



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

Техническая документация к прототипу библиотеки libdomain проекта

Москва 2023

Содержание

1. Назначение и цель документа.....	4
2. Глоссарий.....	4
3. Назначение Решения.....	5
4. Функционал библиотеки libdomain.....	6
5. Объекты библиотеки.....	7
5.1. Модули библиотеки.....	7
5.2. Классы библиотеки.....	8
5.3. Структуры.....	16
5.3.1. Структура attribute_value_pair_s.....	16
5.3.2. Структура csm_state_value_t.....	16
5.3.3. Структура ld_config_s.....	16
5.3.4. Структура ld_entry_s.....	17
5.3.5. Структура ldap_connection_config_t.....	17
5.3.6. Структура ldap_connection_ctx_t.....	19
5.3.7. Структура ldap_global_context_t.....	20
5.3.8. Структура ldap_request_t.....	20
5.3.9. Структура ldap_sasl_default_t.....	21
5.3.10. Структура ldap_sasl_options_t.....	21
5.3.11. Структура ldap_sasl_params_t.....	22
5.3.12. Структура ldap_schema_t.....	22
5.3.13. Структура ldap_search_request_t.....	23
5.3.14. Структура LDAPAttribute_s.....	23
5.3.15. Структура ldhandle.....	23
5.3.16. Структура option_value_t.....	24
5.3.17. Структура Queue_Node_s.....	24
5.3.18. Структура request_queue.....	24
5.3.19. Структура state_machine_ctx_t.....	25

5.4. Файлы.....	28
5.4.1. Файл attribute.h.....	28
5.4.2. Файл common.h.....	29
5.4.3. Файл computer.h.....	31
5.4.4. Файл connection.h.....	33
5.4.5. Файл connection_state_machine.h.....	39
5.4.6. Файл directory.h.....	43
5.4.7. Файл domain.h.....	44
5.4.8. Файл entry.h.....	50
5.4.9. Файл group.h.....	61
5.4.10. Файл ldap_syntaxes.h.....	66
5.4.11. Файл organization_unit.h.....	78
5.4.12. Файл request_queue.h.....	80
5.4.13. Файл schema.h.....	84
5.4.14. Файл user.h.....	87
6. Возвращаемые значения.....	90
7. Синтаксис фильтра поиска.....	90
8. Инструкция по разворачиванию стенда.....	91
8.1. Контроллер домена (Samba AD DC).....	92
8.1.1. Установка ОС «Альт Сервер» 10.x.....	92
8.1.2. Разворачивание сервера Samba AD DC.....	92
8.2. Настройка сервера LDAP.....	95
8.3. Настройка узла с libdomain.....	105
8.4. Тестирование libdomain.....	105
8.5. Использование библиотеки libdomain.....	105

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 ^R	Группа семейств коммерческих операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса
ASN.1	Abstract Syntax Notation One, абстрактная нотация синтаксиса, разработанная ИТУ-Т (стандарты серии X.208/X.680), представляет собой язык описания и кодирования правил для представления

Термин	Описание
	данных. ASN.1 используется для кодирования блоков данных протокола (protocol data unit (PDU), известных также как сообщения (message), блоки (block) или фреймы (frame)) с помощью разных систем кодирования, включая BER (Basic Encoding Rules X.690), CER (Canonical Encoding Rules), DER (Distinguished Encoding Rules), XER (XML Encoding Rules) и PER (Packed Encoding Rules X.691). В случае LDAP используется только простейший BER
DN (Distinguished Name, уникальное имя)	DN состоит из серии RDN, уникально описывающих атрибуты именования по пути ВВЕРХ по DIT от требуемой записи до корневой записи каталога
entry (запись)	Объект, содержащийся в каталоге LDAP. У каждой записи (за исключением базовой, корневой или суффикса DIT) есть одна родительская запись и ноль или более дочерних записей (объектов)
OID (Object Identifier, идентификатор объекта)	Представляет собой разделённое точками значение, например 2.5.6.2 (OID объектного класса country), которое уникально определяет объект и того, кто несет ответственность за определение этого объекта
unauthorized (неавторизованное соединение)	Сессия описывается как неавторизованная, если при её инициализации (посылке bind) передаётся DN без пароля (данных для проверки подлинности). В OpenLDAP реальный эффект такого подсоединения — создание анонимной сессии
IA5	Международный алфавит 5 – эквивалентно ASCII
bind	При установке соединения с сервером LDAP первая операция в последовательности операций называется подсоединением (bind). Операция подсоединения посылает dn записи, которая будет использоваться для аутентификации, и пароль, который будет использоваться (обычно хранящийся в атрибуте userPassword). В случае анонимного соединения оба этих значения будут NULL. При операции bind не разрешён поиск, поэтому используемый для аутентификации DN должен совпадать с DN, с которым запись создавалась изначально.
атрибут	В записи данные содержатся в парах атрибут-значение. Каждый атрибут имеет имя и принадлежит одному или нескольким объектным классам (входит в их состав). Характеристики атрибутов полностью описываются в определениях ASN.1. Один или несколько атрибутов могут быть включены в набор схемы.

