Red Hat Directory Server 12

ディレクトリースキーマの管理

カスタムスキーマの作成と管理

Last Updated: 2024-09-19

Red Hat Directory Server 12 ディレクトリースキーマの管理

カスタムスキーマの作成と管理

法律上の通知

概要

dsconf ユーティリティーと Web コンソールを使用して作成したカスタムスキーマを追加することで、追加のデータを Directory Server に保存できます。スキーマを拡張して、既存の属性値の構文を検証することもできます。

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 DSCONF ユーティリティーを使用したカスタムスキーマの作成

スキーマを拡張することにより、カスタム属性とオブジェクトクラスを Directory Server に追加できます。以下のように、スキーマを拡張できます。

- コマンドラインで **dsconf** ユーティリティーを使用した拡張(このセクションでは、このプロセスを説明します)
- Directory Server Web コンソールを使用した拡張
- スキーマファイルの作成による手動での拡張

1.1. スキーマ拡張のワークフロー

新しいスキーマ要素の追加には、以下が必要です。

1. 新しいスキーマの一意のオブジェクト識別子 (OID) を計画し、定義します。Directory Server は OID によってスキーマ要素を認識しますが、OID を手動で管理する必要があります。
   OID は、サーバーへのスキーマ要素を識別するドットで区切られた番号です。OID は、異なるブランチに対応するために拡張できるベース OID を使用して階層化することができます。たとえば、ベース OID は **1** で、属性のブランチを **1.1** にし、オブジェクトクラスのブランチを **1.2** にすることもできます。

   重要

   必須ではない場合でも、上位互換性とパフォーマンスを向上させるために、カスタムスキーマに数値 OID を使用することを推奨します。

2. Internet Assigned Numbers Authority (IANA) に OID をリクエストします。詳細は、https://pen.iana.org/pen/PenApplication.page を参照してください。
3. OID レジストリーを作成して、OID 割り当てを追跡し、複数の目的で OID が使用されないようにします。OID レジストリーは、説明を含むディレクトリースキーマで使用されるすべての OID のリストです。カスタムスキーマで OID レジストリーを公開します。
4. 新しい属性を定義します。
5. 新しい属性を含むオブジェクトクラスを定義します。ただし、デフォルトのスキーマは更新しないでください。新しい属性を作成する場合は、常にそれらをカスタムオブジェクトクラスに追加してください。

Directory Server は、インスタンスの起動時にスキーマをロードします。新しいスキーマファイルをロードするには、インスタンスを再起動するか、リロードタスクを開始します。

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ます。
- 複数のオブジェクトクラスまたは属性を同じ目的で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。

警告

他のディレクトリーまたは LDAP クライアントアプリケーションとの互換性の問題を回避するために、標準スキーマを更新または削除しないでください。

1.2. DIRECTORY SERVER がレプリケーション環境でスキーマの更新を管理する方法

ディレクトリースキーマが **cn=schema** ツリーで更新されると、Directory Server は変更状態番号 (CSN) を含む **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに変更を保存します。

Directory Server は、スキーマの変更を他のレプリカに直接複製しません。スキーマレプリケーションは、ディレクトリーのコンテンツが複製されたツリーで更新されると開始します。たとえば、スキーマの変更後にユーザーエントリーを更新すると、**nsSchemaCSN** 属性に保存されている CSN と、コンシューマーにある CSN が比較されます。コンシューマーの **nsSchemaCSN** 属性の値がサプライヤーの値よりも低い場合、Directory Server はスキーマをコンシューマーに複製します。レプリケーションに成功すると、サプライヤーにあるすべてのオブジェクトクラスと属性タイプはコンシューマーの定義のスーパーセットである必要があります。

例1.1 スキーマのサブセットとスーパーセット

- **server1** では、**example** オブジェクトクラスが **a1** 属性、**a2** 属性、および **a3** 属性を許可します。
- **server2** では、**example** オブジェクトクラスが **a1** 属性および **a3** 属性を許可します。

前の例では、**server1** の **example** オブジェクトクラスのスキーマ定義は、**server2** のオブジェクトクラスのスーパーセットです。検証フェーズで、Directory Server がスキーマを複製または受け入れると、サーバーはスーパーセット定義を取得します。たとえば、ローカルスキーマのオブジェクトクラスがサプライヤースキーマのオブジェクトクラスよりも少ない属性を許可していることをコンシューマーが検出すると、Directory Server がローカルスキーマを更新します。

スキーマ定義が正常に複製された場合、**nsSchemaCSN** 属性は両サーバーで同一になり、オブジェクトクラスや属性タイプなどのスキーマ定義はレプリケーションセッションの開始時に比較されなくなります。

