

Автоматизация задач администрирования через LDAP запросы

ALD Pro

Exported on 08/11/2023

Table of Contents

1	Введение	4
2	Раздел 1. Технология LDAP	5
3	Раздел 2. Взаимодействие с каталогом через LDAP-протокол.....	11
3.1	Раздел 2.1. Графическое приложение для работы с LDAP-каталогом (Apache Directory Studio)	11
3.1.1	Раздел 2.1.1. Настройка соединения.....	11
3.1.2	Раздел 2.1.2. Просмотр и экспорт объектов каталога	13
3.1.3	Раздел 2.1.3. Просмотр схемы каталога	19
3.2	Раздел 2.2. Утилиты для работы с LDAP-каталогом	21
3.2.1	Раздел 2.2.1. Подключение к LDAP.....	21
3.2.2	Раздел 2.2.2. Поиск по каталогу ldapsearch	23
3.2.2.1	Фильтры запроса	25
3.2.2.2	Получение схемы объектных классов	27
3.2.2.3	Получение схемы атрибутов	28
3.2.3	Раздел 2.2.3. Формат данных LDIF	29
3.2.4	Раздел 2.2.4. Добавление записи в каталог через ldapadd	29
3.2.5	Раздел 2.2.5. Модификация записей в каталоге через ldapmodify.....	31
3.2.5.1	Директивы changetype: modify и add:.....	31
3.2.5.2	Директивы changetype: modify и replace:.....	31
3.2.5.3	Директивы changetype: modify и delete:.....	32
3.2.6	Раздел 2.2.6. Работа с кириллицей и Unicode	34
4	Раздел 3. Примеры автоматизации.....	36
4.1	Раздел 3.1. Работа с объектами из командной строки bash	36
4.1.1	Раздел 3.1.1 Конвертер ldif2csv и генерация отчетов.....	36
4.1.2	Раздел 3.1.2. Конвертер ldif2json и работа с JSONPath.....	40
4.2	Раздел 3.2. Добавление пользователей	43
4.2.1	Раздел 3.4. Смена пароля пользователей.....	47
4.3	Раздел 3.5. Проверка просроченных паролей	48
5	Заключение.....	51

Авторы: Афанасьев Михаил, Саблин Эдуард, Лысов Анатолий

1 Введение

Автоматизация позволяет ускорить выполнение массовых операций администрирования, например, если вам нужно завести тысячу пользователей, вы можете написать скрипт, чтобы импортировать учетные записи из CSV-файла, автоматически распределить пользователей по подразделениям и включить их в соответствующие группы. Это не только сэкономит рабочее время, но и позволит исключить ошибки, связанные с человеческим фактором.

Из настоящего документа вы узнаете об устройстве LDAP-каталога и о том, как управлять доменом с помощью прямых запросов к каталогу. Мы также рассмотрим несколько полезных запросов для решения реальных задач администрирования.

2 Раздел 1. Технология LDAP

Служба каталога ALD Pro построена на базе 389 Directory Server, который реализует функции каталога и поддерживает протокол LDAP v3. Облегченный протокол доступа к данным каталога (Lightweight Directory Access Protocol, LDAP) является упрощенной модификацией более строгих стандартов для построения службы распределенного каталога сети X.500.

Каталог LDAP является специализированной не реляционной базой данных, файлы которой вы найдете в папке /var/lib/dirsrv/slapd-ald-company-local/db. Информация каталога представлена в виде древовидной структуры, которую также называют Directory Information Tree или сокращенно DIT, см. Рис 1.

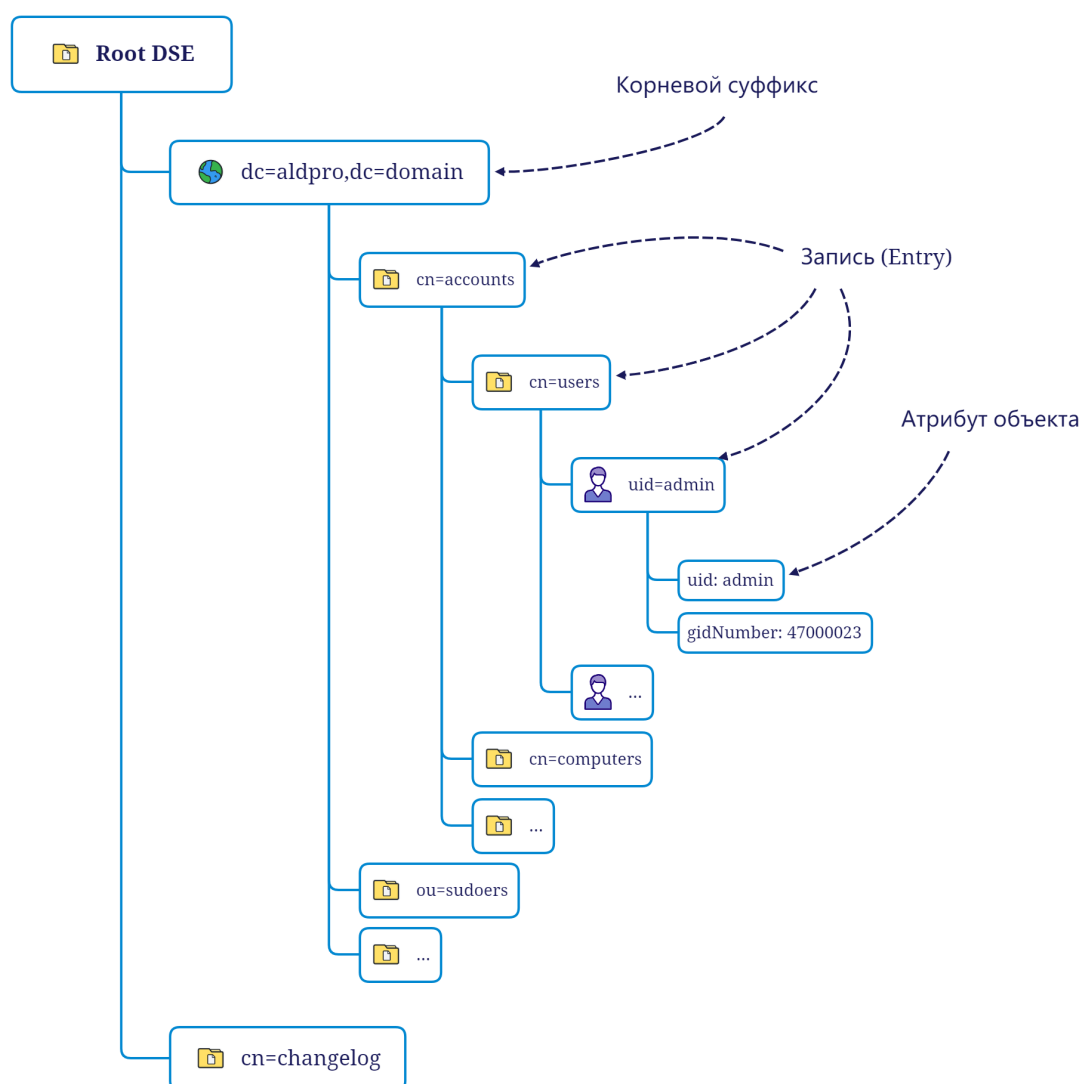


Рис 1. Структура каталога Directory Information Tree

Корень каталога называется Root DSE, где DSE означает Directory system agent Specific Entry, то есть специализированная запись агента системы директорий. Эта запись не имеет родителя, и она описана в файле `/etc/dirsrv/slapd-ALD-COMPANY-LOCAL/dse.ldif`, где `slapd-ALD-COMPANY-LOCAL` директория сервиса `slapd` с именем домена.

Корень Root DSE является родителем для записей **«dc=ald,dc=company,dc=local»** и **«cn=changelog»**, которые называют так же базовыми записями, корневыми суффиксами или контекстами именования (base entry, root suffix, naming context). В контексте **«dc=ald,dc=company,dc=local»** хранятся все объекты каталога, а в **«cn=changelog»** – журнал изменений для работы плагина «Retro Changelog Plugin». Все контексты, определенные в каталоге, указаны в операционном атрибуте **namingContexts** записи **Root DSE**, однако спецификация LDAP позволяет также использовать служебные контексты, например, в записи **«cn=config»** представлены настройки каталога, а через запись **«cn=monitor»** можно получить доступ к информации о состоянии сервера в режиме реального времени.

Существуют разные подходы к наименованию корневых суффиксов, в ALD Pro (FreeIPA) используются правила спецификации RFC-2247, поэтому для домена `ald.company.local` имя корневого суффикса будет **«dc=ald,dc=company,dc=local»**, где **dc** – компонент имени домена, сокращение от англ. Domain Component. Правила наименования необходимы для возможности преобразования DNS-имени в имя LDAP-записи и наоборот.

От корневого суффикса **«dc=ald,dc=company,dc=local»** ответвляются дочерние записи (Entry), которые, в свою очередь, могут быть контейнерами для других записей, за счет чего и образуется древовидная структура. Таким образом записи каталога можно сравнить с директориями файловой системы.

У каждой записи каталога есть имя, которое должно быть уникальным в пределах родительского контейнера, поэтому оно называется относительно уникальным именем Relative Distinguished Name или кратко RDN. Учетные записи пользователей, например, имеют имена **«uid=admin»**, **«uid=ivan.kuznetsov»** и т.д. Особенность имен объектов в LDAP заключается в том, что они хранят не конкретные значения, а только ссылки на хранимые атрибуты записей, которые используются для идентификации объектов. Например, для идентификации учетных записей пользователей используют атрибут **uid**, для учетных записей компьютеров **fqdn** (fully qualified domain name, полное доменное имя хоста), а в именах контейнеров обычно присутствует **cn** (common name, общее имя).

В приведенном примере RDN учетной записи доменного администратора **«uid=admin»** состоит из названия атрибута **«uid»**, после которого идет символ присвоения «=» и далее значение атрибута **«admin»**. Вся эта запись вместе называется Определением Значения Атрибута от англ. Attribute Value Assertion (AVA). Обычно имена записей задаются значением одного атрибута, но могут использоваться и несколько, тогда в имени RDN эти определения AVA будут объединяться знаком «+», например: **«cn=ivan+I=Moscow»**. Каталог 389 Directory Server поддерживает такой способ именования записей, но в ALD Pro (FreeIPA) он не используется.

Для идентификации объекта в пределах всего каталога используют уникальное имя Distinguished Name или сокращенно DN. Уникальное имя представляет из себя цепочку RDN, которые записывают через запятую слева направо, начиная с целевой записи и до корневого суффикса вверх по иерархии, см. Рис 2. Если привести аналогию с объектами файловой системы, то RDN будет соответствовать имени объекта директории или файла, а DN – полному имени объекта файловой системы, которое включает путь к родительской объекту и имени объекта. И также, как на одном диске не может быть двух директорий с одинаковыми полными именами, в каталоге LDAP не может быть двух записей с одинаковыми DN. Описание формата DN можно найти RFC 4514.

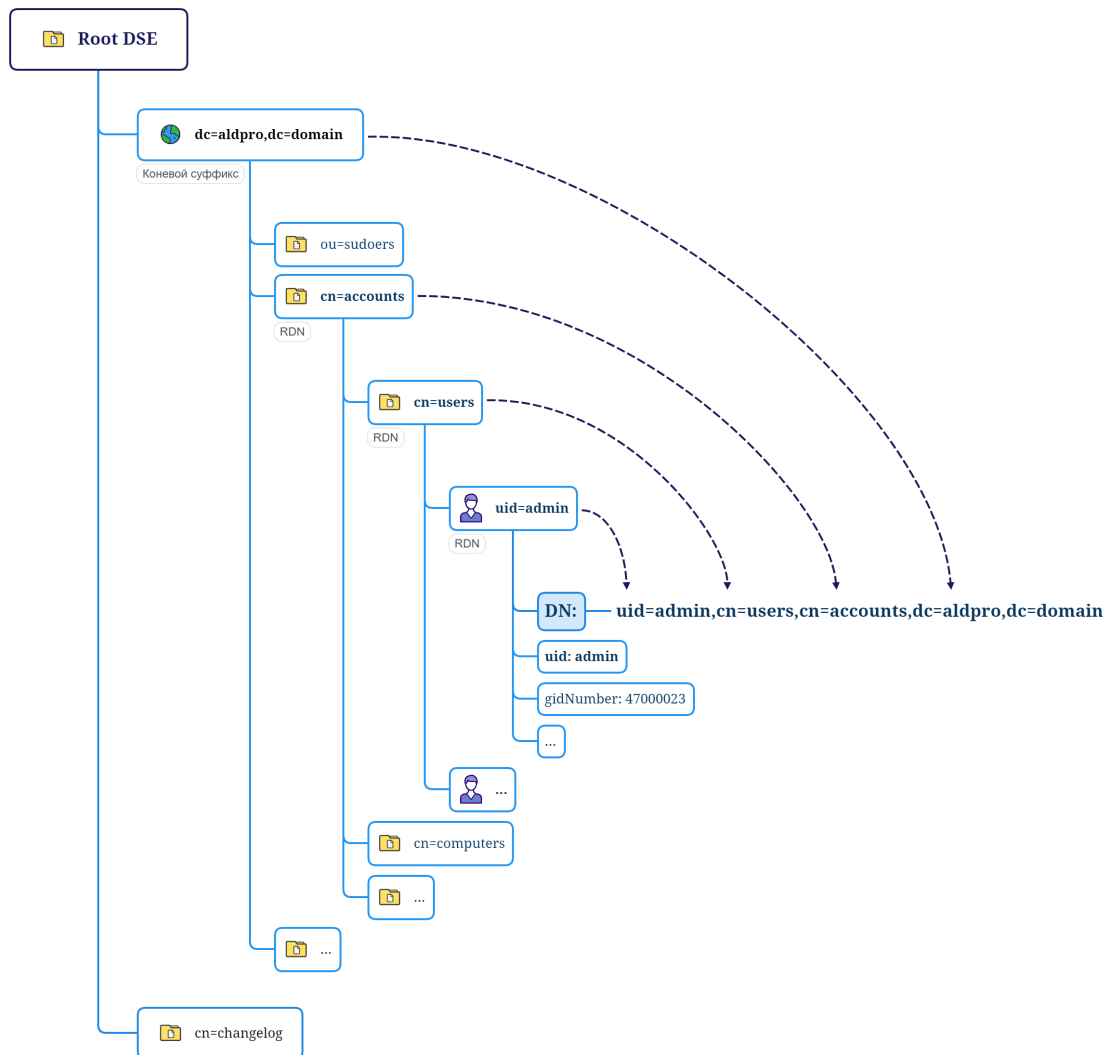


Рис 2. Пример формирования уникального имени записи DN

Запись каталога может хранить не только дочерние записи, но и набор атрибутов, описывающих ее свойства. Например, для учетной записи пользователя это могут быть ФИО, должность, номер телефона и т.д.

