると、ACI マクロは LDAP 操作の対象となるリソースと照合されます。一致する場合、マクロはターゲットリソースの DN の値に置き換え

られます。

*マクロ ACI の詳細は、『*Red Hat Directory Server Administrator's Guide *』を参照してください。*

Example Corp. は、自社のエクストラネットをサポートするために、以下のアクセス制御を追加します。

- *Example Corp. は、すべてのエクストラネットアクティビティーに証明書ベースの認証を使用するかどうかを決定します。ユーザーがエクストラネットにログインする際には、デジタル証明書が必要となります。ディレクトリーは証明書を保存するために使用されます。ディレクトリーは証明書を保存するため、ディレクトリーに保存された公開鍵を検索することで、暗号化されたメールを送信することができます。*

- *Example Corp. は、エクストラネットへの匿名アクセスを禁止する ACI を作成します。これにより、サービス拒否攻撃からエクストラネットを守ることができます。*

- *Example Corp. は、Example Corp.がホストするアプリケーションのみから、ディレクトリーデータの更新を行うようにしています。つまり、エクストラネットを利用するパートナーやサプライヤーは、Example Corp. が提供するツールのみを利用できることを意味します。エクストラネットのネットユーザーを Example Corp. の推奨するツールに制限することで、Example Corp. の管理者は監査ログを使用してディレクトリーの使用状況を追跡することができ、Example Corp. 以外のエクストラネットのユーザーがもたらす可能性のある問題の種類を制限することができます。国際。*

付録A DIRECTORY SERVER の RFC サポート

注記

本章では、注目すべきサポート対象の LDAP 関連の RFC を紹介します。これは、Directory Server がサポートする RFC の完全なリストではありません。

A.1. LDAPV3 の機能

Technical Specification Road Map (RFC 4510)

これは追跡ドキュメントであり、要件は含まれていません。

Protocol (RFC 4511)

サポート対象例外:

- RFC 4511 Section 4.4.1.Notice of Disconnection: この場合、Directory Server は接続を終了します。
- RFC 4511 Section 4.5.1.3.SearchRequest.derefAliases: LDAP エイリアスはサポートされません。
- RFC 4511 Section 4.13.IntermediateResponse メッセージ

Directory Information Models (RFC 4512)

サポート対象例外:

- RFC 4512 Section 2.4.2.Structural Object Classes: Directory Server は、複数の構造オブジェクトクラスを持つエントリーをサポートします。
- RFC 4512 Section 2.6.エイリアスエントリー
- RFC 4512 Section 4.1.2.Attribute Types: 属性タイプ **COLLECTIVE** はサポートされていません。
- RFC 4512 Section 4.1.4.Matching Rule Uses
- RFC 4512 Section 4.1.6.DIT Content Rules
- RFC 4512 Section 4.1.7.DIT Structure Rules and Name Forms
- RFC 4512 Section 5.1.1. altServer

RFC 4512 により、LDAP サーバーが前述の例外に対応できないことに注意してください。詳細は、RFC 4512 Section 7.1. を参照してください。サーバーのガイドライン.

Authentication Methods and Security Mechanisms (RFC 4513)

サポート対象

String Representation of Distinguished Names (RFC 4514)

サポート対象

String Representation of Search Filters (RFC 4515)

サポート対象

Uniform Resource Locator (RFC 4516)

サポート対象ただし、この RFC は主に LDAP クライアントに焦点を当てています。

Syntaxes and Matching Rules (RFC 4517)

サポート対象例外:

- ***directoryStringFirstComponentMatch***
- ***integerFirstComponentMatch***
- ***objectIdentifierFirstComponentMatch***
- ***objectIdentifierFirstComponentMatch***
- ***keywordMatch***
- ***wordMatch***

Internationalized String Preparation (RFC 4518)

サポート対象

Schema for User Applications (RFC 4519)

サポート対象

A.2. 認証方法

Anonymous SASL Mechanism (RFC 4505)

サポート対象外。RFC 4512 は、***ANONYMOUS*** SASL メカニズムを必要としない点に留意してください。ただし、Directory Server は LDAP 匿名バインドをサポートします。

External SASL Mechanism (RFC 4422)

サポート対象

Plain SASL Mechanism (RFC 4616)

サポート対象外。RFC 4512 は、***PLAIN*** SASL メカニズムを必要としない点に留意してください。ただし、Directory Server は LDAP 匿名バインドをサポートします。

SecurID SASL Mechanism (RFC 2808)

サポート対象外。ただし、Cyrus SASL プラグインが存在する場合は、Directory Server はそれを使用することができます。

Kerberos V5 (GSSAPI) SASL Mechanism (RFC 4752)

サポート対象

CRAM-MD5 SASL Mechanism (RFC 2195)

サポート対象

Digest-MD5 SASL Mechanism (RFC 2831)

サポート対象

One-time Password SASL Mechanism (RFC 2444)

サポート対象外。ただし、Cyrus SASL プラグインが存在する場合は、Directory Server はそれを使用することができます。

A.3. X.509 証明書のスキーマおよび属性サポート

X.509 証明書の LDAP スキーマ定義 (RFC 4523)

- *属性タイプとオブジェクトクラス: サポート対象*
- *構文: サポート対象外(Directory Server はバイナリーおよび octet 構文を使用)*
- *一致規則: サポートされていません。*

付録B 改訂履歴

改訂番号は本ガイドに関するものであり、Red Hat Directory Server のバージョン番号ではありません。

改訂 11.5-1 **Tue May 10 2022** **Marc Muehlfeld**
Red Hat Directory Server 11.5 版のガイドをリリース

改訂 11.4-1 **Tue Nov 09 2021** **Marc Muehlfeld**
Red Hat Directory Server 11.4 版のガイドをリリース

改訂 11.3-1 **Tue May 11 2021** **Marc Muehlfeld**
Red Hat Directory Server 11.3 版のガイドをリリース

改訂 11.2-1 **Tue Nov 03 2020** **Marc Muehlfeld**
Red Hat Directory Server 11.2 版のガイドをリリース

改訂 11.1-1 **Tue Apr 28 2020** **Marc Muehlfeld**
Red Hat Directory Server 11.1 版のガイドをリリース

改訂 11.0-1 **Thu Nov 21 2019** **Marc Muehlfeld**
Red Hat Directory Server 11.0 版のガイドをリリース