次のシナリオでは、Directory Server はスキーマを複製しません。

- あるホストのスキーマが、別のホストのスキーマのサブセットの場合
  たとえば、**server2** にある **example** オブジェクトクラスのスキーマ定義は **server1** のオブジェクトクラスのサブセットです。サブセットは、属性 (単一値属性は多値属性のサブセット) および属性の構文に対しても発生する可能性があります。
- サプライヤースキーマとコンシューマースキーマの定義をマージする必要がある場合
- Directory Server がマージするスキーマをサポートしない場合。たとえば、1 台のサーバーのオブジェクトクラスが **a1** 属性、**a2** 属性、および **a3** 属性を許可し、別のサーバーのオブジェクト

クラスが **a1** 属性、**a3** 属性、および **a4** 属性を許可する場合、スキーマはサブセットではないので、マージできません。

- **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 以外のスキーマファイルを使用する場合
  Directory Server を使用すると、**/etc/dirsrv/slapd-instance_name/schema/** ディレクトリーにスキーマファイルを追加できます。ただし、**/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルの CSN のみが更新されます。このため、他のスキーマファイルはローカルでのみ使用され、レプリケーションパートナーに自動的に転送されません。

重要

Directory Server がスキーマを自動的に複製できるようにし、スキーマ定義の重複を回避するには、カスタムスキーマを **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに保存します。

1.3. DSCONF を使用した属性およびオブジェクトクラスのカスタムスキーマの作成

この手順は、**dsconf** ユーティリティーを使用して次のカスタムスキーマを作成する方法を示しています。

- OID **2.16.840.1.1133730.2.1.123** および構文 **Directory String** (OID **1.3.6.1.4.1.1466.115.121.1.15**) を持つ **dateOfBirth** という名前の単一値属性
- 親オブジェクトクラス (**SUP top**) のない **exampleperson** という名前のオブジェクトクラス、**dateOfBirth** 属性が含まれている必要がある OID **2.16.840.1.1133730.2.1.99**

手順

1. **dateOfBirth** 属性を作成します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema attributetypes add --oid="2.16.840.1.1133730.2.1.123" --desc="For employee birthdays" --syntax="1.3.6.1.4.1.1466.115.121.1.15" --single-value --x-origin="Example defined" dateOfBirth
```

2. **exampleperson** オブジェクトクラスを作成します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema objectclasses add --oid="2.16.840.1.1133730.2.1.99" --desc="An example person object class" --sup="top" --must="dateOfBirth" examplePerson
```

3. スキーマの再読み込みタスクを実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema reload
```

検証

- **/var/log/dirsrv/slapd-instance_name/errors** ファイルを監視します。
  - ビルドが成功すると、Directory Server は次のログを記録します。