Данные в LDAP-каталоге строго структурированы и все атрибуты должны быть определены в схеме данных заранее. Перечень доступных для конкретного объекта атрибутов задается списком назначенных ему классов, см. атрибут **objectClass**. Например, учетные записи пользователей могут содержать атрибут **gidNumber** по той причине, что им назначен класс объектов **posixAccount**, см. Рис 3.

Attribute Description	Value
objectClass	organizationalPerson (structural)
objectClass	person (structural)
objectClass	posixAccount (auxiliary)
objectClass	rbta-address (auxiliary)
objectClass	rbta-custom-user-attrs (auxiliary)
objectClass	rbta-inetorgperson-ext (auxiliary)
objectClass	rbta-unit (structural)
objectClass	ruPostMailAccount (auxiliary)
objectClass	top (abstract)
objectClass	x-ald-audit-policy (structural)
objectClass	x-ald-user (structural)
objectClass	x-ald-user-parsec14 (auxiliary)
cn	Administrator
gidNumber	959800000
homeDirectory	/home/admin
ipaNtSecurityIdentifier	S-1-5-21-1724891028-2898148248-1736958143-500
ipaUniqueId	98535a9e-65d5-11ed-bf9c-0800279d572d
sn	Administrator
uid	admin
uidNumber	959800000
gecos	Administrator

objectClass: posixAccount
gidNumber: 959800000

Рис. 3 Возможность указать gidNumber определяется наличием класса объектов posixAccount

Всем объектам каталога назначен как минимум один класс **top**, т.к. в нем описан атрибут **objectClass**, с помощью которого работает механизм назначения классов. Но практической пользы от объектов с одним классом не много, поэтому обычно в каталоге у объектов два и более классов.

В качестве примера класса рассмотрим класс объектов **posixGroup**, который определен в схеме следующим образом:

```
( 1.3.6.1.1.1.2.2 NAME 'posixGroup' DESC 'Standard LDAP objectclass' SUP top
STRUCTURAL MUST ( cn $ gidNumber ) MAY ( userPassword $ memberUid $ description ) X-
ORIGIN 'RFC 2307' )
```

где:

- **1.3.6.1.1.1.2.2** – это идентификатор объекта (object id, OID). Глобальные идентификаторы присваиваются международными организациями (IANA, ISO, ITU-T, ANSI, BSI), а для расширения схемы в прикладных системах используется пространство номеров с префиксом «1.3.6.1.4.1.X», где X – это внутренний номер организации. Например, объекты РусБИТех-Астра имеют префикс «1.3.6.1.4.1.52616.*»
- **NAME** '...' – инструкция NAME задает имя класса
- **DESC** '...' – инструкция DESC задает описание класса
- **SUP** '...' – инструкция SUP указывает родительский класс. В приведенном примере наследование идет от класса «top», поэтому объектам будет доступен его атрибут objectClass.
- **STRUCTURAL** – инструкция указывает, что класс будет относиться к виду структурных. Существуют также абстрактные (ABSTRACT) и вспомогательные (AUXILIARY) классы, но в контексте автоматизации различия между видами классов не принципиальны.
- **MUST** и **MAY** – инструкции, которые позволяют задать списки обязательных и дополнительных атрибутов. Полный перечень атрибутов, доступных объекту, расширяется атрибутами, которые наследуются от всех родительских классов.
- **X-ORIGIN** '...' – инструкция, которая позволяет задать комментарий с ссылкой на документацию, из которой можно почерпнуть дополнительную информацию об этом классе. В приведенном примере информацию следует искать в документе RFC 2307.

Обратите внимание на то, что один и тот же атрибут может быть использован в нескольких классах, поэтому пользователям можно задать значение **gidNumber**, т.к. им назначен класс **posixAccount**, и, в тоже время, он может быть задан у групп, т.к. им назначен класс **posixGroup**, см. Рис. 4

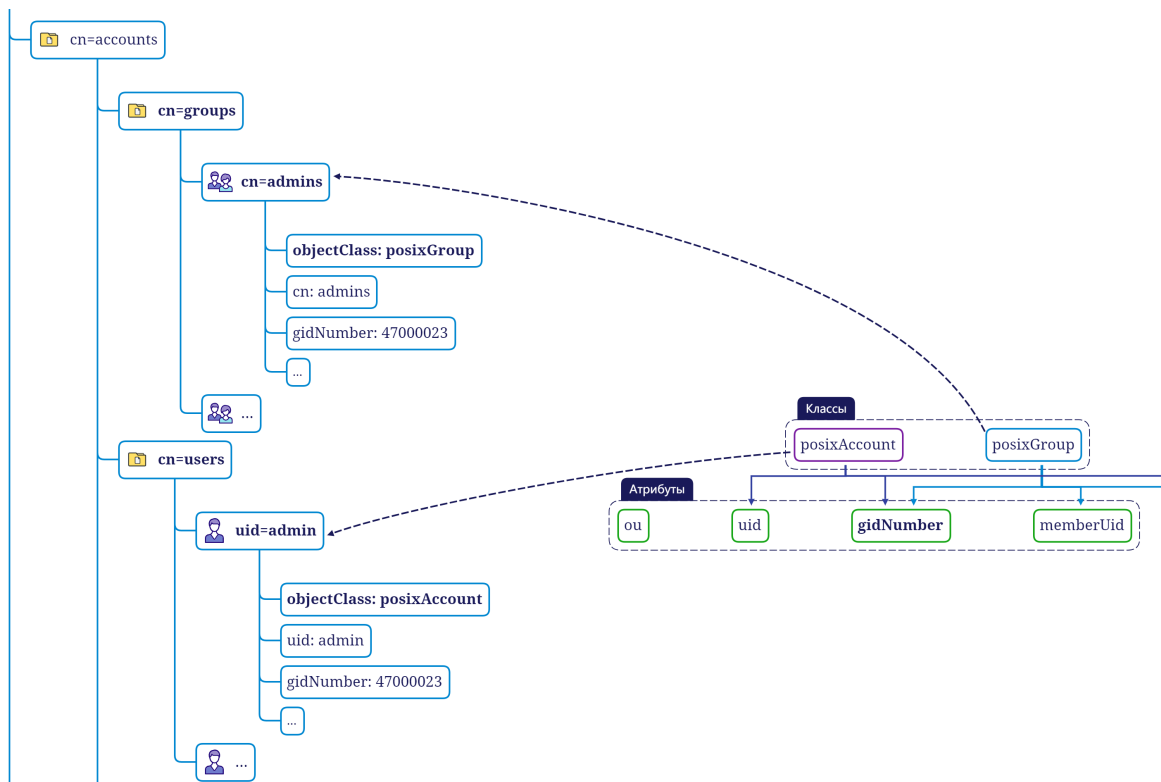


Рис 4. Использование атрибута **gidNumber** классами **posixAccount** и **posixGroup**

Теперь рассмотрим описание атрибута **gidNumber** подробнее:

```
( 1.3.6.1.1.1.1.1 NAME 'gidNumber' DESC 'Standard LDAP attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE X-ORIGIN 'RFC 2307' )
```

где:

- **1.3.6.1.1.1.1.1** – это уникальный идентификатор атрибута.
- **NAME '...'** – инструкция NAME задает атрибута
- **DESC '...'** – инструкция DESC задает описание атрибута
- **SYNTAX '...'** – инструкция задает тип хранимого в атрибуте значения. В приведенном примере 1.3.6.1.4.1.1466.115.121.1.27 соответствует целым числам. Описание всех типов данных можно найти в RFC4517.
- **SINGLE-VALUE** – инструкция указывает, что атрибут может хранить только одно значение. Если этой инструкции не будет, то объектам можно будет присваивать несколько значений этого атрибута.
- **X-ORIGIN '...'** – инструкция, которая позволяет задать комментарий с ссылкой на документацию.

Описание схемы данных хранится в файлах на диске в директориях:

- /usr/share/dirsrv/schema/
- /etc/dirsrv/schema/
- /usr/share/dirsrv/updates/
- /etc/dirsrv/slapd-ALD-COMPANY-LOCAL/

при обращении к каталогу по LDAP информацию можно получить из операционного DIT «cn=schema»

3 Раздел 2. Взаимодействие с каталогом через LDAP-протокол

3.1 Раздел 2.1. Графическое приложение для работы с LDAP-каталогом (Apache Directory Studio)

Для работы с LDAP-каталогом из графического интерфейса – просмотра структуры каталога, редактирования записей, импорта и экспорта данных – можно воспользоваться таким бесплатным инструментом как Apache Directory Studio, см. Рис 5. Загрузить приложение можно с официального сайта directory.apache.org/studio/¹, для работы потребуется java runtime.

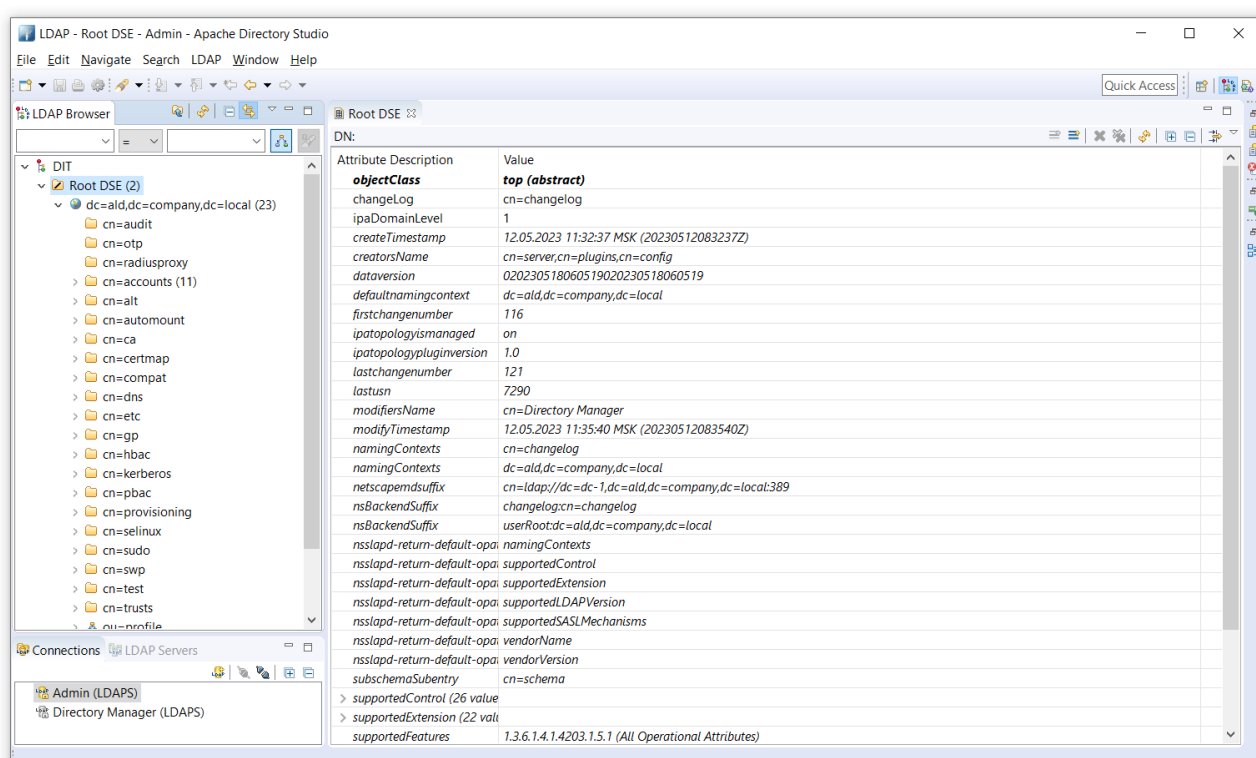


Рис 5. Apache Directory Studio

3.1.1 Раздел 2.1.1. Настройка соединения

Чтобы подключиться к LDAP каталогу нужно создать новое соединение через меню «LDAP > New connection». Откроется окно для создания нового подключения см. Рис 6.

¹ <https://directory.apache.org/studio/>

Рис 6. Настройка сети для нового LDAP-подключения

ВАЖНО!

По умолчанию Apache Directory Studio предлагает создать незащищенное подключение на порт 389, что допустимо только при обращении к каталогу по localhost, т.е. приложение должно быть установлено непосредственно на контроллер домена, т.к. при подключении пароль будет передаваться в открытом виде.

Если вы подключаетесь к каталогу с другого компьютера в сети, обязательно используйте порт 636 и метод шифрования SSL, чтобы перехват пароля был невозможен.

