ООО «Базальт СПО»
Общество с ограниченной ответственностью
«Базальт свободное программное обеспечение»
127015, г. Москва, ул. Бутырская, д. 75, офис 307
Тел./факс: +7 495 123-4799

ОГРН 1157746734837
ИНН 7714350892
КПП 77140100

Приложение 3

Техническая документация к прототипу библиотеки libdomain проекта

«РАЗРАБОТКА ОТКРЫТОЙ БИБЛИОТЕКИ ДЛЯ УПРАВЛЕНИЯ ДОМЕННОЙ ИНФРАСТРУКТУРОЙ НА ОСНОВЕ СЛУЖБЫ КАТАЛОГОВ SAMBA»

Разработал:
Старший разработчик технической документации
Мишина Е. В. _____

Согласовал:
Системный аналитик (Руководитель проекта)
Глуховская А.Е.

_____

Москва 2023

Содержание

1. Назначение и цель документа

Документ содержит описание объектов и их свойств открытой библиотеки для управления доменной инфраструктурой на основе службы каталогов для линейки ОС «Альт».

2. Глоссарий

Перечень терминов и сокращений представлен в таблице 1.

Таблица 1. Термины и сокращения

Термин	Описание
LDAP	(от англ. Lightweight Directory Access Protocol) – «легковесный протокол доступа к каталогам» – открытый протокол, используемый для хранения и получения данных из каталога с иерархической структурой
LDIF	(от англ. LDAP Data Interchange Format) – формат представления записей службы каталогов или их изменений в текстовой форме
API	(от англ. Application programming interface) – интерфейс программирования приложений, интерфейс прикладного программирования
Qt	Кроссплатформенная библиотека разработки элементов интерфейса
GTK	Кроссплатформенная библиотека разработки элементов интерфейса
ОС	Операционная система
Active Directory (AD)	(от англ. Active Directory) – служба каталогов корпорации Microsoft для операционных систем семейства Windows Server
FreeIPA	(от англ. Free Identity, Policy and Audit) – открытый проект для создания централизованной системы для идентификации пользователей, задания политик доступа и аудита, система обеспечения безопасности в виртуализированных средах
OpenLDAP	Открытая реализация LDAP
OU	(от англ. Organizational Unit) – организационная единица, контейнерный объект внутри домена (может содержать в себе другие объекты, объединенные в древовидную структуру)
Домен Active Directory	Группа компьютеров, совместно использующих общую базу данных каталога
ОС Microsoft Windows®	Группа семейств коммерческих операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса

3. Назначение Решения

Решение, представляющее собой открытую библиотеку libdomain, которая абстрагирует доступ к доменной инфраструктуре и предоставляет высокоуровневое API для управления объектами службы каталогов.

Разрабатываемая библиотека имеет перспективы в области импортозамещения в части миграции с импортных доменов Microsoft Active Directory на отечественные разработки доменных инфраструктур.

4. Функционал библиотеки libdomain

Функциональные требования реализуются для следующих серверов каталогов:

- Microsoft AD DS version >= Windows Server 2008 R2
- Samba >= 4.14.0
- OpenLDAP >= 2.4.59-alt0.p9.1

Библиотека libdomain выполняет следующие функциональные требования:

1. Возможность подключения к серверу каталогов:
   - 1.1. Возможность подключения к серверу каталогов, используя различные настройки подключения:
     - 1.1.1. Анонимное подключение.
     - 1.1.2. Подключение, использующее SASL аутентификацию.
     - 1.1.3. Подключение, использующее TLS аутентификацию.
   - 1.2. Возможность загружать настройки подключения из файла.
2. Осуществление контроля установленного соединения:
   - 2.1. Установка параметров соединения:
     - 2.1.1. Установка времени ожидания (timeout)
   - 2.2. Возможность переподключения к серверу при потере соединения.
3. Возможность автоматического конфигурирования на основе kerberos:
   - 3.1. Получение принципала клиента по умолчанию из kerberos.
4. Получение и редактирование записей в LDAP каталоге:
   - 4.1. Добавление записи в LDAP каталог.

4.2. Редактирование записи в LDAP каталоге.

4.3. Переименовывание записи в LDAP каталоге.

4.4. Перемещение записи в LDAP каталоге.

4.5. Удаление записи в LDAP каталоге.

5. Получение и редактирование списка атрибутов:

5.1. Добавление атрибута.

5.2. Удаление атрибута.

6. Возможность осуществлять поиск данных в LDAP каталоге:

6.1. Установка базового DN для поиска.

6.2. Установка фильтра по типу объекта для поиска.

6.3. Установка списка атрибутов возвращаемых при поиске.

7. Осуществление следующих действий с пользователями и группами:

7.1. Изменение пароля пользователя.

7.2. Создание нового пользователя.

7.3. Создание новой группы.

7.4. Удаление и добавление пользователя в группу.

7.5. Блокирование/Разблокирование учётной записи пользователя.

8. Осуществление следующих действий с компьютерами:

8.1. Добавление компьютера.

8.2. Удаление компьютера.

5. Объекты библиотеки

В процессе выполнения задач первого этапа были собраны сведения и проанализированы объекты в домене Active Directory и их свойства.

По результатам анализа спроектирована системная архитектура прототипа.

5.1. Модули библиотеки

Библиотека libdomain состоит из следующих модулей:

- directory.{h,c} — обеспечивает определение текущей реализации сервера каталогов;
- domain.{h,c} — основной набор служебных функций библиотеки;
- connection.{h,c} — представляет набор методов для установки соединения с сервером каталогов;
- connection_state_machine.{h,c} — предоставляет методы для управления состоянием подключения;
- entry.{h,c} — предоставляет операции для работы с записями в LDAP каталоге;
- common.{h,c} — предоставляют методы общего назначения такие как сообщение об ошибках и набор общих флагов;
- user.{h,c} — представляет методы для работы с пользователями;
- group.{h,c} — представляет методы для работы с группами;
- computer.{h,c} — представляет методы для работы с компьютерами;
- organizational_unit.{h,c} — представляет методы для работы с организационными подразделениями.

5.2. Классы библиотеки

На рис. 1-4 представлена диаграмма классов библиотеки libdomain.

