Red Hat Directory Server 12

디렉터리 스키마 관리

사용자 지정 스키마 생성 및 관리

Last Updated: 2024-08-25

Red Hat Directory Server 12 디렉터리 스키마 관리

사용자 지정 스키마 생성 및 관리

법적 공지

초록

dsconf 유틸리티 및 웹 콘솔을 사용하여 생성된 사용자 지정 스키마를 추가하여 Directory Server에 추가 데이터를 저장할 수 있습니다. 스키마를 확장하고 기존 특성 값의 구문을 검증할 수도 있습니다.

차례

RED HAT DIRECTORY SERVER에 대한 피드백 제공

Red Hat의 문서 및 제품에 대한 의견을 제공해 주셔서 감사합니다. Red Hat이 어떻게 이를 개선할 수 있는지 알려 주십시오. 이렇게 하려면 다음을 수행합니다.

- Jira (계정 필요)를 통해 Red Hat Directory Server 설명서에 피드백을 제출하려면 다음을 수행합니다.
  1. Red Hat 문제 추적기 로 이동하십시오.
  2. **요약** 필드에 설명 제목을 입력합니다.
  3. **설명** 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
  4. 대화 상자 하단에서 **생성** 을 클릭합니다.
- Jira를 통해 Red Hat Directory Server 제품에 대한 피드백을 제출하기 위해 필요한 경우:
  1. Red Hat 문제 추적기 로 이동하십시오.
  2. **문제 생성** 페이지에서 **다음을** 클릭합니다.
  3. **Summary** 필드를 입력합니다.
  4. **Component** 필드에서 구성 요소를 선택합니다.
  5. 다음을 포함하여 **Description** 필드를 작성합니다.
     a. 선택한 구성 요소의 버전 번호입니다.
     b. 문제 또는 개선을 위한 제안을 재현하는 단계입니다.
  6. **생성**을 클릭합니다.

1장. DSCONF 유틸리티를 사용하여 사용자 정의 스키마 생성

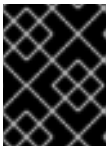
스키마를 확장하여 사용자 지정 특성 및 개체 클래스를 Directory Server에 추가할 수 있습니다. 스키마를 확장할 수 있습니다.

- 명령줄에서 **dsconf** 유틸리티 사용. 이 프로세스는 이 섹션에 설명되어 있습니다.
- 디렉터리 서버 웹 콘솔 사용.
- 스키마 파일을 생성하여 수동으로.

1.1. 스키마 확장 워크플로

새 스키마 요소를 추가하려면 다음이 필요합니다.

1. 새 스키마의 고유 OID(오브젝트 식별자)를 계획하고 정의합니다. Directory Server는 OID로 스키마 요소를 인식하지만 OID를 수동으로 관리해야 합니다.
   OID는 서버의 스키마 요소를 식별하는 점으로 구분된 번호입니다. OID는 다양한 분기를 수용하도록 확장할 수 있는 기본 OID를 사용하여 계층화할 수 있습니다. 예를 들어 기본 OID는 **1** 일 수 있으며 **1.1** 에서 속성과 **1.2** 의 오브젝트 클래스에 대한 분기가 있을 수 있습니다.

   중요

   필요하지 않은 경우에도 Red Hat은 사용자 정의 스키마에 숫자 OID를 사용하여 호환성과 성능을 향상시키는 것이 좋습니다.

2. INA(Internet Assigned Numbers Authority)에서 OID를 요청합니다. 자세한 내용은 https://pen.iana.org/pen/PenApplication.page 을 참조하십시오.
3. OID 레지스트리를 생성하여 OID 할당을 추적하고 둘 이상의 용도로 OID가 사용되지 않도록 합니다. OID 레지스트리는 설명을 포함하여 디렉터리 스키마에 사용되는 모든 OID의 목록입니다. OID 레지스트리를 사용자 지정 스키마와 함께 게시합니다.
4. 새 속성을 정의합니다.
5. 새 속성을 포함하는 개체 클래스를 정의합니다. 그러나 기본 스키마를 업데이트하지 마십시오. 새 특성을 생성하는 경우 항상 사용자 지정 오브젝트 클래스에 추가합니다.

인스턴스가 시작될 때 Directory Server가 스키마를 로드합니다. 새 스키마 파일을 로드하려면 인스턴스를 다시 시작하거나 다시 로드 작업을 시작합니다.

Directory Server 스키마를 사용자 정의할 때 다음 규칙을 고려하십시오.

- 스키마를 최대한 간단하게 유지합니다.
- 가능한 경우 기존 스키마 요소를 재사용합니다.
- 각 오브젝트 클래스에 정의된 필수 특성 수를 최소화합니다.
- 동일한 목적을 위해 둘 이상의 오브젝트 클래스 또는 속성을 정의하지 마십시오.
- 특성 또는 개체 클래스의 기존 정의를 수정하지 마십시오.

주의

다른 디렉터리 또는 LDAP 클라이언트 애플리케이션과의 호환성 문제가 발생하지 않도록 표준 스키마를 업데이트하거나 삭제하지 마십시오.

1.2. DIRECTORY SERVER가 복제 환경의 스키마 업데이트를 관리하는 방법

cn=schema 트리에서 디렉터리 스키마를 업데이트하면 Directory Server는 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 파일의 변경 사항(CSN)에 변경 사항을 저장합니다.