Аутентификация по паролю называется связыванием (Bind), вы можете подключиться к каталогу как супер пользователь «**cn=Directory Manager**» или доменным администратором ALD Pro «**uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local**», см. Рис 7. В зависимости от учетной записи у вас будут разные права на доступ к записям и атрибутам.

НА ЗАМЕТКУ!

Сразу после установки системы пароли этих учетных записей совпадают. Чтобы сбросить пароль Directory Manager вам потребуется вручную менять хэш, записанный в файле dse.ldif, в строке, начинающейся с nsslapd-rootpw.

Перед закрытием окна проверьте корректность подключения нажатием кнопки «Check Authentication».

New LDAP Connection

Authentication

Please select an authentication method and input authentication data.

Authentication Method: Simple Authentication

Authentication Parameter:

Bind DN or user: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local

Authorization ID (SASL): SASL PLAIN only

Bind password: [masked]

☒ Save password

Check Authentication

▶ **SASL Settings**
 ▶ **Kerberos Settings**

? < Back Next > Finish Cancel

Рис 7. Настройка аутентификации для нового LDAP-подключения

После добавления подключения новый сервер появится в списке Connections и вы сможете подключиться к серверу двойным кликом по этой записи.

3.1.2 Раздел 2.1.2. Просмотр и экспорт объектов каталога

Вы можете просмотреть записи в дереве каталога, выбрав нужный RDN из окна LDAP browse. Например, нажав на пункт «cn=accounts» из списка слева, основные атрибуты отобразятся в центральном окне с именем «cn=accounts,dc=ald,dc=company,dc=local».

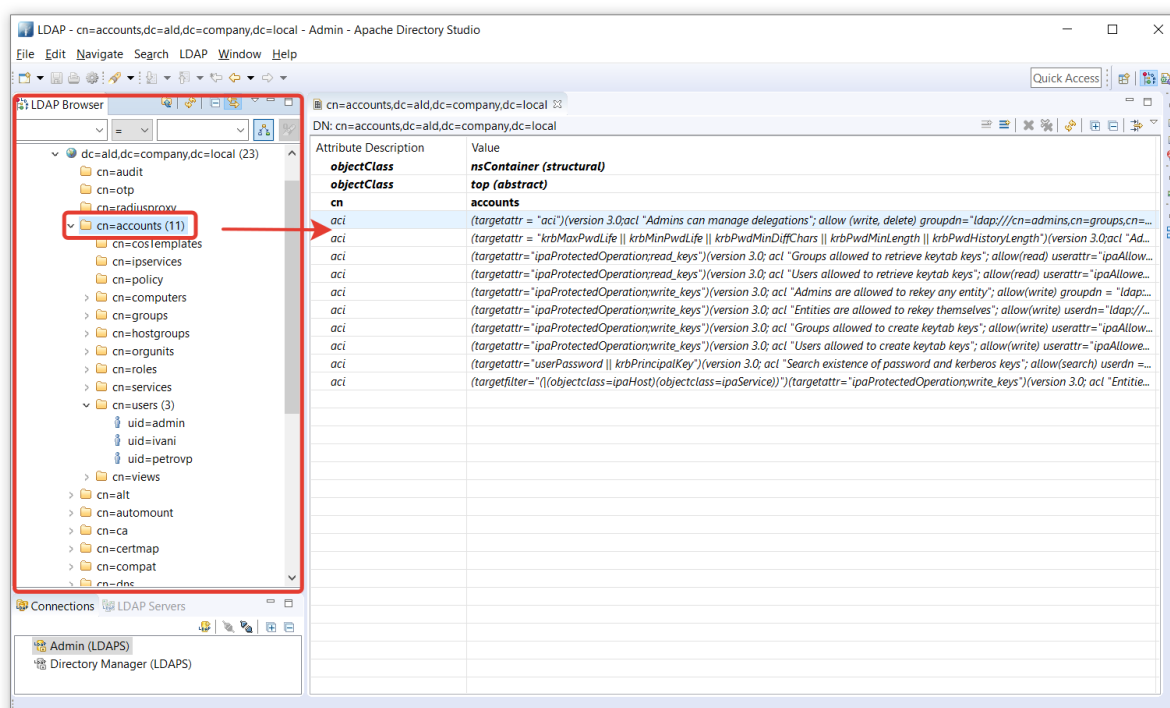


Рис 8. Окно LDAP browse

А также можно открыть любой DN из меню «Navigate > Go to DN», например «cn=config», но для его просмотра нужна учетная запись «cn=Directory Manager». Рассмотрим некоторые полезные DN, см Таблице 1.

Запись DN	Описание
cn=accounts,dc=ald,dc=company,dc=local	Контейнер который содержит дочерние записи контейнеров учетных записей, компьютеров, групп и Д.Р
cn=groups,cn=accounts,...	Содержит список групп пользователей
cn=computers,cn=accounts,...	Содержит список компьютеры в домене
cn=hostgroups, cn=accounts,...	Содержит группы компьютеров домена, например ipaservers
cn=users,cn=accounts,...	Содержит список пользователей домена
cn=orgunits,cn=accounts,...	Содержит список подразделений, которые отображаются у других записей в атрибуте rbtadp , например у пользователя или компьютера

Запись DN	Описание
cn=dns,...	Содержит информацию о записях DNS

Таблица 1. Описание некоторых DN записей

Экспортировать объекты можно, нажав контекстное меню по требуемой записи, например, cn=users в дереве см. Рис 9, а затем Export и нужный формат рассмотрим CSV.

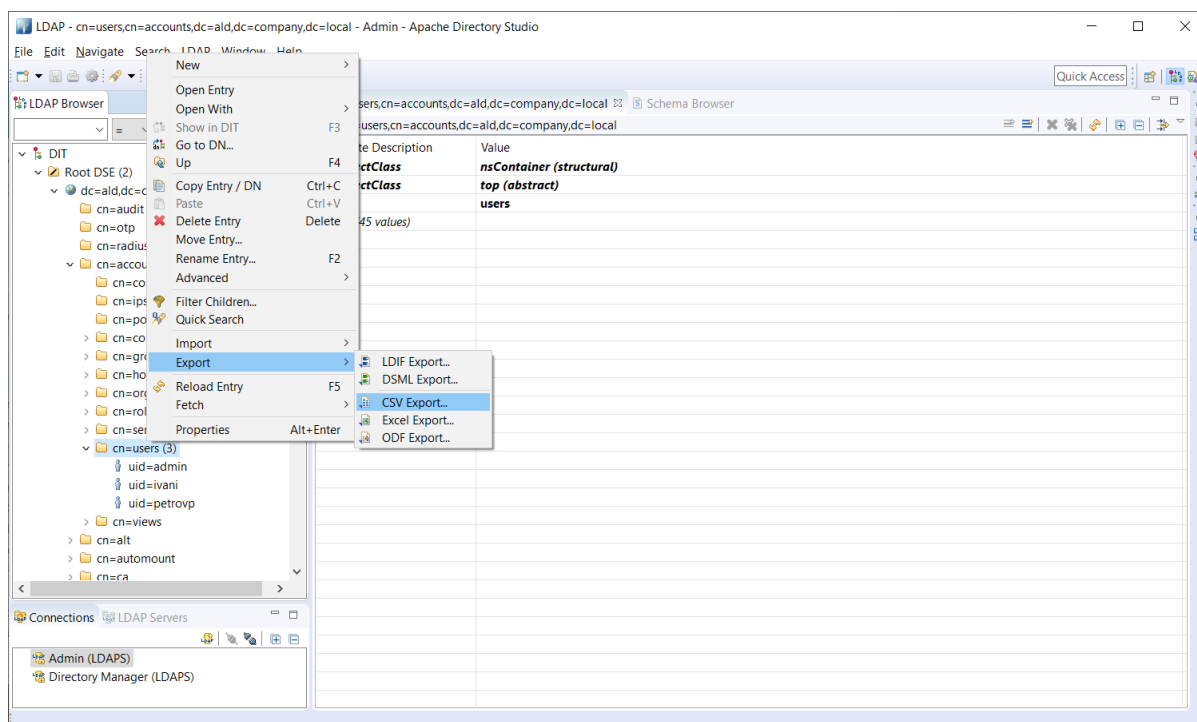


Рис 9. Окно LDAP browse

После выбора формата CSV мы видим диалог настроек параметров экспорта данных, см Рис. 8. В окне параметров настройте нужные фильтры и список требуемых атрибутов для вывода.

CSV Export

Data to Export
Please define search parameters for the export.

Connection:

Search Base:

Filter:

Returning Attributes:

☒ Export DN

Controls

☐ ManageDsaIT

☐ Subentries

☐ Paged Search Page Size: ☒ Scroll Mode

Scope

☐ Object

☐ One Level

☒ Subtree

Limits

Count Limit:

Time Limit (s):

Aliases Dereferencing

☒ Finding Base DN

☒ Search

Referrals Handling

☒ Follow Referrals automatically

☐ Ignore Referrals

Рис 10. Окно параметров CSV Export

Следующим шагом укажите имя файла, в который вы хотите записать CSV, см. Рис 11, а затем нажмите Finish.

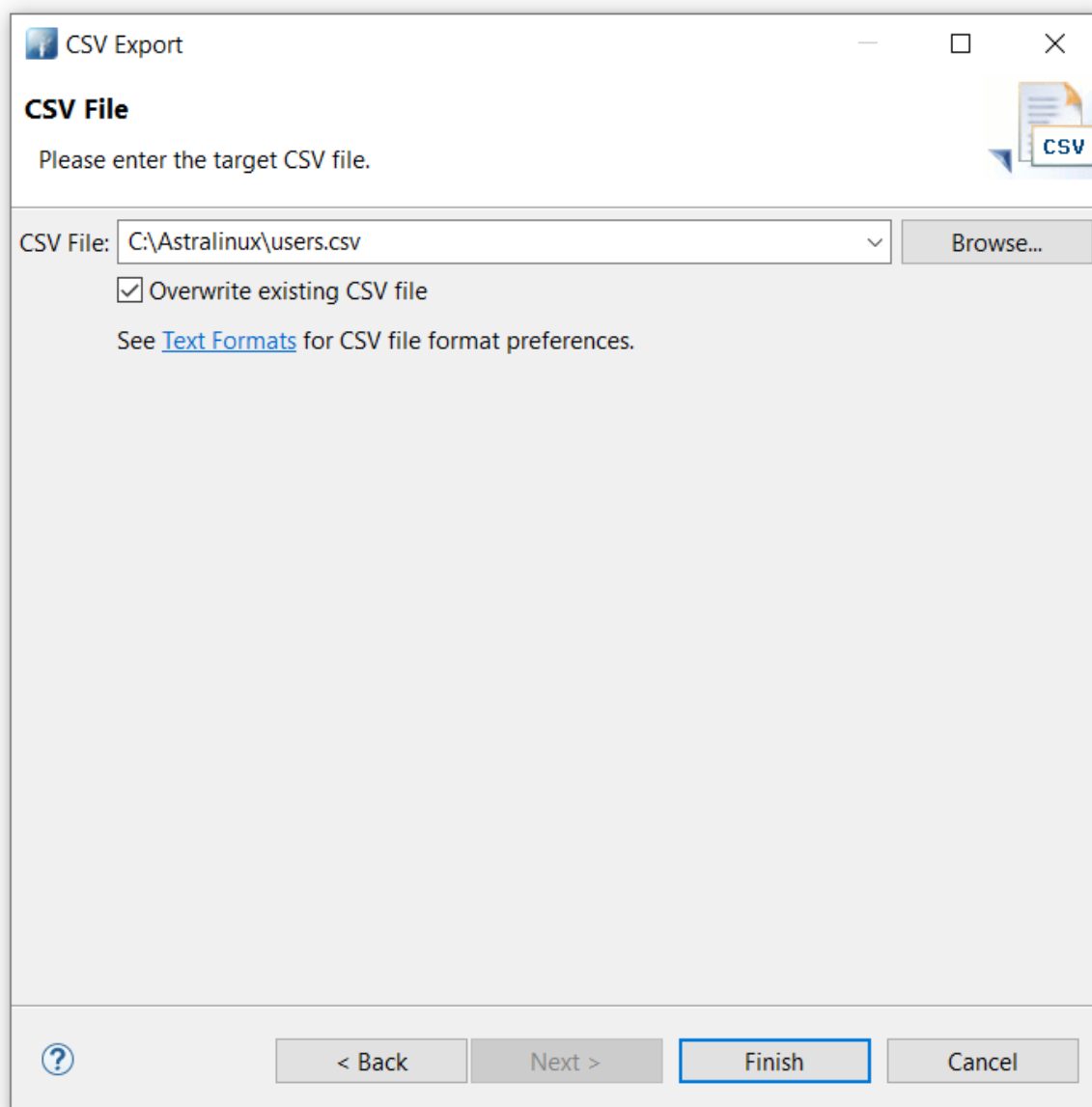


Рис 11. Окно CSV Export выбор имени файла

Если нажать на ссылку Text Formats см. Рис. 12, в которой описаны настройки генерации текстовых данных, таких как разделитель, обрамление данных кавычками, разделитель строк и др.