```
[23/Sep/2021:13:47:33.334241406 +0200] - INFO - schemareload -
schemareload_thread - Schema reload task starts (schema dir: default) ...
[23/Sep/2021:13:47:33.415692558 +0200] - INFO - schemareload -
schemareload_thread - Schema validation passed.
[23/Sep/2021:13:47:33.454768148 +0200] - INFO - schemareload -
schemareload_thread - Schema reload task finished.
```

- ビルドが失敗した場合、Directory Server は失敗した手順とその理由をログに記録します。

第2章 WEB コンソールを使用したカスタムスキーマの作成

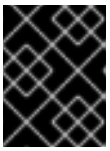
スキーマを拡張することにより、カスタム属性とオブジェクトクラスを Directory Server に追加できます。以下のように、スキーマを拡張できます。

- Directory Server Web コンソールを使用した拡張(このセクションでは、このプロセスを説明します)
- コマンドラインで dsconf ユーティリティーを使用した拡張
- スキーマファイルの作成による手動での拡張

2.1. スキーマ拡張のワークフロー

新しいスキーマ要素の追加には、以下が必要です。

1. 新しいスキーマの一意のオブジェクト識別子 (OID) を計画し、定義します。Directory Server は OID によってスキーマ要素を認識しますが、OID を手動で管理する必要があります。
   OID は、サーバーへのスキーマ要素を識別するドットで区切られた番号です。OID は、異なるブランチに対応するために拡張できるベース OID を使用して階層化することができます。たとえば、ベース OID は **1** で、属性のブランチを **1.1** にし、オブジェクトクラスのブランチを **1.2** にすることもできます。

   重要

   必須ではない場合でも、上位互換性とパフォーマンスを向上させるために、カスタムスキーマに数値 OID を使用することを推奨します。

2. Internet Assigned Numbers Authority (IANA) に OID をリクエストします。詳細は、https://pen.iana.org/pen/PenApplication.page を参照してください。
3. OID レジストリーを作成して、OID 割り当てを追跡し、複数の目的で OID が使用されないようにします。OID レジストリーは、説明を含むディレクトリースキーマで使用されるすべての OID のリストです。カスタムスキーマで OID レジストリーを公開します。
4. 新しい属性を定義します。
5. 新しい属性を含むオブジェクトクラスを定義します。ただし、デフォルトのスキーマは更新しないでください。新しい属性を作成する場合は、常にそれらをカスタムオブジェクトクラスに追加してください。

Directory Server は、インスタンスの起動時にスキーマをロードします。新しいスキーマファイルをロードするには、インスタンスを再起動するか、リロードタスクを開始します。

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ます。
- 複数のオブジェクトクラスまたは属性を同じ目的で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。

警告

他のディレクトリーまたは LDAP クライアントアプリケーションとの互換性の問題を回避するために、標準スキーマを更新または削除しないでください。

2.2. DIRECTORY SERVER がレプリケーション環境でスキーマの更新を管理する方法

ディレクトリースキーマが **cn=schema** ツリーで更新されると、Directory Server は変更状態番号 (CSN) を含む **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに変更を保存します。

Directory Server は、スキーマの変更を他のレプリカに直接複製しません。スキーマレプリケーションは、ディレクトリーのコンテンツが複製されたツリーで更新されると開始します。たとえば、スキーマの変更後にユーザーエントリーを更新すると、**nsSchemaCSN** 属性に保存されている CSN と、コンシューマーにある CSN が比較されます。コンシューマーの **nsSchemaCSN** 属性の値がサプライヤーの値よりも低い場合、Directory Server はスキーマをコンシューマーに複製します。レプリケーションに成功すると、サプライヤーにあるすべてのオブジェクトクラスと属性タイプはコンシューマーの定義のスーパーセットである必要があります。

例2.1 スキーマのサブセットとスーパーセット

- **server1** では、**example** オブジェクトクラスが **a1** 属性、**a2** 属性、および **a3** 属性を許可します。
- **server2** では、**example** オブジェクトクラスが **a1** 属性および **a3** 属性を許可します。

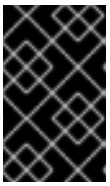
前の例では、**server1** の **example** オブジェクトクラスのスキーマ定義は、**server2** のオブジェクトクラスのスーパーセットです。検証フェーズで、Directory Server がスキーマを複製または受け入れると、サーバーはスーパーセット定義を取得します。たとえば、ローカルスキーマのオブジェクトクラスがサプライヤースキーマのオブジェクトクラスよりも少ない属性を許可していることをコンシューマーが検出すると、Directory Server がローカルスキーマを更新します。

スキーマ定義が正常に複製された場合、**nsSchemaCSN** 属性は両サーバーで同一になり、オブジェクトクラスや属性タイプなどのスキーマ定義はレプリケーションセッションの開始時に比較されなくなります。

次のシナリオでは、Directory Server はスキーマを複製しません。

- あるホストのスキーマが、別のホストのスキーマのサブセットの場合
  たとえば、**server2** にある **example** オブジェクトクラスのスキーマ定義は **server1** のオブジェクトクラスのサブセットです。サブセットは、属性 (単一値属性は多値属性のサブセット) および属性の構文に対しても発生する可能性があります。
- サプライヤースキーマとコンシューマースキーマの定義をマージする必要がある場合
- Directory Server がマージするスキーマをサポートしない場合。たとえば、1 台のサーバーのオブジェクトクラスが **a1** 属性、**a2** 属性、および **a3** 属性を許可し、別のサーバーのオブジェクトクラスが **a1** 属性、**a3** 属性、および **a4** 属性を許可する場合、スキーマはサブセットではないので、マージできません。

- **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 以外のスキーマファイルを使用する場合
  Directory Server を使用すると、**/etc/dirsrv/slapd-instance_name/schema/** ディレクトリーにスキーマファイルを追加できます。ただし、**/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルの CSN のみが更新されます。このため、他のスキーマファイルはローカルでのみ使用され、レプリケーションパートナーに自動的に転送されません。



重要

Directory Server がスキーマを自動的に複製できるようにし、スキーマ定義の重複を回避するには、カスタムスキーマを **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに保存します。

2.3. WEB コンソールを使用した属性およびオブジェクトクラスのカスタムスキーマの作成

この手順では、Web コンソールを使用して次のカスタムスキーマを作成する方法を示します。

- OID **2.16.840.1.1133730.2.1.123** および構文 **Directory String** (OID **1.3.6.1.4.1.1466.115.121.1.15**) を持つ **dateOfBirth** という名前の単一値属性
- 親オブジェクトクラス (**SUP top**) のない **exampleperson** という名前のオブジェクトクラス、**dateOfBirth** 属性が含まれている必要がある OID **2.16.840.1.1133730.2.1.99**

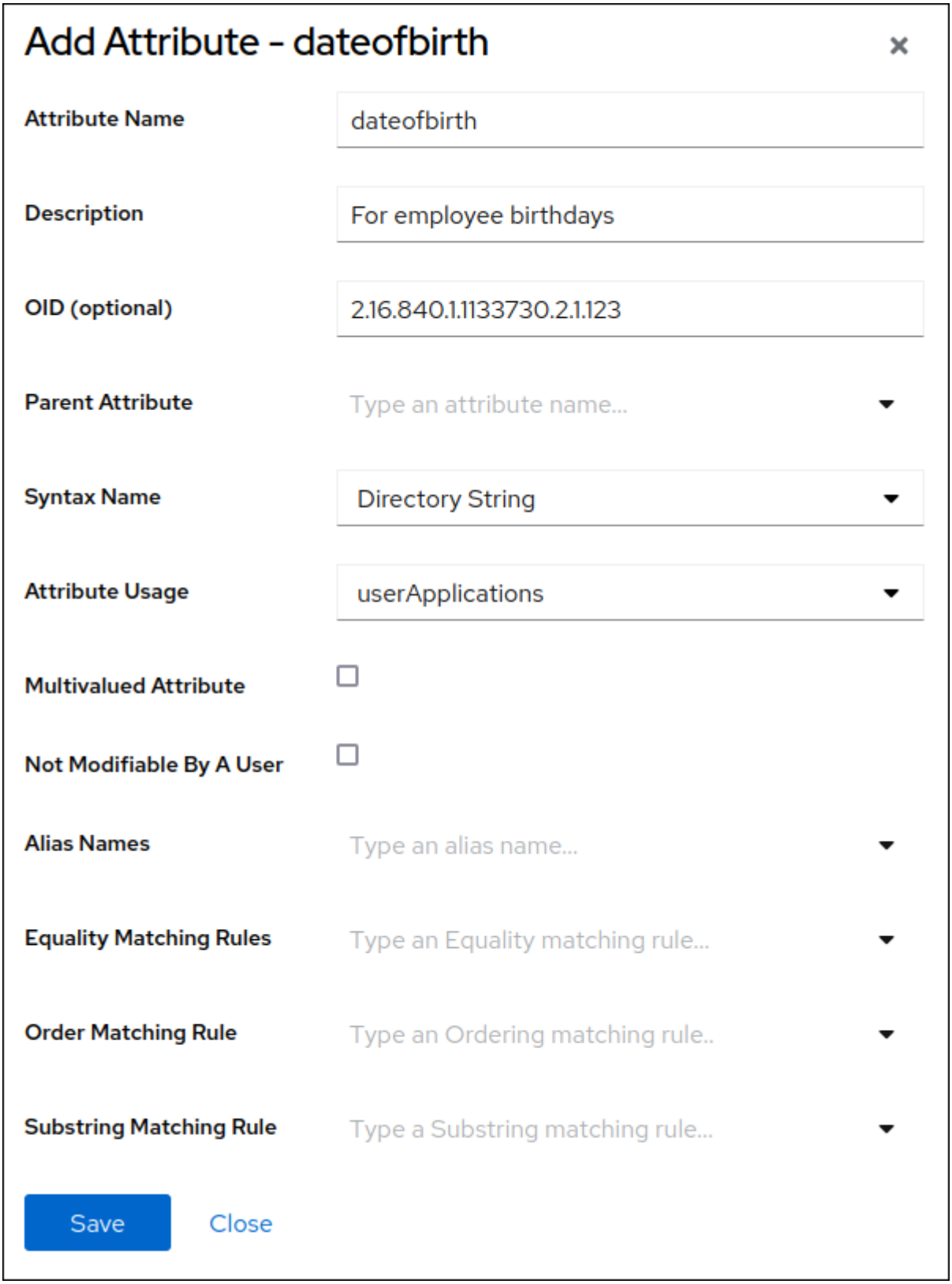
Web コンソールを使用してスキーマを更新する場合、Directory Server はスキーマを自動的に再ロードします。

前提条件

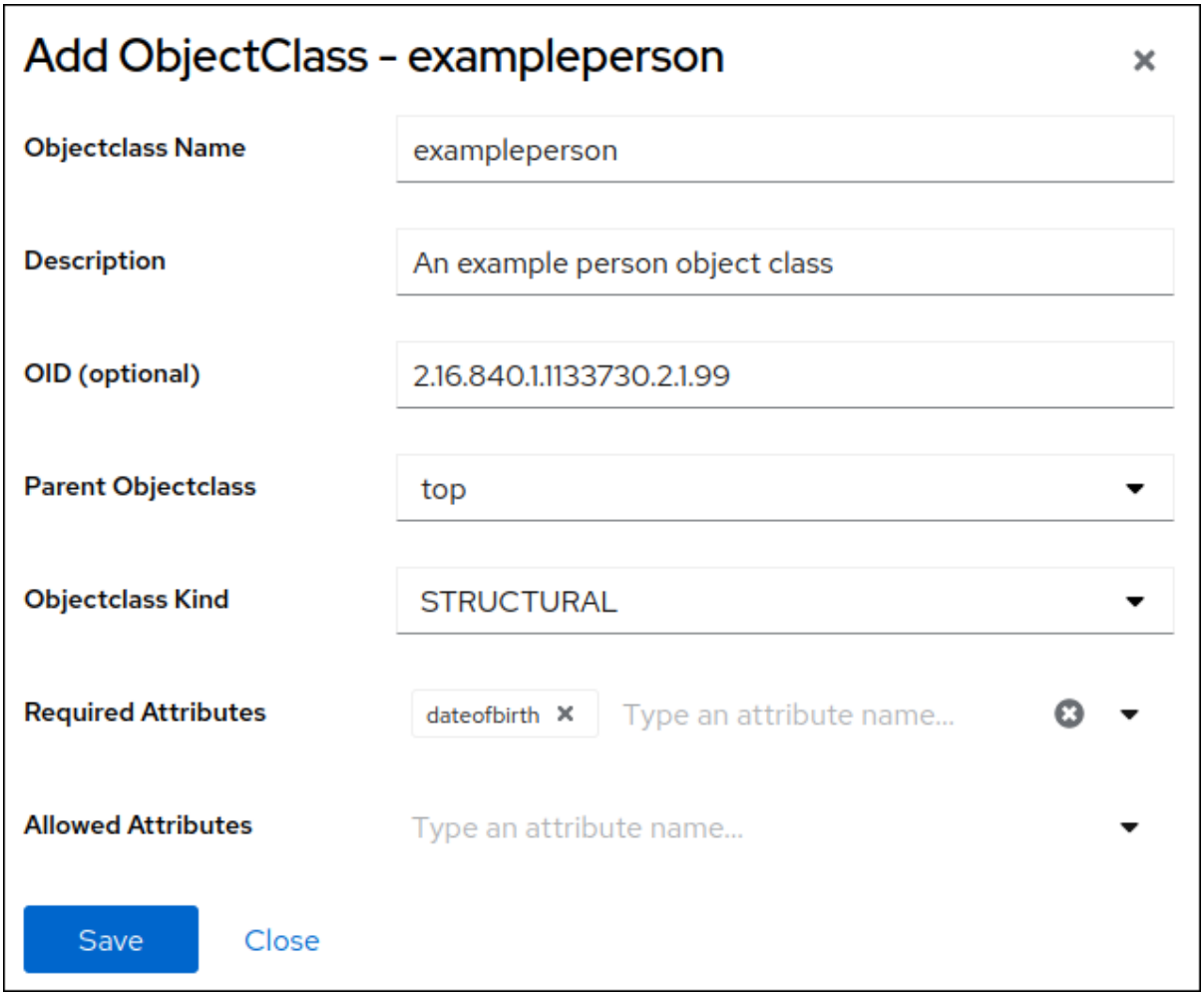
- Web コンソールでインスタンスにログインしている。

手順

1. **Schema → Attributes** に移動し、**Add Attribute** をクリックします。
2. 追加する属性の設定を入力します。



3. **Save**をクリックします。
4. Schema → Objectclasses に移動し、**Add ObjectClass** をクリックします。
5. 追加するオブジェクトクラスの設定を入力します。



6. **Save**をクリックします。

検証

- **Monitoring → Logging → Errors Log** に移動します。
  - ビルドが成功すると、Directory Server は次のログを記録します。