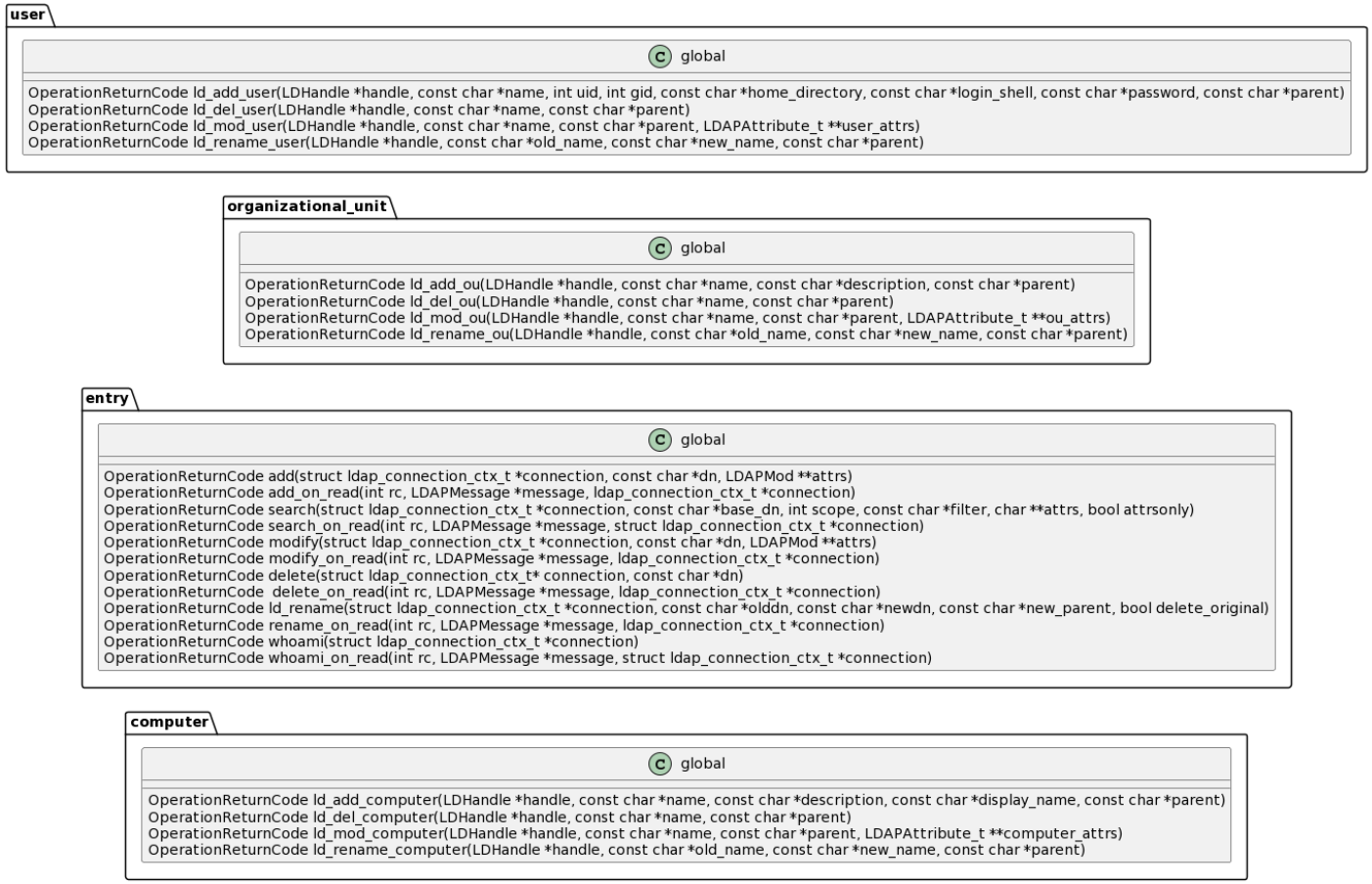

Рисунок 1. Диаграмма классов библиотеки libdomain

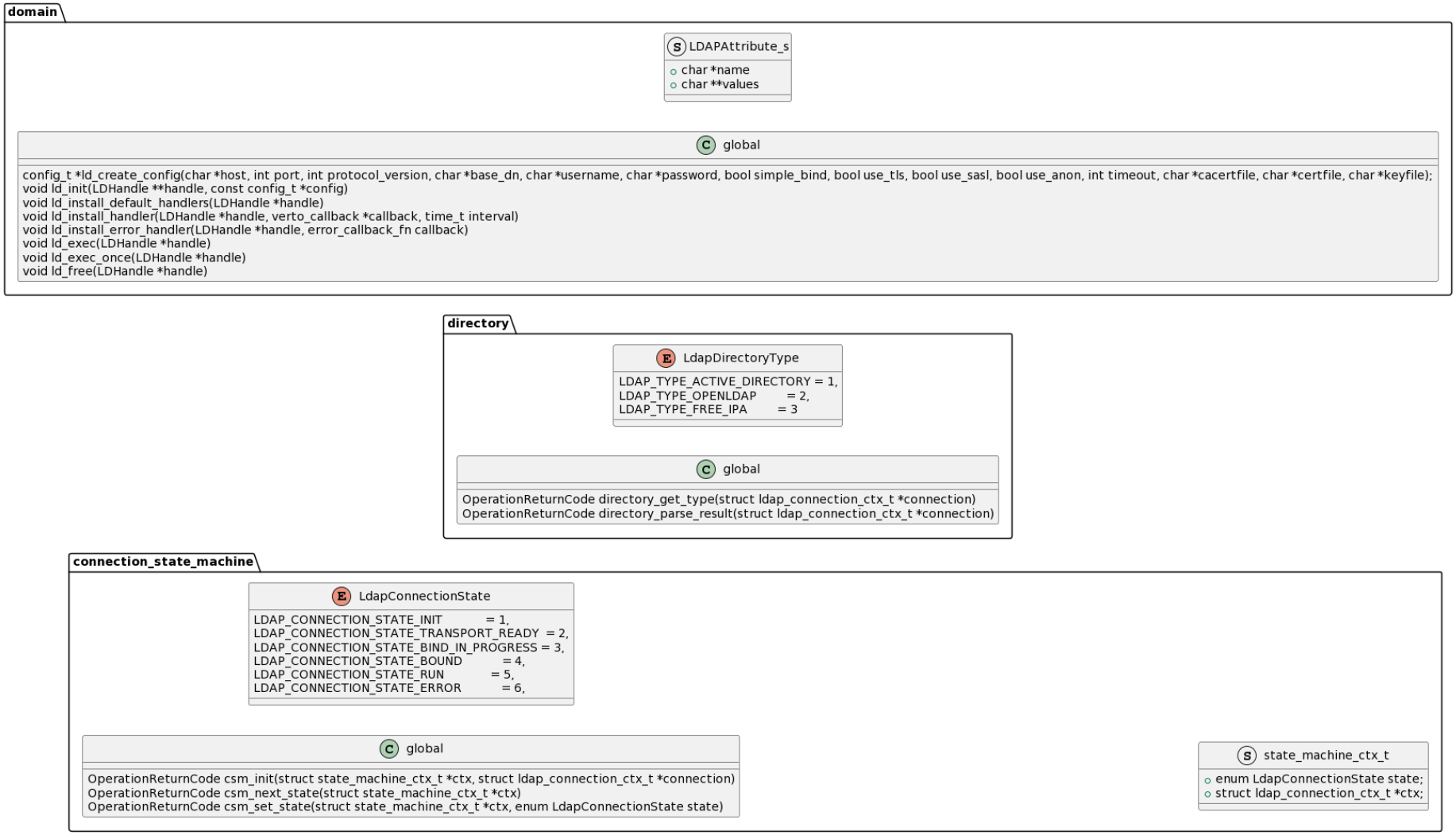

Рисунок 2. Диаграмма классов библиотеки libdomain

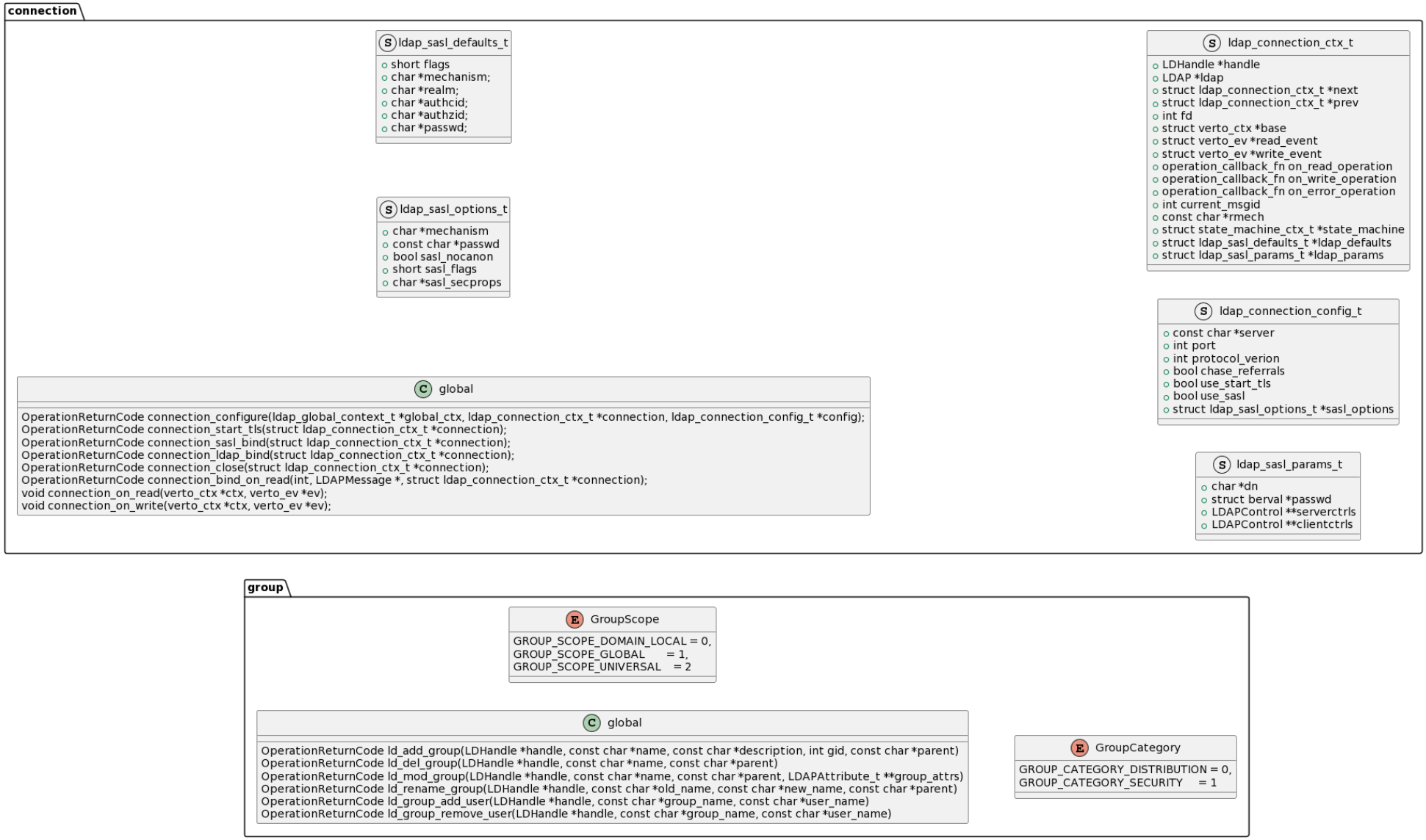

Рисунок 3. Диаграмма классов библиотеки libdomain

Рисунок 4. Диаграмма классов библиотеки libdomain

5.3. Структуры

5.3.1. Структура attribute_value_pair_s

Структура attribute_value_pair_s содержит пару: атрибут и его значение.