Directory Server는 스키마 변경 사항을 다른 복제본에 직접 복제하지 않습니다. 스키마 복제는 복제된 트리에서 디렉터리 콘텐츠가 업데이트되면 시작됩니다. 예를 들어 스키마를 수정한 후 사용자를 업데이트하는 경우 공급자는 **nsSchemaCSN** 속성에 저장된 CSN을 소비자의 속성과 비교합니다. 소비자의 **nsSchemaCSN** 속성 값이 공급자의 nsSchemaCSN 속성 값보다 작으면 Directory Server는 스키마를 소비자에게 복제합니다. 복제에 성공하려면 공급자의 모든 오브젝트 클래스 및 특성 유형이 소비자 정의의 상위 세트여야 합니다.

예 1.1. 스키마 하위 집합 및 슈퍼셋

- **server1** 에서 **예제** 오브젝트 클래스는 **a1**,**a2** 및 **a3** 특성을 허용합니다.
- **server2** 에서 **예제** 오브젝트 클래스는 **a1** 및 **a3** 속성을 허용합니다.

이전 예에서 **server1** 의 **예제** 오브젝트 클래스의 스키마 정의는 **server2** 의 오브젝트 클래스의 상위 세트입니다. 검증 단계에서 Directory Server가 스키마를 복제하거나 수락할 때 서버는 슈퍼 세트 정의를 검색합니다. 예를 들어, 소비자가 로컬 스키마의 오브젝트 클래스가 공급자 스키마의 오브젝트 클래스보다 적은 속성을 허용하는 것을 감지하면 Directory Server는 로컬 스키마를 업데이트합니다.

스키마 정의가 성공적으로 복제되면 **nsSchemaCSN** 특성이 서버 및 오브젝트 클래스 및 속성 유형과 같은 스키마 정의에서 더 이상 복제 세션 시작 시 비교되지 않습니다.

다음 시나리오에서는 Directory Server가 스키마를 복제하지 않습니다.

- 한 호스트의 스키마는 다른 호스트의 스키마의 하위 집합입니다.
  예를 들어 **server2** 의 **예제** 오브젝트 클래스의 스키마 정의는 **server1** 의 오브젝트 클래스의 하위 집합입니다. 하위 집합은 속성에 대해 발생할 수도 있습니다(단일 값 속성은 다중 값 속성의 하위 집합임) 및 특성 구문입니다.
- 공급업체 스키마 및 소비자 스키마의 정의를 병합해야 하는 경우
- Directory Server는 병합 스키마를 지원하지 않습니다. 예를 들어 한 서버의 오브젝트 클래스가 **a1**,**a2** 및 **a3** 특성과 **a1**, a3 ,**a3**, **a4** 를 허용하는 경우 스키마는 하위 집합이 아니며 병합할 수 없습니다.
- **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 이외의 스키마 파일을 사용합니다.
  Directory Server를 사용하면 **/etc/dirsrv/slapd-instance_name/schema/** 디렉터리에 스키마 파일을 추가할 수 있습니다. 그러나 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 파일의 CSN만 업데이트됩니다. 이러한 이유로 다른 스키마 파일은 로컬에서만 사용되며 복제 파트너로 자동 전송되지 않습니다.

중요

Directory Server에서 스키마를 자동으로 복제하고 중복 스키마 정의를 방지하기 위해 사용자 지정 스키마를 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 파일에 저장합니다.

1.3. DSCONF를 사용하여 특성 및 오브젝트 클래스에 대한 사용자 지정 스키마 생성

다음 절차에서는 **dsconf** 유틸리티를 사용하여 사용자 지정 스키마를 생성하는 방법을 설명합니다.

- OID **2.16.840.1.1133730.2.1.123** 및 구문 디렉터리 **문자열** (OID **1.3.6.1.4.1.1466.115.121.1.15**)을 사용하는 **dateOfBirth** 라는 단일 값 속성
- 상위 오브젝트 클래스(**SUP top**)가 없는 **example person** 이라는 오브젝트 클래스, **dateOfBirth** 속성을 포함해야 하는 OID **2.16.840.1.1133730.2.1.99** 라는 오브젝트 클래스입니다.

절차

1. **dateOfBirth** 특성을 생성합니다.

```
# 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** 오브젝트 클래스를 생성합니다.

```
# 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. 스키마 다시 로드 작업을 실행합니다.

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

검증

- **/var/log/dirsrv/slapd-*instance_name*/errors** 파일을 모니터링합니다.
  - 빌드에 성공하면 Directory Server 로그:

```
[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.
```

  - 빌드에 실패하면 Directory Server에서 어떤 단계가 실패했는지와 이유를 기록합니다.

2장. 웹 콘솔을 사용하여 사용자 정의 스키마 생성

스키마를 확장하여 사용자 지정 특성 및 개체 클래스를 Directory Server에 추가할 수 있습니다. 스키마를 확장할 수 있습니다.

- 디렉터리 서버 웹 콘솔 사용. 이 프로세스는 이 섹션에 설명되어 있습니다.
- 명령줄에서 dsconf 유틸리티 사용.
- 스키마 파일을 생성하여 수동으로.

