Red Hat Directory Server 12

ディレクトリーデータベースの設定

Red Hat Directory Server データベースの設定および管理

Last Updated: 2024-09-19

Red Hat Directory Server 12 ディレクトリーデータベースの設定

Red Hat Directory Server データベースの設定および管理

法律上の通知

概要

データベース、接尾辞、チェーンポリシー、データベースリンク、および参照を設定できます。仮想ディレクトリーツリーを使用して、エントリーをカスタムグループまたは階層に整理します。

目次

RED HAT DIRECTORY SERVER に関するフィードバックの提供

Red Hat のドキュメントおよび製品に関するご意見をお待ちしております。ドキュメントの改善点があればお知らせください。以下の方法で送信してください。

- Jira を通じて Red Hat Directory Server ドキュメントに関するフィードバックを送信する場合 (アカウントが必要):
  1. Red Hat Issue Tracker にアクセスしてください。
  2. **Summary** フィールドにわかりやすいタイトルを入力します。
  3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
  4. ダイアログの下部にある **Create** をクリックします。
- Jira を通じて Red Hat Directory Server 製品に関するフィードバックを送信する場合 (アカウントが必要):
  1. Red Hat Issue Tracker にアクセスしてください。
  2. **Create Issue** ページで、**Next** をクリックします。
  3. **Summary** フィールドに入力します。
  4. **Component** フィールドでコンポーネントを選択します。
  5. **Description** フィールドに以下の内容を入力します。
     a. 選択したコンポーネントのバージョン番号。
     b. 問題を再現するための手順、または改善のための提案。
  6. **Create** をクリックします。

第1章 接尾語を別のデータベースに格納する

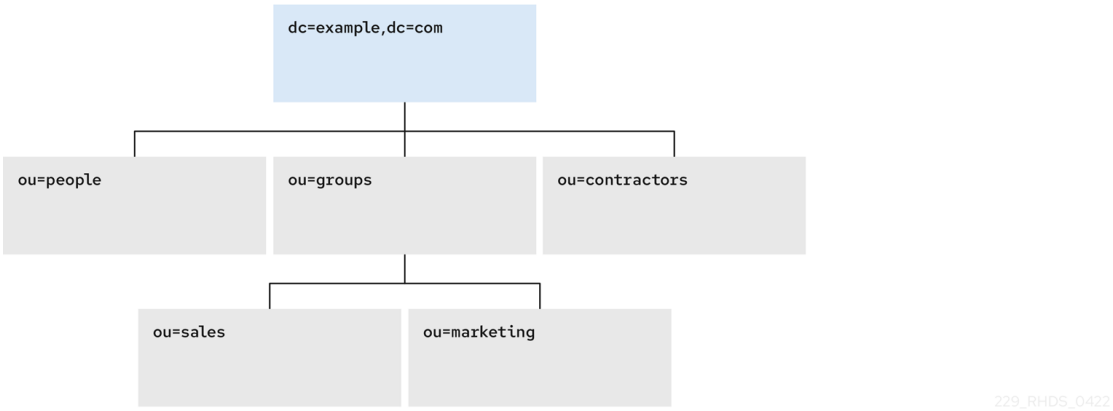
Directory Server では、インスタンス内のデータを複数のデータベースに分割して、データの分散保存ロジックを設計することができます。データ分割の方法として、ディレクトリーツリーの接尾辞を利用することができます。

複数のディレクトリーツリーを作成し、ルートの接尾辞で別々のデータベースに格納することができます。また、1つのディレクトリーツリーをブランチに分割し、サブ接尾辞によってブランチを別々のデータベースに格納することもできます。

1.1. データ構造における接尾語のロール

ディレクトリーサーバーは、データをディレクトリーツリー (DIT) と呼ばれる階層構造で表示します。以下は、簡単なディレクトリーツリーです。

図1.1 単一のルート接尾辞を持つシンプルなディレクトリーツリー



各ディレクトリーツリーには、**dc=example,dc=com** などのディレクトリーツリーの命名コンテキストを定義する単一の root エントリーがあります。

さまざまなディレクトリーツリーを異なるデータベースに保存し、これらのデータベースを複数のサーバーに分散できます。

接尾辞を使用して、データストレージの分散ロジックを定義できます。接尾辞は、ディレクトリーツリーのブランチ (サブツリー) と特定のデータベースを関連付けます。

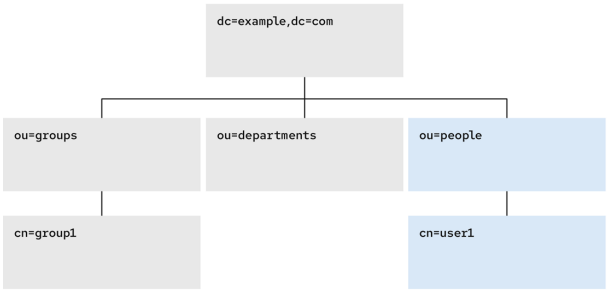
これにより、サーバーの1つのインスタンス内に複数のデータベースを指定できます。一つのデータベースに縛られることはありません。

1.2. ルート接尾辞とサブ接尾辞の比較

ルート接尾辞は、ディレクトリーツリー (DIT) 全体とデータベースを関連付けます。ルート接尾辞には、親の接尾辞がありません。

ディレクトリーツリーのブランチを別のデータベースに保存する場合、サブ接尾辞を作成します。これは、ツリーのブランチをブランチの祖先とは別のデータベースに関連付けるものです。サブ接尾辞は、親接尾辞に付けなければなりません。親接尾辞はルート接尾辞でもサブ接尾辞でも構いません。つまり、任意のサブツリーのブランチを別のデータベースに保存することができます。

図1.2 別のデータベースにサブ接尾辞を持つディレクトリーツリー



この例では、**ou=people,dc=example,dc=com** のサブ接尾辞が 1 つのデータベースに格納され、ルート接尾辞の下にある残りのディレクトリーツリーが別のデータベースに格納されています。

サブ接尾辞を使用する利点:

- データベースのメンテナンス (インポート/エクスポート/インデックス作成) を細かいレベルで実行できます。
- サブ接尾辞を別のディスクに保存できるため、ディスク領域の問題を解決できます。

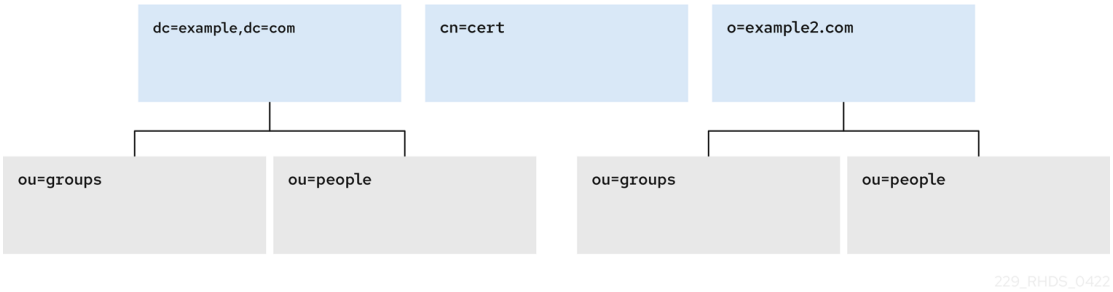
サブ接尾辞を使うことのデメリット

- セットアップ中にさらに多くの管理アクションが必要になります。
- レプリケーションには、サブ接尾辞ごとに個別の設定とレプリカ合意が必要です。

1.3. いくつかのルート接尾辞

また、1 つのインスタンスに異なるルート接尾辞を持つ複数のディレクトリーツリー (DIT) を持つことができます。例えば、ある部分のデータをユーザールートから分離したい場合などです。

図1.3 ルート接尾辞で定義された複数のディレクトリーツリー



クライアントが **dc=example,dc=com** のツリーを検索すると、他のツリーからのエントリーは検索アルゴリズムでは制限されているため、検索では返されません。

次に、インスタンスのデフォルトとなるディレクトリーツリーとネーミングコンテキストを選択できます。

1.4. コマンドラインを使用したルート接尾辞の作成

この手順では、コマンドラインでディレクトリーツリーのルート接尾辞を作成する方法を説明します。

手順

1. オプションとして、すでに使用されている接尾辞とバックエンドデータベースをリストアップします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
```

括弧内の名前は、対応する接尾辞のデータを保存するバックエンドデータベースです。次のステップでルート接尾辞を作成する際に、既存のデータベース名を使用することはできません。

2. ルート接尾辞の DN を **--suffix** 引数で指定し、**--be-name** 引数で新しいデータベースと関連付ける。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend create --
suffix="dc=example,dc=net" --be-name="example"
```

検証

- この手順の最初のステップにあるコマンドを使用して、接尾辞とデータベースをリスト表示します。

1.5. WEB コンソールによるルート接尾辞の作成

この手順では、ブラウザー上でディレクトリーツリーのルート接尾辞を作成する方法を説明します。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database** で、設定ツリーの下にある**Create Suffix**ボタンをクリックします。
2. **Suffix DN** と **Database Name** を記入する。
3. **Create The Top Suffix Entry** を選択し、**Create Suffix** をクリックします。

検証

- 新しい接尾辞は、接尾辞のツリーに表示されるはずです。

1.6. デフォルトのネーミングコンテキストの変更

ネーミングコンテキストは、その DIT のエントリーに対するルート名前空間を定義するディレクトリーツリー (DIT) の属性です。複数のルート接尾辞を持つインスタンスにデータを生成する場合、インスタンスには複数の DIT があり、それぞれ命名コンテキストは異なります。