Публичные переменные attribute_value_pair_s представлены в Таблице 2.

Таблица 2. Публичные переменные attribute_value_pair_s

Имя	**Тип**	**Описание**
name	char*	Имя атрибута
value	char*	Значение атрибута

5.3.2. Структура config_s

Структура config_s содержит данные, необходимые для подключения.

Публичные переменные config_s представлены в Таблице 3.

Таблица 3. Публичные переменные config_s

Имя	**Тип**	**Описание**
host	char*	Узел
protocol_version	int	Версия протокола LDAP
base_dn	char*	Базовый DN
username	char*	Имя пользователя
password	char*	Пароль
simple_bind	bool	Подключение, использующее простую привязку
use_tls	bool	Подключение, использующее TLS аутентификацию
use_sasl	bool	Подключение, использующее SASL аутентификацию
use_anon	bool	Анонимное подключение
timeout	int	Время ожидания (тайм-аут)
cacertfile	char*	Файл, содержащий сертификаты всех удостоверяющих центров
certfile	char*	Файл, содержащий сертификат клиента

keyfile	char*	Файл, содержащий закрытый ключ

5.3.3. Структура csm_state_value_t

Структура csm_state_value_t содержит значения состояний подключений.

Публичные переменные csm_state_value_t представлены в Таблице 4.

Таблица 4. Публичные переменные csm_state_value_t

Имя	**Тип**	**Описание**
state	int	Состояние
value	char*	Значение

5.3.4. Структура ldap_connection_config_t

Структура ldap_connection_config_t содержит параметры для SASL, TLS и т.д

Публичные переменные ldap_connection_config_t представлены в Таблице 5.

Таблица 5. Публичные переменные ldap_connection_config_t

Имя	**Тип**	**Описание**
server	const char*	Сервер
port	int	Порт
protocol_version	int	Версия протокола LDAP
chase_referrals	bool	Указывает, должен ли клиент автоматически следовать по отсылкам, возвращаемым LDAP-серверами.
use_start_tls	bool	Подключение, использующее TLS аутентификацию
use_sasl	bool	Подключение, использующее SASL аутентификацию
sasl_options	struct ldap_sasl_options_t*	Параметры SASL аутентификации

5.3.5. Структура ldap_connection_ctx_t

Структура ldap_connection_ctx_t содержит данные сконфигурированного соединения.

Публичные переменные ldap_connection_ctx_t представлены в Таблице 6.

Таблица 6. Публичные переменные ldap_connection_ctx_t

Имя	**Тип**	**Описание**
handle	LDHandle*	Указатель на дескриптор сеанса libdomain
ldap	LDAP*	
next	ldap_connection_ctx_t*	
prev	ldap_connection_ctx_t*	
fd	int	
base	verto_ctx*	Контекст события
read_event	verto_ev*	Событие чтения
write_event	verto_ev*	Событие записи
on_read_operation	operation_callback_fn	Обратный вызов, который выполняется при операции чтения.
on_write_operation	operation_callback_fn	Обратный вызов, который выполняется при операции записи.
on_error_operation	operation_callback_fn	Обратный вызов, выполняющийся во время операции привязки.
current_msgid	int	Идентификатор сообщения
rmech	char*	
state_machine	state_machine_t*	Состояние подключения
ldap_defaults	ldap_sasl_defaults_t*	Параметры по умолчанию SASL
ldap_params	ldap_sasl_params_t*	Данные для проверки подлинности клиента на сервере LDAP с помощью SASL

5.3.6. Структура ldap_global_context_t

Структура ldap_global_context_t содержит глобальный контекст.

Публичные переменные ldap_global_context_t представлены в Таблице 7.

Таблица 7. Публичные переменные ldap_global_context_t

Имя	**Тип**	**Описание**
global_ldap	LDAP	
talloc_ctx	TALLOC_CTX	

5.3.7. Структура ldap_sasl_default_t

Структура ldap_sasl_default_t содержит параметры по умолчанию SASL.

Публичные переменные ldap_sasl_default_t представлены в Таблице 8.

Таблица 8. Публичные переменные ldap_sasl_default_t

Имя	**Тип**	**Описание**
flags	short	Флаги SASL для подключения
mechanism	char*	Указывает, какой механизм SASL следует использовать
realm	char*	Указывает SASL-realm
authcid	char*	Указывает аутентификационную идентификационную сущность
authzid	char*	Указывает прокси-авторизационную идентификационную сущность
passwd	char*	Пароль

5.3.8. Структура ldap_sasl_options_t

Структура ldap_sasl_options_t содержит параметры SASL.

Публичные переменные ldap_sasl_options_t представлены в Таблице 9.

Таблица 9. Публичные переменные ldap_sasl_options_t

Имя	**Тип**	**Описание**
mechanism	char*	Механизм SASL
passwd	char*	Пароль
sasl_nocanon	bool	Выполнять обратные DNS-запросы для поиска канонической формы имён хостов SAS
sasl_flags	short	Флаги SASL для подключения
sasl_secprops	char*	Параметры безопасности Cyrus SASL

5.3.9. Структура ldap_sasl_params_t

Структура ldap_sasl_params_t содержит данные для проверки подлинности клиента на сервере LDAP с помощью SASL.

Публичные переменные ldap_sasl_params_t представлены в Таблице 10.

Таблица 10. Публичные переменные ldap_sasl_params_t

Имя	**Тип**	**Описание**
dn	char*	Различающееся имя записи, используемой для привязки
passwd	struct berval*	Учетные данные, используемые для проверки подлинности
serverctrls	LDAPControl**	Список серверных элементов управления LDAP
clientctrls	LDAPControl**	Список клиентских элементов управления LDAP

5.3.10. Структура LDAPAttribute_s

Структура LDAPAttribute_s содержит атрибут и его значения.

Публичные переменные LDAPAttribute_s представлены в Таблице 11.

Таблица 11. Публичные переменные LDAPAttribute_s

Имя	**Тип**	**Описание**
name	char	Имя атрибута
values	char	Значения атрибута

5.3.11. Структура ldhandle

Структура ldhandle содержит указатель на дескриптор сеанса libdomain.

Публичные переменные ldhandle представлены в Таблице 12.

Таблица 12. Публичные переменные ldhandle

Имя	**Тип**	**Описание**
talloc_ctx	TALLOC_CTX	
global_ctx	struct ldap_global_context_t	Глобальный контекст
connection_ctx	struct ldap_connection_ctx_t	Соединение
config_ctx	struct ldap_connection_config_t	Конфигурация подключения
global_config	config_t	Структура конфигурации

5.3.12. Структура option_value_t

Структура option_value_t содержит пару параметр-значение.

Публичные переменные option_value_t представлены в Таблице 13.

Таблица 13. Публичные переменные option_value_t

Имя	**Тип**	**Описание**
option	int	Параметр
value	char*	Значение

5.3.13. Структура state_machine_ctx_t

Структура state_machine_ctx_t определяет состояние подключения.

Публичные переменные state_machine_ctx_t представлены в Таблице 14.

Таблица 14. Публичные переменные state_machine_ctx_t

Имя	**Тип**	**Описание**
state	enum LdapConnectionState	Состояние подключения. Возможные значения: LDAP_CONNECTION_STATE_INIT = 1 LDAP_CONNECTION_STATE_TRANSPORT_READY = 2 LDAP_CONNECTION_STATE_BIND_IN_PROGRESS = 3 LDAP_CONNECTION_STATE_BOUND = 4 LDAP_CONNECTION_STATE_RUN = 5 LDAP_CONNECTION_STATE_ERROR = 6
ctx	struct ldap_connection_ctx_t	Соединение

Процесс подключения библиотеки libdomain к LDAP сервису имеет несколько отдельных фаз:

- LDAP_CONNECTION_STATE_INIT — начальное состояние;
- LDAP_CONNECTION_STATE_TRANSPORT_READY — состояние используемое для инициации операции bind;
- LDAP_CONNECTION_STATE_BIND_IN_PROGRESS — специальное состояние для выполнения интерактивного подсоединения;
- LDAP_CONNECTION_BOUND — состояние в которое переключается соединение при завершении операции интерактивного подсоединения;
- LDAP_CONNECTION_RUN — рабочее состояние соединения из этого состояния нет прямых переходов;
- LDAP_CONNECTION_ERROR — состояние ошибки если не исчерпан лимит переподключений запускает процесс повторного соединения.