2.1. 스키마 확장 워크플로

새 스키마 요소를 추가하려면 다음이 필요합니다.

1. 새 스키마의 고유 OID(오브젝트 식별자)를 계획하고 정의합니다. Directory Server는 OID로 스키마 요소를 인식하지만 OID를 수동으로 관리해야 합니다.
   OID는 서버의 스키마 요소를 식별하는 점으로 구분된 번호입니다. OID는 다양한 분기를 수용하도록 확장할 수 있는 기본 OID를 사용하여 계층화할 수 있습니다. 예를 들어 기본 OID는 **1** 일 수 있으며 **1.1** 에서 속성과 **1.2** 의 오브젝트 클래스에 대한 분기가 있을 수 있습니다.

   중요

   필요하지 않은 경우에도 Red Hat은 사용자 정의 스키마에 숫자 OID를 사용하여 호환성과 성능을 향상시키는 것이 좋습니다.

2. INA(Internet Assigned Numbers Authority)에서 OID를 요청합니다. 자세한 내용은 https://pen.iana.org/pen/PenApplication.page 을 참조하십시오.
3. OID 레지스트리를 생성하여 OID 할당을 추적하고 둘 이상의 용도로 OID가 사용되지 않도록 합니다. OID 레지스트리는 설명을 포함하여 디렉터리 스키마에 사용되는 모든 OID의 목록입니다. OID 레지스트리를 사용자 지정 스키마와 함께 게시합니다.
4. 새 속성을 정의합니다.
5. 새 속성을 포함하는 개체 클래스를 정의합니다. 그러나 기본 스키마를 업데이트하지 마십시오. 새 특성을 생성하는 경우 항상 사용자 지정 오브젝트 클래스에 추가합니다.

인스턴스가 시작될 때 Directory Server가 스키마를 로드합니다. 새 스키마 파일을 로드하려면 인스턴스를 다시 시작하거나 다시 로드 작업을 시작합니다.

Directory Server 스키마를 사용자 정의할 때 다음 규칙을 고려하십시오.

- 스키마를 최대한 간단하게 유지합니다.
- 가능한 경우 기존 스키마 요소를 재사용합니다.
- 각 오브젝트 클래스에 정의된 필수 특성 수를 최소화합니다.
- 동일한 목적을 위해 둘 이상의 오브젝트 클래스 또는 속성을 정의하지 마십시오.
- 특성 또는 개체 클래스의 기존 정의를 수정하지 마십시오.

주의

다른 디렉터리 또는 LDAP 클라이언트 애플리케이션과의 호환성 문제가 발생하지 않도록 표준 스키마를 업데이트하거나 삭제하지 마십시오.

2.2. DIRECTORY SERVER가 복제 환경의 스키마 업데이트를 관리하는 방법

cn=schema 트리에서 디렉터리 스키마를 업데이트하면 Directory Server는 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 파일의 변경 사항(CSN)에 변경 사항을 저장합니다.

Directory Server는 스키마 변경 사항을 다른 복제본에 직접 복제하지 않습니다. 스키마 복제는 복제된 트리에서 디렉터리 콘텐츠가 업데이트되면 시작됩니다. 예를 들어 스키마를 수정한 후 사용자를 업데이트하는 경우 공급자는 **nsSchemaCSN** 속성에 저장된 CSN을 소비자의 속성과 비교합니다. 소비자의 **nsSchemaCSN** 속성 값이 공급자의 nsSchemaCSN 속성 값보다 작으면 Directory Server는 스키마를 소비자에게 복제합니다. 복제에 성공하려면 공급자의 모든 오브젝트 클래스 및 특성 유형이 소비자 정의의 상위 세트여야 합니다.

예 2.1. 스키마 하위 집합 및 슈퍼셋

- **server1** 에서 **예제** 오브젝트 클래스는 **a1**,**a2** 및 **a3** 특성을 허용합니다.
- **server2** 에서 **예제** 오브젝트 클래스는 **a1** 및 **a3** 속성을 허용합니다.

이전 예에서 **server1** 의 **예제** 오브젝트 클래스의 스키마 정의는 **server2** 의 오브젝트 클래스의 상위 세트입니다. 검증 단계에서 Directory Server가 스키마를 복제하거나 수락할 때 서버는 슈퍼 세트 정의를 검색합니다. 예를 들어, 소비자가 로컬 스키마의 오브젝트 클래스가 공급자 스키마의 오브젝트 클래스보다 적은 속성을 허용하는 것을 감지하면 Directory Server는 로컬 스키마를 업데이트합니다.

스키마 정의가 성공적으로 복제되면 **nsSchemaCSN** 특성이 서버 및 오브젝트 클래스 및 속성 유형과 같은 스키마 정의에서 더 이상 복제 세션 시작 시 비교되지 않습니다.

다음 시나리오에서는 Directory Server가 스키마를 복제하지 않습니다.