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. Получение и редактирование записей в LDAP каталоге:
 - 3.1. Добавление записи в LDAP каталог.
 - 3.2. Редактирование записи в LDAP каталоге.
 - 3.3. Переименовывание записи в LDAP каталоге.
 - 3.4. Перемещение записи в LDAP каталоге.
 - 3.5. Удаление записи в LDAP каталоге.
4. Получение и редактирование списка атрибутов:
 - 4.1. Добавление атрибута.

- 4.2. Удаление атрибута.
- 5. Возможность осуществлять поиск данных в LDAP каталоге:
 - 5.1. Установка базового DN для поиска.
 - 5.2. Установка фильтра по типу объекта для поиска.
 - 5.3. Установка списка атрибутов возвращаемых при поиске.
- 6. Осуществление следующих действий с пользователями и группами:
 - 6.1. Изменение пароля пользователя.
 - 6.2. Создание нового пользователя.
 - 6.3. Создание новой группы.
 - 6.4. Удаление и добавление пользователя в группу.
 - 6.5. Блокирование и разблокирование учётной записи пользователя.
- 7. Осуществление следующих действий с компьютерами:
 - 7.1. Добавление компьютера.
 - 7.2. Удаление компьютера.

5. Объекты библиотеки

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.
- attribute.{h,c} – представляет методы для работы с атрибутами записи LDAP;
- ldap_syntaxes.{h,c} – представляет набор методов для проверки пользовательских данных на соответствие синтаксису LDAP;
- request_queue.{h,c} – представляет методы очереди сообщений;
- schema.{h,c} – представляет методы для работы с LDAP схемой;

5.2. Классы библиотеки

На рис. 1-7 представлена диаграмма классов библиотеки libdomain.

user**C** global

```
OperationReturnCode ld_add_user(LDHandle *handle, const char *name, LDAPAttribute_t **user_attrs, const char *parent)
OperationReturnCode ld_del_user(LDHandle *handle, const char *name, const char *parent)
OperationReturnCode ld_mod_user(LDHandle *handle, const char *name, const char *parent, LDAPAttribute_t **user_attrs)
OperationReturnCode ld_rename_user(LDHandle *handle, const char *old_name, const char *new_name, const char *parent)
OperationReturnCode ld_block_user(LDHandle *handle, const char *name, const char *parent)
OperationReturnCode ld_unblock_user(LDHandle *handle, const char *name, const char *parent)
```

schema**C** global

```
ldap_schema_t *ldap_schema_new(TALLOC_CTX *ctx);
LDAPObjectClass **ldap_schema_object_classes(const ldap_schema_t *schema)
bool ldap_schema_append_attributetype(ldap_schema_t *schema, LDAPAttributeType *attributetype)
LDAPAttributeType **ldap_schema_attribute_types(const ldap_schema_t *schema)
bool ldap_schema_append_objectclass(ldap_schema_t *schema, LDAPObjectClass *objectclass)
```

organizational_unit**C** global

```
OperationReturnCode ld_add_ou(LDHandle *handle, const char *name, LDAPAttribute_t **ou_attrs, const char *parent)
OperationReturnCode ld_del_ou(LDHandle *handle, const char *name, const char *parent)
OperationReturnCode ld_mod_ou(LDHandle *handle, const char *name, const char *parent, LDAPAttribute_t **ou_attrs)
OperationReturnCode ld_rename_ou(LDHandle *handle, const char *old_name, const char *new_name, const char *parent)
```