```
[23/Sep/2021:13:47:33.334241406 +0200] - INFO - schemareload -
schemareload_thread - Schema reload task starts (schema dir: default) ...
[23/Sep/2021:13:47:33.415692558 +0200] - INFO - schemareload -
schemareload_thread - Schema validation passed.
[23/Sep/2021:13:47:33.454768148 +0200] - INFO - schemareload -
schemareload_thread - Schema reload task finished.
```

  - ビルドが失敗した場合、Directory Server は失敗した手順とその理由をログに記録します。

第3章 カスタムスキーマファイルの手動での作成

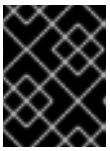
スキーマを拡張することにより、カスタム属性とオブジェクトクラスを Directory Server に追加できます。以下のように、スキーマを拡張できます。

- スキーマファイルの作成による手動での拡張(このセクションでは、このプロセスを説明します)
- コマンドラインで dsconf ユーティリティーを使用した拡張
- Directory Server Web コンソールを使用した拡張

3.1. スキーマ拡張のワークフロー

新しいスキーマ要素の追加には、以下が必要です。

1. 新しいスキーマの一意のオブジェクト識別子 (OID) を計画し、定義します。Directory Server は OID によってスキーマ要素を認識しますが、OID を手動で管理する必要があります。
   OID は、サーバーへのスキーマ要素を識別するドットで区切られた番号です。OID は、異なるブランチに対応するために拡張できるベース OID を使用して階層化することができます。たとえば、ベース OID は **1** で、属性のブランチを **1.1** にし、オブジェクトクラスのブランチを **1.2** にすることもできます。

   重要

   必須ではない場合でも、上位互換性とパフォーマンスを向上させるために、カスタムスキーマに数値 OID を使用することを推奨します。

2. Internet Assigned Numbers Authority (IANA) に OID をリクエストします。詳細は、https://pen.iana.org/pen/PenApplication.page を参照してください。
3. OID レジストリーを作成して、OID 割り当てを追跡し、複数の目的で OID が使用されないようにします。OID レジストリーは、説明を含むディレクトリースキーマで使用されるすべての OID のリストです。カスタムスキーマで OID レジストリーを公開します。
4. 新しい属性を定義します。
5. 新しい属性を含むオブジェクトクラスを定義します。ただし、デフォルトのスキーマは更新しないでください。新しい属性を作成する場合は、常にそれらをカスタムオブジェクトクラスに追加してください。

Directory Server は、インスタンスの起動時にスキーマをロードします。新しいスキーマファイルをロードするには、インスタンスを再起動するか、リロードタスクを開始します。

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ます。
- 複数のオブジェクトクラスまたは属性を同じ目的で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。

警告

他のディレクトリーまたは LDAP クライアントアプリケーションとの互換性の問題を回避するために、標準スキーマを更新または削除しないでください。

3.2. スキーマファイルの要件

スキーマファイルは、**cn=schema** エントリーを定義する LDIF 形式を使用します。各属性タイプとオブジェクトクラスがこのエントリーに追加されます。

スキーマファイルの要件は次のとおりです。

- ファイルは次のエントリーで始まる必要があります。