この手順では、インスタンスで複数のルート接尾辞を使用する際に、コマンドラインでデフォルトのネーミングコンテキストを変更する方法を説明します。

インスタンスにアクセスするクライアントでは、使用する必要のあるネーミングコンテキストがわからない可能性があります。Directory Server は、デフォルトのネーミングコンテキストが既知の他の設定がない場合に、クライアントに対して通知します。

cn=config の **nsslapd-defaultnamingcontext** 属性でデフォルトのネーミングコンテキストを設定します。Directory Server はこの値を Directory Server Agent Service Entry(root DSE) に伝播し、クライアントは匿名にクエリーできます。

前提条件

- インスタンスのデフォルトのネーミングコンテキストを定義する root 接尾辞を作成している。

手順

1. オプション: 現在のデフォルトネーミングコンテキストを表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config get nsslapd-defaultnamingcontext
nsslapd-defaultnamingcontext: dc=example,dc=com
```

2. **nsslapd-defaultnamingcontext** パラメーターの値を、必要なネーミングコンテキストに置き換えます。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace nsslapd-defaultnamingcontext=dc=example,dc=net
```

検証

- 現在のデフォルトネーミングコンテキストを表示します。値を更新する必要があります。

関連情報

- コマンドラインを使用したルート接尾辞の作成
- Web コンソールによるルート接尾辞の作成

1.7. コマンドラインによるサブ接尾辞の作成

コマンドラインを使用して、ディレクトリーツリーのサブ接尾辞を作成できます。

前提条件

- サブ接尾辞の親接尾辞が作成されている。

手順

1. オプションとして、すでに使用されている接尾辞とバックエンドデータベースをリストアップします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
```

括弧内の名前は、対応する接尾辞のデータを保存するバックエンドデータベースです。次のステップでサブ接尾辞を作成する際に、既存のデータベース名を使用することはできません。

2. **--suffix** 引数でサブ接尾辞の完全な DN を指定し、**--be-name** 引数を使用してその DN を新しいデータベースに関連付け、**--parent-suffix** 引数で親接尾辞を指定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend create --
suffix="ou=People,dc=example,dc=com" --be-name="example" --parent-
suffix="dc=example,dc=com" --create-suffix
```

--create-suffix 引数を指定すると、コマンドはサブ接尾辞の設定エントリーとサブ接尾辞エントリー **ou=People,dc=example,dc=com** を作成します。

--create-suffix 引数は、RDN 属性タイプ **c**、**cn**、**dc**、**o**、および **ou** を使用した接尾辞の作成をサポートします。**l** などの RDN を使用して接尾辞を作成する場合は、**--create-suffix** オプションを指定せずに **dsconf backend create** コマンドを使用し、LDAP 追加操作を使用して接尾辞エントリーを追加するか、LDIF ファイルからエントリーをインポートします。

検証

- この手順の最初のステップにあるコマンドを使用して、接尾辞とデータベースをリスト表示します。

1.8. WEB コンソールによるサブ接尾辞の作成

この手順では、ブラウザーでディレクトリーツリーのサブ接尾辞を作成する方法を説明します。

前提条件

- Web コンソールでインスタンスにログインしている。
- サブ接尾辞の親接尾辞が作成されている。

手順

1. **Database** で、サブ接尾辞の親である設定ツリーから接尾辞を選択します。
2. **Suffix Tasks** をクリックし、Create Sub-Suffix を選択します。
3. **ou=People** および **Database Name** などの **Sub-Suffix DN** を入力します。
4. **Create the Top Suffix Entry** を選択し、**Create Sub-Suffix** をクリックします。

検証

- 新しいサブ接尾辞は、設定ツリーの接尾辞の間に表示されるはずです。

第2章 ビューを使用した仮想ディレクトリー階層の作成

仮想ディレクトリーツリー (DIT) ビューを作成することで、エントリーを独自のグループや階層で整理し、様々な視点から標準の DIT を操作することができます。これにより、ディレクトリー管理のコストを節約し、エントリーによる移動をより直感的にサービスのユーザーに直感的に行うことができます。

2.1. ビューについて

仮想ディレクトリーツリービュー、または **ビュー** とは、DIT でエントリーをカテゴリー分け、検索するための標準ディレクトリーツリー (DIT) に加えて構造の任意のレイヤーです。

ビューを使用すると、仮想ディレクトリー階層を作成できます。そのため、標準の DIT に配置しても、エントリーの移動が簡単になります。ビューは、フィルターされたロールまたは動的グループのメンバーと同様に、エントリーの属性を使用して仮想階層に追加します。クライアントアプリケーションにとって、ビューは通常のコンテナー階層として表示されます。

この方法では、最初はフラットな DIT にエントリーを配置し、ビューを使用して複雑な階層でエントリーを分類することができ、エントリーを移動させる必要がありません。また、標準的な DIT では実現できない、複数のビューでエントリーを表示することができます。

ビューは、**名前付きのフィルター**と考えることができます。各ビューは **nsView** オブジェクトクラスのエントリーであり、どのエントリーをそのビューで表示するかを示す **nsViewFilter** 属性を持つことができます。フィルターにオブジェクトクラスを指定して、返さ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。

ビューを他のビューのコンテナーとして使用し、仮想階層を作成できます。入れ子になったビューは、その祖先からフィルターを継承し、そのフィルターと祖先のフィルターを **(&(container filter)(view filter))** のように **AND** で結合してビューを制限します。

ビューをベースとして検索が実行されると、フィルターに一致するエントリーがこの仮想検索スペースから返されます。エントリーは、ほぼビュー下でネスト化されるように見えますが、実際には DIT の任意の場所に保存できます。

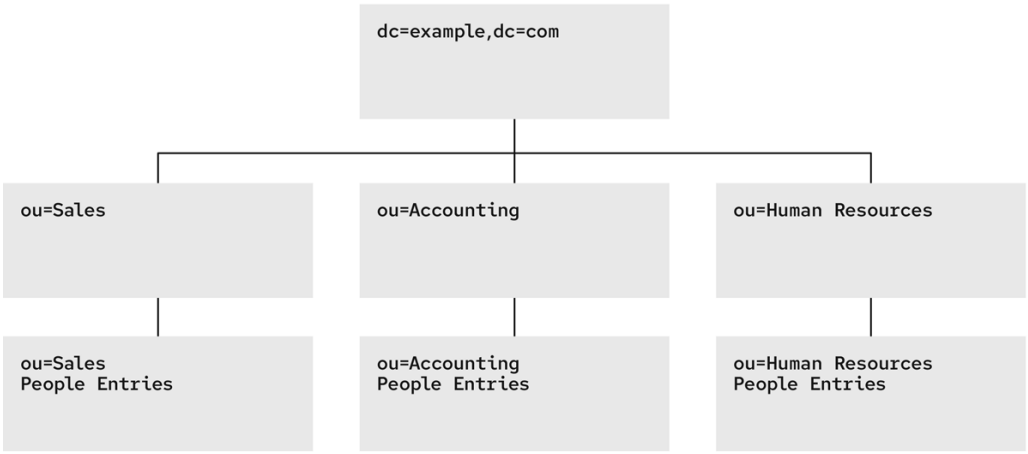
注記

テストインスタンスを作成し、**/usr/share/dirsrv/data/Example-views.ldif** にあるサンプルファイルからインポートされたデータでビューがどのように機能するかを確認することができます。

2.2. ディレクトリーの設計に関する考慮事項

ディレクトリーツリー (DIT) を設計する場合は、階層でエントリーを組織内の階層を反映するために、自然にお気づけられます。DIT のエントリーの位置をエントリーの識別名 (DN) に表示しない標準 DIT は、固定階層で使用するのに適しています。

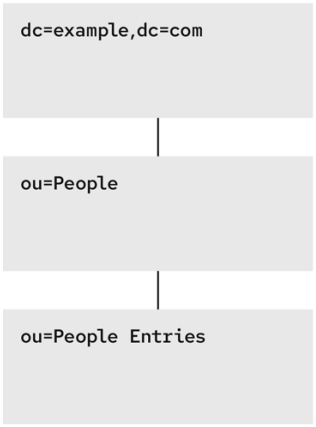
図2.1機能的な単位に基づく標準階層 DIT



ただし、組織内の階層の性質は動的なものです。標準的な DIT のエントリーの移動は、エントリーの位置を変更するたびに、エントリーとその子すべての子の名前を変更する必要があるためです。これにより、サービスの中断や追加費用 (特に最上位サブツリーの変更) が発生します。

リソースタイプ (people、等価性など) など、変更しない特性によってリソースの分類が含まれるフラット階層を設計し、標準のフラット DIT でこの階層を取得することが推奨されます。