Рисунок 1. Диаграмма классов библиотеки *libdomain*

domain**S** LDAPAttribute_s

- o char *name
- o char **values

C global

```
char *username, char *password, bool simple_bind, bool use_tls, bool use_sasl, bool use_anon, int timeout,  
config_t *ld_load_config(TALLOC_CTX *ctx, const char *filename);  
config_t *ld_create_config(TALLOC_CTX *talloc_ctx, char *host, int port, int protocol_version, char *base_dn,  
char *cacertfile, char *certfile, char *keyfile);  
void ld_init(LDHandle **handle, const config_t *config)  
void ld_install_default_handlers(LDHandle *handle)  
void ld_install_handler(LDHandle *handle, verto_callback *callback, time_t interval)  
void ld_install_error_handler(LDHandle *handle, error_callback_fn callback)  
void ld_exec(LDHandle *handle)  
void ld_exec_once(LDHandle *handle)  
void ld_free(LDHandle *handle)
```

directory**C** global

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

E LdapDirectoryType

LDAP_TYPE_UNINITIALIZED	= -1,
LDAP_TYPE_UNKNOWN	= 0,
LDAP_TYPE_ACTIVE_DIRECTORY	= 1,
LDAP_TYPE_OPENLDAP	= 2,
LDAP_TYPE_FREE_IPA	= 3