- 한 호스트의 스키마는 다른 호스트의 스키마의 하위 집합입니다.
  예를 들어 **server2** 의 **예제** 오브젝트 클래스의 스키마 정의는 **server1** 의 오브젝트 클래스의 하위 집합입니다. 하위 집합은 속성에 대해 발생할 수도 있습니다(단일 값 속성은 다중 값 속성의 하위 집합임) 및 특성 구문입니다.
- 공급업체 스키마 및 소비자 스키마의 정의를 병합해야 하는 경우
- Directory Server는 병합 스키마를 지원하지 않습니다. 예를 들어 한 서버의 오브젝트 클래스가 **a1**,**a2** 및 **a3** 특성과 **a1**, a3 ,**a3**, **a4** 를 허용하는 경우 스키마는 하위 집합이 아니며 병합할 수 없습니다.
- **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 이외의 스키마 파일을 사용합니다.
  Directory Server를 사용하면 **/etc/dirsrv/slapd-instance_name/schema/** 디렉터리에 스키마 파일을 추가할 수 있습니다. 그러나 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 파일의 CSN만 업데이트됩니다. 이러한 이유로 다른 스키마 파일은 로컬에서만 사용되며 복제 파트너로 자동 전송되지 않습니다.

중요

Directory Server에서 스키마를 자동으로 복제하고 중복 스키마 정의를 방지하기 위해 사용자 지정 스키마를 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 파일에 저장합니다.

2.3. 웹 콘솔을 사용하여 특성 및 오브젝트 클래스에 대한 사용자 정의 스키마 생성

다음 절차에서는 웹 콘솔을 사용하여 다음과 같이 사용자 정의 스키마를 생성하는 방법을 보여줍니다.

- OID **2.16.840.1.1133730.2.1.123** 및 구문 디렉터리 **문자열** (OID **1.3.6.1.4.1.1466.115.121.1.15**)을 사용하는 **dateOfBirth** 라는 단일 값 속성
- 상위 오브젝트 클래스가 없는 **example person** (**SUP top**), OID **2.16.840.1.1133730.2.1.99** 라는 오브젝트 클래스로 **dateOfBirth** 특성을 포함해야 합니다.

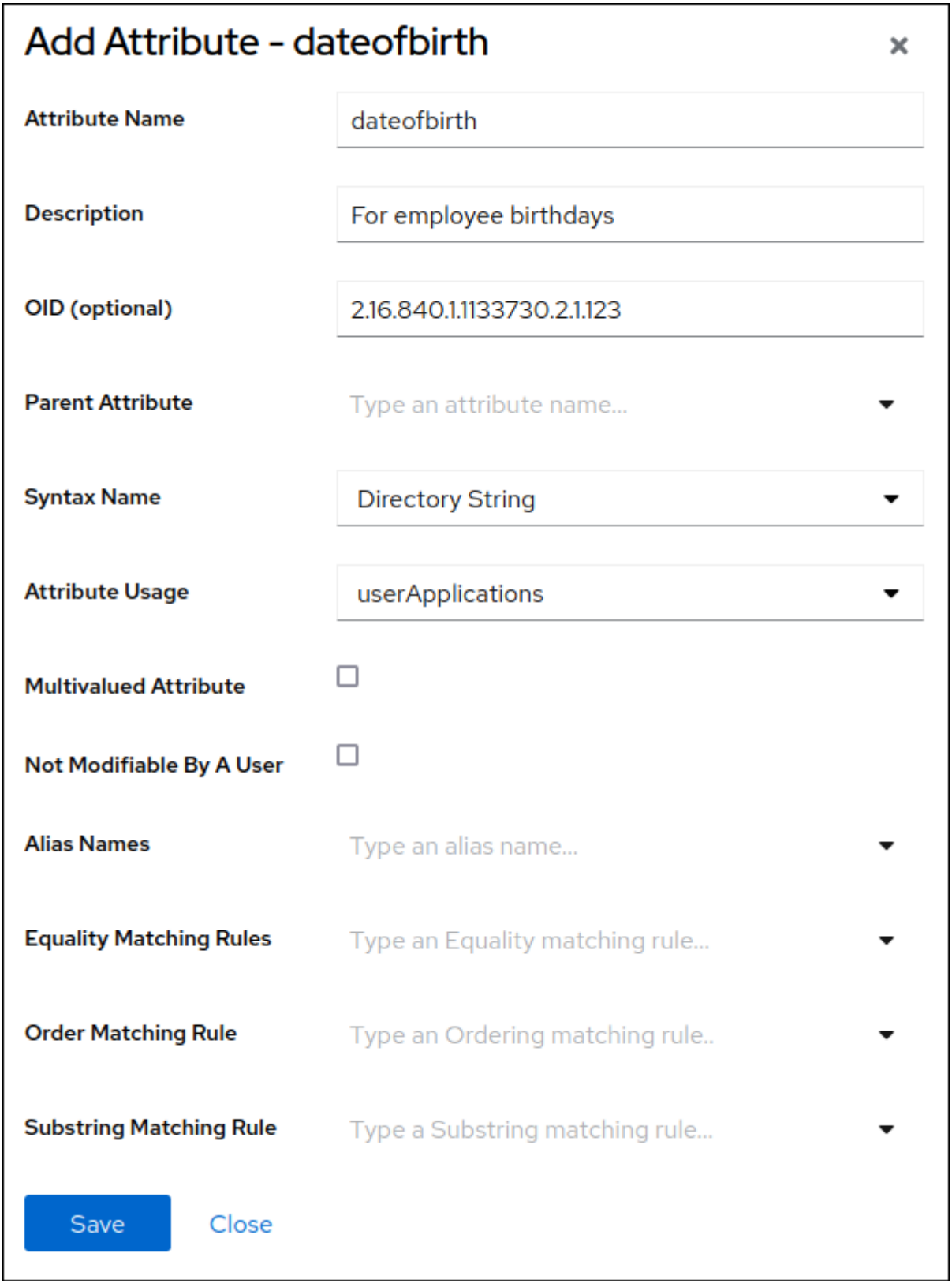
웹 콘솔을 사용하여 스키마를 업데이트하면 디렉터리 서버가 스키마를 자동으로 다시 로드합니다.