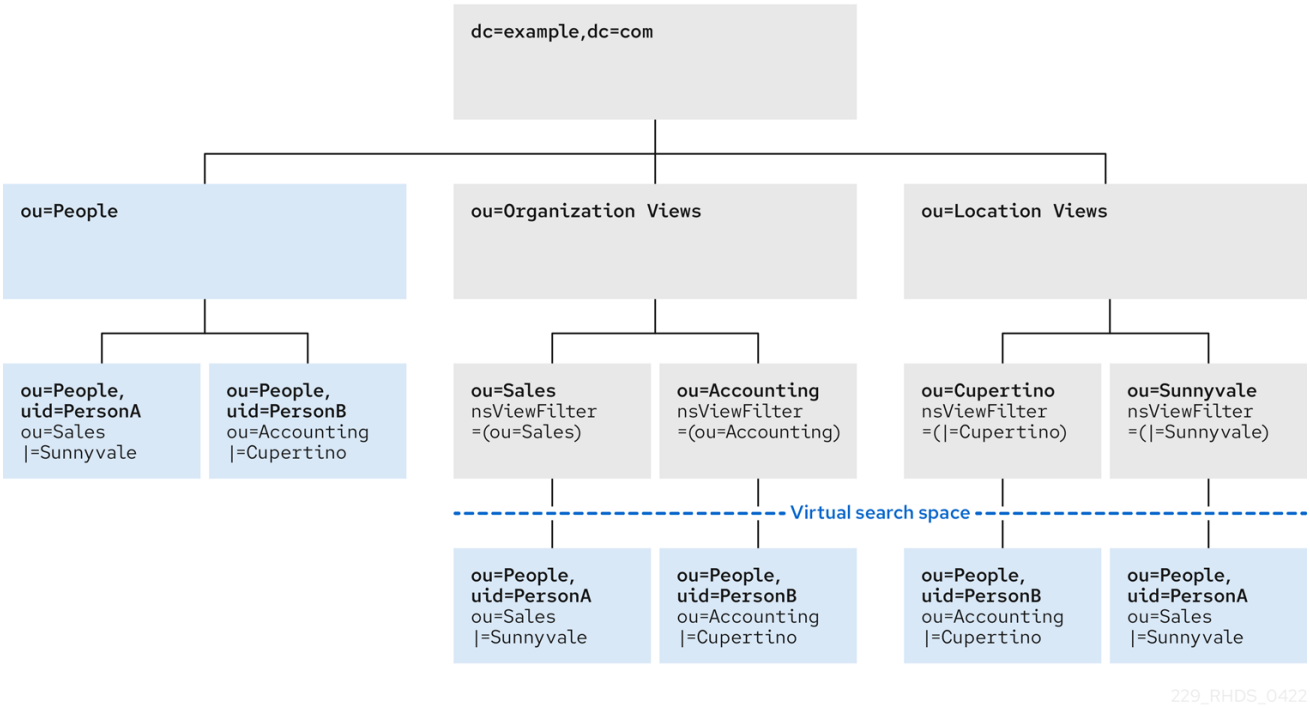
図2.2 リソースタイプに基づく標準フラット DIT



ただし、リソースを移動するのにフラット DIT は便利ではありません。異なるユーザーは、機能性ユニットや地理的なロケーションとの関連付けなど、異なるパースペクティブからリソースをナビゲートする必要があります。これには、フラット DIT の場合は追加のツールや複雑な検索クエリーが必要になる場合があります。

フラット DIT の制限を解消する解決策は、ビューの仮想階層で提供されます。ビューを使用すると、階層内のエントリーの位置からエントリーの名前を分離して、柔軟な階層を作成できます。仮想階層は、代わりに属性に基づいています。

図2.3 仮想階層のビューを持つ DIT



2.3. ビューを使用する利点

仮想ディレクトリーツリービューを使用すると、ユーザーが直感的に操作でき、管理者にとっては深く入れ子になった標準ディレクトリーツリー (DIT) よりも効率的にメンテナンスできる、柔軟なカスタム階層を構築できるというメリットがあります。

フラットで柔軟な命名

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。
DIT に変更がある場合は、エントリーを移動する必要はありません。DIT ビュー階層の変更のみが変更されます。これらの階層には実際のエントリーが含まれていないため、簡単に変更できます。

設計エラーの削減

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。

迅速かつ cheap maintenance

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。

全体的な柔軟性の強化

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 ビューが提供する機能を使用すると、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、エントリーを DIT の特定の位置に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。

直感的なユーザーナビゲーション

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。

検索クエリーを頻繁に行うためのショートカット

仮想 DIT ビュー階層は、頻繁に必要とされる情報の検索を容易にするための、一種の既製のクエリーとして作成することができます。

2.4. 他の機能とのビューの互換性

ビューを使用する場合、検索領域は単一の接尾辞のエントリーに制限されます。ユーザーは、仮想階層からの結果を取得するために、ビューで検索クエリーに基づく必要があります。アクセス制御には若干異なる方法を使用する必要があります。ビューでは、ロールとサービスのクラスと共にエントリーグループを使用できます。

複数のバックエンド

バーチャル DIT 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。
検索は単一のバックエンドに制限されています。つまり、ビューによって返されるエントリーは同じ接尾辞内になければなりません。

探索空間

仮想検索領域は、標準の検索スペースとは異なります。仮想検索領域は、検索がフィルターを持つビューノードにある場合にのみアクセスできます。それ以外の場合は、仮想 DIT 階層に含まれるエントリーが返されない標準のディレクトリーツリー (DIT) に対する規則的な検索です。
たとえば、**dc=example,dc=com** がベースの検索を実行すると、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。Views の処理は、検索ベースが **ou=Cupertino,ou=Location Views,dc=example,dc=com** のような場合に発生します。

このようにして、Directory Server では、検索が両方の場所からエントリーが発生しないようにします。

Access control

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、ビューにはアクセス制御リスト (ACL) の明示的なサポートがないため、親のビューにロールベースの 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。

エントリーのグループ化

Directory Server の**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。ビュー階層の下に **class of service** または **role** 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**や**サービスクラス**を適用することができますが、フラットな名前空間を照会しても、その適用の効果を見ることができます。

2.5. クライアントアプリケーションとのビューの互換性

仮想ディレクトリーツリー (DIT) ビューは、標準の DIT を高レベルに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** 運用属性を使用して、ノードの下に何個のエントリーが存在するかを判断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。

2.6. ビューの作成

この手順では、コマンドラインで仮想ディレクトリーツリービューを作成する方法を説明します。

手順

- **ldapadd** ユーティリティーを使用してビューエントリーを追加します。
  nsView オブジェクトクラスを指定し、**nsViewFilter** 属性でビューフィルターを定義します。

```
# ldapadd -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: ou=PeopleInRoom0466,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
objectClass: nsView
ou: PeopleInRoom0466
description: People in the room 0466
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=0466))
```

検証

- ビューで検索ベースとして実行します。

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
ou=PeopleInRoom0466,dc=example,dc=com
```

2.7. コマンドラインを使用してビューのパフォーマンスを改善するためのインデックスの作成

ビューは指定のフィルターに基づいて検索結果から派生します。フィルターの一部は **nsViewFilter** で明示的に指定される属性です。フィルターの残りの部分はエントリー階層に基づいており、ビューに含まれる実際のエントリーの **entryid** および **parentid** 操作属性を検索します。

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

検索された属性 (**entryid**、**parentid**、または **nsViewFilter** の属性) のいずれかにインデックスが付けられていない場合、検索は部分的にインデックスが解除され、Directory Server はディレクトリーツリー全体で一致するエントリーを検索します。

ビューのパフォーマンスを改善するには、以下のようにインデックスを作成します。

- **entryid** の**等式インデックス** (**eq**) を作成します。**parentid** 属性は、デフォルトでシステムインデックスでインデックス化されます。
- **nsViewFilter** テストにフィルターがある場合 (**attribute=***) は、テスト中の属性について、**存在インデックス**(**pres**) を作成します。このインデックスタイプは、少数のディレクトリーエントリーに表示される属性でのみ使用する必要があります。
- **nsViewFilter** テスト等価 (**attribute=value**) のフィルターの場合、テストする属性に対して **等価インデックス** (**eq**) を作成します。
- **nsViewFilter** のフィルターがサブ文字列をテストする場合 (**attribute=value***) は、テストする属性の substring index (**sub**) を作成します。
- **nsViewFilter** でフィルターが近似値をテストする場合 (**attribute~=value**)、テストされている属性のapproximate index (**approximate**) を作成します。

たとえば、以下の view フィルターを使用する場合は、以下を実行します。

```
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=*66))
```

デフォルトで行われる**等価インデックス**で **objectClass** にインデックスを付け、**部分文字列インデックス**で **roomNumber** にインデックスを付ける必要があります。

前提条件

- ビューフィルターで使用する属性に注意してください。

手順

1. オプション: バックエンドをリスト表示し、データベースをインデックス化します。

   ```
   # dsconf -D "cn=Directory Manager" instance_name backend suffix list
   dc=example,dc=com (userroot)
   ```

   (括弧で) 選択したデータベース名を書き留めておきます。

2. 選択したバックエンドのデータベースの **dsconfig** ユーティリティーを使用してインデックス設定を作成します。
   特に国際化されたインスタンスの場合、属性名、インデックスタイプと、任意で照合順序 (OID) を設定するためのマッチングルールを指定します。

```
# dsconf -D "cn=Directory Manager" instance_name backend index add --attr
roomNumber --index-type sub userroot
```

view フィルターで使用される属性ごとに、このステップを繰り返します。

3. 新規インデックスを適用するためにデータベースを再インデックスします。

```
# dsconf -D "cn=Directory Manager" instance_name backend index reindex userroot
```

検証

1. ビューで使用するフィルターが同じ標準ディレクトリーツリーに基づく検索を実行します。

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
dc=example,dc=com (&(objectClass=inetOrgPerson)(roomNumber=*66))
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
dc=example,dc=com "(&(objectClass=inetOrgPerson)(roomNumber=*66))"
```

2. **/var/log/dirsrv/slapd-instance_name/access**.の**アクセス**ログを見る。
検索の **RESULT** には **note=U** または **Partially Unindexed Filter** が詳細に含まれていないはずです。

