



Red Hat Directory Server 12

管理目录模式

创建和管理自定义模式

创建和管理自定义模式

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

您可以通过添加自定义模式（使用 `dsconf` 工具和 Web 控制台创建）来将额外的数据存储于目录服务器中。您还可以扩展架构，并验证现有属性值的语法。

目录

对红帽文档提供反馈	3
第 1 章 使用 DSCONF 工具创建自定义模式	4
1.1. 模式扩展的工作流	4
1.2. 目录服务器如何在复制环境中管理模式更新	5
1.3. 使用 DSCONF 为属性和对象类创建自定义 SCHEMA	6
第 2 章 使用 WEB 控制台创建自定义模式	7
2.1. 模式扩展的工作流	7
2.2. 目录服务器如何在复制环境中管理模式更新	8
2.3. 使用 WEB 控制台为属性和对象类创建自定义 SCHEMA	9
第 3 章 手动创建自定义模式文件	12
3.1. 模式扩展的工作流	12
3.2. 架构文件的要求	13
3.3. 自定义模式文件中的属性定义	13
3.4. 自定义模式文件中的对象类定义	14
3.5. 目录服务器如何在复制环境中管理模式更新	15
3.6. 为属性和对象类手动创建自定义架构文件	15
第 4 章 验证现有属性值的语法	17
4.1. 使用 DSCONF 模式 VALIDATE-SYNTAX 命令创建语法验证任务	17
4.2. 使用 CN 任务条目创建语法验证任务	17

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请让我们了解如何改进文档。要做到这一点：

- 要通过 JIRA 提交反馈（需要帐户）：
 1. 登录到 [Jira](#) 网站。
 2. 在顶部导航栏中点 **Create**
 3. 在 **Summary** 字段中输入描述性标题。
 4. 在 **Description** 字段中输入您对改进的建议。包括到文档相关部分的链接。
 5. 点对话框底部的 **Create**。
- 要通过 Bugzilla 提交反馈（需要帐户）：
 1. 进入 [Bugzilla](#) 网站。
 2. 在 Component 中选择 **Documentation**。
 3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
 4. 点 **Submit Bug**。

第 1 章 使用 DSCONF 工具创建自定义模式

您可以通过扩展 schema 将自定义属性和对象类添加到目录服务器中。您可以扩展模式：

- 在命令行中使用 **dsconf** 工具。本节描述了此过程。
- 使用 [Directory Server Web 控制台](#)。
- [通过创建架构文件 手动](#)。

1.1. 模式扩展的工作流

添加新架构元素需要：

1. 为新架构规划和定义唯一对象标识符(OID)。目录服务器通过其 OID 识别模式元素，但您必须手动管理 OID。
OID 是一个点分隔的数字，用于标识到服务器的 schema 元素。OID 可以使用基础 OID 进行分层，以容纳不同的分支。例如，基础 OID 可以是 1，在 1.1 上可以有一个属性的分支，以及在 1.2 中的对象类。
-
- 重要**
- 即使不需要，红帽建议将数字 OID 用于自定义模式，以更好地转发兼容性和性能。
2. 从互联网编号分配机构(IANA)请求 OID。详情请查看 <https://pen.iana.org/pen/PenApplication.page>。
 3. 创建 OID registry 来跟踪 OID 分配，并确保没有 OID 用于多个目的。OID registry 是目录 schema 中使用的所有 OID 列表，包括描述。使用自定义模式发布 OID registry。
 4. 定义新属性。
 5. 定义包含新属性的对象类。但是，永远不会更新默认模式。如果创建新属性，请始终将它们添加到自定义对象类中。

目录服务器在实例启动时加载架构。要加载新的架构文件，重启实例或启动重新加载任务。

在自定义目录服务器模式时请注意以下规则：

- 使架构尽可能简单。
- 尽可能重复使用现有的架构元素。
- 尽可能减少为每个对象类定义的必要属性数量。
- 不要为相同的目的定义多个对象类或属性。
- 不要修改属性或对象类的任何现有定义。

**警告**

不要更新或删除标准模式，以避免与其他目录或 LDAP 客户端应用程序的兼容性问题。

1.2. 目录服务器如何在复制环境中管理模式更新

当您更新 **cn=schema** 树中的目录模式时，Directory 服务器会将更改存储在 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 文件中，包括更改状态号(CSN)。