Переходы между фазами зависят от результатов выполнения функций которые располагаются в connection.c. Управление состоянием соединения выполняется при помощи connection_state_machine.c.

Переходы между фазами могут в любой момент перейти к состоянию LDAP_CONNECTION_ERROR, но будут продвигаться только вперед, кроме состояния LDAP_CONNECTION_ERROR, которое запускает повторную установку соединения.

Схема процедуры установки соединения изображена на Рис. 5.

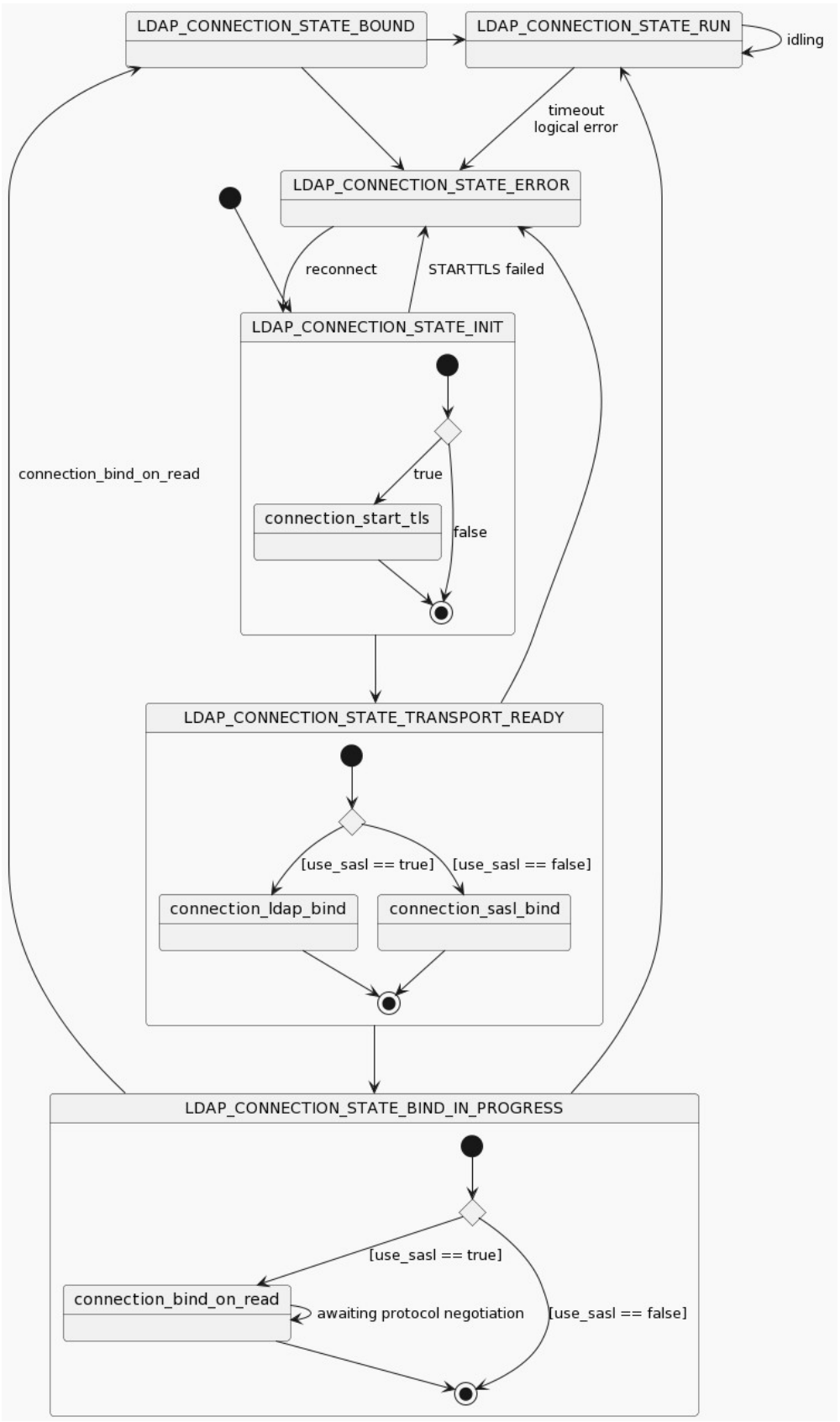

Рисунок 5. Диаграмма машины состояний управлением соединением

5.4. Функции

5.4.1. Функции src/common.h

Таблица 15. Функции src/common.h

Имя	**Описание**
error	Записывает ошибку в stderr
warning	Записывает предупреждение в stderr
info	Записывает информацию в stderr

5.4.1.1. Функция error (src/common.h)

Функция error записывает ошибку в stderr.

Синтаксис функции error:

```
void error(const char* format, …)
```

Параметры функции error приведены в Таблице 16.

Таблица 16. Параметры функции error

Имя	**Тип**	**Описание**
format	char	Формат, используемый в функции printf

5.4.1.2. Функция warning (src/common.h)

Функция warning записывает предупреждение в stderr.

Синтаксис функции warning:

```
void warning(const char* format, …)
```

Параметры функции warning приведены в Таблице 17.

Таблица 17. Параметры функции warning

Имя	**Тип**	**Описание**
format	char	Формат, используемый в функции printf

5.4.1.3. Функция info (src/common.h)

Функция info записывает информацию в stderr.

Синтаксис функции info:

```
void info(const char* format, …)
```

Параметры функции info приведены в Таблице 18.

Таблица 18. Параметры функции info

Имя	**Тип**	**Описание**
format	char	Формат, используемый в функции printf.

5.4.2. Функции src/computer.h

Таблица 19. Функции computer.h

Имя	**Описание**
ld_add_computer	Добавить учётную запись компьютера
ld_del_computer	Удалить учётную запись компьютера
ld_mod_computer	Изменить учётную запись компьютера
ld_rename_computer	Переименовать учётную запись компьютера

5.4.2.1. Функция ld_add_computer (src/computer.h)

Функция ld_add_computer добавляет учётную запись компьютера.

Синтаксис функции ld_add_computer:

```
enum OperationReturnCode ld_add_computer(LDHandle* handle, const char* name, const char* description, const char* display_name, const char* parent)
```

Параметры функции ld_add_computer приведены в Таблице 20.

Таблица 20. Параметры функции ld_add_computer

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название компьютера
description [in]	char	Описание компьютера
display_name [in]	char	Отображаемое имя компьютера
parent [in]	char	Родительский контейнер для компьютера

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.2.2. Функция ld_del_computer (src/computer.h)

Функция ld_del_computer удаляет учётную запись компьютера.

Синтаксис функции ld_del_computer:

```
enum OperationReturnCode ld_del_computer(LDHandle* handle, const char* name, const char* parent)
```

Параметры функции ld_del_computer приведены в Таблице 21.

Таблица 21. Параметры функции ld_del_computer

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название компьютера
parent [in]	char	Родительский контейнер компьютера

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.2.3. Функция ld_mod_computer (src/computer.h)

Функция ld_mod_computer изменяет учётную запись компьютера.

Синтаксис функции ld_mod_computer:

```
enum OperationReturnCode ld_mod_computer(LDHandle* handle, const char* name, const char* parent, LDAPAttribute_t* computer_attrs)
```

Параметры функции ld_mod_computer приведены в Таблице 22.

Таблица 22. Параметры функции ld_mod_computer

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название компьютера
parent [in]	char	Родительский контейнер для компьютера
computer_attrs [in]	LDAPAttribute_t	Список атрибутов для изменения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.2.4. Функция ld_rename_computer (src/computer.h)

Функция ld_rename_computer переименовывает учётную запись компьютера.

Синтаксис функции ld_rename_computer:

```
enum OperationReturnCode ld_rename_computer(LDHandle* handle, const char* old_name, const char* new_name, const char* parent)
```

Параметры функции ld_rename_computer приведены в Таблице 23.

Таблица 23. Параметры функции ld_rename_computer

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
old_name [in]	char	Текущее название компьютера
new_name [in]	char	Новое название компьютера
parent [in]	char	Родительский контейнер для компьютера

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.3. Функции src/connection.h

Таблица 24. Функции connection.h

Имя	**Описание**
connection_configure	Настраивает соединение (подключение)
connection_start_tls	Настройка TLS
connection_sasl_bind	Пытается выполнить неинтерактивное подключение с использованием привязки SASL. Устанавливает обработчик операции connection_bind_on_read.
connection_ldap_bind	Выполняет интерактивную привязку и устанавливает обработчик операции connection_bind_on_read.
connection_close	Закрывает соединение и освобождает ресурсы, связанные с указанным соединением.
connection_on_read	Обратный вызов, который выполняется при операции чтения.
connection_on_write	Обратный вызов, который выполняется при операции записи.

connection_bind_on_read	Обратный вызов, выполняющийся во время операции привязки.