関連情報

- インデックスの管理

2.8. WEB コンソールを使用してビューのパフォーマンスを改善するためのインデックスの作成

ビューは指定のフィルターに基づいて検索結果から派生します。フィルターの一部は **nsViewFilter** で明示的に指定される属性です。フィルターの残りの部分はエントリー階層に基づいており、ビューに含まれる実際のエントリーの **entryid** および **parentid** 操作属性を検索します。

```
(|(parentid=search_base_id)(entryid=search_base_id)
```

検索された属性 (**entryid**、**parentid**、または **nsViewFilter** の属性) のいずれかにインデックスが付けられていない場合、検索は部分的にインデックスが解除され、Directory Server はディレクトリーツリー全体で一致するエントリーを検索します。

ビューのパフォーマンスを改善するには、以下のようにインデックスを作成します。

- **entryid** の**等式インデックス** (**eq**) を作成します。**parentid** 属性は、デフォルトでシステムインデックスでインデックス化されます。
- **nsViewFilter** テストにフィルターがある場合 (**attribute=***) は、テスト中の属性について、**存在インデックス**(**pres**) を作成します。このインデックスタイプは、少数のディレクトリーエントリーに表示される属性でのみ使用する必要があります。
- **nsViewFilter** テスト等価 (**attribute=value**) のフィルターの場合、テストする属性に対して **等価インデックス** (**eq**) を作成します。
- **nsViewFilter** のフィルターがサブ文字列をテストする場合 (**attribute=value***) は、テストする属性の substring index (**sub**) を作成します。

- **nsViewFilter** でフィルターが近似値をテストする場合 (**attribute~=value**)、テストされている属性のapproximate index (**approximate**) を作成します。

たとえば、以下の view フィルターを使用する場合は、以下を実行します。

```
nsViewFilter: (&(objectClass=inetOrgPerson)(roomNumber=*66))
```

デフォルトで行われる**等価インデックス**で **objectClass** にインデックスを付け、**部分文字列インデックス**で **roomNumber** にインデックスを付ける必要があります。

前提条件

- Web コンソールでインスタンスにログインしている。
- ビューフィルターで使用する属性に注意してください。

手順

1. **Database** で、インデックスを作成する設定ツリーから接尾辞を選択します。
2. **Indexes** および **Database Indexes** に移動します。
3. **Add Index** ボタンをクリックします。
4. 属性の名前を入力し、属性を選択します。
5. この属性に対して作成する必要のある **Index Types** を選択します。
6. 必要に応じて、特に国際化されたインスタンスの場合に、照合順序 (OID) を指定するための **Matching Rules** を追加します。
7. **Index attribute after creation** を選択し、後でインデックスを再ビルドします。
8. **Create Index** をクリックします。
9. インデックス化される各属性に対して手順を繰り返します。

検証

- 追加された属性の名前を入力して、**Filter Indexes** します。
- 新たにインデックス化された属性が結果に表示されるはずです。

関連情報

- インデックスの管理

第3章 読み取り専用モードへのデータベースの切り替え

デフォルトでは、Directory Server のデータベースは、読み取り/書き込みモードで稼働します。このモードでは、ユーザーはデータの取得と保存の両方が可能です。

たとえば、バックアップの前やコンシューマーの手動初期化の前にデータベースの 5 倍のイメージが必要な場合に、ユーザーがエントリーの作成、変更、または削除を禁止する、データベースを読み取り専用モードに切り替える場合があります。

3.1. 前提条件

- データベースは読み書きモードです。
- 読み取り専用モードを有効にするとレプリケーションが無効になるため、データベースはレプリケーションに使用されません。

3.2. コマンドラインを使用した読み取り専用モードでのデータベースの設定

この手順では、Directory Server データベースをコマンドラインで読み取り専用モードに切り替える方法を説明します。

手順

1. 接尾辞とそれに対応するデータベースをリスト表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
o=test (test_database)
```

   切り替えるデータベースの名前または接尾辞を書き留めておきます。

2. **--enable-readonly** パラメーターで読み取り専用モードを有効にし、名前または接尾辞でデータベースを指定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix set --enable-readonly "test_database"
```

検証

- ディレクトリーへの書き込み操作を試行します。以下に例を示します。

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: description
description: foo
```

  サーバーは、実行を拒否するはずです。

```
modifying entry "dc=example,dc=com"
ldap_modify: Server is unwilling to perform (53)
 additional info: Server is read-only
```

関連情報

- インスタンス全体を読み取り専用モードに切り替え

3.3. WEB コンソールを使用したデータベースの読み取り専用モードへの切り替え

この手順では、Directory Server データベースをブラウザーで読み取り専用モードに切り替える方法を説明します。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database** で、設定ツリーの接尾辞を選択します。
2. **Database Read-Only Mode** オプションを確認します。
3. **Save Configuration** をクリックします。

検証

- ディレクトリーへの書き込み操作を試行します。以下に例を示します。

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: description
description: foo
```

サーバーは、実行を拒否するはずです。

```
modifying entry "dc=example,dc=com"
ldap_modify: Server is unwilling to perform (53)
 additional info: Server is read-only
```

3.4. 関連情報

- Directory Server のバックアップ

第4章 インスタンスの読み取り専用モードへの切り替え

デフォルトでは、インスタンスは、ユーザーがデータの取得と保存の両方を行うことができる読み書きモードで動作します。レプリケーションを禁止したり、インデックス作成時にデータの修正を無効にしたいが、ディレクトリーは利用可能な状態にしておきたいなどの緊急時には、一時的にインスタンスを読み取り専用モードに切り替えることができます。

Directory Server が複数のデータベースを管理していて、すべてのデータベースを読み取り専用に切り替える必要がある場合、コマンドラインまたはウェブコンソールで、1回の操作でこれを行うことができます。

警告

read-only モードでは、インスタンスを再起動できませんが、設定を変更できます。

読み取り専用モードでインスタンスを停止すると、手動で読み取り専用モードを無効にするまで起動することはできません。

読み取り専用モードを手動で無効にするには、**/etc/dirsrv/slapd-instance_name/dse.ldif** ファイルを開き、**cn=config** セクションに移動して、**nsslapd-readonly** パラメーターを **off** に設定します。

4.1. 前提条件

- インスタンスは読み取り/書き込みモードです。
- 読み取り専用モードを有効にするとレプリケーションが無効になるため、インスタンスはレプリケーションに使用されません。

4.2. コマンドラインを使用したインスタンスの読み取り専用モードへの切り替え

この手順では、Directory Server インスタンスをコマンドラインで読み取り専用モードに切り替える方法を説明します。

手順

- **nsslapd-readonly** パラメーターを **on** に設定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-readonly=on
```

検証

- ディレクトリーへの書き込み操作を試行します。以下に例を示します。

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: dc=example,dc=com
```

```
changetype: modify
add: description
description: foo
```

サーバーは、実行を拒否するはずです。

```
modifying entry "dc=example,dc=com"
ldap_modify: Server is unwilling to perform (53)
 additional info: Server is read-only
```

関連情報

- 読み取り専用モードへのデータベースの切り替え

4.3. WEB コンソールを使用したインスタンスの読み取り専用モードへの切り替え

この手順では、Directory Server インスタンスをブラウザーで読み取り専用モードに切り替える方法を説明します。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Server** で、**Advanced Settings** タブを選択します。
2. **Server Read-Only** オプションを確認します。
3. **Save** をクリックします。

検証

- ディレクトリーへの書き込み操作を試行します。以下に例を示します。

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: description
description: foo
```

サーバーは、実行を拒否するはずです。

```
modifying entry "dc=example,dc=com"
ldap_modify: Server is unwilling to perform (53)
 additional info: Server is read-only
```

関連情報

- 読み取り専用モードへのデータベースの切り替え

第5章 不要になった接尾辞のデータベースの削除

Directory Server ホストでディスク領域を回収する必要がある場合は、使用していない接尾辞のデータベースを削除できます。

5.1. コマンドラインを使用したデータベースの削除

この手順では、コマンドラインで Directory Server データベースを削除する方法を説明します。

手順

1. 接尾辞とそれに対応するデータベースをリスト表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
o=test (test_database)
```

   削除するデータベースの名前を書き留めます。

2. **dsconf backend delete** コマンドを入力し、データベースの名前を指定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend delete
"test_database"
```

3. プロンプトで "Yes I am sure" を入力して、削除を確認します。

```
Deleting Backend cn=test_database,cn=ldbm database,cn=plugins,cn=config :
Type 'Yes I am sure' to continue: Yes I am sure
```

検証

- 接尾辞/データベースをリスト表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userroot)
```

5.2. WEB コンソールを使用したデータベースの削除

この手順では、ブラウザーで Directory Server データベースを削除する方法を説明します。

前提条件

- Web コンソールでインスタンスにログインしている。

手順

1. **Database** で、削除する接尾辞を選択します。
2. **Suffix Tasks → Delete Suffix** に移動します。
3. **Yes, I am sure**.を選択します。

4. **Delete Suffix**をクリックします。

検証

- **Database** で、設定ツリーの接尾辞のリストを確認します。

第6章 バックエンドデータベースの整合性の確認

Directory Server データベースの整合性チェックは、破損したメタデータページや重複キーのソートなどの問題を検出できます。問題が検出される場合は、問題によっては、データベースのインデックスを再作成したり、バックアップを復元したりすることができます。