사전 요구 사항

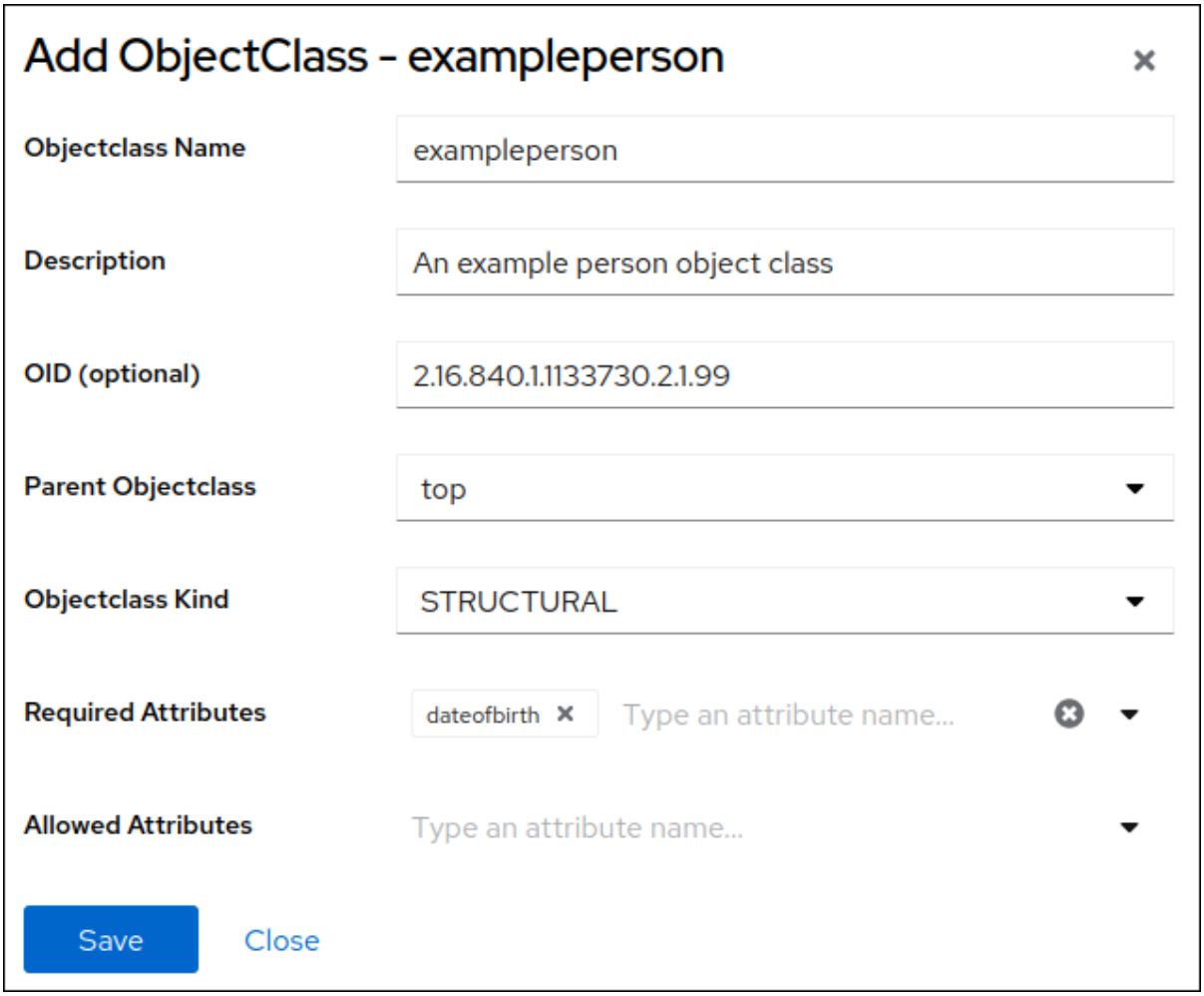
- 웹 콘솔에서 인스턴스에 로그인되어 있습니다.

절차

1. **Schema+ 속성** 를 클릭합니다.
2. 추가할 속성의 설정을 입력합니다.



3. **저장**을클릭합니다.
4. **Schema → Objectclasses** 로 이동하여 **ObjectClass 추가** 를 클릭합니다.
5. 추가할 오브젝트 클래스의 설정을 입력합니다.



6. **저장**을클릭합니다.

검증

- **Monitoring → Logging → Errors Log** 로 이동합니다.
  - 빌드에 성공하면 Directory Server 로그:

    ```
    [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.
    ```

  - 빌드에 실패하면 Directory Server에서 어떤 단계가 실패했는지와 이유를 기록합니다.