5.4.3.1. Функция connection_configure (src/connection.h)

Функция connection_configure настраивает соединение при выполнении следующих действий:

- создает дескриптор LDAP и устанавливает версию протокола, включает флаг асинхронного подключения;
- если используется SASL, настраивает флаги SASL для подключения. Выделяет структуру для хранения параметров SASL;
- если используется TLS, настраивает флаги TLS для подключения.
- создает базу событий для соединения.

Синтаксис функции connection_configure:

```
enum OperationReturnCode connection_configure(struct ldap_global_context_t* global_ctx, struct ldap_connection_ctx_t* connection,struct ldap_connection_config_t* config)
```

Параметры функции connection_configure приведены в Таблице 25.

Таблица 25. Параметры функции connection_configure

Имя	**Тип**	**Описание**
global_ctx [in]	ldap_global_context_t	Глобальный контекст
connection [out]	ldap_connection_ctx_t	Сконфигурированное соединение готово к передаче в конечный автомат соединения
config [in]	ldap_connection_config_t	Конфигурация подключения (содержит параметры для SASL, TLS и т.д.)

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.3.2. Функция connection_start_tls (src/connection.h)

Функция connection_start_tls настраивает TLS соединение.

Синтаксис функции connection_start_tls:

```
enum OperationReturnCode connection_start_tls(struct ldap_connection_ctx_t* connection)
```

Параметры функции connection_start_tls приведены в Таблице 26.

Таблица 26. Параметры функции connection_start_tls

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Сконфигурированное соединение

5.4.3.3. Функция connection_sasl_bind (src/connection.h)

Функция connection_sasl_bind пытается выполнить неинтерактивное подключение с использованием привязки SASL. Устанавливает обработчик операции connection_bind_on_read.

Синтаксис функции connection_sasl_bind:

```
enum OperationReturnCode connection_sasl_bind(struct ldap_connection_ctx_t* connection)
```

Параметры функции connection_ sasl_bind приведены в Таблице 27.

Таблица 27. Параметры функции connection_start_tls

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Сконфигурированное соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.3.4. Функция connection_ldap_bind (src/connection.h)

Функция connection_ldap_bind выполняет интерактивную привязку и устанавливает обработчик операции connection_bind_on_read.

Синтаксис функции connection_ldap_bind:

```
enum OperationReturnCode connection_ldap_bind(struct ldap_connection_ctx_t* connection)
```

Параметры функции connection_ldap_bind приведены в Таблице 28.

Таблица 28. Параметры функции connection_ldap_bind

Имя	**Тип**	**Описание**

connection [in]	ldap_connection_ctx_t	Сконфигурированное соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_OPERATION_IN_PROGRESS, если функция все еще выполняется;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.3.5. Функция connection_close (src/connection.h)

Функция connection_close закрывает соединение и освобождает ресурсы, связанные с указанным соединением.

Синтаксис функции connection_close:

```
enum OperationReturnCode connection_close(struct ldap_connection_ctx_t* connection)
```

Параметры функции connection_close приведены в Таблице 29.

Таблица 29. Параметры функции connection_close

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Сконфигурированное соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно.

5.4.3.6. Функция connection_on_read (src/connection.h)

Функция connection_on_read это обратный вызов, выполняемый при операции чтения.

Синтаксис функции connection_on_read:

```
void connection_on_read(verto_ctx* ctx, verto_ev* ev)
```

Параметры функции connection_on_read приведены в Таблице 30.

Таблица 30. Параметры функции connection_on_read

Имя	**Тип**	**Описание**
ctx [in]	verto_ctx	Контекст события
ev [in]	verto_ev	Событие

5.4.3.7. Функция connection_on_write (src/connection.h)

Функция connection_on_write это обратный вызов, выполняемый при операции записи.

Синтаксис функции connection_on_write:

```
void connection_on_write(verto_ctx* ctx, verto_ev* ev)
```

Параметры функции connection_on_write приведены в Таблице 31.

Таблица 31. Параметры функции connection_on_write

Имя	**Тип**	**Описание**
ctx [in]	verto_ctx	Контекст события
ev [in]	verto_ev	Событие

5.4.3.8. Функция connection_bind_on_read (src/connection.h)

Этот обратный вызов выполняется во время операции привязки.

Синтаксис функции connection_bind_on_read:

```
enum OperationReturnCode connection_bind_on_read (int rc, LDAPMessage*
message, ldap_connection_ctx_t* connection)
```

Параметры функции connection_bind_on_read приведены в Таблице 32.

Таблица 32. Параметры функции connection_bind_on_read

Имя	**Тип**	**Описание**
rc [in]	int	Код возврата операции привязки.
message [in]	LDAPMessage	Сообщение полученное во время работы.
connection [in]	ldap_connection_ctx_t	Соединение, используемое во время операции привязки.

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.4. Функции src/connection_state_machine.h

Таблица 33. Функции

Имя	**Описание**
csm_init	Инициализирует состояние подключения, устанавливает состояние

	подключения в LDAP_CONNECTION_STATE_INIT.
csm_next_state	Изменяет состояние подключения на основе текущего состояния подключения.
csm_set_state	Устанавливает новое состояние подключения, выводит переход между состояниями.

5.4.4.1. Функция csm_init (src/connection_state_machine.h)

Функция csm_init инициализирует состояние подключения, устанавливает состояние подключения в LDAP_CONNECTION_STATE_INIT.

Синтаксис функции csm_init:

```
enum OperationReturnCode csm_init(struct state_machine_ctx_t* ctx, struct ldap_connection_ctx_t* connection)
```

Параметры функции csm_init приведены в Таблице 34.

Таблица 34. Параметры функции csm_init

Имя	**Тип**	**Описание**
ctx [in]	state_machine_ctx_t	Состояние подключения для инициализации
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно.

5.4.4.2. Функция csm_next_state (src/connection_state_machine.h)

Функция csm_next_state изменяет состояние подключения на основе текущего состояния подключения.

Синтаксис функции csm_next_state:

```
enum OperationReturnCode csm_next_state(struct state_machine_ctx_t* ctx)
```

Параметры функции csm_next_state приведены в Таблице 35.

Таблица 35. Параметры функции csm_next_state

Имя	**Тип**	**Описание**
ctx [in]	state_machine_ctx_t	Текущее состояние подключения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;

- RETURN_CODE_OPERATION_IN_PROGRESS, если функция все еще выполняется;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.4.3. Функция csm_set_state (src/connection_state_machine.h)

Функция csm_set_state устанавливает новое состояние подключения, выводит переход между состояниями.

Синтаксис функции csm_set_state:

```
enum OperationReturnCode csm_set_state(struct state_machine_ctx_t* ctx, enum LdapConnectionState state)
```

Параметры функции csm_set_state приведены в Таблице 36.

Таблица 36. Параметры функции csm_set_state

Имя	**Тип**	**Описание**
ctx [in]	state_machine_ctx_t	Текущее состояние подключения
state [in]	LdapConnectionState	Новое состояние подключения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно.

5.4.5. Функции src/directory.h

Таблица 37. Функции directory.h

Имя	**Описание**
directory_get_type	Запрашивает тип каталога LDAP у сервиса.
directory_parse_result	Анализирует результаты запроса типа каталога и инициализирует подключение к данному типу каталога.

Возможные типы каталогов LDAP:

- LDAP_TYPE_ACTIVE_DIRECTORY — Samba/Microsoft Active Directory;
- LDAP_TYPE_OPENLDAP — OpenLDAP;
- LDAP_TYPE_FREE_IPA — FreeIPA.

5.4.5.1. Функция directory_get_type (src/directory.h)

Функция directory_get_type запрашивает тип каталога LDAP у сервиса.

Синтаксис функции directory_get_type:

```
enum OperationReturnCode directory_get_type(struct
ldap_connection_ctx_t* connection)
```

Параметры функции directory_get_type приведены в Таблице 38.

Таблица 38. Параметры функции directory_get_type

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.5.2. Функция directory_parse_result (src/directory.h)

Функция directory_parse_result анализирует результаты запроса типа каталога и инициализирует подключение к данному типу каталога.

Синтаксис функции directory_parse_result:

```
enum OperationReturnCode directory_parse_result(struct
ldap_connection_ctx_t* connection)
```

Параметры функции directory_parse_result приведены в Таблице 39.