Рисунок 2. Диаграмма классов библиотеки libdomain

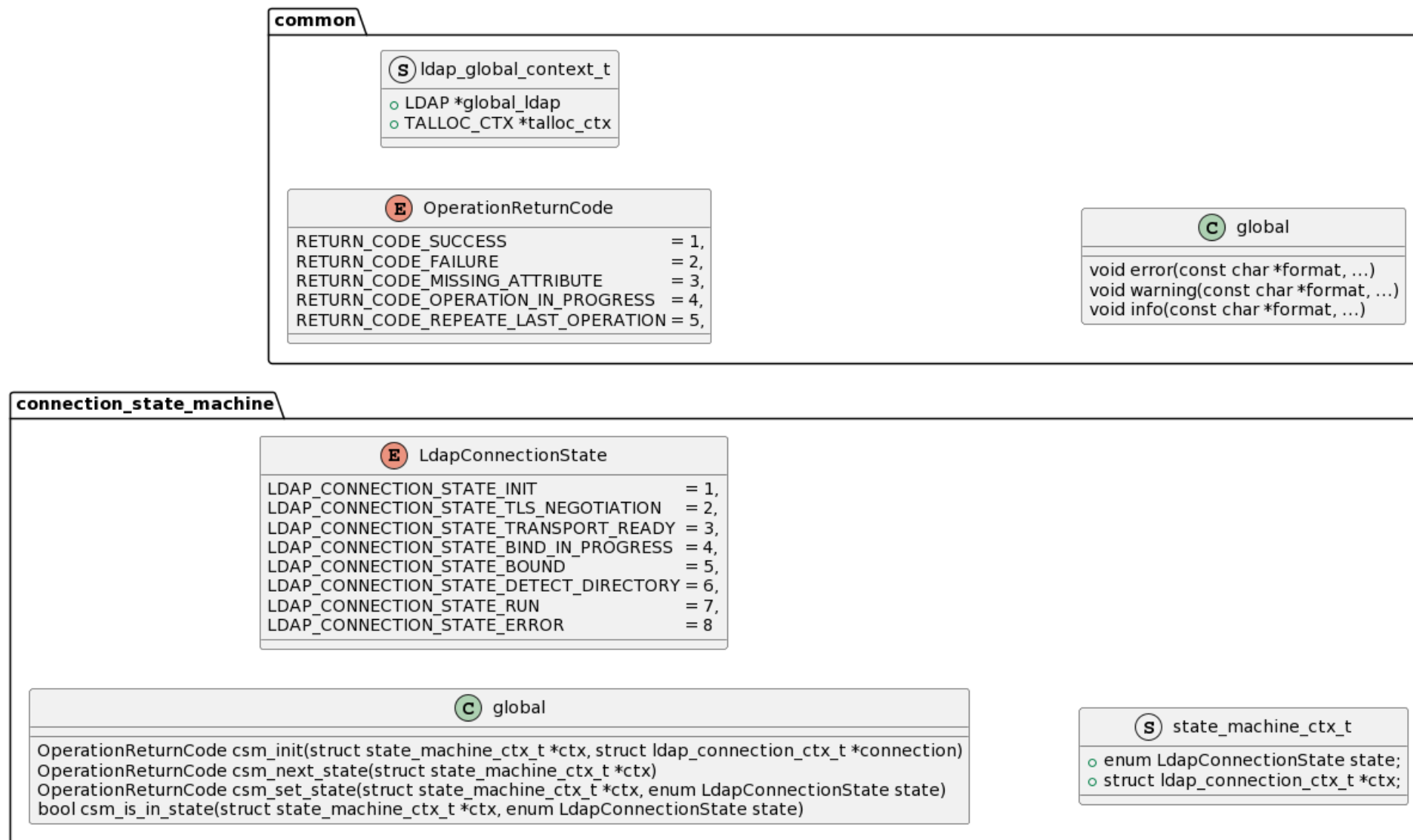


Рисунок 3. Диаграмма классов библиотеки libdomain

connection

(S) ldap_sasl_options_t

- o char *mechanism
- o const char *passwd
- o bool sasl_nocanon
- o short sasl_flags
- o char *sasl_secprops

(S) ldap_sasl_defaults_t

- o short flags
- o char *mechanism;
- o char *realm;
- o char *authcid;
- o char *authzid;
- o char *passwd;

(S) ldap_sasl_params_t

- o char *dn
- o struct berval *passwd
- o LDAPControl **serverctrls
- o LDAPControl **clientctrls

(S) ldap_search_request_t

- o int msgid
- o search_callback_fn on_search_operation

(S) ldap_request_t

- o int msgid
- o operation_callback_fn on_read_operation
- o operation_callback_fn on_write_operation
- o struct Queue_Node_s node

(S) ldap_connection_config_t

- o const char *server
- o int port
- o int protocol_verion
- o bool chase_referrals
- o bool use_start_tls
- o bool use_sasl
- o int bind_type
- o struct ldap_sasl_options_t *sasl_options
- o int ssl_mode
- o const char *tls_ca_cert_file
- o const char *tls_ca_cert_path
- o const char *tls_cert_file
- o const char *tls_key_file
- o bool *tls_require_cert
- o int tls_min_protocol_version
- o int search_timelimit
- o int network_timeout

(S) ldap_connection_ctx_t

- o LDHandle *handle
- o LDAP *ldap
- o struct ldap_connection_ctx_t *next
- o struct ldap_connection_ctx_t *prev
- o int fd
- o struct verto_ctx *base
- o struct verto_ev *read_event
- o struct verto_ev *write_event
- o operation_callback_fn on_error_operation
- o int bind_type
- o int directory_type
- o int msgid
- o const char *rmec
- o struct request_queue *callqueue
- o struct ldap_request_t read_requests
- o struct ldap_request_t write_requests
- o struct ldap_search_request_t search_requests
- o int n_read_requests
- o int n_write_requests
- o int n_search_requests
- o int n_reconnect_attempts
- o struct state_machine_ctx_t *state_machine
- o struct ldap_sasl_defaults_t *ldap_defaults
- o struct ldap_sasl_params_t *ldap_params
- o struct ldap_connection_config_t *config

(E) BindType

- BIND_TYPE_INTERACTIVE = 1,
- BIND_TYPE_SIMPLE = 2


(C) global

```
OperationReturnCode connection_configure(struct ldap_global_context_t *global_ctx, struct ldap_connection_ctx_t *connection, struct ldap_connection_config_t *config);
OperationReturnCode connection_start_tls(struct ldap_connection_ctx_t *connection);
OperationReturnCode connection_sasl_bind(struct ldap_connection_ctx_t *connection);
OperationReturnCode connection_ldap_bind(struct ldap_connection_ctx_t *connection);
OperationReturnCode connection_close(struct ldap_connection_ctx_t *connection);
OperationReturnCode connection_bind_on_read(int rc, LDAPMessage *message, struct ldap_connection_ctx_t *connection);
OperationReturnCode connection_start_tls_on_read(int rc, LDAPMessage *message, struct ldap_connection_ctx_t *connection);
void connection_on_read(verto_ctx *ctx, verto_ev *ev);
void connection_on_write(verto_ctx *ctx, verto_ev *ev);
```