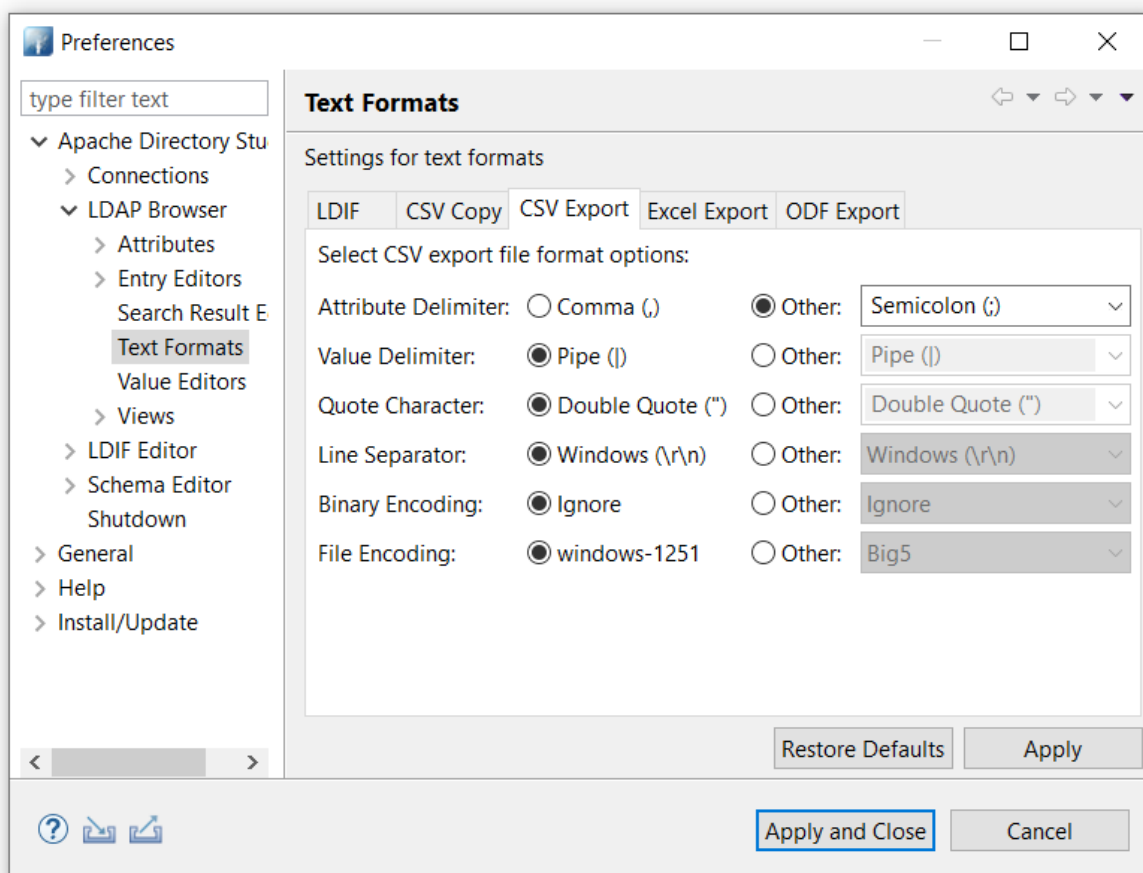


Рис 12. Настройки вывода текстового форматирования

Откроем результат выполнения экспорта данных в программе LibreOffice Calc см. Рис. 13.

1	dn	uid	displayName	rbtadp	mail	krbPasswordExpiration	nsAccountLock
2	uid=admin.cn=users.cn=accounts	admin		ou=ald.company.local.cn=orgunits.cn=accounts	dc=ald.dc=company	20230810083425Z	FALSE
3	uid=petrov.pn=users.cn=accounts	petrov	Петров	petr	ou=ald.company.local.cn=orgunits.cn=petrov@ald.company.local	20230522135449Z	FALSE
4	uid=ivani.cn=users.cn=accounts	ivani	Иван Иванов		ou=ald.company.local.cn=orgunits.cn=ivani@ald.company.local	20230522095730Z	FALSE
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

Рис 13. Результат выполнения экспорта данных

Вы можете использовать этот файл для автоматизации как входную информацию или подготовить любой отчет по объектам из каталога LDAP.

3.1.3 Раздел 2.1.3. Просмотр схемы каталога

Для просмотра схемы каталога выберите Root DSE в дереве LDAP Browser и выполните команду «LDAP > Open Schema Browser» см. Рис 14.

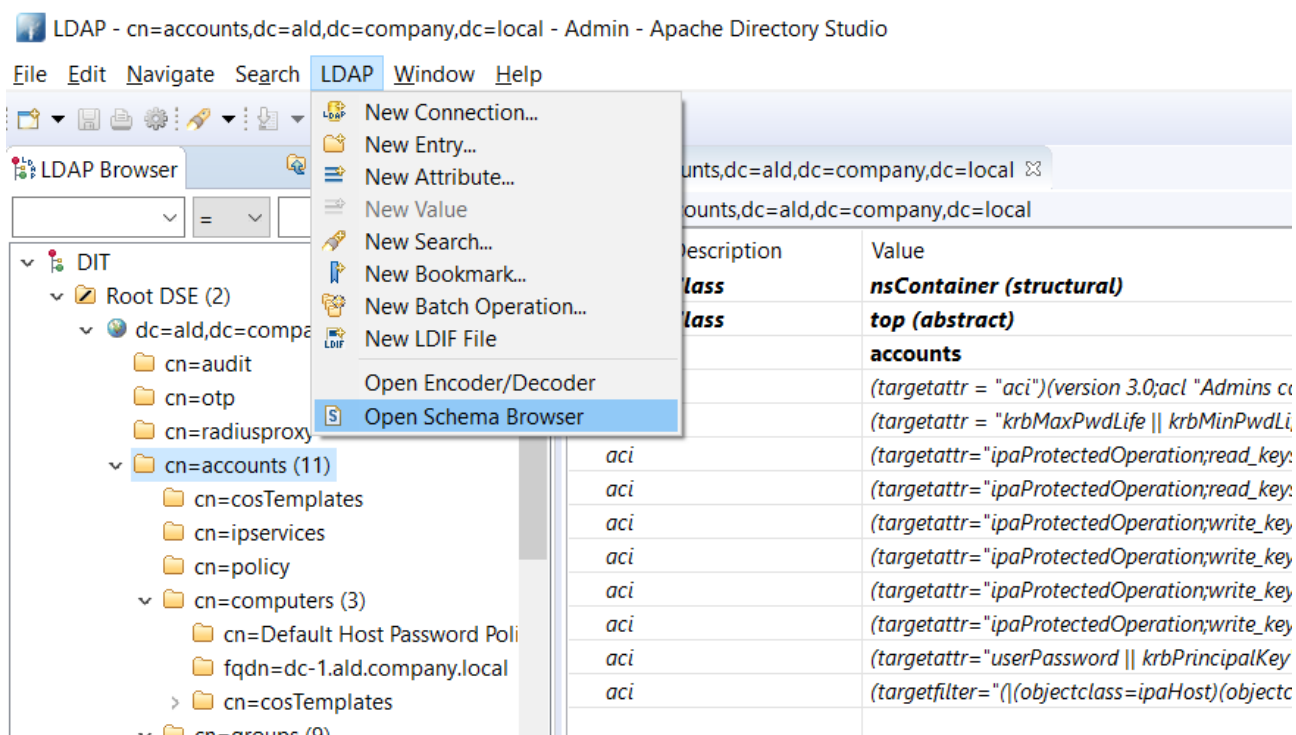


Рис 14. Меню Open Schema Browser

По умолчанию в Schema Browser открывается страница для просмотра классов объектов (Object Classes). Вы можете делать поиск и просматривать детали (Details). Например, введите в фильтр «posix» и вы найдете обсуждаемые ранее классы **posixAccount** и **posixGroup**, см. Рис 15.

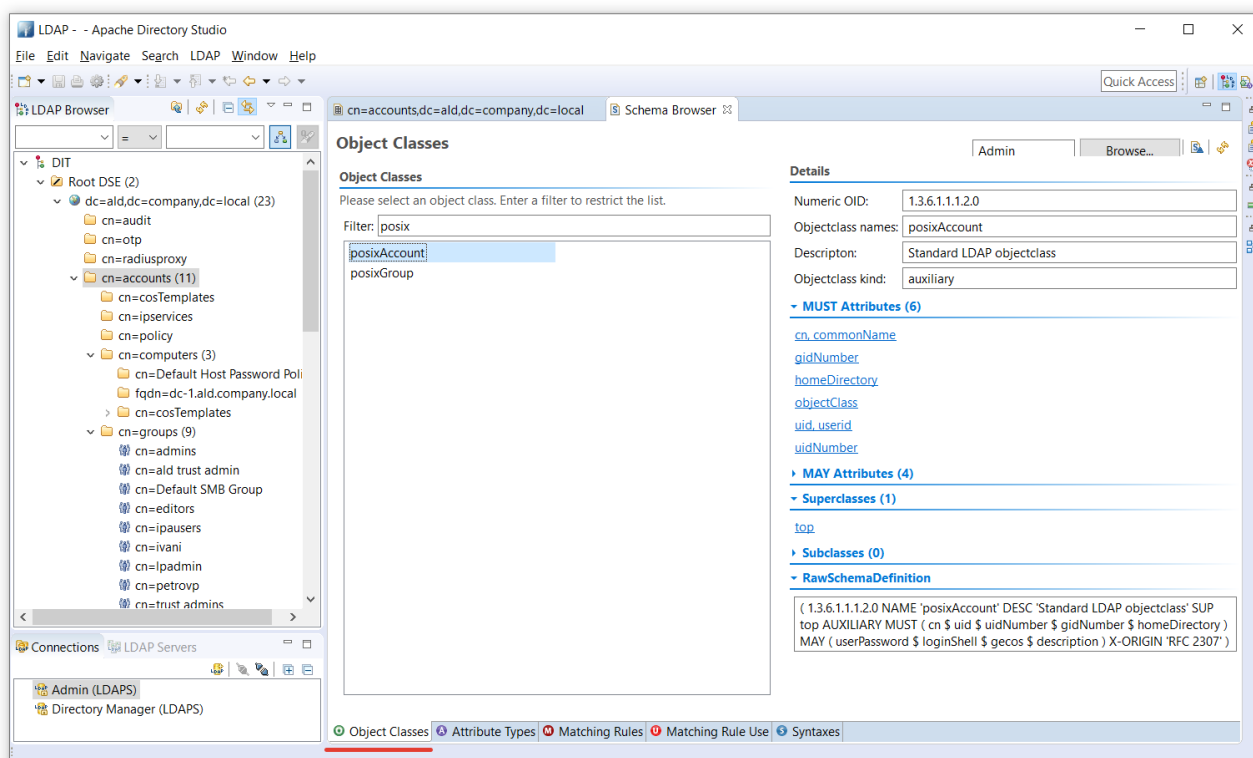


Рис 15. Schema Browser и вкладка Object Classes

Для перехода к другим группам объектов используйте ярлычки в нижней части вкладки. Для просмотра атрибутов откройте страницу Attribute Types. В этом окне можно искать по списку всех атрибутов и просматривать детали выбранного атрибута. Например, введите в поле фильтра gid и вы увидите несколько атрибутов, в именах которых содержится эта подстрока, см. Рис 16.

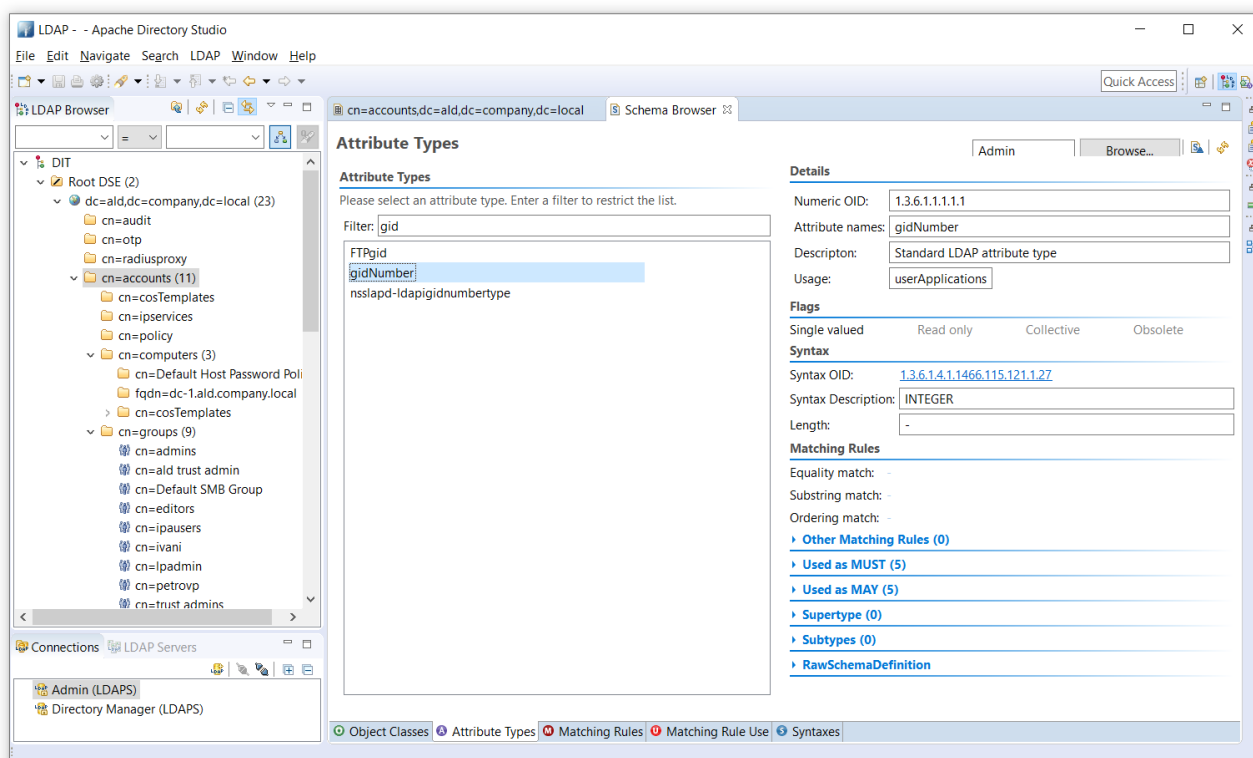


Рис 16. Schema Browser и вкладка Attribute Types

3.2 Раздел 2.2. Утилиты для работы с LDAP-каталогом

Для работы с LDAP-каталогом из командной строки используются утилиты пакета **ldap-utils**:

- **ldapwhoami** – выполняет подключение к каталогу и возвращает имя текущего пользователя;
- **ldapsearch** – выполняет поиск по каталогу с указанными параметрами;
- **ldapadd** – позволяет создать новый объект в каталоге;
- **ldapdelete** – позволяет удалить объект из каталога;
- **ldapmodify** – позволяет изменить атрибуты существующего объекта в каталоге;
- **ldapcompare** – позволяет сравнить текущее значение атрибута конкретной записи с желаемым значением и получить результат в формате TRUE/FALSE;
- **ldapexop** – позволяет выполнять расширенные операции, добавленные разработчиками конкретного LDAP-сервера. Расширенные операции подобны хранимым процедурам SQL, и позволяют расширять возможности взаимодействия с сервером без внесения изменений в LDAP-протокол;
- **ldappasswd** – позволяет сбросить пароль для учетной записи;
- **ldapmodrdn** – позволяет переименовывать существующие объекты каталога.