```
dn: cn=schema
```

- スキーマファイルには、属性タイプまたはオブジェクトクラス、あるいはその両方を含めることができます。
- オブジェクトクラス定義は、他のスキーマファイルで定義された属性を使用できます。
- カスタムスキーマファイルを使用するインスタンスに応じて、次のいずれかの場所に保存します。
  - **/etc/dirsrv/slapd-instance_name/schema/** は、この特定のインスタンスでスキーマファイルを使用できるようにします。
  - **/usr/share/dirsrv/schema/** は、このホストで実行しているすべてのインスタンスでスキーマファイルを使用できるようにします。
- デフォルトでは、Directory Server は **99user.ldif** ファイルのカスタムスキーマを想定しています。別のファイル名を使用する場合は、以下のようにします。
  - 名前はアルファベット順に **99user.ldif** より小さくする必要があります。たとえば、**99aaa.ldif** は使用できますが、**99zzz.ldif** は使用できません。
  - カスタムスキーマファイルは、**00** から **98** までのコアスキーマファイルの後にロードする必要があるため、名前は 2 桁で始まり、**01** より大きくなければなりません。
    Directory Server は、スキーマファイルをアルファベット順に読み取ります。したがって、たとえば、定義 **99user.ldif** を保存すると、名前が **00** と **01** で始まる標準ファイルの定義がオーバーライドされます。
- **/usr/share/dirsrv/data/** ディレクトリーの標準スキーマファイルを使用する場合は、ファイルを使用するインスタンスに応じて、ファイルを **/etc/dirsrv/slapd-instance_name/schema/** または **/usr/share/dirsrv/schema/** にコピーします。ただし、宛先ディレクトリーで別のファイル名を使用してください。それ以外の場合は、Directory Server はアップグレード中にファイルの名前を変更し、**.bak** 接尾辞を追加します。

例3.1 カスタムスキーマファイルの例