Таблица 39. Параметры функции directory_parse_result

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.6. Функции src/domain.h

Таблица 40. Функции domain.h

Имя	**Описание**
ld_create_config	Заполняет поля структуры конфигурации.
ld_init	Инициализирует библиотеку, позволяя выполнять различные

	операции.
ld_install_default_handlers	Устанавливает обработчики по умолчанию для управления соединением. Этот метод необходимо вызывать перед выполнением каких-либо операций.
ld_install_handler	Позволяет установить собственный обратный вызов ошибки
ld_install_error_handler	Позволяет установить собственный дескриптор ошибки
ld_exec	Начать основной цикл событий. Не нужно вызывать эту функцию, если уже существует цикл событий, например. внутри приложения Qt.
ld_exec_once	Один раз перебирает список событий. Может заблокировать.
ld_free	Освобождает дескриптор библиотеки и связанные с ней ресурсы. После освобождения дескриптора будет невозможно выполнять какие-либо операции.

5.4.6.1. Функция ld_create_config (src/domain.h)

Функция ld_create_config Заполняет поля структуры конфигурации.

Синтаксис функции ld_create_config:

```
config_t* ld_create_config(char* host, int port, int protocol_version, char* base_dn, char* username, char* password, bool simple_bind, bool use_tls, bool use_sasl, bool use_anon, int timeout, char* cacertfile, char* certfile, char* keyfile)
```

Параметры функции ld_create_config приведены в Таблице 41.

Таблица 41. Параметры функции ld_create_config

Имя	**Тип**	**Описание**
host [in]	char	Имя узла LDAP
port [in]	int	Порт
protocol_version [in]	int	Версия протокола LDAP
base_dn [in]	char	Базовый DN
username [in]	char	Имя пользователя
password [in]	char	Пароль пользователя
simple_bind [in]	bool	Подключение, использующее простую привязку
use_tls [in]	bool	Подключение, использующее TLS аутентификацию

use_sasl [in]	bool	Подключение, использующее SASL аутентификацию
use_anon [in]	bool	Анонимное подключение
timeout [in]	int	Время ожидания (тайм-аут)
cacertfile [in]	char	Файл, содержащий сертификаты всех удостоверяющих центров
certfile [in]	char	Файл, содержащий сертификат клиента
keyfile [in]	char	Файл, содержащий закрытый ключ

Возвращаемое значение:

- config_t*, если функция завершается успешно;
- NULL, если функция выполняется неудачно.

5.4.6.2. Функция ld_init (src/domain.h)

Функция ld_init инициализирует библиотеку, позволяя выполнять различные операции.

Синтаксис функции ld_init:

```
void ld_create_config(LDHandle** handle, const config_t* config)
```

Параметры функции ld_init приведены в Таблице 42.

Таблица 42. Параметры функции ld_init

Имя	**Тип**	**Описание**
handle [out]	LDHandle	Указатель на дескриптор сеанса libdomain
config [in]	config_t	Используемое соединение

5.4.6.3. Функция ld_install_default_handlers (src/domain.h)

Функция ld_install_default_handlers устанавливает обработчики по умолчанию для управления соединением.

Синтаксис функции ld_install_default_handlers:

```
void ld_create_config(LDHandle** handle)
```

Параметры функции ld_install_default_handlers приведены в Таблице 43.

Таблица 43. Параметры функции ld_install_default_handlers

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain

5.4.6.4. Функция ld_install_handler (src/domain.h)

Функция ld_install_handler позволяет установить собственный обратный вызов ошибки.

Синтаксис функции ld_install_handler:

```
void ld_install_handler(LDHandle** handle, verto_callback* callback, time_t interval)
```

Параметры функции ld_install_handler приведены в Таблице 44.

Таблица 44. Параметры функции ld_install_handler

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
callback [in]	verto_callback	Обратный вызов
interval [in]	time_t	Время ожидания

5.4.6.5. Функция ld_install_error_handler (src/domain.h)

Функция ld_install_error_handler позволяет установить собственный дескриптор ошибки.

Синтаксис функции ld_install_error_handler:

```
void ld_install_error_handler(LDHandle** handle, error_callback_fn callback)
```

Параметры функции ld_install_ error_handler приведены в Таблице 45.

Таблица 45. Параметры функции ld_install_handler

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
callback [in]	verto_callback	Обратный вызов

5.4.6.6. Функция ld_exec (src/domain.h)

Функция ld_exec позволяет начать основной цикл событий.

Синтаксис функции ld_exec:

```
void ld_exec(LDHandle** handle)
```

Параметры функции ld_exec приведены в Таблице 46.

Таблица 46. Параметры функции ld_exec

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain

5.4.6.7. Функция ld_exec_once (src/domain.h)

Функция ld_exec_once один раз перебирает список событий.

Синтаксис функции ld_exec_once:

```
void ld_exec_once(LDHandle** handle)
```

Параметры функции ld_exec_once приведены в Таблице 47.

Таблица 47. Параметры функции ld_exec_once

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain

5.4.6.8. Функция ld_free (src/domain.h)

Функция ld_ free освобождает дескриптор библиотеки и связанные с ней ресурсы.

Синтаксис функции ld_free:

```
void ld_free(LDHandle** handle)
```

Параметры функции ld_ free приведены в Таблице 48.

Таблица 48. Параметры функции ld_free

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain

5.4.7. Функции src/entry.h

Таблица 49. Функции entry.h

Имя	**Описание**
add	Оборачивает функцию ldap_add_ext, связывая ее с подключением
add_on_read	Обратный вызов, вызываемый при завершении операции добавления ldap
search	Оборачивает операцию ldap_search, связывая ее с подключением
search_on_read	Обратный вызов, вызываемый после завершения операции поиска ldap.

modify	Оборачивает операцию ldap_modify_ext, связывая ее с подключением
modify_on_read	Обратный вызов, вызываемый после завершения операции изменения ldap.
delete	Функция удаления оборачивает ldap_delete_ext, связывая ее с подключением
delete_on_read	Обратный вызов, определяющий результат операции удаления.
id_rename	Оборачивает функцию ldap_rename, связывая ее с подключением
rename_on_read	Обратный вызов, определяющий результат операции переименования.
whoami	Определяет, текущего пользователя. Эта операция поддерживается только в OpenLDAP
whoami_on_read	Обратный вызов, определяющий результат операции whoami.

5.4.7.1. Функция add (src/entry.h)

Функция add оборачивает (декорирует) функцию ldap_add_ext, связывая ее с подключением. Функция ldap_add_ext инициирует асинхронную операцию добавления в дерево LDAP.

Синтаксис функции add:

```
enum OperationReturnCode add(struct ldap_connection_ctx_t* connection,
const char* dn, LDAPMod** attrs)
```

Параметры функции add приведены в Таблице 50.

Таблица 50. Параметры функции add

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение
dn [in]	char	Имя добавляемой записи. Если NULL, на сервер отправляется DN нулевой длины.
attrs [in]	LDAPMod	Атрибуты записи, указанные с использованием структуры LDAPMod, определенной для ldap_modify(). Поля mod_type и mod_vals ДОЛЖНЫ быть заполнены. Поле mod_op игнорируется до тех пор, пока не будет выполнено ИЛИ с константой LDAP_MOD_BVALUES, используемой для выбора случая mod_bvalues объединения mod_vals.

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.2. Функция add_on_read (src/entry.h)

Этот обратный вызов выполняется после завершения операции добавления ldap.

Синтаксис функции add_on_read:

```
enum OperationReturnCode add_on_read(int rc, LDAPMessage* message, ldap_connection_ctx_t* connection)
```

Параметры функции add_on_read приведены в Таблице 51.

Таблица 51. Параметры функции add_on_read

Имя	**Тип**	**Описание**
rc [in]	int	Код возврата ldap_result
message [in]	LDAPMessage	Сообщение полученное от LDAP
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.3. Функция search (src/entry.h)

Функция search оборачивает (декорирует) функцию ldap_search, связывая ее с подключением. Функция ldap_search выполняет поиск в каталоге LDAP и возвращает запрошенный набор атрибутов для каждой соответствующей записи.

Синтаксис функции search:

```
enum OperationReturnCode search(struct ldap_connection_ctx_t* connection, const char* base_dn, int scope, const char* filter, char** attrs, bool attrsonly)
```

Параметры функции search приведены в Таблице 52.