3장. 수동으로 사용자 지정 스키마 파일 생성

스키마를 확장하여 사용자 지정 특성 및 개체 클래스를 Directory Server에 추가할 수 있습니다. 스키마를 확장할 수 있습니다.

- 스키마 파일을 생성하여 수동으로 수행합니다. 프로세스는 이 섹션에 설명되어 있습니다.
- 명령줄에서 dsconf 유틸리티 사용.
- 디렉터리 서버 웹 콘솔 사용.

3.1. 스키마 확장 워크플로

새 스키마 요소를 추가하려면 다음이 필요합니다.

1. 새 스키마의 고유 OID(오브젝트 식별자)를 계획하고 정의합니다. Directory Server는 OID로 스키마 요소를 인식하지만 OID를 수동으로 관리해야 합니다.
   OID는 서버의 스키마 요소를 식별하는 점으로 구분된 번호입니다. OID는 다양한 분기를 수용하도록 확장할 수 있는 기본 OID를 사용하여 계층화할 수 있습니다. 예를 들어 기본 OID는 **1** 일 수 있으며 **1.1** 에서 속성과 **1.2** 의 오브젝트 클래스에 대한 분기가 있을 수 있습니다.

   중요

   필요하지 않은 경우에도 Red Hat은 사용자 정의 스키마에 숫자 OID를 사용하여 호환성과 성능을 향상시키는 것이 좋습니다.

2. INA(Internet Assigned Numbers Authority)에서 OID를 요청합니다. 자세한 내용은 https://pen.iana.org/pen/PenApplication.page 을 참조하십시오.
3. OID 레지스트리를 생성하여 OID 할당을 추적하고 둘 이상의 용도로 OID가 사용되지 않도록 합니다. OID 레지스트리는 설명을 포함하여 디렉터리 스키마에 사용되는 모든 OID의 목록입니다. OID 레지스트리를 사용자 지정 스키마와 함께 게시합니다.
4. 새 속성을 정의합니다.
5. 새 속성을 포함하는 개체 클래스를 정의합니다. 그러나 기본 스키마를 업데이트하지 마십시오. 새 특성을 생성하는 경우 항상 사용자 지정 오브젝트 클래스에 추가합니다.

인스턴스가 시작될 때 Directory Server가 스키마를 로드합니다. 새 스키마 파일을 로드하려면 인스턴스를 다시 시작하거나 다시 로드 작업을 시작합니다.

Directory Server 스키마를 사용자 정의할 때 다음 규칙을 고려하십시오.

- 스키마를 최대한 간단하게 유지합니다.
- 가능한 경우 기존 스키마 요소를 재사용합니다.
- 각 오브젝트 클래스에 정의된 필수 특성 수를 최소화합니다.
- 동일한 목적을 위해 둘 이상의 오브젝트 클래스 또는 속성을 정의하지 마십시오.
- 특성 또는 개체 클래스의 기존 정의를 수정하지 마십시오.

주의

다른 디렉터리 또는 LDAP 클라이언트 애플리케이션과의 호환성 문제가 발생하지 않도록 표준 스키마를 업데이트하거나 삭제하지 마십시오.

3.2. 스키마 파일의 요구 사항

스키마 파일은 **cn=schema** 항목을 정의하는 LDIF 형식을 사용합니다. 각 특성 유형 및 오브젝트 클래스는 이 항목에 추가됩니다.

스키마 파일에 대한 요구 사항은 다음과 같습니다.

- 파일은 다음 항목으로 시작해야 합니다.

  ```
  dn: cn=schema
  ```

- 스키마 파일에는 특성 유형 또는 오브젝트 클래스 또는 둘 다 포함할 수 있습니다.
- 오브젝트 클래스 정의는 다른 스키마 파일에 정의된 특성을 사용할 수 있습니다.
- 사용자 지정 스키마 파일을 사용해야 하는 인스턴스에 따라 다음 위치 중 하나에 저장합니다.
  - **/etc/dirsrv/slapd-*instance_name*/schema/** 를 사용하여 이 특정 인스턴스에서 스키마 파일을 사용할 수 있도록 합니다.
  - **/usr/share/dirsrv/schema/** 를 사용하여 이 호스트에서 실행 중인 모든 인스턴스에서 스키마 파일을 사용할 수 있도록 합니다.
- 기본적으로 Directory Server는 **99user.ldif** 파일에서 사용자 지정 스키마를 예상합니다. 다른 파일 이름을 사용하는 경우:
  - 이름은 **99user.ldif** 보다 알파벳순이어야 합니다. 예를 들어 **99aaa.ldif** 는 ok이지만 **99zzz.ldif** 는 그렇지 않습니다.
  - 이름은 두 자리 숫자로 시작해야 하며 **01** 보다 커야 합니다. 사용자 정의 스키마 파일은 **00** 부터 **98**으로 시작하는 코어 스키마 파일 뒤에 로드되어야 하기 때문입니다.
    Directory Server는 스키마 파일을 알파벳순으로 읽습니다. 따라서 예를 들어 정의 **99user.ldif** 를 저장하는 경우 이름이 **00** 및 **01** 로 시작하는 표준 파일의 정의를 재정의합니다.
- **/usr/share/dirsrv/data/** 디렉터리의 표준 스키마 파일을 사용하려면 파일을 **/etc/dirsrv/slapd-*instance_name*/schema/** 또는 **/usr/share/dirsrv/schema/** 에 복사하십시오. 그러나 대상 디렉터리에서 다른 파일 이름을 사용합니다. 그렇지 않으면 Directory Server는 업그레이드 중에 파일의 이름을 바꾸고 **.bak** 접미사를 추가합니다.