```
dn: cn=schema
```

```
objectClasses: ( 2.16.840.1.1133730.2.1.123 NAME 'exampleperson' DESC 'An example
  person object class' SUP top STRUCTURAL MUST dateOfBirth X-ORIGIN 'user defined' )
attributeTypes: ( 2.16.840.1.1133730.2.1.99 NAME 'dateOfBirth' DESC 'For employee
  birthday' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined' )
```

3.3. カスタムスキーマファイルの属性の定義

スキーマファイルの属性は、**attributeTypes** 属性の値として定義します。

例3.2 属性の定義

```
attributeTypes: ( 2.16.840.1.1133730.2.1.123 NAME 'dateOfBirth' DESC 'For employee birthday'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined' )
```

属性定義には、次のコンポーネントが含まれています。

- ドット区切りの番号として指定された一意のオブジェクト識別子 (OID)。
- **NAME attribute_name** 形式の一意の名前。
- **DESC description** 形式の説明。
- **SYNTAX OID** 形式の属性値の OID。LDAP 属性構文の詳細は、RFC 4517 を参照してください。
- オプション: 属性が定義されているソース。

3.4. カスタムスキーマファイルでのオブジェクトクラスの定義

スキーマファイルのオブジェクトクラスは、**objectClasses** 属性の値として定義します。

例3.3 オブジェクトクラスの定義