Рисунок 4. Диаграмма классов библиотеки libdomain

entry global

```
OperationReturnCode add(struct ldap_connection_ctx_t *connection, const char *dn, LDAPMod **attrs)
OperationReturnCode add_on_read(int rc, LDAPMessage *message, ldap_connection_ctx_t *connection)
OperationReturnCode search(struct ldap_connection_ctx_t *connection, const char *base_dn, int scope,
const char *filter, char **attrs, bool attrsonly, search_callback_fn search_callback)
OperationReturnCode search_on_read(int rc, LDAPMessage *message, struct ldap_connection_ctx_t *connection)
OperationReturnCode modify(struct ldap_connection_ctx_t *connection, const char *dn, LDAPMod **attrs)
OperationReturnCode modify_on_read(int rc, LDAPMessage *message, ldap_connection_ctx_t *connection)
OperationReturnCode delete(struct ldap_connection_ctx_t *connection, const char *dn)
OperationReturnCode delete_on_read(int rc, LDAPMessage *message, ldap_connection_ctx_t *connection)
OperationReturnCode ld_rename(struct ldap_connection_ctx_t *connection, const char *olddn, const char *newdn, const char *new_parent, bool delete_original)
OperationReturnCode rename_on_read(int rc, LDAPMessage *message, ldap_connection_ctx_t *connection)
OperationReturnCode whoami(struct ldap_connection_ctx_t *connection)
OperationReturnCode whoami_on_read(int rc, LDAPMessage *message, struct ldap_connection_ctx_t *connection)
ld_entry_t *ld_entry_new(TALLOC_CTX *ctx, const char *dn)
const char *ld_entry_get_dn(ld_entry_t *entry)
OperationReturnCode ld_entry_add_attribute(ld_entry_t *entry, const LDAPAttribute_t *attr)
LDAPAttribute_t *ld_entry_get_attribute(ld_entry_t *entry, const char *name_or_oid)
LDAPAttribute_t **ld_entry_get_attributes(ld_entry_t *entry)
```

computer global

```
OperationReturnCode ld_add_computer(LDHandle *handle, const char *name, LDAPAttribute_t **attrs, const char *parent)
OperationReturnCode ld_del_computer(LDHandle *handle, const char *name, const char *parent)
OperationReturnCode ld_mod_computer(LDHandle *handle, const char *name, const char *parent, LDAPAttribute_t **computer_attrs)
OperationReturnCode ld_rename_computer(LDHandle *handle, const char *old_name, const char *new_name, const char *parent)
```

Рисунок 5. Диаграмма классов библиотеки libdomain

request_queue

E RequestQueueErrorCode

OPERATION_SUCCESS	= 0,
OPERATION_ERROR_FULL	= 1,
OPERATION_ERROR_INVALID_PARAMETER	= 2,
OPERATION_ERROR_FAULT	= 3

S Queue_Node_s

o Queue_Node_s *prev

C global

```
request_queue *ldap_schema_new(TALLOC_CTX *ctx, unsigned int capacity);
RequestQueueErrorCode request_queue_push(request_queue *queue, struct Queue_Node_s *node)
Queue_Node_s *request_queue_pop(request_queue *queue)
Queue_Node_s *request_queue_peek(request_queue *queue)
bool request_queue_empty(request_queue *queue)
RequestQueueErrorCode request_queue_move(request_queue *from, request_queue *to)
```

group

E GroupScope

GROUP_SCOPE_DOMAIN_LOCAL	= 0,
GROUP_SCOPE_GLOBAL	= 1,
GROUP_SCOPE_UNIVERSAL	= 2

E GroupCategory

GROUP_CATEGORY_DISTRIBUTION	= 0,
GROUP_CATEGORY_SECURITY	= 1

C global

```
OperationReturnCode ld_add_group(LDHandle *handle, const char *name, LDAPAttribute_t **attributes, const char *parent)
OperationReturnCode ld_del_group(LDHandle *handle, const char *name, const char *parent)
OperationReturnCode ld_mod_group(LDHandle *handle, const char *name, const char *parent, LDAPAttribute_t **group_attrs)
OperationReturnCode ld_rename_group(LDHandle *handle, const char *old_name, const char *new_name, const char *parent)
OperationReturnCode ld_group_add_user(LDHandle *handle, const char *group_name, const char *user_name)
OperationReturnCode ld_group_remove_user(LDHandle *handle, const char *group_name, const char *user_name)
```