目录服务器不会直接将模式更改复制到其他副本。当复制树中更新目录内容时，架构复制将开始。例如，如果您在修改 schema 后更新用户，则供应商会将存储在 **nsSchemaCSN** 属性中的 CSN 与消费者上的 CSN 进行比较。如果消费者上的 **nsSchemaCSN** 属性的值低于供应商上的 **nsSchemaCSN** 属性的值，则目录服务器会将模式复制到消费者。对于成功复制，供应商上的所有对象类和属性类型都必须是消费者定义的超集。

例 1.1. 模式子集和超集

- 在 **server1** 上，**示例** 对象类允许 **a1**、**a2** 和 **a3** 属性。
- 在 **server2** 上，**示例** 对象类允许 **a1** 和 **a3** 属性。

在上例中，**server1** 上 **示例** 对象类的 schema 定义是 **server2** 上对象类的超集。在验证阶段，当目录服务器复制或接受 schema 时，服务器会检索 superset 定义。例如，如果消费者检测到本地 schema 中的对象类允许的属性小于供应商模式中的对象类，Directory 服务器会更新本地 schema。

如果架构定义被成功复制，则 **nsSchemaCSN** 属性在服务器和架构定义（如对象类和属性类型）上相同，也不会复制会话开始时进行比较。

在以下场景中，目录服务器不会复制模式：

- 一个主机上的 schema 是另一主机的模式的子集。
例如，**server2** 上 **示例** 对象类的架构定义是 **server1** 上对象类的子集。子集也可以为属性（单值属性是多值属性的子集）和属性语法发生。
- 当供应商模式和消费者模式中的定义需要合并时。
- 目录服务器不支持合并模式。例如，如果一个服务器上的对象类允许 **a1**、**a2** 和 **a3** 属性以及 **a1**、**a3** 和 **a4** 在另一个服务器上，则架构不是子集且不能合并。
- 您可以使用 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 以外的模式文件。
目录服务器允许您在 **/etc/dirsrv/slapd-*instance_name*/schema/** 目录中添加额外的模式文件。但是，只更新 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 文件中的 CSN。因此，其他架构文件仅在本地使用，不会自动传送到复制合作伙伴。

**重要**

要使目录服务器自动复制模式并避免重复的模式定义，请将自定义模式存储在 **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** 文件中。

1.3. 使用 DSCONF 为属性和对象类创建自定义 SCHEMA

此流程演示了如何使用 **dsconf** 工具来创建自定义模式：

- 名为 **dateOfBirth** 的单值属性，带有 OID **2.16.840.1.1133730.2.1.123** 和语法目录字符串 (OID **1.3.6.1.4.1.1466.115.121.1.15**)
- 名为 **exampleperson** 没有父对象类(SUP top), OID **2.16.840.1.1133730.2.1.99** 的对象类必须包含 **dateOfBirth** 属性。

流程

1. 创建 **dateOfBirth** 属性：

```
# 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. 创建 **exampleperson** 对象类：

```
# 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. 运行 schema reload 任务：

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

验证

- 监控 **/var/log/dirsrv/slapd-instance_name/errors** 文件：

- 如果构建成功，Directory 服务器日志：

```
[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.
```

- 如果构建失败，Directory 服务器会记录步骤失败以及原因。

第 2 章 使用 WEB 控制台创建自定义模式

您可以通过扩展 schema 将自定义属性和对象类添加到目录服务器中。您可以扩展模式：

- 使用 Directory Server Web 控制台。本节描述了此过程。
- [在命令行中使用 dsconf 工具。](#)
- [通过创建架构文件手动。](#)

2.1. 模式扩展的工作流

添加新架构元素需要：

1. 为新架构规划和定义唯一对象标识符(OID)。目录服务器通过其 OID 识别模式元素，但您必须手动管理 OID。
OID 是一个点分隔的数字，用于标识到服务器的 schema 元素。OID 可以使用基础 OID 进行分层，以容纳不同的分支。例如，基础 OID 可以是 1，在 1.1 上可以有一个属性的分支，以及在 1.2 中的对象类。