6.1. データベース整合性チェックの実行

dsctl dbverify コマンドを使用すると、管理者はバックエンドデータベースの整合性を検証できます。

手順

1. 必要に応じて、インスタンスのバックエンドデータベースをリスト表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend suffix list
dc=example,dc=com (userRoot)
```

2. インスタンスを停止します。

```
# dsctl instance_name stop
```

3. データベースを確認します。たとえば、**userRoot** データベースを確認するには、以下のコマンドを実行します。

```
# dsctl instance_name dbverify userRoot
[04/Feb/2022:13:11:02.453624171 +0100] - INFO -
ldbm_instance_config_cachememsize_set - force a minimal value 512000
[04/Feb/2022:13:11:02.465339507 +0100] - WARN -
ldbm_instance_add_instance_entry_callback - ldbm instance userroot already exists
[04/Feb/2022:13:11:02.468060144 +0100] - ERR - ldbm_config_read_instance_entries -
Failed to add instance entry cn=userroot,cn=ldbm database,cn=plugins,cn=config
[04/Feb/2022:13:11:02.471079045 +0100] - ERR - bdb_config_load_dse_info - failed to read
instance entries
[04/Feb/2022:13:11:02.476173304 +0100] - ERR - libdb - BDB0522 Page 0: metadata page
corrupted
[04/Feb/2022:13:11:02.481684604 +0100] - ERR - libdb - BDB0523 Page 0: could not check
metadata page
[04/Feb/2022:13:11:02.484113053 +0100] - ERR - libdb - /var/lib/dirsrv/slapd-
instance_name/db/userroot/entryrdn.db: BDB0090 DB_VERIFY_BAD: Database verification
failed
[04/Feb/2022:13:11:02.486449603 +0100] - ERR - dbverify_ext - verify failed(-30970):
/var/lib/dirsrv/slapd-instance_name/db/userroot/entryrdn.db
dbverify failed
```

4. 検証プロセスで問題が報告された場合は、手動で修正するか、バックアップを復元します。

5. インスタンスを起動します。

```
# dsctl instance_name start
```

関連情報

- Directory Server の復元

第7章 属性暗号化の管理

Directory Server は、ディレクトリー内の機密データへのアクセスを保護するための多数のメカニズムを提供します。ただし、デフォルトでは、サーバーは暗号化されていないデータをデータベースに格納します。機密性の高い情報の場合、攻撃者がデータベースにアクセスできるという潜在的なリスクは、重大なリスクになる可能性があります。

属性の暗号化機能により、管理者は特定の属性を機密データ (政府識別番号など) と共に暗号化してデータベースに保存できます。接尾辞を有効にすると、これら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、データベース内のこの属性に格納されているすべてのエントリーに対して暗号化されます。接尾辞の属性暗号化を有効にできることに注意してください。サーバー全体でこの機能を有効にするには、サーバー上の各接尾辞の属性暗号化を有効にする必要があります。属性の暗号化は、**eq** および **pres** インデックス作成と完全に互換性があります。

重要

エントリーの識別名 (DN) 内で使用する属性は、効率的に暗号化できません。たとえば、**uid** 属性を暗号化するように設定した場合、値はエントリーでは暗号化されますが、DN では暗号化されません。

```
dn: uid=demo_user,ou=People,dc=example,dc=com
...
uid::Sf04P9nJWGU1qiW9JJCGRg==
```

7.1. DIRECTORY SERVER が属性の暗号化に使用するキー

属性の暗号化を使用するには、TLS を使用して暗号化された接続を設定する必要があります。Directory Server は、サーバーの TLS 暗号化キーと、属性の暗号化に同じ PIN 入力方法を使用します。

サーバーは、ランダムに生成された対称暗号鍵を使用して、属性データを暗号化および復号化します。サーバーは、サーバーの TLS 証明書の公開鍵を使用してこれらの鍵をラップします。結果として、属性暗号化の有効な強度は、サーバーの TLS キーの強度より高くすることはできません。

警告

サーバーの秘密鍵にアクセスできないと、ラップ済みのコピーから対称キーを復旧することができません。そのため、サーバーの証明書データベースを定期的にバックアップしてください。キーを紛失すると、データベースに保存されているデータを復号化および暗号化できなくなります。

7.2. コマンドラインを使用して属性暗号化を有効にする

この手順では、コマンドラインを使用して **userRoot** データベースの **telephoneNumber** 属性の属性暗号化を有効にする方法を示します。この手順を実行すると、サーバーはこの属性の既存の値と新しい値を AES 暗号化して格納します。

前提条件

- Directory Server で TLS 暗号化を有効にした。

手順

1. **userRoot** データベースをエクスポートします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend export -E
userRoot
```

サーバーは、エクスポートを **/var/lib/dirsrv/slapd-instance_name/ldif/** ディレクトリーの LDIF ファイルに保存します。**-E** オプションは、エクスポート中にすでに暗号化されている属性を復号化します。

2. **telephoneNumber** 属性の AES 暗号化を有効にします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend attr-encrypt --
add-attr telephoneNumber dc=example,dc=com
```

3. インスタンスを停止します。

```
# dsctl instance_name stop
```

4. LDIF ファイルをインポートします。

```
# dsctl instance_name ldif2db --encrypted userRoot /var/lib/dirsrv/slapd-
instance_name/ldif/None-userroot-2022_01_24_10_28_27.ldif
```

--encrypted パラメーターを使用すると、スクリプトで属性を暗号化して、インポート中に暗号化を設定できます。

5. インスタンスを起動します。

```
# dsctl instance_name start
```

関連情報

- Directory Server への TLS 暗号化接続の有効化

7.3. WEB コンソールを使用して属性暗号化を有効にする

この手順では、Web コンソールを使用して **userRoot** データベースの **telephoneNumber** 属性の属性暗号化を有効にする方法を示します。この手順を実行すると、サーバーはこの属性の既存の値と新しい値を AES 暗号化して格納します。

Web コンソールのエクスポートおよびインポート機能は、暗号化された属性をサポートしていないことに注意してください。したがって、これらの手順はコマンドラインで実行する必要があります。

前提条件

- Directory Server で TLS 暗号化を有効にした。
- Web コンソールでインスタンスにログインしている。

手順

1. **userRoot** データベースをエクスポートします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend export -E
userRoot
```

サーバーは、エクスポートを **/var/lib/dirsrv/slapd-instance_name/ldif/** ディレクトリーの LDIF ファイルに保存します。**-E** オプションは、エクスポート中にすでに暗号化されている属性を復号化します。

2. Web コンソールで、**Database → Suffixes → suffix_entry → Encrypted Attributes** に移動します。

3. 暗号化する属性を入力し、**Add Attribute** をクリックします。

4. **Actions** メニューで、**Stop Instance** を選択します。

5. コマンドラインで、LDIF ファイルをインポートします。

```
# dsctl instance_name ldif2db --encrypted userRoot /var/lib/dirsrv/slapd-
instance_name/ldif/None-userroot-2022_01_24_10_28_27.ldif
```

--encrypted パラメーターを使用すると、スクリプトで属性を暗号化して、インポート中に暗号化を設定できます。

6. Web コンソールで **Actions** メニューを開き、**Start Instance** を選択します。

関連情報

- Directory Server への TLS 暗号化接続の有効化

7.4. 属性暗号化の有効化後の一般的な考慮事項

データベースにすでに存在するデータの暗号化を有効にした後、次の点を考慮してください。

- 暗号化されていないデータは、サーバーのデータベースページプールのバッキングファイルで保持できます。このデータを削除するには、以下を実行します。

  a. インスタンスを停止します。

  ```
  # dsctl instance_name stop
  ```

  b. **/var/lib/dirsrv/slapd-instance_name/db/guardian** ファイルを削除します。

  ```
  # **rm /var/lib/dirsrv/slapd-instance_name/db/guardian``
  ```

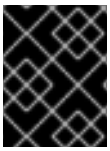
  c. インスタンスを起動します。

  ```
  # dsctl instance_name start
  ```

- 暗号化を有効にし、データが正常にインポートされた後に、暗号化されていないデータで LDIF ファイルを削除します。

- Directory Server はレプリケーションログファイルを暗号化しません。このデータを保護するには、レプリケーションログを暗号化されたディスクに保存します。

- サーバーのメモリー (RAM) のデータは暗号化されず、swap パーティションに一時的に保存できます。このデータを保護するには、暗号化されたスワップ領域を設定します。



重要

暗号化されていないデータを含むファイルを削除すると、このデータは特定の状況で復元できます。

7.5. 属性暗号化に使用される TLS 証明書の更新

属性の暗号化は、サーバーの TLS 証明書に基づいています。TLS 証明書を更新または置き換えた後に属性の暗号化が失敗しないようにするには、次の手順に従います。

前提条件

- 属性の暗号化を設定しました。
- TLS 証明書の有効期限がまもなく切れる。

手順

1. **userRoot** データベースをエクスポートします。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend export -E
userRoot
```

サーバーは、エクスポートを **/var/lib/dirsrv/slapd-instance_name/ldif/** ディレクトリーの LDIF ファイルに保存します。**-E** オプションは、エクスポート中にすでに暗号化されている属性を復号化します。

2. プライベートキーおよび証明書署名要求 (CSR) を作成します。外部ユーティリティーを使用して作成する場合は、この手順を省略します。

   - ホストが 1 つの名前のみで到達可能である場合は、以下を実行します。

```
# dsctl instance_name tls generate-server-cert-csr -s
"CN=server.example.com,O=example_organization"
```

   - 複数の名前でホストにアクセスできる場合は、以下を行います。