Таблица 52. Параметры функции search

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение
base_dn [in]	char	DN записи, с которой следует начать поиск. Если NULL, на сервер отправляется DN нулевой длины.

scope [in]	int	Одно из следующих значений для указания области поиска: − LDAP_SCOPE_BASE (0x00) — поиск только базовой записи; − LDAP_SCOPE_ONELEVEL (0x01) — поиск всех записей на первом уровне ниже базовой записи, за исключением базовой записи; − LDAP_SCOPE_SUBTREE (0x02) — поиск по базовой записи и всем записям в дереве.
filter [in]	char	Строка символов, представляющая фильтр поиска. Значение NULL может быть передано, чтобы указать, что должен использоваться фильтр "(objectclass=*)", который соответствует всем записям. Если вызывающая сторона API использует LDAPv2, можно успешно использовать только подмножество функций фильтрации, описанных в разделе Возвращаемые значения .
attrs [in]	char	Массив строк, завершающихся значением NULL, указывающий, какие атрибуты следует возвращать для каждой соответствующей записи. Передача значения NULL для этого параметра приводит к извлечению всех доступных атрибутов. Строка LDAP_NO_ATTRS ("1.1") МОЖЕТ использоваться как единственная строка в массиве, указывающая, что сервер не должен возвращать никакие типы атрибутов. Строка LDAP_ALL_USER_ATTRS ("*") может использоваться в массиве attrs вместе с именами некоторых операционных атрибутов, чтобы указать, что должны быть возвращены все пользовательские атрибуты плюс перечисленные операционные атрибуты.
attrsonly [in]	bool	Логическое значение, которое должно быть равно нулю, если должны быть возвращены как типы атрибутов, так и значения, и не должно быть нулевым, если нужны только типы.

Возвращаемое значение:

− RETURN_CODE_SUCCESS, если функция завершается успешно;

− RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.4. Функция search_on_read (src/entry.h)

Этот обратный вызов выполняется после завершения операции поиска ldap.

Синтаксис функции search_on_read:

```
enum OperationReturnCode search_on_read(int rc, LDAPMessage* message,
struct ldap_connection_ctx_t* connection)
```

Параметры функции search_on_read приведены в Таблице 53.

Таблица 53. *Параметры функции search_on_read*

Имя	Тип	Описание
rc [in]	int	Код возврата ldap_result
message [in]	LDAPMessage	Сообщение полученное от LDAP
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.5. Функция modify (src/entry.h)

Функция modify оборачивает (декорирует) функцию ldap_modify_ext, связывая ее с подключением. Функция ldap_modify_ext редактирует существующую запись в дереве LDAP.

Синтаксис функции modify:

```
enum OperationReturnCode modify(struct ldap_connection_ctx_t* connection, const char* dn, char** attrs)
```

Параметры функции modify приведены в Таблице 54.

Таблица 54. Параметры функции modify

Имя	Тип	Описание
connection [in]	ldap_connection_ctx_t	Используемое соединение
dn [in]	char	DN записи, которую необходимо изменить. Если NULL, на сервер отправляется DN нулевой длины.
attrs [in]	char	Массив строк, завершающихся значением NULL, указывающий, какие изменения следует внести для записи.

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.6. Функция modify_on_read (src/entry.h)

Этот обратный вызов выполняется после завершения операции изменения ldap.

Синтаксис функции modify_on_read:

```
enum OperationReturnCode modify_on_read(int rc, LDAPMessage* message,
struct ldap_connection_ctx_t* connection)
```

Параметры функции modify_on_read приведены в Таблице 55.

Таблица 55. *Параметры функции modify_on_read*

Имя	**Тип**	**Описание**
rc [in]	int	Код возврата ldap_result
message [in]	LDAPMessage	Сообщение полученное от LDAP
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.7. Функция delete (src/entry.h)

Функция delete оборачивает (декорирует) функцию ldap_delete_ext, связывая ее с подключением. Функция ldap_delete_ext удаляет конечную запись из дерева LDAP.

Синтаксис:

```
enum OperationReturnCode delete(struct ldap_connection_ctx_t*
connection, const char* dn)
```

Параметры функции delete приведены в Таблице 56.

Таблица 56. Параметры функции delete

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение
dn [in]	char	DN записи, которую необходимо удалить. Если NULL, на сервер отправляется DN нулевой длины.

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.8. Функция delete_on_read (src/entry.h)

Этот обратный вызов выполняется после завершения операции удаления ldap.

Синтаксис:

```
enum OperationReturnCode delete_on_read(int rc, LDAPMessage* message,
struct ldap_connection_ctx_t* connection)
```

Параметры функции delete_on_read приведены в Таблице 57.

Таблица 57. *Параметры функции delete_on_read*

Имя	**Тип**	**Описание**
rc [in]	int	Код возврата ldap_result
message [in]	LDAPMessage	Сообщение полученное от LDAP
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.9. Функция ld_rename (src/entry.h)

Функция ld_rename оборачивает (декорирует) функцию ldap_rename, связывая ее с подключением. Функция ldap_rename изменяет различающееся имя записи в каталоге LDAP.

Синтаксис функции ld_rename:

```
enum OperationReturnCode ld_rename(struct ldap_connection_ctx_t*
connection, const char* olddn, const char* newdn, const char*
new_parent, bool delete_original)
```

Параметры функции ld_rename приведены в Таблице 58.

Таблица 58. Параметры функции ld_rename

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение
olddn [in]	char	DN записи, которую необходимо переименовать. Если NULL, на сервер отправляется DN нулевой длины.
newdn [in]	char	Новое относительное различающееся имя для записи. Если NULL, на сервер отправляется DN нулевой длины.
new_parent [in]	char	Имя нового родительского элемента для этой записи. Этот параметр позволяет переместить

		запись в новый родительский контейнер.
delete_original [in]	bool	TRUE, если необходимо удалить старое относительное различающееся имя; FALSE, если старое относительное различающееся имя должно сохраниться.

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.10. Функция rename_on_read (src/entry.h)

Этот обратный вызов выполняется после завершения операции переименования ldap.

Синтаксис функции rename_on_read:

```
enum OperationReturnCode rename_on_read(int rc, LDAPMessage* message, struct ldap_connection_ctx_t* connection)
```

Параметры функции rename_on_read приведены в Таблице 59.

Таблица 59. *Параметры функции rename_on_read*

Имя	**Тип**	**Описание**
rc [in]	int	Код возврата ldap_result
message [in]	LDAPMessage	Сообщение полученное от LDAP
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.7.11. Функция whoami (src/entry.h)

Функция whoami определяет, кто является текущим пользователем. Эта операция поддерживается только в OpenLDAP.

Синтаксис функции whoami:

```
enum OperationReturnCode whoami(struct ldap_connection_ctx_t* connection)
```

Параметры функции whoami приведены в Таблице 60.

Таблица 60. Параметры функции whoami

Имя	**Тип**	**Описание**
connection [in]	ldap_connection_ctx_t	Используемое соединение

5.4.7.12. Функция whoami_on_read (src/entry.h)

Этот обратный вызов определяет результат операции whoami.

Синтаксис функции whoami_on_read:

```
enum OperationReturnCode whoami_on_read(int rc, LDAPMessage* message,
struct ldap_connection_ctx_t* connection)
```

Параметры функции whoami_on_read приведены в Таблице 61.

Таблица 61. *Параметры функции whoami_on_read*

Имя	**Тип**	**Описание**
rc [in]	int	Код возврата ldap_result
message [in]	LDAPMessage	Сообщение полученное от LDAP
connection [in]	ldap_connection_ctx_t	Используемое соединение

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.8. Функции src/group.h

Таблица 62. Функции group.h

Имя	**Описание**
ld_add_group	Добавить группу
ld_del_group	Удалить группу
ld_mod_group	Изменить группу
ld_rename_group	Переименовать группу
ld_group_add_user	Добавить пользователя в группу
ld_group_remove_user	Удалить пользователя из группы

Возможные области действия группы:

- GROUP_SCOPE_DOMAIN_LOCAL — домен локальная;
- GROUP_SCOPE_GLOBAL — глобальная;
- GROUP_SCOPE_UNIVERSAL — универсальная.

Возможные категория группы:

- GROUP_CATEGORY_DISTRIBUTION — рассылка;
- GROUP_CATEGORY_SECURITY — безопасность.

5.4.8.1. Функция ld_add_group (src/group.h)

Функция ld_add_group добавляет группу.

Синтаксис функции ld_add_group:

```
enum OperationReturnCode ld_add_group(LDHandle* handle, const char* name, const char* description, int gid, const char* parent)
```

Параметры функции ld_add_group приведены в Таблице 63.

Таблица 63. Параметры функции ld_add_group

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название группы
description [in]	char	Описание группы
gid [in]	int	ID группы
parent [in]	char	Родительский контейнер для группы

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.8.2. Функция ld_del_group (src/group.h)

Функция ld_del_group удаляет группу.

Синтаксис функции ld_del_group:

```
enum OperationReturnCode ld_del_group(LDHandle* handle, const char* name, const char* parent)
```

Параметры функции ld_del_group приведены в Таблице 64.