```
objectClasses: ( 2.16.840.1.1133730.2.1.99 NAME 'exampleperson' DESC 'An example person
object class' SUP top STRUCTURAL MUST dateOfBirth X-ORIGIN 'user defined' )
```

オブジェクトクラス定義には、次のコンポーネントが含まれています。

- ドット区切りの番号として指定された一意のオブジェクト識別子 (OID)。
- **NAME attribute_name** 形式の一意の名前。
- **DESC description** 形式の説明。
- **SUP object_class** 形式のこのオブジェクトクラスの上位 (親) オブジェクトクラス。関連する親がない場合は、**SUP top** を使用します。
- **STRUCTURAL** という単語は、オブジェクトクラスが適用されるエントリーのタイプを定義します。すべてのエントリーは、少なくとも 1 つの **STRUCTURAL** オブジェクトクラスに属している必要があります。**AUXILIARY** は、すべてのエントリーに適用できることを意味します。

- 必須属性のリスト。前に **MUST** キーワードが付きます。複数の属性を含めるには、グループを括弧で囲み、属性を [command]`$ ` (ドル記号とスペース) で区切ります。
- **MAY** キーワードが前に付いたオプションの属性のリスト。複数の属性を含めるには、グループを括弧で囲み、属性を [command]`$ ` (ドル記号とスペース) で区切ります。

名前と OID のみが必要であり、その他の設定はオブジェクトクラスのニーズによって異なります。

関連情報

- RFC 4512 のセクション 2.4

3.5. DIRECTORY SERVER がレプリケーション環境でスキーマの更新を管理する方法

ディレクトリースキーマが **cn=schema** ツリーで更新されると、Directory Server は変更状態番号 (CSN) を含む **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに変更を保存します。

Directory Server は、スキーマの変更を他のレプリカに直接複製しません。スキーマレプリケーションは、ディレクトリーのコンテンツが複製されたツリーで更新されると開始します。たとえば、スキーマの変更後にユーザーエントリーを更新すると、**nsSchemaCSN** 属性に保存されている CSN と、コンシューマーにある CSN が比較されます。コンシューマーの **nsSchemaCSN** 属性の値がサプライヤーの値よりも低い場合、Directory Server はスキーマをコンシューマーに複製します。レプリケーションに成功すると、サプライヤーにあるすべてのオブジェクトクラスと属性タイプはコンシューマーの定義のスーパーセットである必要があります。

> **例3.4 スキーマのサブセットとスーパーセット**
>
> - **server1** では、**example** オブジェクトクラスが **a1** 属性、**a2** 属性、および **a3** 属性を許可します。
> - **server2** では、**example** オブジェクトクラスが **a1** 属性および **a3** 属性を許可します。

前の例では、**server1** の **example** オブジェクトクラスのスキーマ定義は、**server2** のオブジェクトクラスのスーパーセットです。検証フェーズで、Directory Server がスキーマを複製または受け入れると、サーバーはスーパーセット定義を取得します。たとえば、ローカルスキーマのオブジェクトクラスがサプライヤースキーマのオブジェクトクラスよりも少ない属性を許可していることをコンシューマーが検出すると、Directory Server がローカルスキーマを更新します。

スキーマ定義が正常に複製された場合、**nsSchemaCSN** 属性は両サーバーで同一になり、オブジェクトクラスや属性タイプなどのスキーマ定義はレプリケーションセッションの開始時に比較されなくなります。

次のシナリオでは、Directory Server はスキーマを複製しません。

- あるホストのスキーマが、別のホストのスキーマのサブセットの場合
  たとえば、**server2** にある **example** オブジェクトクラスのスキーマ定義は **server1** のオブジェクトクラスのサブセットです。サブセットは、属性 (単一値属性は多値属性のサブセット) および属性の構文に対しても発生する可能性があります。
- サプライヤースキーマとコンシューマースキーマの定義をマージする必要がある場合
- Directory Server がマージするスキーマをサポートしない場合。たとえば、1 台のサーバーのオブジェクトクラスが **a1** 属性、**a2** 属性、および **a3** 属性を許可し、別のサーバーのオブジェクト

クラスが **a1** 属性、**a3** 属性、および **a4** 属性を許可する場合、スキーマはサブセットではないので、マージできません。

- **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 以外のスキーマファイルを使用する場合
  Directory Server を使用すると、**/etc/dirsrv/slapd-instance_name/schema/** ディレクトリーにスキーマファイルを追加できます。ただし、**/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルの CSN のみが更新されます。このため、他のスキーマファイルはローカルでのみ使用され、レプリケーションパートナーに自動的に転送されません。

重要

Directory Server がスキーマを自動的に複製できるようにし、スキーマ定義の重複を回避するには、カスタムスキーマを **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに保存します。

3.6. 属性およびオブジェクトクラスのカスタムスキーマファイルの手動での作成

カスタムスキーマを手動で作成する場合は、それを **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルに保存します。別のファイル名を使用することも可能ですが、他のファイルに格納されているスキーマ定義が複製され、レプリカの **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** に格納されるなどの欠点があります。Directory Server がレプリケーション環境でスキーマの更新を管理する方法 を参照してください。

この手順により、以下が追加されます。

- OID **2.16.840.1.1133730.2.1.123** および構文 **Directory String** (OID **1.3.6.1.4.1.1466.115.121.1.15**) を持つ **dateOfBirth** という名前の単一値属性
- **exampleperson** という名前のオブジェクトクラス (親オブジェクトクラス (**SUP top**) がなく、**dateOfBirth** 属性が含まれる必要あり)

手順

1. **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** ファイルの **dn: cn=schema** エントリーの下に次のコンテンツを追加します。