重要

即使不需要，红帽建议将数字 OID 用于自定义模式，以更好地转发兼容性和性能。

2. 从互联网编号分配机构(IANA)请求 OID。详情请查看 <https://pen.iana.org/pen/PenApplication.page>。
3. 创建 OID registry 来跟踪 OID 分配，并确保没有 OID 用于多个目的。OID registry 是目录 schema 中使用的所有 OID 列表，包括描述。使用自定义模式发布 OID registry。
4. 定义新属性。
5. 定义包含新属性的对象类。但是，永远不会更新默认模式。如果创建新属性，请始终将它们添加到自定义对象类中。

目录服务器在实例启动时加载架构。要加载新的架构文件，重启实例或启动重新加载任务。

在自定义目录服务器模式时请注意以下规则：

- 使架构尽可能简单。
- 尽可能重复使用现有的架构元素。
- 尽可能减少为每个对象类定义的必要属性数量。
- 不要为相同的目的定义多个对象类或属性。
- 不要修改属性或对象类的任何现有定义。

**警告**

不要更新或删除标准模式，以避免与其他目录或 LDAP 客户端应用程序的兼容性问题。

2.2. 目录服务器如何在复制环境中管理模式更新

当您更新 **cn=schema** 树中的目录模式时，Directory 服务器会将更改存储在 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 文件中，包括更改状态号(CSN)。

目录服务器不会直接将模式更改复制到其他副本。当复制树中更新目录内容时，架构复制将开始。例如，如果您在修改 schema 后更新用户，则供应商会将存储在 **nsSchemaCSN** 属性中的 CSN 与消费者上的 CSN 进行比较。如果消费者上的 **nsSchemaCSN** 属性的值低于供应商上的 **nsSchemaCSN** 属性的值，则目录服务器会将模式复制到消费者。对于成功复制，供应商上的所有对象类和属性类型都必须是消费者定义的超集。

例 2.1. 模式子集和超集

- 在 **server1** 上，**示例** 对象类允许 **a1**、**a2** 和 **a3** 属性。
- 在 **server2** 上，**示例** 对象类允许 **a1** 和 **a3** 属性。

在上例中，**server1** 上 **示例** 对象类的 schema 定义是 **server2** 上对象类的超集。在验证阶段，当目录服务器复制或接受 schema 时，服务器会检索 superset 定义。例如，如果消费者检测到本地 schema 中的对象类允许的属性小于供应商模式中的对象类，Directory 服务器会更新本地 schema。

如果架构定义被成功复制，则 **nsSchemaCSN** 属性在服务器和架构定义（如对象类和属性类型）上相同，也不会在复制会话开始时进行比较。

在以下场景中，目录服务器不会复制模式：

- 一个主机上的 schema 是另一主机的模式的子集。
例如，**server2** 上 **示例** 对象类的架构定义是 **server1** 上对象类的子集。子集也可以为属性（单值属性是多值属性的子集）和属性语法发生。
- 当供应商模式和消费者模式中的定义需要合并时。
- 目录服务器不支持合并模式。例如，如果一个服务器上的对象类允许 **a1**、**a2** 和 **a3** 属性以及 **a1**、**a3** 和 **a4** 在另一个服务器上，则架构不是子集且不能合并。
- 您可以使用 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 以外的模式文件。
目录服务器允许您在 **/etc/dirsrv/slapd-instance_name/schema/** 目录中添加额外的模式文件。但是，只更新 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 文件中的 CSN。因此，其他架构文件仅在本地使用，不会自动传送到复制合作伙伴。

**重要**

要使目录服务器自动复制模式并避免重复的模式定义，请将自定义模式存储在 **/etc/dirsrv/slapd-instance_name/schema/99user.ldif** 文件中。

2.3. 使用 WEB 控制台为属性和对象类创建自定义 SCHEMA

此流程演示了如何使用 Web 控制台创建自定义模式：

- 名为 **dateOfBirth** 的单值属性，带有 OID **2.16.840.1.1133730.2.1.123** 和语法目录 **字符串** (OID **1.3.6.1.4.1.1466.115.121.1.15**)
- 名为 **exampleperson** 没有父对象类(**SUP top**), OID **2.16.840.1.1133730.2.1.99** 的对象类，必须包含 **dateOfBirth** 属性