3.2.1 Раздел 2.2.1. Подключение к LDAP

Подключение к LDAP-каталогу происходит в рамках вызова каждой утилиты. Вы можете задавать параметры подключения в явном виде, или полагаться на настройки по умолчанию, которые описаны в файле **/etc/ldap/ldap.conf**. При вводе компьютера в домен указанный файл настраивается

автоматически, подключение будет выполняться к одному из контроллеров домена по протоколу LDAPS с использованием Kerberos-билета из связки ключей.

Проверим наш доступ в каталог командой **ldapwhoami**:

```
ldapwhoami
```

Результат выполнения:

```
SASL/GSSAPI authentication started
ldap_sasl_interactive_bind_s: Local error (-2)
    additional info: SASL(-1): generic failure: GSSAPI Error: Unspecified GSS
failure. Minor code may provide more information (No Kerberos credentials available
(default cache: KEYRING:persistent:1000))
```

Данный результат говорит, что мы не имеем учетных записей в кэше Kerberos. Пробуем выполнить вход **kinit admin** и проверим повторно **ldapwhoami**:

```
kinit admin
ldapwhoami
```

Результат выполнения:

```
Password for admin@ALDPRO.DOMAIN:
SASL/GSSAPI authentication started
SASL username: admin@ALDPRO.DOMAIN
SASL SSF: 256
SASL data security layer installed.
dn: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local
```

Теперь у нас получилось выполнить запрос к LDAP серверу, т.к. у нас был валиден TGT Kerberos билет и система смогла с его помощью успешно получить сервисный билет на доступ к LDAP серверу. Проверить текущие билеты в кэше можно командой **klist**:

```
klist
```

Результат выполнения:

```
Ticket cache: KEYRING:persistent:1000:1000
Default principal: admin@ALD.COMPANY.LOCAL

Valid starting    Expires          Service principal
22.05.2023 09:58:25 23.05.2023 09:58:01 ldap/dc-1.ald.company.local@ALD.COMPANY.LOCAL
22.05.2023 09:58:08 23.05.2023 09:58:01 krbtgt/ALD.COMPANY.LOCAL@ALD.COMPANY.LOCAL
```

Как видите, в кеше есть билет ldap: ldap/dc-1.ald.company.local@ALD.COMPANY.LOCAL, поэтому мы можем использовать утилиты из пакета **ldap-utils**.

При подключении к серверу мы можем переопределять любой из параметров по умолчанию, задавая их в явном виде:

```
ldapsearch -H dc-1.ald.company.local -ZZ -x -W -D
"uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local" -b
"cn=users,cn=accounts,dc=ald,dc=company,dc=local" '(uid=*)' uid givenName sn
```

где:

Параметр **-H** – для указания адреса LDAP-сервера, например dc-1.ald.company.local. Параметр позволяет указать нешифрованный протокол ldap или шифрованный ldaps, например, «[ldap://dc-1.ald.company.local](#)». Вы можете выполнять подключение к LDAP каталогу также через unix-сокет, если приложение выполняется на том же сервере, тогда нужно указать «[ldapi://%2fvar%2frun%2fslapd-ALD-COMPANY-LOCAL.socket](#)»

- Параметр **-ZZ** – это параметр включения шифрованного соединения, где первая Z означает отправку серверу запроса STARTTLS. Если сервер не поддерживает TLS, соединение продолжится, и оно не будет шифроваться. Вторая Z устанавливает требование использовать только шифрованное соединение, если сервер не поддерживает TLS, соединение прервется. Рекомендуется использовать двойной ключ -ZZ для установления шифрованного соединения, если по каким-либо причинам 636 порт недоступен, а также требование использовать безопасный протокол TLS на 636 порту можно задать с помощью ключа -H, указав порт явно ldaps://dc-1.ald.company.local;
- Параметр **-x** – указывает на необходимость выполнить простую аутентификацию по логину/паролю;
- Параметр **-D** – задает Bind DN пользователя для аутентификации, например, "uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local" или супер пользователя, "cn=Directory Manager";
- Параметр **-W** – указывает, что пароль должен быть предоставлен в интерактивном режиме (а если нужно передать пароль в параметре, то это можно сделать с помощью ключа -w)

Используя параметры **-D** и **-W**, вы можете подключиться к нужному серверу по IP или имени сервера и файл конфигурации задействован не будет.

3.2.2 Раздел 2.2.2. Поиск по каталогу ldapsearch

Для поиска информации вы можете использовать утилиту **ldapsearch**, которая извлекает из каталога набор записей с указанными атрибутами в соответствии с заданными критериями фильтрации и заданного списка атрибутов. Синтаксис команды в общем виде выглядит следующим образом:

```
ldapsearch [options] [filter [attributes...]]
```

где:

- options — параметры вызова, в т.ч. параметры подключения. Для простоты мы будем использовать параметры подключения по умолчанию;
- filter — служит для точного указания критериев поиска;
- attributes — указывает, какие атрибуты необходимо запросить.

Для примера напишем запрос, который будет выдавать список всех пользователей домена из «**cn=users**», которая в свою очередь является потомком записи «cn=accounts», и все они находятся в пространстве корневого суффикса «dc=ald,dc=company,dc=local». Таким образом полным уникальным именем записи (Distinguished name, DN), в которой хранятся пользователи, является «cn=users,cn=accounts,dc=ald,dc=company,dc=local».

Выполним поиск по каталогу с помощью команды **ldapsearch**:

```
ldapsearch -Q -s sub -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local" '(uid=*)'
uid givenName sn
```

где:

- Параметр **-Q** – это тихий режим SASL, не выводится информация о SASL подключении, который полезен при автоматизации для отчитки данных от технической информации.
- Параметр **-s** – это область поиска (scope). Может принимать следующие значения:
 - one - поиск идет по дочерним записям на один уровень ниже в иерархии
 - sub - поиск идет по всем дочерним записям на всю глубину иерархии, параметр по умолчанию
 - children - то же, что и sub, но ограничивает поиск только дочерними записями
 - base - ограничивает поиск по текущей записи, заданной параметром -b. Если задан one, поиск идет по дочерним записям на один уровень ниже в иерархии. Если задан sub, то поиск идет по всем дочерним записям на всю глубину иерархии, начиная с записи, заданной параметром -b, при этом включая саму базовую запись. children - то же, что и sub, но ограничивает поиск только дочерними записями, не включая базовую запись.
- Параметр **-b** – это базовая запись (base), которая будет использоваться в качестве начальной точки для поиска по дереву.
- Параметр **'(uid=*)'** – это фильтр, в котором мы ищем все записи, имеющие атрибут uid. Фильтры и составные фильтры мы рассмотрим далее.
- Параметры **uid givenName sn** – это атрибуты, которые нужно вывести в результат. Если их не указывать, то будут отображены все атрибуты найденных записей.

Результат поиска по каталогу:

```
# extended LDIF
#
# LDAPv3
# base <cn=users,cn=accounts,dc=ald,dc=company,dc=local> with scope subtree
# filter: (uid=*)
# requesting: uid givenName sn
#
# admin, users, accounts, ald.company.local
dn: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local
uid: admin
sn: Administrator
# petrovp, users, accounts, ald.company.local
dn: uid=petrovp,cn=users,cn=accounts,dc=ald,dc=company,dc=local
uid: petrovp
givenName:: 0J/RkdGC0YA=
sn:: 0J/QtdGC0YDQvtCy
```



```
# ivani, users, accounts, ald.company.local
dn: uid=ivani,cn=users,cn=accounts,dc=ald,dc=company,dc=local
uid: ivani
givenName:: 0JjQstCw0L0=
sn:: 0JjQstCw0L3QvtCy

# search result
search: 4
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Результат выводится в поток **stdout** в текстовом формате LDIF и его можно использовать по конвейеру pipeline или перенаправить в файл для дальнейшей обработки. Подробнее о формате LDIF мы поговорим в разделе 2.2.3.

Существуют операционные атрибуты (operational attributes), которые встроены в LDAP сервер и управляются им самим, например, **entrydn**, **entryid**, **parentid** или **nsAccountLock**. Большинство операционных атрибутов для пользователей доступны только для чтения. Чтобы их увидеть необходимо указать "+" в конце команды, однако не все операционные атрибуты доступны для просмотра таким образом:

```
ldapsearch -Q -LLL -s base -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local" "+"
```

Результат выполнения:

```
dn: cn=users,cn=accounts,dc=ald,dc=company,dc=local
creatorsName: cn=Directory Manager
modifiersName: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local
createTimestamp: 20230324160645Z
modifyTimestamp: 20230324161206Z
nsUniqueId: df91d884-ca5d11ed-b5f0ee72-e6acffe1
parentid: 2
entryid: 3
entryusn: 6055
numSubordinates: 6
...
```

Например, мы видим, что запись **"cn=users"** имеет операционный атрибут **numSubordinates**, который показывает число дочерних записей.

3.2.2.1 Фильтры запроса

Рассмотрим, как сделать поиск **ldapsearch** более гибким с помощью фильтров, задав условие для отбора нужной нам информации. Синтаксис фильтра выглядит следующим образом:

(attribute operator value)

В Таблице 2 приведены операторы (operator), используемые в фильтрах:

Оператор	Комментарий
>=	Больше или равно. Возвращает результат, если атрибут больше или равен какому-то значению, например (uidNumber>=180003)
<=	Меньше или равно. Возвращает результат, если атрибут меньше или равен какому-то значения, например: (uidNumber<=180003)
=*	Наличие. Возвращает результат, если атрибут содержит одно или более значений, например: (cn=*)
=	Равенство. Возвращает результат, если атрибут равен некоторому значению, например: (sn=petrov)
~=	Примерное равенство. Возвращает результат, если атрибут содержит приблизительно схожее значение, например: (cn~=petrof)
=string*string	Включает в себя. Возвращает значение, если атрибут содержит определенную подстроку. Где знак "*" означает ноль или более символов, например: (cn=pet*ov)

Таблица 2. Операторы для фильтрации запросов

Например, выведем список пользователей с фамилией petrov. Для этого нам потребуется указать фильтр по атрибуту sn (surname):

```
ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(sn=Петров)" cn
```

Результат выполнения:

```
dn: uid=ppetrov,cn=users,cn=accounts,dc=ald,dc=company,dc=local
cn: Petr Petrov
```

В результате запрос отфильтрован по атрибуту sn. Таким образом мы можем подготовить любые данные для обработки.

Составные фильтры запроса

Для большей гибкости вы можете использовать несколько фильтров, объединяя их с помощью логических операторов. Синтаксис составного фильтра выглядит следующим образом:

```
(Boolean-operator(filter)(filter)(filter)...) 
```

В Таблице 3 приведены логические операторы (Boolean-operator), используемые в составных фильтрах:

Логический оператор	Комментарий
&	Логическое И. Фильтр возвращает те записи, которые удовлетворяют всем указанным условиям, например: (&(uid=user)(uidNumber=180003))
	Логическое ИЛИ. Фильтр возвращает те записи, которые удовлетворяют одному из указанных условий, например: ((uid=user1)(uid=user2))
!	Логическое НЕ. Фильтр возвращает те записи, которые не удовлетворяют указанному условию, например: (!(uid=admin))

Таблица 3 Логические операторы для составного фильтра

Например, найдем пользователей из группы admins, которые два и более месяца не меняли пароль:

```
ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(&(memberof=cn=admins*)(krbLastPwdChange>=$(date +%Y%m%d000000Z" -d "-60 days")))"
cn krbLastPwdChange
```

Результат выполнения:

```
dn: uid=admin,cn=users,cn=accounts,
cn: Administrator
krbLastPwdChange: 20230324160900Z
```

В результате запроса, мы видим список администраторов, у которых пароль изменялся за последние 60 дней, обратите внимание на то, что можно добавлять подкоманды через конструкцию: `$(date +%Y%m%d000000Z" -d "-60 days")` между двойными кавычками фильтра. Так мы вычислим дату и подставим ее в фильтр в нужном формате.

3.2.2.2 Получение схемы объектных классов

Объектные классы описаны в операционном атрибуте `objectClasses` из специальной записи схемы «`cn=schema`». Посмотреть список всех классов можно командой:

```
ldapsearch -Q -LLL -o ldif-wrap=no -s base -b "cn=schema" objectClasses
```

Результат выполнения:

```
dn: cn=schema
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass X-ORIGIN 'RFC 45
12' )
objectClasses: ( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST aliasedObjectNam
```

```
e X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.20.1 NAME 'subschema' AUXILIARY MAY ( dITStructureRules $
  nameForms $ dITContentRules $ objectClasses $ attributeTypes $ matchingRules
  $ matchingRuleUse ) X-ORIGIN 'RFC 4512' )
objectClasses: ( 1.3.6.1.4.1.1466.101.120.111 NAME 'extensibleObject' SUP top
  AUXILIARY X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.11 NAME 'applicationProcess' SUP top STRUCTURAL MUST cn
  MAY ( seeAlso $ ou $ l $ description ) X-ORIGIN 'RFC 4519' )
...
```

В результате выведется список всех классов, в котором можно искать через команду grep:

```
ldapsearch -Q -LLL -o ldif-wrap=no -s base -b "cn=schema" objectClasses | grep --
color posix
```

Результат выполнения:

```
objectClasses: ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' DESC 'Standard LDAP objectclass'
  SUP top AUXILIARY MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory ) MAY
  ( userPassword $ loginShell $ gecos $ description ) X-ORIGIN 'RFC 2307' )
objectClasses: ( 1.3.6.1.1.1.2.2 NAME 'posixGroup' DESC 'Standard LDAP objectclass'
  SUP top STRUCTURAL MUST ( cn $ gidNumber ) MAY ( userPassword $ memberUid $
  description ) X-ORIGIN 'RFC 2307' )
objectClasses: ( 2.16.840.1.113730.3.2.326 NAME 'dynamicGroup' DESC 'Group containing
  internal dynamically-generated members' SUP posixGroup AUXILIARY MAY dsOnlyMemberUid
  X-ORIGIN 'Red Hat Directory Server' )
```

3.2.2.3 Получение схемы атрибутов

Также, как и объектные классы, атрибуты описаны через операционный атрибут **«attributeTypes»** корневой записи схемы **«cn=schema»**. Посмотреть список всех атрибутов можно командой:

```
ldapsearch -Q -LLL -o ldif-wrap=no -s base -b "cn=schema" attributeTypes
```

Результат выполнения:

```
dn: cn=schema
attributeTypes: ( 2.16.840.1.113730.3.1.582 NAME 'nsDS5ReplicaCredentials' DES
  C 'Netscape defined attribute type' SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGL
  E-VALUE X-ORIGIN 'Netscape Directory Server' )
attributeTypes: ( 2.16.840.1.113730.3.8.22.1.2 NAME 'ipaCertMapMapRule' DESC '
  Certificate Mapping Rule' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X
  -ORIGIN 'IPA v4.5' )
attributeTypes: ( 1.3.6.1.4.1.15953.9.1.1 NAME 'sudoUser' DESC 'User(s) who ma
  y run sudo' EQUALITY caseExactIA5Match SUBSTR caseExactIA5SubstringsMatch SY
  NTAX 1.3.6.1.4.1.1466.115.121.1.26 X-ORIGIN 'SUDO' )
...
```

Результат команды также можно перенаправить в файл командой перенаправления «>»:

```
ldapsearch -Q -LLL -o ldif-wrap=no -s base -b "cn=schema" attributeTypes > attrs
```

3.2.3 Раздел 2.2.3. Формат данных LDIF

Как упоминалось ранее, утилита `ldapsearch` возвращает результаты в формате LDIF, LDAP Data Interchange Format – это формат представления записей службы каталогов или их изменений в текстовой форме. Записи каталога или их изменения представляются набором LDIF-записей, по одной на каждую запись каталога или изменение. LDIF был разработан в начале 90-х годов и был доработан для использования с LDAPv3. Описание формата опубликовано в RFC 4525. Рассмотрим пример структуры LDIF:

```
dn: уникальное имя
имя атрибута: значение атрибута
имя атрибута:: base64 значение атрибута

dn: уникальное имя
имя атрибута: значение атрибута
имя атрибута:< значение атрибута url
```

Записи каталога представляются группами строк, разделенных пустой строкой, при этом каждая строка в группе представляет отдельное значение атрибута записи. Первая строка в группе должна представлять уникальное имя записи DN. Значение атрибута записывается в кодировке ASCII и отделяется от его имени символом «:». Значения, не подходящие под эту кодировку, записываются в кодировке base64 и отделяются от имени атрибута символами «::». Данный формат можно использовать для хранения данных, добавления и модификации записей в каталоге. Рассмотрим пример LDIF файла добавления записи в каталог нового подразделения **add_dp.ldif** с таким содержанием:

```
dn:
ou=marketing,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
objectClass: rbta-org-unit
objectClass: top
ou: marketing
displayName: Маркетинг
```

НА ЗАМЕТКУ!

В конце LDIF стоит пустая строка, которая определяет разделение между записями. Даже если запись одна нужно ставить разделитель из пустой строки.

При модификации используется директива `changetype`, которая может принимать значения: `add`, `modify`, `replace`, `delete` и `moddn`. Более подробно мы рассмотрим в примере **ldapmodify**.

3.2.4 Раздел 2.2.4. Добавление записи в каталог через `ldapadd`

Добавление в каталог информации можно утилитами **ldapadd** и **ldapmodify**, где **ldapadd** это символическая ссылка на команду **ldapmodify**, поэтому можно использовать **ldapmodify -a**. Давайте

добавим новое подразделение через описанный выше LDIF файл **add_dp.ldif** командой перенаправления «<»:

```
Ldapadd -Q < add_dp.ldif
```

Результат выполнения:

```
adding new entry "ou=marketing,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local"
```

В результате выводится сообщение adding new entry и DN новой записи.

Второй способ добавления – это работа с утилитой в интерактивном режиме для этого запускаем команду **ldapadd**:

```
ldapadd -Q
```

Далее напечатаем или вставим из буфера обмена текст записи в формате LDIF, в конце которого должна быть пустая строка:

```
dn:
ou=develop,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
objectClass: rbta-org-unit
objectClass: top
ou: develop
displayName: Разработка ПО
```

Как мы видим курсор ждет ввода символов или сигналов, поэтому отправим терминалу сигнал EOF командой **Ctrl+<d>**

```
Ctrl + <d>
```

Результат выполнения:

```
objectClass: rbta-org-unit
objectClass: top
ou: Develop
displayName: Разработка ПО
adding new entry "ou=Develop,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local"
```

Программа сообщит об успешном добавлении «adding new entry». Кстати, для выхода без изменений можно отправить другой сигнал SIGINT **Ctrl + <c>**. Использовать интерактивный режим для скриптов не рекомендуется, потому что может произойти событие ожидания ввода и скрипт зависнет, ожидая ввода пользователя. Поэтому в написании скриптов для команд используют параметр -u для подтверждения всех запросов. Далее мы будем использовать первый подход с перенаправлением.

3.2.5 Раздел 2.2.5. Модификация записей в каталоге через ldapmodify

Как мы уже говорили есть директива **changetype**, она должна следовать сразу после **dn**. Это нужно для того, чтобы утилита понимала, которую запись мы изменяем. Если в LDIF отсутствует директива **changetype**, то по умолчанию подразумевается «**changetype: add**» добавление записи в каталог.

3.2.5.1 Директивы changetype: modify и add:

Данная директива LDIF позволяет добавить атрибут к записи, но есть атрибуты, которые могут быть только в единичном числе, например, атрибут **displayName**, то при добавлении второго атрибута возникнет ошибка «ldap_add: Already exists (68)». В качестве примера добавим к нашему пользователю admin новую локацию атрибут «L» файл **add_loc.ldif**:

```
dn: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local
changetype: modify
add: l
l: Казань
```

Обратите внимание на пустую строку - это разделитель между записями и ее нужно добавлять даже мы оперируем с одной записью. А затем запустим команду модификации **ldapmodify**:

```
ldapmodify -Q < add_loc.ldif
```

Или можно передать по конвейеру

```
cat add_loc.ldif | ldapmodify -Q
```

Результат выполнения:

```
modifying entry "uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
```

Мы видим успешное изменение записи

3.2.5.2 Директивы changetype: modify и replace:

Переместим пользователя в другой департамент изменив ему атрибут **rbtadb**. Создадим файл **changedp.ldif**:

```
dn: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local
changetype: modify
replace: rbtadp
rbtadp:
ou=marketing,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
```

А затем запустим команду модификации `ldapmodify`:

```
ldapmodify -Q < changedp.ldif
```

Результат выполнения:

```
modifying entry "uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
```

Мы видим успешное изменение записи, которое можно проверить в Apache Directory Studio.

3.2.5.3 Директивы **changetype: modify** и **delete**:

Если нам необходимо изменить несколько атрибутов у записи, то можно несколько операций разделить с новой строки символом «-». Пример изменений нескольких атрибутов удаляет атрибут «l: Казань» и добавляет «l: Москва». Создадим файл **change_two_attrs.ldif**:

```
dn: uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local
changetype: modify
delete: l
l: Казань
-
add: l
l: Москва
```

Запустим команду **ldapmodify**, указав файл **change_two_attrs.ldif** через параметр **-f**:

```
ldapmodify -Q -f change_two_attrs.ldif
```

Результат выполнения:

```
modifying entry "uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
```

Мы видим успешное изменение записи

Директивы **changetype: modrdn** и **newrdn**:

Директива **modrdn** изменяет RDN записи, другими словами переименовывает запись. Например, переименуем подразделение Разработки ПО «ou=develop» на «ou=arch» архитекторов через файл **rename_rdn.ldif**, в котором директива «deleteoldrdn: 1» удалит старый DN:

```
dn:
ou=develop,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
changetype: modrdn
newrdn: ou=arch
deleteoldrdn: 1

dn: ou=arch,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
```



```
changetype: modify
replace: displayName
displayName: Архитекторы
```

Запустим команду `ldapmodify`, перенаправив файл `rename_rdn.ldif`:

```
ldapmodify -Q < rename_rdn.ldif
```

Результат выполнения:

```
modifying rdn of entry
"ou=develop,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local"

modifying entry
"ou=arch,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local"
```

Мы видим успешное переименование RDN и изменение атрибута `displayName` для нового подразделения

Директива **changetype: delete**

Если нам необходимо удалить запись из каталога, то используем директиву `changetype: delete`, однако, у записи не должно быть ни одной дочерней записи. Например, удалим группу «`ou=arch`» через файл **`del_ou.ldif`**:

```
dn: ou=arch,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
changetype: delete
```

Проверим результат, запустив команду:

```
ldapmodify -Q < del_ou.ldif
```

Результат выполнения:

```
deleting entry
"ou=arch,ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local"
```

Мы видим успешное удаление записи, а если запись будет не найдена, то выведется сообщение об ошибке:

```
ldap_delete: No such object (32)
    matched DN:
ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
```

3.2.6 Раздел 2.2.6. Работа с кириллицей и Unicode

Если атрибуты содержат значения не в ASCII кодировке, то утилита `ldapsearch` при выводе значений представляет их в base64 кодировке. В нашем примере, при создании пользователя `ppetrov` атрибуту `displayName` было присвоено значение "Петров П.П.". Посмотрим на вывод команды `ldapsearch`:

```
ldapsearch -Q -LLL -s base -b
"uid=ppetrov,cn=users,cn=accounts,dc=ald,dc=company,dc=local" displayName
```

Результат выполнения:

```
dn: uid=petrov.pp,cn=users,cn=accounts,dc=ald,dc=company,dc=local
displayName: 0J/QtdGC0YDQvtCyINCfLtCfLg==
```

Поскольку `ldapsearch` понимает только ASCII символы, она представила значение `displayName` в неудобной для чтения base64 строке. Мы сможем увидеть исходное значение, только сделав обратное преобразование:

```
echo '0J/QtdGC0YDQvtCyINCfLtCfLg==' | base64 -d
```

Результат выполнения:

```
Петров П.П.
```

Для того, чтобы получить результат сразу в удобном для автоматизации виде, можно использовать команду:

```
ldapsearch -Q -LLL -s base -b
"uid=ppetrov,cn=users,cn=accounts,dc=ald,dc=company,dc=local" displayName | perl
-MMIME::Base64 -Mutf8 -pe 's/^([-a-zA-Z0-9;]+):(:\s+)(\S+)\$/$1.$2.&decode_base64($3)/e'
```

Результат выполнения:

```
dn: uid=petrov.pp,cn=users,cn=accounts,dc=ald,dc=company,dc=local
displayName: Петров П.П.
```

Отметим, что не требуется производить какие-то дополнительные действия в фильтрах при поиске по строкам, не содержащим ASCII символы:

```
ldapsearch -Q -LLL -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(displayName=Петров П.П.)" cn
```

Результат выполнения:

```
dn: uid=petrov.pp,cn=users,cn=accounts,dc=ald,dc=company,dc=local  
cn: Petr Petrov
```

4 Раздел 3. Примеры автоматизации

В предыдущих разделах объясняется, как можно управлять каталогом через LDAP-протокол, это своего рода «кирпичики», из которых можно «строить дома», т.е. решать сложные задачи автоматизации. Давайте рассмотрим примеры на языках `bash` и `python`, а также поработаем с форматами CSV, LDIF, JSON.

4.1 Раздел 3.1. Работа с объектами из командной строки `bash`

В операционных системах Microsoft Windows для решения задач автоматизации раньше использовали командную оболочку и интерпретатор командной строки CMD. На замену ему пришел более мощный объектно-ориентированный PowerShell, который предоставляет удобные инструменты для работы с датами, хеш-таблицами, словарями, CSV и JSON объектами, а также имеет множество дополнительных модулей, например, для взаимодействия с каталогом Active Directory, объектной моделью офисных приложений и пр.

В мире Linux есть множество оболочек и интерпретаторов, например, `sh`, `bash`, `zsh` и др., каждый из которых предлагает свой вариант синтаксиса и может вызывать утилиты, доступные в операционной системе, например, `ls`, `ping`, `sed`, `cat`, `cut` и др. Но все они в какой-то степени являются эквивалентом CMD, т.к. работают только с текстовыми переменными и не поддерживают объекты.

Обрабатывать текстовый поток, получаемый от других приложений, средствами командной строки довольно сложно, поэтому `bash` целесообразно дополнить возможностями языка программирования Python, который в мире Linux можно считать эквивалентом PowerShell, по крайней мере в части удобства работы с объектами. Мы покажем вам примеры скриптов Python для конвертации LDIF в JSON/CSV, и как с этими данными можно дальше работать с помощью утилит `jq/cut`.

4.1.1 Раздел 3.1.1 Конвертер `ldif2csv` и генерация отчетов

Обрабатывать LDIF довольно сложно, т.к. данные представлены в блоках строк, а нужные атрибуты могут отсутствовать. Вот пример Python скрипта, который позволяет конвертировать поток LDIF в CSV, для его использования создайте файл **`ldif2csv.py`** и скопируйте туда следующее содержимое:

```
#!/usr/bin/python3
import sys
import base64
import re

if sys.version_info[0] < 3:
    raise Exception("Use Python 3: python3 ldif2csv.py")

# read from stdin
data = sys.stdin.readlines()
header_string = ""
atrcheck = ""
entry = ""
dic_entries = {}
dic_entry = {}
current_dn = ""
val_base64 = False
max_size_cell = 32000 # limit chars in excel cell
```

```

# main loop for parse headers and collect dict
for line in data:
    ln = line.replace("\n", "").replace("\r", "")
    if len(ln) == 0:
        if not current_dn == "":
            dic_entries[current_dn] = dic_entry
            dic_entry = {}
            entry = ""
    else:
        if ln.startswith("version"):
            #dic_entries["version"] = ln.split(": ")[1]
            continue
        elif ln.lstrip()[0] == "#":
            continue
        elif ln[0] == " ":
            # if line wrapped line starts with " " then add line to last attr
            dic_entry[atrcheck] += ln.lstrip()
            if val_base64:
                val_to_decode = dic_entry[atrcheck]
                try:
                    dic_entry[atrcheck] = base64.b64decode(val_to_decode).decode(
                        'utf-8').strip()
                except:
                    dic_entry[atrcheck] = val_to_decode
            continue
        entry += ln
        attribute = []
        attribute_name = ""
        attribute_value = ""
        if ln.find(":: ") > 0:
            val_base64 = True
            attribute = ln.split(":: ")
            try:
                attribute_value = base64.b64decode(
                    attribute[1]).decode('utf-8').strip()
            except:
                attribute_value = attribute[1]
        elif ln.find("< ") > 0:
            val_base64 = False
            attribute = ln.split("< ")
            attribute_value = attribute[1]
        else:
            val_base64 = False
            attribute = ln.split(": ")
            try:
                attribute_value = re.sub(r"^.*?: ", "", ln)
            except:
                attribute_value = ""

        atrcheck = attribute[0].replace(":", "")
        # get attribute and check if attribute exist

        if dic_entry.get(atrcheck):

```

```

        new_len = len(dic_entry[atrcheck]) + len("|" + str(attribute_value))
        if new_len < max_size_cell:
            dic_entry[atrcheck] = str(
                dic_entry[atrcheck]) + "|" + str(attribute_value)
        else:
            dic_entry[atrcheck] = attribute_value
        if atrcheck == "dn":
            current_dn = attribute[1]
    if header_string.find(atrcheck) < 0:
        if header_string == "":
            header_string += atrcheck
        else:
            header_string += ";" + atrcheck
# add row if row not empty
if entry != "":
    dic_entries[current_dn] = dic_entry
    dic_entry = {}
    entry = ""

# print utf-8-BOM and headers
print('\u00ff' + "\"" + header_string.replace(";", "\\;") + "\"")
# print data
for d in dic_entries:
    od = dic_entries[d]
    csv_row = "" # row csv value
    split_char = "" # spliter for values
    for column in header_string.split(";"):
        if len(csv_row) > 0:
            split_char = ";" # need split because csv row have chars
            find_column = od.get(column)
            if find_column and find_column.strip():
                csv_row += split_char + "\"" + od[column].replace("\", "\\") + "\""
            else:
                csv_row += split_char + "\"\""
    print(csv_row)

```

Рассмотрим пример конвертации данных из **ldapsearch** через конвейер:

```

ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(uid=*)" uid displayName sn mail rbtadb | python3 ldif2csv.py

```

Результат выполнения:

```

"dn";"uid";"sn";"displayName";"mail"
"uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"admin";"Administrator";
"";""
"uid=petrovvp,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"petrovvp";"Петров";
"Петров петр";"petrovvp@ald.company.local"
"uid=ivani,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"ivani";"Иванов";"Иван
Иванов";"ivani@ald.company.local"

```

```
"uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"petrovss";"Сидоров";
"Петр С.";petrov.ss@ald.company.local"
```

Вы можете сохранить этот вывод в файл **users.csv** простым перенаправлением:

```
ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(uid=*)" uid displayName sn mail rbtadb | python3 ldif2csv.py > users.csv
```

На рисунке 17 можно увидеть, как будет выглядеть содержимое файла, если его открыть в LibreOffice Calc.

	A	B	C	D	E	F	G
1	dn	uid	sn	displayName	mail		
2	uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local	admin	Administrator				
3	uid=petrov,cn=users,cn=accounts,dc=ald,dc=company,dc=local	petrov	Петров	Петров петр	petrov@ald.company.local		
4	uid=ivani,cn=users,cn=accounts,dc=ald,dc=company,dc=local	ivani	Иванов	Иван Иванов	ivani@ald.company.local		
5	uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local	petrovss	Сидоров	Петр С.	petrov.ss@ald.company.local		
6							
7							
8							
9							
10							
11							
12							
13							
14							

Рис 17. Файл, экспортированный с помощью *ldif2csv.py*, открытый в LibreOffice Calc

Далее вы можете использовать полученный CSV-файл следующим образом:

```
#!/bin/bash
csv_file=$1
cat $csv_file | while read line
do
    if [ ! -z "${line}" ]; then
        if [[ "${line:0:1}" == "\"" ]]; then
            column1=$(echo ${line} | cut -d ";" -f 1 | sed "s/^\"//g" | sed "s/\"$/g")
            column2=$(echo ${line} | cut -d ";" -f 2 | sed "s/^\"//g" | sed "s/\"$/g")
            column3=$(echo ${line} | cut -d ";" -f 3 | sed "s/^\"//g" | sed "s/\"$/g")
            echo -e "${column1}\t${column2}\t${column3}"
            # можете добавить свои действия по обработке данных из файла
        fi
    fi
done
```

Если сохранить приведенный скрипт в файл *parse_csv.sh*, то ему можно передать имя csv-файла для обработки, как параметр. Не забудьте назначить скрипту права на выполнение:

```
chmod +x ./parse_csv.sh
./parse_csv.sh users.csv
```

Результат выполнения:

uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local	admin	Administrator
uid=petrov, cn=users, cn=accounts, dc=ald, dc=company, dc=local	petrov	Петров
uid=ivani, cn=users, cn=accounts, dc=ald, dc=company, dc=local	ivani	Иванов
uid=petrovss, cn=users, cn=accounts, dc=ald, dc=company, dc=local	petrovss	

Сидоров

Как мы видим, выводится несколько колонок с разделителем табуляции «\t».

4.1.2 Раздел 3.1.2. Конвертер ldif2json и работа с JSONPath

Для работы со структурированными данными можно также использовать формат JSON, и в Linux есть очень удобный процессор JSON, который называется jq. Вот пример Python скрипта, который позволяет конвертировать поток LDIF в JSON, для его использования создайте файл **ldif2json.py** и скопируйте туда следующее содержимое:

```
#!/usr/bin/python3
import sys
import base64
import json
import re
# parse bool and int to json
def parse_value(value):
    try:
        return int(value)
    except:
        if value == "FALSE" or value == "FALSE": return bool(value)
        else: return value

if sys.version_info[0] < 3:
    raise Exception("Use Python 3: python3 ldif2json.py")

data = sys.stdin.readlines()
header_string = ""
atrcheck = ""
entry = ""
dic_entries = {}
dic_entry = {}
current_dn = ""
val_base64 = False
# main loop for parse headers and collect dict
for line in data:
    ln = line.replace("\n", "").replace("\r", "")
    if len(ln) == 0:
        if not current_dn == "":
```



```

dic_entries[current_dn] = dic_entry
dic_entry = {}
entry = ""
else:
    if ln.startswith("version"):
        dic_entries["version"] = ln.split(": ")[1]
        continue
    elif ln.lstrip()[0] == "#":
        continue

# if line wrapped line starts with " " then add line to last attr
elif ln[0] == " ":
    dic_entry[atrcheck] += ln.lstrip()
    if val_base64:
        val_to_decode = dic_entry[atrcheck]
        try:
            dic_entry[atrcheck] = base64.b64decode(val_to_decode).decode(
                'utf-8').strip()
        except:
            dic_entry[atrcheck] = val_to_decode
        continue
    entry += ln
    attribute = []
    attribute_name = ""
    attribute_value = ""
    if ln.find(":: ") > 0:
        val_base64 = True
        attribute = ln.split ":: ")
        try:
            attribute_value = base64.b64decode(
                attribute[1]).decode('utf-8').strip()
        except:
            attribute_value = attribute[1]
    elif ln.find("< ") > 0:
        val_base64 = False
        attribute = ln.split("< ")
        attribute_value = attribute[1]
    else:
        val_base64 = False
        attribute = ln.split(": ")
        try:
            attribute_value = re.sub(r"^\.*?: ", "", ln)
        except:
            attribute_value = ""
    atrcheck = attribute[0].replace(":", "")
    # get attribute and check if attribute exist

    if dic_entry.get(atrcheck):
        dic_entry[atrcheck] = str(
            dic_entry[atrcheck]) + "|" + str(attribute_value)
    else:
        dic_entry[atrcheck] = parce_value(attribute_value)
    if atrcheck == "dn":

```

```

        current_dn = attribute[1]
    if header_string.find(atrcheck) < 0:
        if header_string == "":
            header_string += atrcheck
        else:
            header_string += ";" + atrcheck
# add row if row not empty
if entry != "":
    dic_entries[current_dn] = dic_entry
    dic_entry = {}
    entry = ""

# print stdout json from dict
print(json.dumps(dic_entries, ensure_ascii=False))

```

Рассмотрим пример конвертации данных из **ldapsearch** через конвейер:

```

ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(uid=*)" uid displayName sn mail rbtadb | python3 ldif2json.py

```

Результат выполнения:

```

{
  "uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local": {
    "dn": "uid=admin,cn=users,cn=accounts,dc=ald,dc=company,dc=local",
    "uid": "admin",
    "sn": "Administrator"
  },
  "uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local": {
    "dn": "uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local",
    "uid": "petrovss",
    "displayName": "Петр С.",
    "sn": "Сидоров",
    "mail": "petrov.ss@ald.company.local"
  },...
}

```

Вы можете сохранить этот вывод в файл **users.json** простым перенаправлением и далее с помощью утилиты **jq** из одноименного пакета извлекать любые данные с помощью запросов JSONPath. Например, узнаем значение атрибута **mail** для пользователя **petrovss**:

```

ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(uid=*)" uid displayName sn mail rbtadb | python3 ldif2json.py > users.json && jq -r
'."uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local".mail' users.json

```

где:

- параметр **-r** – означает, что выводить нужно сырые данные без кавычек;
- **'."uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local".mail'** - это текст запроса JSONPath

Результат выполнения:

```
petrov.ss@ald.company.local
```

Давайте обработаем полученные JSON данные в цикле, создайте файл `json_cycle.sh` со следующим содержимым:

```
#!/bin/bash
echo -e "uid\tdisplayName\tsn\tmail"
ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local"
"(uid=*)" uid displayName sn mail rbtadb | python3 ldif2json.py > users.json
cat users.json | jq -r 'delpaths([paths | select(length > 1)])' | jq -r '[paths |
join(".")] | jq -r 'join("\n")' | while read key
do
uid=$(jq -r ".$key.uid" users.json)
mail=$(jq -r ".$key.mail" users.json)
displayName=$(jq -r ".$key.displayName" users.json)
sn=$(jq -r ".$key.sn" users.json)
echo -e "$uid\t$displayName\t$sn\t$mail"
# можете добавить свои действия по обработке данных из файла
done
```

Результат выполнения:

```
uid      displayName      sn      mail
admin    null      Administrator null
petrovp  Петров петр  Петров  petrov@ald.company.local
ivani    Иван Иванов  Иванов  ivani@ald.company.local
petrovss      Петр С. Сидоров petrov.ss@ald.company.local
```

4.2 Раздел 3.2. Добавление пользователей

Добавим нового пользователя `petrov.ss` с помощью `ldapadd`. В этом примере при добавлении пользователя `objectClass` класс с именем **`ipantuserattrs`** не добавляется, чтобы атрибут **`ipaNTSecurityIdentifier`** сгенерировался автоматически. Рассмотрим файл **`add_user.ldif`**:

```
dn: uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local
objectClass: top
objectClass: person
objectClass: organizationalperson
objectClass: inetorgperson
objectClass: inetuser
objectClass: posixaccount
objectClass: krbprincipalaux
objectClass: krbticketpolicyaux
objectClass: ipaobject
objectClass: ipasshuser
```