```
attributeTypes: ( 2.16.840.1.1133730.2.1.123 NAME 'dateOfBirth' DESC 'For employee
  birthday' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined'
)
objectClasses: ( 2.16.840.1.1133730.2.1.99 NAME 'exampleperson' DESC 'An example
  person object class' SUP top STRUCTURAL MUST dateOfBirth X-ORIGIN 'user defined' )
```

2. スキーマの再読み込みタスクを実行します。

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com schema reload
```

検証手順:

- **/var/log/dirsrv/slapd-instance_name/errors** ファイルを監視します。
  - ビルドが成功すると、Directory Server は次のログを記録します。

```
[23/Sep/2021:13:47:33.334241406 +0200] - INFO - schemareload -
schemareload_thread - Schema reload task starts (schema dir: default) ...
[23/Sep/2021:13:47:33.415692558 +0200] - INFO - schemareload -
schemareload_thread - Schema validation passed.
[23/Sep/2021:13:47:33.454768148 +0200] - INFO - schemareload -
schemareload_thread - Schema reload task finished.
```

- ビルドが失敗した場合、Directory Server は失敗した手順とその理由をログに記録します。

第4章 既存の属性値の構文の検証

構文検証により、ディレクトリーサーバーは、属性値がその属性の定義で指定された構文の規則に従っているかどうかを確認します。ディレクトリーサーバーは、構文検証タスクの結果を **/var/log/dirsrv/slapd-instance_name/errors** ファイルに記録します。

次の場合、手動の構文検証が必要です。

- **nsslapd-syntaxcheck** パラメーターで構文検証が無効化されている場合。

  注記

  Red Hat では、構文検証を無効化しないことを推奨しています。

- 構文検証が無効化されている、または構文検証を行わずにサーバーからデータを移行する場合。

4.1. DSCONF スキーマの VALIDATE-SYNTAX コマンドを使用した構文検証タスクの作成

dsconf schema validate-syntax コマンドを使用すると、構文検証タスクを作成して、変更されたすべての属性をチェックし、新しい値が必要な構文を持つことを確認できます。

手順

- 構文検証タスクを作成するには、次のように入力します。

  ```
  # dsconf -D "cn=Directory Manager" ldap://server.example.com schema validate-syntax -f
  '(objectclass=inetorgperson)' ou=People,dc=example,dc=com
  ```

  出力例では、コマンドは **(objectclass=inetorgperson)** フィルターに一致する **ou=People,dc=example,dc=com** サブツリー内のすべての値の構文を検証するタスクを作成します。

4.2. CN タスクエントリーを使用した構文検証タスクの作成

ディレクトリーサーバー設定の **cn=tasks,cn=config** エントリーは、サーバーがタスクを管理するために使用する一時エントリーのコンテナーエントリーです。**cn=syntax validate,cn=tasks,cn=config** エントリーでタスクを作成することにより、構文検証操作を開始できます。

手順

- 構文の検証操作を開始するには、次のように **cn=syntax validate,cn=tasks,cn=config** エントリーにタスクを作成します。

  ```
  # ldapadd -D "cn=Directory Manager" -W -p 389 -H ldap://server.example.com -x

  dn: cn=example_syntax_validate,cn=syntax validate,cn=tasks,cn=config
  objectclass: extensibleObject
  cn: cn=example_syntax_validate
  basedn: ou=People,dc=example,dc=com
  filter: (objectclass=inetorgperson)
  ```

出力例では、コマンドは **(objectclass=inetorgperson)** フィルターに似た **ou=People,dc=example,dc=com** サブツリーのすべての値の構文を検証するタスクを作成します。タスクが完了すると、ディレクトリーサーバーはディレクトリー設定からエントリーを削除します。

関連情報

- 設定およびスキーマ参照