如果使用 Web 控制台更新 schema，Directory 服务器会自动重新载入 schema。

前提条件

- 在 web 控制台中登录到实例。

流程

1. 导航到 **Schema → Attributes**，再单击 **Add Attribute**。
2. 输入您要添加的属性的设置：

Add Attribute - dateofbirth ✕

Attribute Name	<input type="text" value="dateofbirth"/>
Description	<input type="text" value="For employee birthdays"/>
OID (optional)	<input type="text" value="2.16.840.1.1133730.2.1.123"/>
Parent Attribute	<input type="text" value="Type an attribute name..."/> ▼
Syntax Name	<input type="text" value="Directory String"/> ▼
Attribute Usage	<input type="text" value="userApplications"/> ▼
Multivalued Attribute	<input type="checkbox"/>
Not Modifiable By A User	<input type="checkbox"/>
Alias Names	<input type="text" value="Type an alias name..."/> ▼
Equality Matching Rules	<input type="text" value="Type an Equality matching rule..."/> ▼
Order Matching Rule	<input type="text" value="Type an Ordering matching rule.."/> ▼
Substring Matching Rule	<input type="text" value="Type a Substring matching rule..."/> ▼

3. 点 **Save**
4. 导航到 **Schema → Objectclasses**, 再单击 **Add ObjectClass**。
5. 输入您要添加的对象类的设置：

Add ObjectClass - exampleperson ✕

Objectclass Name	<input type="text" value="exampleperson"/>
Description	<input type="text" value="An example person object class"/>
OID (optional)	<input type="text" value="2.16.840.1.1133730.2.1.99"/>
Parent Objectclass	<input style="border-bottom: 1px solid #ccc;" type="text" value="top"/>
Objectclass Kind	<input style="border-bottom: 1px solid #ccc;" type="text" value="STRUCTURAL"/>
Required Attributes	<input style="border-bottom: 1px solid #ccc;" type="text" value="dateofbirth"/> ✕ <input style="border-bottom: 1px solid #ccc;" type="text" value="Type an attribute name..."/> ✕
Allowed Attributes	<input style="border-bottom: 1px solid #ccc;" type="text" value="Type an attribute name..."/>

6. 点 **Save**

验证

- 导航到 **Monitoring** → **Logging** → **Errors Log**。

- 如果构建成功，Directory 服务器日志：

```

[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.
```

- 如果构建失败，Directory 服务器会记录步骤失败以及原因。

第 3 章 手动创建自定义模式文件

您可以通过扩展 schema 将自定义属性和对象类添加到目录服务器中。您可以扩展模式：

- 通过创建架构文件手动。本节描述了这个过程。
- 在命令行中使用 `dsconf` 工具。
- 使用 Directory Server Web 控制台。

3.1. 模式扩展的工作流

添加新架构元素需要：

1. 为新架构规划和定义唯一对象标识符(OID)。目录服务器通过其 OID 识别模式元素，但您必须手动管理 OID。
OID 是一个点分隔的数字，用于标识到服务器的 schema 元素。OID 可以使用基础 OID 进行分层，以容纳不同的分支。例如，基础 OID 可以是 1，在 1.1 上可以有一个属性的分支，以及在 1.2 中的对象类。
- 
- 重要**
- 即使不需要，红帽建议将数字 OID 用于自定义模式，以更好地转发兼容性和性能。
2. 从互联网编号分配机构(IANA)请求 OID。详情请查看 <https://pen.iana.org/pen/PenApplication.page>。
 3. 创建 OID registry 来跟踪 OID 分配，并确保没有 OID 用于多个目的。OID registry 是目录 schema 中使用的所有 OID 列表，包括描述。使用自定义模式发布 OID registry。
 4. 定义新属性。
 5. 定义包含新属性的对象类。但是，永远不会更新默认模式。如果创建新属性，请始终将它们添加到自定义对象类中。

目录服务器在实例启动时加载架构。要加载新的架构文件，重启实例或启动重新加载任务。

在自定义目录服务器模式时请注意以下规则：

- 使架构尽可能简单。
- 尽可能重复使用现有的架构元素。
- 尽可能减少为每个对象类定义的必要属性数量。
- 不要为相同的目的定义多个对象类或属性。
- 不要修改属性或对象类的任何现有定义。