```
# dsctl instance_name tls generate-server-cert-csr -s
"CN=server.example.com,O=example_organization" server.example.com
server.example.net
```

   最後のパラメーターとしてホスト名を指定した場合、このコマンドは **DNS:server.example.com, DNS:server.example.net** エントリーで SAN (Subject Alternative Name) 拡張を CSR に追加します。

-s subject パラメーターで指定した文字列は、RFC 1485 に従って有効なサブジェクト名である必要があります。サブジェクトの **CN** フィールドが必要で、サーバーの完全修飾ドメイン名 (FQDN) の 1 つに設定する必要があります。このコマンドは、**/etc/dirsrv/slapd-instance_name/Server-Cert.csr** ファイルに CSR を保存します。

3. 認証局 (CA) に CSR を送信し、発行した証明書を取得します。詳細は、CA のドキュメントを参照してください。

4. CA が発行するサーバー証明書を NSS データベースにインポートします。

   - **dsctl tls generate-server-cert-csr** コマンドを使用して秘密鍵を作成した場合は、以下を入力します。

     ```
     # dsconf -D "cn=Directory Manager" ldap://server.example.com security certificate add --file /root/instance_name.crt --name "server-cert" --primary-cert
     ```

     --name _certificate_nickname パラメーターで設定した証明書の名前を書き留めておきます。これは後のステップで必要になります。

   - 外部ユーティリティーを使用して秘密鍵を作成した場合は、サーバー証明書および秘密鍵をインポートします。

     ```
     # dsctl instance_name tls import-server-key-cert /root/server.crt /root/server.key
     ```

     このコマンドでは、最初にサーバー証明書へのパスを指定してから、秘密鍵へのパスを指定する必要があります。このメソッドは、証明書のニックネームを **Server-Cert** に設定します。

5. CA 証明書を NSS データベースにインポートします。

   ```
   # dsconf -D "cn=Directory Manager" ldap://server.example.com security ca-certificate add --file /root/ca.crt --name "Example CA"
   ```

6. CA 証明書の信頼フラグを設定します。

   ```
   # dsconf -D "cn=Directory Manager" ldap://server.example.com security ca-certificate set-trust-flags "Example CA" --flags "CT,,"
   ```

   これにより、Directory Server が、TLS による暗号化および証明書ベースの認証に対して CA を信頼するように設定します。

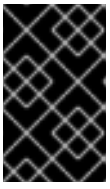
7. インスタンスを停止します。

   ```
   # dsctl instance_name stop
   ```

8. **/etc/dirsrv/slapd-instance_name/dse.ldif** ファイルを編集し、属性を含む以下のエントリーを削除します。

   - **cn=AES,cn=encrypted attribute keys,cn=database_name,cn=ldbm database,cn=plugins,cn=config**

   - **cn=3DES,cn=encrypted attribute keys,cn=database_name,cn=ldbm database,cn=plugins,cn=config**

   重要

   全データベースのエントリーを削除します。**nsSymmetricKey** 属性を含むエントリーが `/etc/dirsrv/slapd-**instance_name**/dse.ldi ファイルに残されると、Directory Server は起動に失敗します。

9. LDIF ファイルをインポートします。

```
# dsctl instance_name ldif2db --encrypted userRoot /var/lib/dirsrv/slapd-
instance_name/ldif/None-userroot-2022_01_24_10_28_27.ldif
```

--encrypted パラメーターを使用すると、スクリプトで属性を暗号化して、インポート中に暗号化を設定できます。

10. インスタンスを起動します。

```
# dsctl instance_name start
```

第8章 データベースリンクの作成および維持

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。**チェーン** とは、サーバーが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接続し、組み合わせた結果を返すことを意味します。このチェーンプロセスは、データベースリンクを使用して実装されます。

8.1. 新規データベースリンクの作成

基本的なデータベースリンクの設定には、以下の情報が必要です。

- **接尾辞情報**
  接尾辞情報は、データを含むリモートサーバー上の接尾辞に対応する必要があります。データベースリンクが管理するディレクトリーツリーに接尾辞を作成できます。

- **バインド認証情報**
  データベースリンクがリモートサーバーにバインドされると、ユーザーのなりすましが行われ、リモートサーバーとバインドするために使用する各データベースリンクの **DN** および認証情報を指定します。

- **LDAP URL**
  LDAP URL 情報は、データベースリンクが接続するリモートサーバーの **LDAP URL** を提供します。URL は、プロトコル (**LDAP** または **LADPS**)、サーバーのホスト名または IP アドレス (**IPv4** または **IPv6**)、およびポートで構成されます。

- **フェイルオーバーサーバーの一覧**
  フェイルオーバーサーバーのリストは、障害が発生した場合にデータベースリンクが接続する代替サーバーのリストです (任意)。

注記

単純なパスワード認証に安全なバインドが必要な場合は、チェーン操作を実行するときに安全な接続 (**TLS** および **STARTTLS** 接続、または **SASL** 認証) を使用することを推奨します。

8.2. コマンドラインを使用した新規データベースリンクの作成

dsconf chaining link-create コマンドを使用して、新しいデータベースリンクを作成できます。

前提条件

- Web コンソールで Directory Server ユーザーインターフェイスを開き、インスタンスを選択している。

手順

1. 新規データベースリンクを作成するには、以下を行います。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-create --suffix="ou=Customers,dc=example,dc=com" --server-url="ldap://remote_server.example.com:389" --bind-mech=SIMPLE --bind-dn="cn=proxy_user,cn=config" --bind-pw="password" "example_chain_name"
```

このコマンドを使用すると、**ou=Customers,dc=example,dc=com** の **example_chain_name** という名前のデータベースリンクが作成されます。これは、**ldap://remote_server.example.com:389** サーバーを参照し、指定のバインド **DN** と **password** を使用して認証します。bind-mech を **SIMPLE** (証明書ベースの認証の場合は **EXTERNAL**) に設定するか、kerberos 認証の場合は **GSSAPI** に設定する必要があります。

1. データベースリンクの作成時に設定できる追加設定を表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-create --help
```

注記

proxy_user にデータへのアクセス権を付与するには、リモートサーバーの **dc=example,dc=com** 接尾辞にプロキシー ACI エントリーを作成する必要があります。

検証

- 新しいデータベースリンクを表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining link-create --suffix="ou=Customers,dc=example,dc=com" --server-url="ldap://remote_server.example.com:389" --bind-mech=SIMPLE --bind-dn="cn=proxy_user,cn=config" --bind-pw="password" "example_chain_name"
```

8.3. WEB コンソールを使用した新規データベースリンクの作成

Web コンソールを使用して、新しいデータベースリンクを作成できます。

前提条件

- Web コンソールで Directory Server ユーザーインターフェイスを開き、インスタンスを選択している。

手順

1. **Database** メニューを開きます。
2. 新しい接尾辞を作成します。
   a. **Create Suffix** ボタンをクリックします。
   b. **DN** 接尾辞およびバックエンド名を入力します。
   c. **Create The Top Suffix Entry** を選択し、**Create Suffix** をクリックします。
3. 接尾辞を選択し、右側の **Suffix Tasks** ボタンをクリックし、**Create Database Link** を選択します。
4. フィールドに、リモートサーバーへのコネクションの詳細を入力します。
5. **Create Database Link** をクリックします。

検証

- **Database** メニューを開き、このメニューに新しいデータベースのリンクが表示されていることを確認します。

8.4. 新規データベースリンクのデフォルト設定の管理

dsconf chaining コマンドを使用して、データベースリンクのデフォルト設定を管理できます。

手順

1. データベースリンクの現在のデフォルト値を表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-get-def
```

2. **dsconf chaining config-set-def** コマンドを使用して、新しいデータベースリンク設定を変更します。たとえば、**response-delay** パラメーターを **30** に設定するには、以下を実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set-def --response-delay 30
```

このサンプルコマンドは、すべてのチェーン接続にデフォルトの応答タイムアウトを設定します。**dsconf instance chaining link-set** コマンドを使用すると、特定のチェーンリンクの応答タイムアウトを上書きできます。

3. 設定可能なすべてのパラメーターの一覧を表示するには、以下を実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set-def --help
```

第9章 データベースリンクの作成に必要な設定

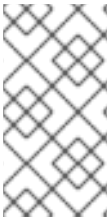
データベースリンクを作成するときは、**接尾辞**、**バインド認証情報**、**バインドメカニズム**、および **LDAP URL** 設定を指定する必要があります。

9.1. バインド認証情報

特定のバインド認証情報を使用して、クライアントアプリケーションからリモートサーバーにリクエストをチェーンできます。リモートサーバー上のチェーン接尾辞には、ユーザーにプロキシー認証を許可する ACI が必要です。バインド認証情報がないと、データベースリンクは **anonymous** としてリモートサーバーにバインドされます。

警告

チェーンを有効にする場合は、アクセス制御を慎重に検討して、ディレクトリーの制限領域にアクセスできないようにしてください。たとえば、データベースリンクを使用して接続するユーザーは、ブランチの下にあるすべてのエントリーを確認できます。すべてのサブツリーをユーザーに表示する必要がない場合はサブツリーへのアクセスを制限するには、セキュリティーの問題を回避するために追加の ACI を作成します。



注記

クライアントアプリケーションがデータベースリンクを使用してエントリーを作成または変更する場合に、**creatorsName** 属性と **modifiersName** 属性はエントリーの実際の作成者または変更者を反映しません。これらの属性には、リモートデータサーバーでプロキシー認証権限が付与されている管理ユーザーの名前が含まれています。

バインド認証情報を指定するには、リモートサーバーで以下の手順が必要です。

- データベースリンク用に、**cn=proxy_user,cn=config** などの管理ユーザーを作成します。
- データベースリンクを使用してチェーンされたターゲットサブツリーに対して、前の手順で作成した管理ユーザーにプロキシーアクセス権を提供します。

たとえば、以下の ACI は、ACI が設定されたサブツリー内のみにリモートサーバーに含まれるデータにアクセスするために、**cn=proxy_admin,cn=config** ユーザーへの読み取り専用アクセスを付与します。