예 3.1. 사용자 정의 스키마 파일의 예

```
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. 사용자 지정 스키마 파일의 속성 정의

스키마 파일에서 속성을 **attributeTypes** 속성 값으로 정의합니다.

예 3.2. 속성 정의

```
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' )
```

속성 정의에는 다음 구성 요소가 포함됩니다.

- 점으로 구분된 숫자로 지정된 고유 OID(오브젝트 식별자)입니다.
- **NAME *attribute_name*형식의 고유 이름입니다**.
- **DESC 설명 형식의 *설명***
- **SYNTAX OID** 형식의 속성 값의 구문에 대한 *OID*입니다. LDAP 특성 구문에 대한 자세한 내용은 RFC 4517 을 참조하십시오.
- 선택 사항: 속성이 정의된 소스입니다.

3.4. 사용자 지정 스키마 파일의 오브젝트 클래스 정의

스키마 파일에서 오브젝트 클래스를 **objectClasses** 속성의 값으로 정의합니다.

예 3.3. 오브젝트 클래스 정의

```
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' )
```

오브젝트 클래스 정의에는 다음 구성 요소가 포함됩니다.

- 점으로 구분된 숫자로 지정된 고유 OID(오브젝트 식별자)입니다.
- **NAME *attribute_name*형식의 고유 이름입니다**.
- **DESC 설명 형식의 *설명***
- **SUP *object_class*형식의 이 오브젝트 클래스에 대한 우수성(parent) 오브젝트 클래스입니다**. 관련 부모가 없는 경우 **SUP top** 을 사용합니다.
- **STRUCTURAL** 이라는 단어는 오브젝트 클래스가 적용되는 항목 유형을 정의합니다. 모든 항목은 하나 이상의 **STRUCTURAL** 개체 클래스에 속해야 합니다. **A CryostatILIARY** 는 모든 입력에 적용할 수 있음을 의미합니다.

- essential 키워드 앞에 있는 필수 속성 목록입니다. 여러 속성을 포함하려면 그룹을 괄호로 묶고 [command]'$ '(dollar 기호 및 공백)로 특성을 구분합니다.

- MAY 키워드 앞에 있는 선택적 속성 목록입니다. 여러 속성을 포함하려면 그룹을 괄호로 묶고 [command]'$ '(dollar 기호 및 공백)로 특성을 구분합니다.

이름 및 OID만 필요하며 기타 설정은 오브젝트 클래스의 요구에 따라 다릅니다.

추가 리소스

- RFC 4512의 섹션 4.2

3.5. DIRECTORY SERVER가 복제 환경의 스키마 업데이트를 관리하는 방법

cn=schema 트리에서 디렉터리 스키마를 업데이트하면 Directory Server는 /etc/dirsrv/slapd-*instance_name*/schema/99user.ldif 파일의 변경 사항(CSN)에 변경 사항을 저장합니다.

Directory Server는 스키마 변경 사항을 다른 복제본에 직접 복제하지 않습니다. 스키마 복제는 복제된 트리에서 디렉터리 콘텐츠가 업데이트되면 시작됩니다. 예를 들어 스키마를 수정한 후 사용자를 업데이트하는 경우 공급자는 nsSchemaCSN 속성에 저장된 CSN을 소비자의 속성과 비교합니다. 소비자의 nsSchemaCSN 속성 값이 공급자의 nsSchemaCSN 속성 값보다 작으면 Directory Server는 스키마를 소비자에게 복제합니다. 복제에 성공하려면 공급자의 모든 오브젝트 클래스 및 특성 유형이 소비자 정의의 상위 세트여야 합니다.

> 예 3.4. 스키마 하위 집합 및 슈퍼셋
>
> - server1 에서 예제 오브젝트 클래스는 a1,a2 및 a3 특성을 허용합니다.
>
> - server2 에서 예제 오브젝트 클래스는 a1 및 a3 속성을 허용합니다.

이전 예에서 server1 의 예제 오브젝트 클래스의 스키마 정의는 server2 의 오브젝트 클래스의 상위 세트입니다. 검증 단계에서 Directory Server가 스키마를 복제하거나 수락할 때 서버는 슈퍼 세트 정의를 검색합니다. 예를 들어, 소비자가 로컬 스키마의 오브젝트 클래스가 공급자 스키마의 오브젝트 클래스보다 적은 속성을 허용하는 것을 감지하면 Directory Server는 로컬 스키마를 업데이트합니다.

스키마 정의가 성공적으로 복제되면 **nsSchemaCSN** 특성이 서버 및 오브젝트 클래스 및 속성 유형과 같은 스키마 정의에서 더 이상 복제 세션 시작 시 비교되지 않습니다.

다음 시나리오에서는 **Directory Server**가 스키마를 복제하지 않습니다.

- 한 호스트의 스키마는 다른 호스트의 스키마의 하위 집합입니다.

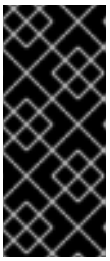
  예를 들어 **server2** 의 예제 오브젝트 클래스의 스키마 정의는 **server1** 의 오브젝트 클래스의 하위 집합입니다. 하위 집합은 속성에 대해 발생할 수도 있습니다(단일 값 속성은 다중 값 속성의 하위 집합임) 및 특성 구문입니다.

- 공급업체 스키마 및 소비자 스키마의 정의를 병합해야 하는 경우

- **Directory Server**는 병합 스키마를 지원하지 않습니다. 예를 들어 한 서버의 오브젝트 클래스가 **a1,a2** 및 **a3** 특성과 **a1, a3 ,a3, a4** 를 허용하는 경우 스키마는 하위 집합이 아니며 병합할 수 없습니다.

- **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 이외의 스키마 파일을 사용합니다.

  Directory Server를 사용하면 **/etc/dirsrv/slapd-instance_name/schema/** 디렉터리에 스키마 파일을 추가할 수 있습니다. 그러나 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 파일의 **CSN**만 업데이트됩니다. 이러한 이유로 다른 스키마 파일은 로컬에서만 사용되며 복제 파트너로 자동 전송되지 않습니다.



중요

Directory Server에서 스키마를 자동으로 복제하고 중복 스키마 정의를 방지하기 위해 사용자 지정 스키마를 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 파일에 저장합니다.

3.6. 특성 및 개체 클래스에 대한 사용자 지정 스키마 파일을 수동으로 생성

사용자 지정 스키마를 수동으로 만들려면 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 파일에 저장합니다. 다른 파일 이름을 사용할 수 있지만 다른 파일에 저장된 스키마 정의와 같은 단점이 복제되지만 복제본의 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 에 저장됩니다. **Directory Server**가 복제 환경에서 스키마 업데이트를 관리하는 방법을 참조하십시오.

이 절차에서는 다음을 추가합니다.

- OID 2.16.840.1.1133730.2.1.123 및 구문 디렉터리 문자열 (OID 1.3.6.1.4.1.1466.115.121.1.15)을 사용하는 dateOfBirth 라는 단일 값 속성
- dateOfBirth 특성을 포함해야 하는 상위 오브젝트 클래스(SUP top)가 없는 example person 이라는 오브젝트 클래스입니다.

절차

1. /etc/dirsrv/slapd-*instance_name*/schema/99user.ldif 파일에 dn: cn=schema 항목 아래에 다음 내용을 추가합니다.