警告

不要更新或删除标准模式，以避免与其他目录或 LDAP 客户端应用程序的兼容性问题。

3.2. 架构文件的要求

模式文件使用 LDIF 格式来定义 **cn=schema** 条目。每个属性类型和对象类添加到此条目中。

以下是模式文件的要求：

- 该文件必须以以下条目开头：

```
dn: cn=schema
```

- 架构文件可以包含属性类型或对象类或两者。
- 对象类定义可以使用其他架构文件中定义的属性。
- 根据哪些实例应该使用自定义模式文件，将其存储在以下位置之一：
 - `/etc/dirsrv/slapd-instance_name/schema/` 使 schema 文件可供这个特定实例使用
 - `/usr/share/dirsrv/schema/` 使 schema 文件可供此主机上运行的所有实例使用
- 默认情况下，Directory 服务器需要 **99user.ldif** 文件中的自定义模式。如果您使用不同的文件名：
 - 名称必须按字母顺序低于 **99user.ldif**。例如，**99aaa.ldif** 是 ok，但 **99zzz.ldif** 不是。
 - 名称必须以两个数字开头，且大于 **01**，因为自定义模式文件必须在核心架构文件后加载，从 **00** 开始到 **98**
目录服务器以字母顺序读取模式文件。因此，例如，如果您存储了定义 **99user.ldif**，它将覆盖名称以 **00** 和 **01** 开头的标准文件中的定义。
- 如果要使用 `/usr/share/dirsrv/data/` 目录中的标准模式文件，请将文件复制到 `/etc/dirsrv/slapd-instance_name/schema/` 或 `/usr/share/dirsrv/schema/`，具体取决于哪些实例应使用该文件。但是，在目标目录中使用不同的文件名。否则，Directory 服务器会在升级过程中重命名文件，并附加 **.bak** 后缀。

例 3.1. 自定义模式文件示例

```
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. 自定义模式文件中的属性定义

您可以在 schema 文件中定义属性，作为 **attributeTypes** 属性的值。

例 3.2. 属性的定义

```
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' )
```

属性定义包含以下组件：

- 以点分开的数字指定的唯一对象标识符(OID)。
- **NAME** *attribute_name* 形式的唯一名称。
- 以 **DESC** *描述形式的描述*。
- 属性值语法的 OID，格式为 **SYNTAX** *OID*。有关 LDAP 属性语法的详情，请参阅 [RFC 4517](#)。
- 可选：定义属性的源。

3.4. 自定义模式文件中的对象类定义

您可以在架构文件中将对象类定义为 **iwles** 属性的值。

例 3.3. 对象类的定义