```

objectClass: x-ald-user
objectClass: x-ald-user-parsecl4
objectClass: x-ald-audit-policy
objectClass: ruPostMailAccount
objectClass: rbtaCustomUserAttrs
objectClass: rbtaUserMeta
objectClass: rbta-unit
objectClass: rbta-address
objectClass: rbta-inetorgperson-ext
objectClass: ipaSshGroupOfPubKeys
objectClass: mepOriginEntry
givenName: Петр
sn: Сидоров
uid: petrovss
cn: petrov.ss
uidNumber: -1
gidNumber: -1
displayName: Петр С.
initials: P.C.
gecos: Тел. +71234567890
rbtamiddlename: Сидоров
rbtadp: ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
loginShell: /bin/bash
homeDirectory: /home/petrov.ss
mail: petrov.ss@ald.company.local
x-ald-user-mac: 0:0x0:0:0x0
krbCanonicalName: petrov.ss@ALD.COMPANY.LOCAL
krbPrincipalName: petrov.ss@ALD.COMPANY.LOCAL
userPassword: somepassword

```

ВАЖНО!

Атрибуты `uidNumber` и `gidNumber` нужно устанавливать равными -1, в этом случае DNA-плагин автоматически сгенерирует значения идентификаторов при добавлении пользователя. Пароль следует передавать открытым текстом. Он будет хеширован алгоритмом PBKDF2_SHA256 автоматически. Записать пользователю, сгенерированный где-то в другом месте hash пароля возможно только при создании пользователя, если сервер переведен в режим миграции.

Выполним команду по добавлению нового пользователя:

```
ldapadd -Q < add_user.ldif
```

Результат выполнения:

```
adding new entry "uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
```

В результате мы увидим успешное выполнение. После входа пользователю нужно будет задать новый пароль, так как пароль назначенный таким способом является временным.

Теперь добавим пользователей в цикле, используя данные из CSV файла. Создадим в `cycleadd.csv` разделителем «;» в любой программе:

```
givenName;sn;uid
Александр;Петров;alexp
Михаил;Иванов;mikhaili
Артём;Сидоров;artems
```

Создадим скрипт **add_from_csv.sh** по добавлению пользователей из CSV файла:

```
#!/bin/bash
csv_file=$1
delim=$2
function template()
{
    psw=$(cat /dev/urandom| tr -dc '0-9a-zA-Z!@#%&*+-' | head -c 14;echo;)
    uid=$1 #установить uid из первого параметра
    givenName=$2 #установить Имя из второго
    sn=$3 #установить Фамилию из третьего
    cat <<EOF
dn: uid=$uid,cn=users,cn=accounts,dc=ald,dc=company,dc=local
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalperson
objectClass: inetorgperson
objectClass: inetuser
objectClass: posixaccount
objectClass: krbprincipalaux
objectClass: krbticketpolicyaux
objectClass: ipaobject
objectClass: ipasshuser
objectClass: x-ald-user
objectClass: x-ald-user-parsec14
objectClass: x-ald-audit-policy
objectClass: ruPostMailAccount
objectClass: rbtaCustomUserAttrs
objectClass: rbtaUserMeta
objectClass: rbta-unit
objectClass: rbta-address
objectClass: rbta-inetorgperson-ext
objectClass: ipaSshGroupOfPubKeys
objectClass: mepOriginEntry
givenName: $givenName
sn: $sn
uid: $uid
cn: $uid
uidNumber: -1
gidNumber: -1
displayName: $givenName ${sn:0:1}.
initials: ${givenName:0:1}. ${sn:0:1}.
rbtamiddlename: $sn
rbtadp: ou=ald.company.local,cn=orgunits,cn=accounts,dc=ald,dc=company,dc=local
loginShell: /bin/bash
```

```

homeDirectory: /home/$uid
mail: $uid@ald.company.local
x-ald-user-mac: 0:0x0:0:0x0
krbCanonicalName: $uid@ALD.COMPANY.LOCAL
krbPrincipalName: $uid@ALD.COMPANY.LOCAL
userPassword: $psw

EOF
}
echo Добавление пользователей из $csv_file
truncate -s 0 cycleadd.ldif
chmod 500 ./cycleadd.ldif
while read line; do let c++;
if [ $c -gt 1 ]; then
    p_givenName=$(echo ${line} | cut -d $delim -f 1 | sed "s/^\\"//g" | sed "s/\\"$//g" |
sed "s/^\\"//g" | sed "s/\\"'$//g")
    p_sn=$(echo ${line} | cut -d $delim -f 2 | sed "s/^\\"//g" | sed "s/\\"$//g" | sed "s/
^\\"//g" | sed "s/\\"'$//g")
    p_uid=$(echo ${line} | cut -d $delim -f 3 | sed "s/^\\"//g" | sed "s/\\"$//g" | sed
"s/^\\"'$//g" | sed "s/\\"'$//g")
    template $p_uid $p_givenName $p_sn >> cycleadd.ldif
fi
done < $csv_file
# добавим пользователей одним пакетом
ldapadd -Q -c < cycleadd.ldif

```

Как мы видим, есть функция `template`, в которой описан шаблон одной записи. Этот шаблон вы можете настроить по своему усмотрению, прописав свои корневые суффиксы и реалмы.

Назначим права запуска скрипту **add_from_csv.sh** и запустим его:

```
chmod +x ./add_from_csv.sh && ./add_from_csv.sh cycleadd.csv ';
```

Результат выполнения:

```

Добавление пользователей из cycleadd.csv
adding new entry "uid=alexp,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
adding new entry "uid=mikhaili,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
adding new entry "uid=artems,cn=users,cn=accounts,dc=ald,dc=company,dc=local"

```

Скрипт **add_from_csv.sh** создал временный LDIF файл **cycleadd.ldif** и передал команде **ldapadd**. После чего мы можем создать новый файл **new_users.csv** уже после добавления пользователей, потому что там уже сохранены новые пароли пользователей:

```
cat cycleadd.ldif | python3 ldif2csv.py > new_users.csv
```

Файл **new_users.csv** можно скачать, чтоб потом отправить коллегам пароли и учетные данные для входа, см Рис 18.

new_users.csv - LibreOffice Calc

ФайлПравкаВидВставкаФорматСтилиЛистДанныеСервисОкноСправка

<

Рис 18. Файл с новыми пароллями для добавленных в цикле пользователей

4.2.1 Раздел 3.4. Смена пароля пользователей

В случае взлома системы нам нужно изменить пароли всех пользователей. Для этого напишем скрипт на языке bash **change_all_pass.sh**:

```
#!/bin/sh
read -p "Вы хотите создать новые пароли всем пользователям (Yes|no)? " yn
if [[ $yn =~ "Yes" ]]; then
    echo '"dn";"password"' > new_passwords.csv
    echo "" > tmp_new_pass
    ldapsearch -Q -LLL -s one -b "cn=users,cn=accounts,dc=ald,dc=company,dc=local" "(!
(uid=admin))" dn | while read line
    do
        if [ ! -z "${line}" ]; then
            psw=$(cat /dev/urandom| tr -dc '0-9a-zA-Z!@#%&*+_- ' | head -c 14;echo;)
            echo "\"$line\";\"$psw\"" >> new_passwords.csv
            echo -e "${line}\nchangetype: modify\nreplace: userPassword\nuserPassword:
$psw\n" >> tmp_new_pass
        fi
    done
    ldapmodify -Q < tmp_new_pass
fi
```

Запустим скрипт:

```
chmod +x change_all_pass.sh && ./change_all_pass.sh
```

Результат выполнения:

```

Вы хотите создать новые пароли всем пользователям (Yes|no)? Yes
modifying entry "uid=petrovvp,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
modifying entry "uid=ivani,cn=users,cn=accounts,dc=ald,dc=company,dc=local"
modifying entry "uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local"

```

Подготовим CSV файл **newpass.csv** с паролями из временного tmp_new_pass LDIF через конвертер **ldif2CSV.py**:

```
cat tmp_new_pass | python3 ldif2csv.py > newpass.csv
```

Результат выполнения:

```

"dn";"changetype";"replace";"userPassword"
"uid=petrovvp,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"modify";"userPassword"
;"l@q%yTU1hf_EP7"
"uid=ivani,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"modify";"userPassword";
"nM^ZER_z_sULpE"
"uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local";"modify";
"userPassword";"oxK*HzeFmGJw5w"

```

А также можно открыть файл **newpass.csv** в редакторе электронных таблиц, см. Рис 19.

	A	B	C	D	E
1	dn	changetype	replace	userPassword	
2	uid=petrovvp,cn=users,cn=accounts,dc=ald,dc=company,dc=local	modify	userPassword	l@q%yTU1hf_EP7	
3	uid=ivani,cn=users,cn=accounts,dc=ald,dc=company,dc=local	modify	userPassword	nM^ZER_z_sULpE	
4	uid=petrovss,cn=users,cn=accounts,dc=ald,dc=company,dc=local	modify	userPassword	oxK*HzeFmGJw5w	
5					
6					
7					
8					
9					
10					

Рис 19. Пакетное изменение паролей пользователей

4.3 Раздел 3.5. Проверка просроченных паролей

Проверим просроченные пароли пользователей с помощью скрипта **check_expired.sh**:

```
#!/bin/bash
```



```

# 1. Получить список пользователей из DN cn=users,cn=accounts,dc=ald,dc=company,dc=local
# 2. Отфильтровать список с полями uid, displayname, mail, krbPasswordExpiration
# 3. Оставить пользователей, у которых дата просрочки пароля (krbPasswordExpiration) от
даты сегодня до даты сегодня + 7 дней
ldapsearch -Q -LLL -o ldif-wrap=no -b
"cn=users,cn=accounts,dc=ald,dc=company,dc=local" "(&(krbPasswordExpiration>=$(date
+"%Y%m%d000000Z"))(krbPasswordExpiration<=$(date +"%Y%m%d000000Z" -d "+7 days")))"
uid displayname mail krbPasswordExpiration > expired_ldif
while read line
do # 4. Цикл пользователь из пользователей
if [ -z "${line}" ]; then
if [ ! -z $user_mail ]; then
# 4.1 Если почта у пользователя есть
timeExp=$(date -d "${date_expire:0:4}-${date_expire:4:2}-${date_expire:6:2} ${date_expire:8:2}:${date_expire:10:2}:${date_expire:12:2}Z" +"%s")
timeNow=$(date +"%s")
seconds=$(( timeExp - timeNow ))
min=$(( seconds / 60 ))
hours=$(( min / 60 ))
days=$(( hours / 24 ))
ago=$(echo "через $daysдн.")
if [ $timeNow -gt $timeExp ]; then
ago="просрочен!!"
fi
dateprint=$(date -d "@$timeExp")
echo "$user_name ($user_mail), пароль просрочится ${dateprint} $ago"
# Тут вы можете добавить свою обработку пользователя.
else
# 4.3 Иначе добавить пользователя в лог
echo "$(date) [expire.sh] uid $user_uid mail is empty" >> "expired.log"
fi
# очистим переменные для следующей записи
user_uid=""
user_name=""
user_mail=""
date_expire=""
else
attr=$(echo $line | cut -d ":" -f 1)
attvalue=$(echo $line | cut -d ":" -f 2)
if [ -z "$attvalue" ]; then
attvalue=$(echo $line | cut -d " " -f 2 | base64 -d)
fi
# проверка атрибута и присвоение переменной значения
case $attr in
uid) user_uid=$(echo $attvalue | xargs echo -n);;
displayname) user_name=$(echo $attvalue | xargs echo -n) ;;
mail) user_mail=$(echo $attvalue | xargs echo -n);;
krbPasswordExpiration) date_expire=$(echo $attvalue | xargs echo -n);;
esac
fi
done < expired_ldif

```

Поставим атрибут выполнения и запустим скрипт:

```
chmod +x ./check_expired.sh && ./check_expired.sh
```

Результат выполнения:

```
Петров петр (petrov@ald.company.local), пароль просрочится Вт мая 30 12:18:21 MSK 2023
через 2дн.
Александр П. (alex@ald.company.local), пароль просрочится Сб мая 27 12:46:42 MSK 2023
просрочен!!
Михаил И. (mikhail@ald.company.local), пароль просрочится Сб мая 27 12:46:42 MSK 2023
просрочен!!
Артём С. (artem@ald.company.local), пароль просрочится Сб мая 27 12:46:42 MSK 2023
просрочен!!
```

Мы обработали даты просрочки пароля, указав просроченные пароли и дни до их завершения. Заметьте, что в данном примере мы использовали только `bash` и вывод `ldapsearch`. Вы также можете добавить в скрипт свою обработку просроченных учетных записей. Например, отправку сообщения в корпоративный мессенджер или на электронную почту.

5 Заключение

LDAP разрабатывался с целью хранения любой информации об объектах на предприятии, таких как пользователи, компьютеры, серверы, подразделения и др. Информация удобно расположена в виде иерархического дерева, которое строго типизировано объектными классами. База данных ориентирована на чтение, где быстро и легко можно получить любую информацию по объектам.

На сегодняшний день LDAP-каталог – это стандарт. Множество продуктов имеют встроенные интеграции с службой каталога, а также существует большое количество библиотек для разных языков для доступа к LDAP серверу, например: `python-ldap` для `python`, `ldaptive` для `java`. В некоторые языки программирования уже встроена работа LDAP, например, в языки `PHP` и `C#`. А также у продукта ALD Pro (FreeIPA) есть REST API, через него можно управлять каталогом, используя простые Web запросы.

Мы рассмотрели автоматизацию через LDAP запросы, используя только некоторые инструменты командной строки. Научились получать данные по запросу, а также добавлять, изменять и удалять данные. Комбинируя разные примеры, написали несколько скриптов `bash` и `python` для решения задач администрирования. Надеемся, что данные примеры помогут Вам написать свои решения для автоматизации.