```
aci: (targetattr = "*")(version 3.0; acl "Proxied authorization
    for database links"; allow (proxy) userdn = "ldap:///cn=proxy_admin
    ,cn=config";)
```

注記

ユーザーがデータベースリンクにバインドすると、ユーザーのアイデンティティーがリモートサーバーに送信されます。アクセス制御は、常にリモートサーバーで評価されます。ユーザーがリモートサーバーにデータを変更したり書き込んだりできるようにするには、リモートサーバー上で正しいアクセス制御を設定します。

関連情報

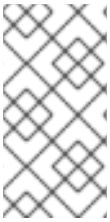
- アクセス制御の管理
- ディレクトリー属性の管理

9.2. LDAP URL

データベースリンクを含むサーバー上の **LDAP URL** を使用して、データベースリンクが接続するリモートサーバーを特定できます。リモートサーバーの URL は接尾辞を指定せず、**ldap://host_name:port** 形式です。

TLS ではなく、**LDAP** を使用してリモートサーバーにデータベースをリンクする場合に、リモートサーバーは、**LDAP** の代わりに **LDAPS** プロトコルを URL で使用し、サーバーのセキュアなポートを参照します。

```
ldaps://africa.example.com:636/
```



注記

ローカルの Directory Server およびリモートの Directory Server で **TLS** を有効にする必要があります。データベースリンクとリモートサーバーが **TLS** を使用して通信する場合に、操作リクエストを作成するクライアントアプリケーションは通常のポートを使用してバインドでき、通信に必ずしも **TLS** を使用する必要はありません。

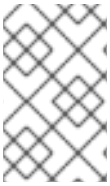
関連情報

- Directory Server への TLS 暗号化接続の有効化

9.3. バインドメカニズム

次のいずれかの方法で、ローカルサーバーをリモートサーバーに接続できます。

- 標準の **LDAP** ポートを使用する。
- 専用の **LDAPS** ポートを使用する。
- **STARTTLS** 接続を使用します。これは、標準ポートよりも安全な接続です。



注記

単純なパスワード認証に安全なバインドが必要な場合は、チェーン操作を実行するときに安全な接続 (**TLS** および **STARTTLS** 接続、または **SASL** 認証) を使用することを推奨します。

ローカルサーバーは、次の方法を使用してリモートサーバーに対する認証を行うことができます。

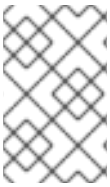
- `EMPTY`
  EMPTY メソッドを使用する場合に、ローカルサーバーは単純な認証を実行し、バインドメカニズムが設定されていない場合はバインド DN とパスワードを必要とします。
- **EXTERNAL**
  EXTERNAL メソッドを使用する場合、ローカルサーバーは TLS 証明書を適用して、ローカルサーバーをリモートサーバーに対して認証します。ローカルサーバー URL を安全な URL

(**ldaps**) に設定するか、**nsUseStartTLS** 属性を **on** に設定する必要があることに注意してください。さらに、ローカルサーバーの証明書をバインド ID にマップするようにリモートサーバーを設定する必要があります。

- **DIGEST-MD5**
  DIGEST-MD5 メソッドを使用する場合は、ローカルサーバーは **DIGEST-MD5** 暗号化を使用した **SASL** 認証を適用します。単純な認証と同様に、このタイプの認証では、バインド情報を提供するために **nsMultiplexorBindDN** 属性と **nsMultiplexorCredentials** 属性が必要です。

- **GSSAPI**
  GSSAPI メソッドを使用する場合、ローカルサーバーは **SASL** 認証を介して **Kerberos** ベースの認証を適用します。

  Kerberos キータブを使用してローカルサーバーを設定できます。リモートサーバーは、ローカルサーバーのバインド ID に対して定義済みの **SASL** マッピングを設定する必要があります。



注記

SASL 接続は、標準の接続または **TLS** 接続で確立できます。**SASL** の使用時に **SASL** およびパスワードポリシーコンポーネントをチェーンするようにローカルサーバーを設定できます。

関連情報

- SASL Identity マッピングの設定

第10章 連鎖ポリシーの設定

Directory Server は、クライアントアプリケーションからデータベースリンクを含む Directory Server にリクエストをチェーンするように設定できます。チェーンポリシーは、Directory Server で作成されたすべてのデータベースリンクに適用されます。

10.1. コンポーネントの動作の連鎖

コンポーネント は、フロントエンドのプラグインや関数など、内部操作を使用するサーバー内の機能単位です。

一部のコンポーネントは、ローカルデータのみにアクセスできることを想定し、内部 LDAP 要求をサーバーに送信します。このようなコンポーネントの場合、コンポーネントがそこでの操作を正常に完了できるように、チェーンポリシーを制御する必要があります。(例: 証明書の検証機能など)。この関数によって作成された LDAP リクエストを連鎖させて、リモートサーバーが信頼されていることを示す証明書を確認できます。リモートサーバーが信頼されていない場合は、セキュリティーの問題があります。

デフォルトでは、すべての内部操作とコンポーネントとを連鎖できませんが、デフォルトのオーバーライドは可能です。

さらに、指定したプラグインがリモートサーバー上で操作を実行できるようにするには、リモートサーバー上に **ACI** を作成する必要があります。**ACI** は、データベースリンクに割り当てられた **接尾辞** に存在している必要があります。

以下は、コンポーネント名、これらのコンポーネントに内部操作の連鎖を許可した場合に発生しえる副次的な影響、およびリモートサーバー上の **ACI** でコンポーネントが必要とするパーミッションです。

- **ACI プラグイン** コンポーネント
  ACI プラグイン コンポーネントはアクセス制御を実装します。ローカル属性とリモート属性を混在させるのは安全ではないため、**ACI** 属性の取得と更新に使用する操作を連鎖させることはできません。ただし、次の連鎖コンポーネント属性を設定して、ユーザーエントリーの取得に使用されるリクエストを連鎖できます。

  ```
  nsActiveChainingComponents: cn=ACI Plugin,cn=plugins,cn=config
  ```

  権限: 読み取り、検索、および比較

- **resource limit** コンポーネント
  resource limits コンポーネントは、ユーザーバインド DN に応じてサーバー制限を設定します。リソース制限コンポーネントをチェーンする場合は、リモートユーザーにリソース制限を適用できます。リソース制限コンポーネント操作を連鎖させるには、以下の連鎖コンポーネント属性を追加します。

  ```
  nsActiveChainingComponents: cn=resource limits,cn=components,cn=config
  ```

  権限: 読み取り、検索、および比較

- **certificate-based authentication** コンポーネント
  外部バインドメソッドでは、**certificate-based authentication** コンポーネントを使用できます。このコンポーネントは、リモートサーバーのデータベースからユーザー証明書を取得します。このコンポーネントの連鎖を許可すると、証明書ベースの認証がデータベースリンクで機能するようになります。このコンポーネントの操作を連鎖させるには、以下の連鎖コンポーネント属性を追加します。

```
nsActiveChainingComponents: cn=certificate-based
authentication,cn=components,cn=config
```

権限: 読み取り、検索、および比較

- **password policy** コンポーネント
  password policy コンポーネントは、リモートサーバーに **SASL** バインドを追加します。一部の形式の SASL 認証には、ユーザー名とパスワードでの認証が不可欠です。パスワードポリシーを有効にすると、サーバーは要求された特定の認証方法を検証および実装し、適切なパスワードポリシーを適用できます。このコンポーネントの動作を連鎖させるには、連鎖コンポーネント属性を追加します。

  ```
  nsActiveChainingComponents: cn=password policy,cn=components,cn=config
  ```

  権限: 読み取り、検索、および比較

- **SASL** コンポーネント
  SASL コンポーネントは、SASL がリモートサーバーにバインドできるようにします。このコンポーネントの動作を連鎖させるには、連鎖コンポーネント属性を追加します。

  ```
  nsActiveChainingComponents: cn=password policy,cn=components,cn=config
  ```

  権限: 読み取り、検索、および比較

- **referential integrity postoperation** コンポーネント
  referential integrity postoperation コンポーネントは、DN を含む属性の更新を、属性へのポインターが含まれるエントリーに伝播します。たとえば、グループが削除されると、グループからエントリーを自動的に削除できます。**referential integrity postoperation** プラグインをチェーンとともに使用すると、グループメンバーが静的グループ定義にリモートな場合に静的グループの管理が簡素化されます。

  ```
  nsActiveChainingComponents: cn=referential integrity postoperation,cn=plugins,cn=config
  ```

  権限: 読み取り、検索、および比較

- **attribute Uniqueness** コンポーネント
  attribute Uniqueness コンポーネントは、指定された属性のすべての値が一意であることを検証します。プラグインを連鎖すると、データベースリンクを通じて属性が変更された場合でも、属性値が一意であることが確認されます。このコンポーネントの動作を連鎖させるには、連鎖コンポーネント属性を追加します。