```
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' )
```

对象类定义包含以下组件：

- 以点分开的数字指定的唯一对象标识符(OID)。
- **NAME** *attribute_name* 形式的唯一名称。
- 以 **DESC** *描述形式的描述*。
- 此对象类的优越（父）对象类，格式为 **SUP** *object_class*。如果没有相关的父级，请使用 **SUP top**。
- **STRUCTURAL** 词语定义对象类应用到的条目类型。任何条目必须至少属于一个 **STRUCTURTURAL** 对象类。**AUXILIARY** 表示它可以应用到任何条目。
- 必要属性列表，前面带有 **MUST** 关键字。要包含多个属性，请将组用括号括起，并使用 [command]\$(dollar sign and space)分隔属性。
- 可选属性列表，前面带有 **MAY** 关键字。要包含多个属性，请将组用括号括起，并使用 [command]\$(dollar sign and space)分隔属性。

只有 name 和 OID 是必需的，其他设置则取决于对象类的需求。

其他资源

- RFC 4512 中的第 4.2 节

3.5. 目录服务器如何在复制环境中管理模式更新

当您更新 `cn=schema` 树中的目录模式时，Directory 服务器会将更改存储在 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 文件中，包括更改状态号(CSN)。

目录服务器不会直接将模式更改复制到其他副本。当复制树中更新目录内容时，架构复制将开始。例如，如果您在修改 schema 后更新用户，则供应商会将存储在 `nsSchemaCSN` 属性中的 CSN 与消费者上的 CSN 进行比较。如果消费者上的 `nsSchemaCSN` 属性的值低于供应商上的 `nsSchemaCSN` 属性的值，则目录服务器会将模式复制到消费者。对于成功复制，供应商上的所有对象类和属性类型都必须是消费者定义的超集。

例 3.4. 模式子集和超集

- 在 `server1` 上，**示例** 对象类允许 `a1`、`a2` 和 `a3` 属性。
- 在 `server2` 上，**示例** 对象类允许 `a1` 和 `a3` 属性。

在上例中，`server1` 上 **示例** 对象类的 schema 定义是 `server2` 上对象类的超集。在验证阶段，当目录服务器复制或接受 schema 时，服务器会检索 superset 定义。例如，如果消费者检测到本地 schema 中的对象类允许的属性小于供应商模式中的对象类，Directory 服务器会更新本地 schema。

如果架构定义被成功复制，则 `nsSchemaCSN` 属性在服务器和架构定义（如对象类和属性类型）上相同，也不会复制会话开始时进行比较。

在以下场景中，目录服务器不会复制模式：

- 一个主机上的 schema 是另一主机的模式的子集。
例如，`server2` 上 **示例** 对象类的架构定义是 `server1` 上对象类的子集。子集也可以为属性（单值属性是多值属性的子集）和属性语法发生。
- 当供应商模式和消费者模式中的定义需要合并时。
- 目录服务器不支持合并模式。例如，如果一个服务器上的对象类允许 `a1`、`a2` 和 `a3` 属性以及 `a1`、`a3` 和 `a4` 在另一个服务器上，则架构不是子集且不能合并。
- 您可以使用 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 以外的模式文件。
目录服务器允许您在 `/etc/dirsrv/slapd-instance_name/schema/` 目录中添加额外的模式文件。但是，只更新 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 文件中的 CSN。因此，其他架构文件仅在本地使用，不会自动传送到复制合作伙伴。



重要

要使目录服务器自动复制模式并避免重复的模式定义，请将自定义模式存储在 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 文件中。

3.6. 为属性和对象类手动创建自定义架构文件

如果要手动创建自定义模式，将其存储在 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 文件中。可能使用不同的文件名，但会导致出现缺陷，如存储在其他文件中的 schema 定义，但随后存储在副本上的 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 中。请参阅 [目录服务器如何在复制环境中管理模式更新](#)。

这个过程添加：

- 名为 **dateOfBirth** 的单值属性，带有 OID **2.16.840.1.1133730.2.1.123** 和语法目录字符串 (OID **1.3.6.1.4.1.1466.115.121.1.15**)
- 名为 **exampleperson** 的对象类没有父对象类(SUP top)，它必须包含 **dateOfBirth** 属性。

流程

1. 在 `/etc/dirsrv/slapd-instance_name/schema/99user.ldif` 文件中的 `dn: cn=schema` 条目中添加以下内容：

```
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. 运行 schema reload 任务：

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

验证步骤：

- 监控 `/var/log/dirsrv/slapd-instance_name/errors` 文件：
 - 如果构建成功，Directory 服务器日志：

```
[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.
```

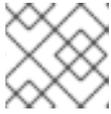
- 如果构建失败，Directory 服务器会记录步骤失败以及原因。

第 4 章 验证现有属性值的语法

通过语法验证，Directory 服务器会检查属性值是否遵循该属性定义中提供的语法规则。目录服务器在 `/var/log/dirsrv/slaped-instance_name/errors` 文件中记录了语法验证任务的结果。

如果需要手动验证语法：

- 您已在 `nsslapd-syntaxcheck` 参数中禁用了语法验证。



注意

红帽建议不要禁用语法验证。

- 您可以从禁用或没有语法验证的服务器迁移数据。

4.1. 使用 DSCONF 模式 VALIDATE-SYNTAX 命令创建语法验证任务

使用 `dsconf` 模式 `validate-syntax` 命令，您可以创建一个语法验证任务来检查每个修改的属性，并确保新值具有所需的语法。

流程

- 要创建语法验证任务，请输入：

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

在示例输出中，命令会创建一个任务来验证 `ou=People,dc=example,dc=com` 子树中所有值的语法，它与 `(objectclass=inetorgperson)` 过滤器匹配。

4.2. 使用 CN 任务条目创建语法验证任务

Directory Server 配置中的 `cn=tasks,cn=config` 条目是用于服务器用于管理任务的临时条目的容器条目。您可以通过在 `cn=syntax validate,cn=tasks,cn=config` 条目中创建任务来启动语法验证操作。

流程

- 要启动语法验证操作，请在 `cn=syntax validate,cn=tasks,cn=config` 条目中创建任务，如下所示：

```
# 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)
```

在示例输出中，命令会创建一个任务来验证 `ou=People,dc=example,dc=com` 子树中所有值的语法，它与 `(objectclass=inetorgperson)` 过滤器类似。任务完成后，Directory 服务器会从目录配置中删除条目。

其他资源

- [配置和架构参考](#)