Таблица 64. Параметры функции ld_del_group

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название группы
parent [in]	char	Родительский контейнер группы

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.8.3. Функция ld_mod_group (src/group.h)

Функция ld_mod_group изменяет группу.

Синтаксис функции ld_mod_group:

```
enum OperationReturnCode ld_mod_group(LDHandle* handle, const char* name, const char* parent, LDAPAttribute_t* group_attrs)
```

Параметры функции ld_mod_group приведены в Таблице 65.

Таблица 65. Параметры функции ld_mod_group

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название группы
parent [in]	char	Родительский контейнер для группы
group_attrs [in]	LDAPAttribute_t	Список атрибутов группы

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.8.4. Функция ld_rename_group (src/group.h)

Функция ld_rename_group переименовывает группу.

Синтаксис функции ld_rename_group:

```
enum OperationReturnCode ld_rename_group(LDHandle* handle, const char* old_name, const char* new_name, const char* parent)
```

Параметры функции ld_rename_group приведены в Таблице 66.

Таблица 66. Параметры функции ld_rename_group

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
old_name [in]	char	Старое название группы
new_name [in]	char	Новое название группы
parent [in]	char	Родительский элемент для группы

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.8.5. Функция ld_group_add_user (src/group.h)

Функция ld_group_add_user добавляет пользователя в группу.

Синтаксис функции ld_group_add_user:

```
enum OperationReturnCode ld_group_add_user(LDHandle* handle, const char* group_name, const char* user_name)
```

Параметры функции ld_group_add_user приведены в Таблице 67.

Таблица 67. Параметры функции ld_group_add_user

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
group_name [in]	char	Название группы, в которую будет добавлен пользователь
user_name [in]	char	Имя пользователя

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.8.6. Функция ld_group_remove_user (src/group.h)

Функция ld_group_remove_user удаляет пользователя из группы.

Синтаксис функции ld_group_remove_user:

```
enum OperationReturnCode ld_group_add_user(LDHandle* handle, const char* group_name, const char* user_name)
```

Параметры функции ld_group_remove_user приведены в Таблице 68.

Таблица 68. Параметры функции ld_group_remove_user

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
group_name [in]	char	Название группы
user_name [in]	char	Имя пользователя

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.9. Функции src/organization_unit.h

Таблица 69. Функции organization_unit.h

Имя	**Описание**
ld_add_ou	Создать подразделение
ld_del_ou	Удалить подразделение
ld_mod_ou	Изменить подразделение
ld_rename_ou	Переименовать подразделение

5.4.9.1. Функция ld_add_ou (src/organization_unit.h)

Функция ld_add_ou создает подразделение.

Синтаксис функции ld_add_ou:

```
enum OperationReturnCode ld_add_ou(LDHandle* handle, const char* name,
const char* city, const char* description, const char* parent)
```

Параметры функции ld_add_ou приведены в Таблице 70.

Таблица 70. Параметры функции ld_add_ou

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название подразделения (OU)
description [in]	char	Описание подразделения
parent [in]	char	Родительский контейнер для подразделения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.9.2. Функция ld_del_ou (src/organization_unit.h)

Функция ld_del_ou удаляет подразделение.

Синтаксис функции ld_del_ou:

```
enum OperationReturnCode ld_del_ou(LDHandle* handle, const char* name,
const char* parent)
```

Параметры функции ld_del_ou приведены в Таблице 71.

Таблица 71. Параметры функции ld_del_ou

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название подразделения (OU)
parent [in]	char	Родительский контейнер подразделения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.9.3. Функция ld_mod_ou (src/organization_unit.h)

Функция ld_mod_ou изменяет подразделение.

Синтаксис функции ld_mod_ou:

```
enum OperationReturnCode ld_mod_ou(LDHandle* handle, const char* name,
const char* parent, LDAPAttribute_t** ou_attrs)
```

Параметры функции ld_mod_ou приведены в Таблице 72.

Таблица 72. Параметры функции ld_mod_ou

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Название подразделения (OU)
parent [in]	char	Родительский контейнер для подразделения
ou_attrs [in]	LDAPAttribute_t	Список атрибутов подразделения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.9.4. Функция ld_rename_ou (src/organization_unit.h)

Функция ld_rename_ou переименовывает группу.

Синтаксис функции ld_rename_ou:

```
enum OperationReturnCode ld_rename_ou(LDHandle* handle, const char* old_name, const char* new_name, const char* parent)
```

Параметры функции ld_rename_ou приведены в Таблице 73.

Таблица 73. Параметры функции ld_rename_ou

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
old_name [in]	char	Старое название подразделения
new_name [in]	char	Новое название подразделения
parent [in]	char	Родительский элемент для подразделения

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.10. Функции src/user.h

Таблица 74. Функции user.h

Имя	**Описание**
ld_add_user	Создать пользователя
ld_del_user	Удалить пользователя
ld_mod_user	Изменить пользователя
ld_rename_user	Переименовать пользователя

5.4.10.1. Функция ld_add_user (src/user.h)

Функция ld_add_user создает пользователя.

Синтаксис функции ld_add_user:

```
enum OperationReturnCode ld_add_user(LDHandle* handle, const char* name, const int uid, const int gid, const char* home_directory, const char* login_shell, const char* password, const char* parent)
```

Параметры функции ld_add_user приведены в Таблице 75.

Таблица 75. Параметры функции ld_add_user

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Имя пользователя
uid [in]	int	Идентификатор пользователя
gid [in]	int	Идентификатор группы
home_directory [in]	char	Домашний каталог пользователя
login_shell [in]	char	Интерпретатор команд
password [in]	char	Пароль пользователя
parent [in]	char	Контейнер, в котором необходимо создать пользователя

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.10.2. Функция ld_del_user (src/user.h)

Функция ld_del_user удаляет пользователя.

Синтаксис функции ld_del_user:

```
enum OperationReturnCode ld_del_user(LDHandle* handle, const char* name, const char* parent)
```

Параметры функции ld_del_user приведены в Таблице 76.

Таблица 76. Параметры функции ld_del_user

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Имя пользователя
parent [in]	char	Контейнер пользователя

Возвращаемое значение:

– RETURN_CODE_SUCCESS, если функция завершается успешно;

– RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.10.3. Функция ld_mod_user (src/user.h)

Функция ld_mod_user изменяет пользователя.

Синтаксис функции ld_mod_user:

```
enum OperationReturnCode ld_mod_user(LDHandle* handle, const char* name, const char* parent, LDAPAttribute_t* user_attrs)
```

Параметры функции ld_mod_user приведены в Таблице 77.

Таблица 77. Параметры функции ld_mod_user

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
name [in]	char	Имя пользователя
parent [in]	char	Контейнер пользователя
user_attrs [in]	LDAPAttribute_t	Список атрибутов пользователя

Возвращаемое значение:

– RETURN_CODE_SUCCESS, если функция завершается успешно;

– RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.10.4. Функция ld_rename_user (src/user.h)

Функция ld_rename_user переименовывает пользователя.

Синтаксис функции ld_rename_user:

```
enum OperationReturnCode ld_rename_ou(LDHandle* handle, const char* old_name, const char* new_name, const char* parent)
```

Параметры функции ld_rename_user приведены в Таблице 78.

Таблица 78. Параметры функции ld_rename_user

Имя	**Тип**	**Описание**
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
old_name [in]	char	Старое имя пользователя
new_name [in]	char	Новое имя пользователя

parent [in]	char	Контейнер пользователя

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;
- RETURN_CODE_FAILURE, если функция выполняется неудачно.

6. Возвращаемые значения

В следующем списке перечислены коды возвращаемых значений:

- 1 (RETURN_CODE_SUCCESS) — успешное завершение функции;
- 2 (RETURN_CODE_FAILURE) — функция завершилась с ошибкой;
- 3 (RETURN_CODE_MISSING_ATTRIBUTE) — пропущен атрибут;
- 4 (RETURN_CODE_OPERATION_IN_PROGRESS) — функция еще выполняется.

7. Синтаксис фильтра поиска

Фильтры поиска позволяют определять критерии поиска и предоставлять более эффективные и эффективные поисковые запросы.

Синтаксис LDAP-фильтра имеет вид:

```
<Атрибут><оператор сравнения><значение>
```

В Таблице 79 приведены примеры фильтров поиска LDAP.

Таблица 79. Примеры фильтров поиска LDAP

Фильтр поиска	**Описание**
"(objectClass=*)"	Все объекты
"((&objectCategory=person)(objectClass=user)(!(cn=ivanov))"	Все пользовательские объекты, кроме пользователя с cn=ivanov
"(sn=sm*)"	Все объекты с cn, начинающимся с sm
"(&(objectClass=user)(email=*))"	Все пользователи с атрибутом электронной почты