```
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. 스키마 다시 로드 작업을 실행합니다.

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

검증 단계:

- /var/log/dirsrv/slapd-*instance_name*/errors 파일을 모니터링합니다.
  - 빌드에 성공하면 Directory Server 로그:

```
[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.
```

  - 빌드에 실패하면 Directory Server에서 어떤 단계가 실패했는지와 이유를 기록합니다.

4장. 기존 속성 값의 구문 검증

구문 유효성 검사를 통해 **Directory Server**는 특성 값이 해당 특성 정의에 제공된 구문의 규칙을 따르는지 확인합니다. **Directory Server**는 구문 검증 작업 결과를 **/var/log/dirsrv/slapd-instance_name/errors** 파일에 기록합니다.

다음과 같은 경우 수동 구문 검증이 필요합니다.

- **nsslapd-syntaxcheck** 매개변수에서 구문 검증이 비활성화되어 있습니다.

  참고

  Red Hat은 구문 검증을 비활성화해서는 안 됩니다.

- 비활성화되거나 구문 검증이 없는 서버에서 데이터를 마이그레이션합니다.

4.1. DSCONF SCHEMA VALIDATE-SYNTAX 명령을 사용하여 구문 검증 작업 생성

dsconf schema validate-syntax 명령을 사용하면 구문 검증 작업을 생성하여 수정된 모든 특성을 확인하고 새 값에 필수 구문이 있는지 확인할 수 있습니다.

절차

- 구문 검증 작업을 생성하려면 다음을 입력합니다.

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

  예제 출력에서 명령은 **(objectclass=inetorgperson)** 필터와 일치하는 **ou=People,dc=example,dc=com** 하위 트리의 모든 값의 구문을 확인하는 작업을 생성합니다.

4.2. CN 작업 항목을 사용하여 구문 검증 작업 생성

Directory Server 구성의 **cn=tasks,cn=config** 항목은 작업을 관리하는 데 서버에서 사용하는 임시 항목의 컨테이너 항목입니다. **cn=syntax validate,cn=tasks,cn=config** 항목에 작업을 생성하여 구문 검증

작업을 시작할 수 있습니다.

절차

- 구문 검증 작업을 시작하려면 다음과 같이 **cn=syntax validate,cn=tasks,cn=config** 항목에 작업을 생성합니다.

```
# 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)
```

예제 출력에서 명령은 **(objectclass=inetorgperson)** 필터와 유사한 **ou=People,dc=example,dc=com** 하위 트리의 모든 값의 구문을 확인하는 작업을 생성합니다. 작업이 완료되면 **Directory Server**가 디렉터리 구성에서 항목을 삭제합니다.

추가 리소스

- 구성 및 스키마 참조