  ```
  nsActiveChainingComponents: cn=attribute uniqueness,cn=plugins,cn=config
  ```

  権限: 読み取り、検索、および比較

- **roles** コンポーネント
  roles コンポーネントは、データベースのエントリーのロールおよびロール割り当てを連鎖させます。このコンポーネントを連鎖すると、連鎖されたデータベースでもロールを維持します。このコンポーネントの動作を連鎖させるには、連鎖コンポーネント属性を追加します。

  ```
  nsActiveChainingComponents: cn=roles,cn=components,cn=config
  ```

  権限: 読み取り、検索、および比較

注記

Roles プラグイン、**Password policy** コンポーネント、**Replication** プラグイン、および **Referential Integrity** プラグインコンポーネントをチェーンすることはできません。チェーン要求を発行するサーバーで **Referential Integrity** プラグインを有効にする場合は、パフォーマンス、リソース、時間、および整合性のニーズを分析していることを確認します。整合性チェックは時間がかかり、メモリーと CPU を浪費する可能性があります。

10.2. コマンドラインを使用したコンポーネント操作の連鎖

コマンドラインを使用して、チェーンを許可するコンポーネントを追加できます。

手順

1. チェーンに追加するコンポーネントを指定します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set \ --
add-comp="cn=referential integrity postoperation,cn=components,cn=config"
```

2. インスタンスを再起動します。

```
# dsctl instance_name restart
```

3. 操作を連鎖させるリモートサーバーの接尾辞に ACI を作成します。

```
# ldapmodify -D "cn=Directory Manager" -W -H 389 remoteserver.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*")(target="ldap:///ou=customers,ou=People,dc=example,dc=com")
(version 3.0; acl "RefInt Access for chaining"; allow
(read,write,search,compare) userdn = "ldap:///cn=referential
integrity postoperation,cn=plugins,cn=config";)
```

検証

- 連鎖が許可されているコンポーネントを表示します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining config-set \ --
add-comp="cn=referential integrity postoperation,cn=components,cn=config"
```

10.3. WEB コンソールを使用したコンポーネントの操作の連鎖

Web コンソールを使用して、連鎖を許可するコンポーネントを追加できます。

前提条件

- Web コンソールで Directory Server ユーザーインターフェイスを開き、インスタンスを選択している。

手順

1. **Database** を開きます。
2. 左側のナビゲーションで、**Chaining Configuration** エントリーを選択します。
3. **Components to Chain** フィールドの下にある **Add** ボタンをクリックします。
4. 連鎖するコンポーネントを選択し、**Add & Save New Components** をクリックします。
5. 操作を連鎖させるリモートサーバーの接尾辞に **ACI** を作成します。

```
# ldapmodify -D "cn=Directory Manager" -W -H 389 remoteserver.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*")(target="ldap:///ou=customers,ou=People,dc=example,dc=com")
(version 3.0; acl "RefInt Access for chaining"; allow
(read,write,search,compare) userdn = "ldap:///cn=referential
integrity postoperation,cn=plugins,cn=config";)
```

検証

- 選択したコンポーネントは、連鎖させる必要があります。

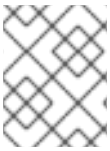
第11章 LDAP 制御チェーン

LDAP 操作には、動作を変更するデータ (コントロールという名前が付けられ、LDAP プロトコルで指定されたもの) が含まれる場合があります。どの LDAP コントロールをリモートサーバーに転送するかを指定できます。

11.1. LDAP コントロールの連鎖について

データベースリンクは、LDAP コントロールを連鎖するために、次のコントロールを含むリクエストをリモートサーバーに転送します。

- **Virtual List View (VLV)** コントロールは、特定のエントリーのリストを提供します。
- **Server-side sorting** コントロールは、通常は特定の一致ルールを使用して、属性値に従ってエントリーを分類します。
- この **Dereferencing** では、参照されたエントリーから指定された属性情報を取得し、この情報を残りの検索結果とともに返します。
- **マネージド DSA** コントロールは、これらの参照に従うのではなく、スマート参照をエントリーとして返します。したがって、スマート参照自体は変更または削除できます。
- **Loop detection** コントロールは、別のサーバーとサーバーチェーンの回数を追跡します。カウントが設定された数に達すると、**Loop detection** によってループが検出され、クライアントアプリケーションに通知されます。



注記

データベースリンクは、クライアントアプリケーションが複数のデータベースにリクエストを行う場合、**Server-side sorting** と **VLV** コントロールをサポートできません。

以下に、チェーンを許可される **LDAP** コントロールとそのオブジェクト識別子 (OID) の一部を示します。

コントロール名	OID
仮想リストビュー (VLV)	2.16.840.1.113730.3.4.9
サーバー側のソート	1.2.840.113556.1.4.473
管理 DSA	2.16.840.1.113730.3.4.2
ループ検出	1.3.6.1.4.1.1466.29539.12
検索の逆参照	1.3.6.1.4.1.4203.666.5.16

11.2. コマンドラインを使用した LDAP コントロールの連鎖

コマンドラインで **dsconf chaining config-set --add-control** を使用すると、**LDAP** コントロールを連鎖できます。

手順

1. **LDAP** コントロールを連鎖します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com chaining \ config-set --
add-control="2.16.840.1.113730.3.4.9"
```

Directory Server のクライアントが独自のコントロールを作成し、そこでの操作をリモートサーバーに連鎖する場合は、カスタムコントロールのオブジェクト識別子 (OID) を追加します。

11.3. WEB コンソールを使用した LDAP 制御の連鎖

Web コンソールを使用して **LDAP** コントロールを連鎖できます。

前提条件

- Web コンソールで Directory Server ユーザーインターフェイスを開き、インスタンスを選択している。

手順

1. **Database** メニューを開きます。
2. **Chaining Configuration** エントリーを選択します。
3. **Forwarded LDAP Controls** フィールドの下にある **Add** ボタンをクリックします。
4. LDAP コントロールを選択し、**Add & Save New Controls** をクリックします。
   Directory Server のクライアントが独自のコントロールを作成し、そこでの操作をリモートサーバーに連鎖する場合は、カスタムコントロールのオブジェクト識別子 (OID) を追加します。
5. **Save** ボタンをクリックします。

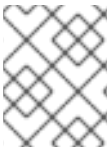
検証

- **Database** メニューをクリックし、選択した **LDAP** コントロールが連鎖されていることを確認します。

第12章 データベースリンクおよびアクセス制御評価

ユーザーがデータベースリンクを含むサーバーにバインドすると、データベースリンクがユーザーの ID をリモートサーバーに送信します。リモートサーバーでアクセス制御を評価できます。

プロキシー認証コントロールを使用して渡されたクライアントアプリケーションの元の ID を使用して、リモートサーバー上の **LDAP** 操作を評価できます。

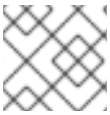


注記

リモートサーバー上で操作を成功させるには、リモートサーバー上に存在するサブツリーに対して正しいアクセス制御が必要です。

次の制限を指定して、通常のアクセス制御をリモートサーバーに追加できます。

- すべてのタイプのアクセス制御を使用できません。たとえば、データベースリンクを介してデータにアクセスするため、ロールベースまたはフィルターベースの ACI はユーザーエントリーにアクセスする必要があります。
- リモートサーバーは、データベースリンクと同じ IP アドレスと DNS ドメインでクライアントアプリケーションを表示します。クライアントの元のドメインが連鎖中に失われるため、クライアントの IP アドレスまたは DNS ドメインに基づくすべてのアクセス制御が機能できなくなります。



注記

Directory Server は、**IPv4** と **IPv6** の IP アドレスの両方に対応します。

データベースリンクで使用される ACI には、以下の制限が適用されます。

- 使用するグループで ACI を特定する必要があります。動的グループの場合、グループのすべてのユーザーが ACI およびグループで特定します。静的グループの場合、ユーザーはリモートサーバーにリンクします。
- 使用するロール定義と、これらのロールを使用する予定のユーザーで ACI を特定必要があります。
- ユーザーのエントリーの値にリンクする ACI は、ユーザーがリモートの場合が機能する必要があります。

アクセス制御の評価は常にリモートサーバーで行われますが、データベースリンクを含むサーバーとリモートサーバーの両方でアクセス制御を評価することもできます。これには、以下の制限事項があります。

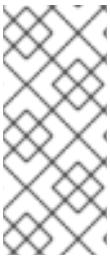
- たとえば、データベースリンクを含むサーバー上でアクセス制御を評価する場合、およびエントリーがリモートサーバー上にある場合、ユーザーエントリーの内容は必ずしも利用できるとは限りません。

注記

パフォーマンス上の理由から、クライアントはリモート問い合わせを実行してアクセス制御を評価することはできません。

- 変更操作を実行する場合、データベースリンクはリモートサーバーに保存されている完全なエントリーにアクセスできず、必然的にクライアントアプリケーションによって変更されているエントリーにもアクセスできません。

- 削除操作を実行すると、データベースリンクはエントリーの **DN** のみを認識します。アクセス制御が特定の属性を指定する場合、データベースリンクを介して実行すると削除操作は必ず失敗します。



注記

デフォルトでは、データベースリンクが含まれるサーバーに設定されたアクセス制御は評価できません。**cn=database_link**、**cn=chaining database**、**cn=plugins**、および **cn=config** エントリーの **nsCheckLocalACI** 属性を使用して、このデフォルト設定を上書きできます。ただし、データベースリンクを含むサーバーでアクセス制御を評価することは、カスケード連鎖を使用する場合を除いて推奨されません。