Рисунок 6. Диаграмма классов библиотеки libdomain

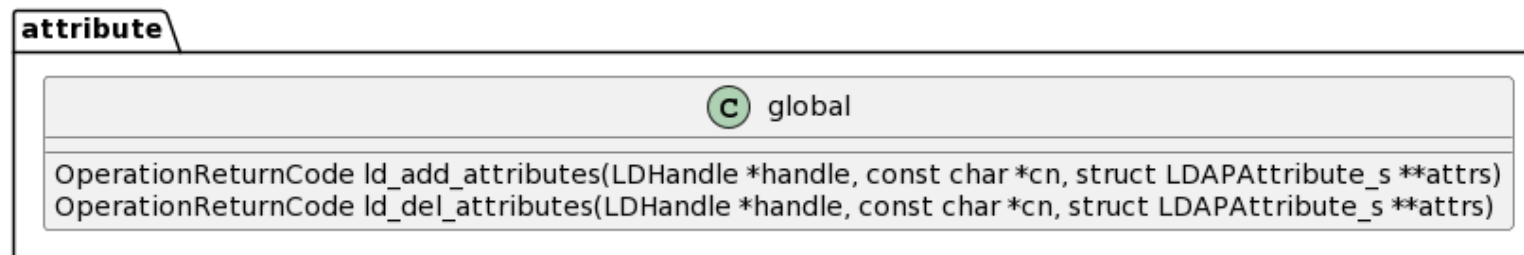
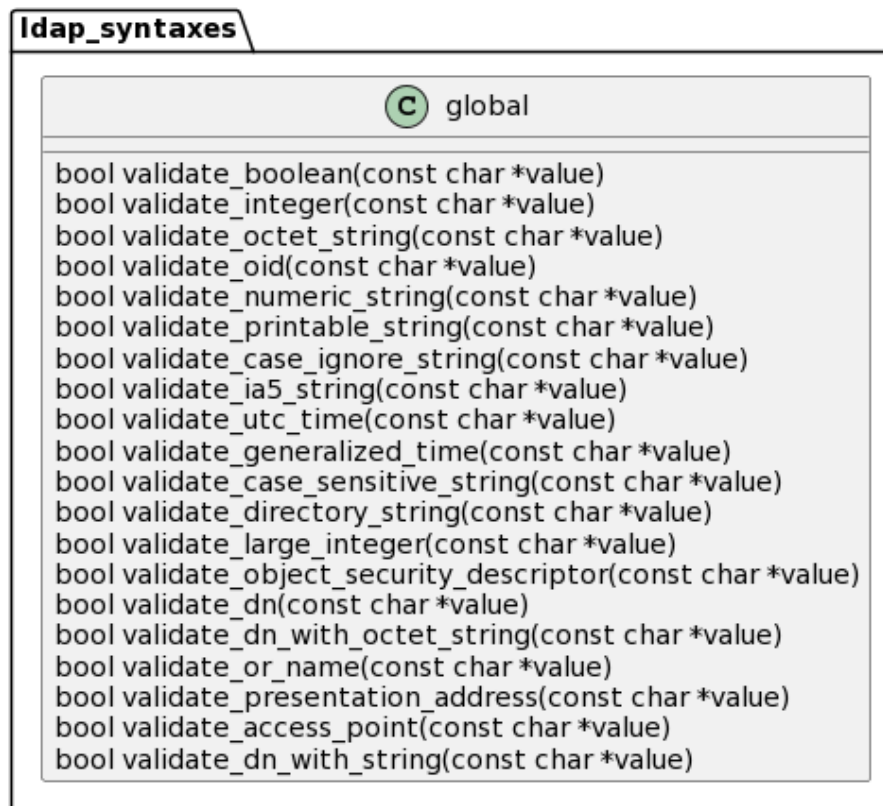


Рисунок 7. Диаграмма классов библиотеки *libdomain*

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 csm_state_value_t

Структура csm_state_value_t содержит значения состояний подключений.

Публичные переменные csm_state_value_t представлены в Таблице 3.

Таблица 3. Публичные переменные csm_state_value_t

Имя	Тип	Описание
state	int	Состояние
value	char*	Значение

5.3.3. Структура ld_config_s

Структура ld_config_s содержит данные, необходимые для подключения (конфигурацию, которая будет использоваться во время соединения).

Публичные переменные ld_config_s представлены в Таблице 4.

Таблица 4. Публичные переменные ld_config_s

Имя	Тип	Описание
host	char*	LDAP-сервер для подключения
protocol_version	int	Версия протокола LDAP
base_dn	char*	Базовый DN для привязки
username	char*	Имя пользователя, с которым происходит подключение. Может быть NULL

password	char*	Пароль для указанного пользователя. Может быть NULL
simple_bind	bool	Возможно использовать два типа привязки: простую и интерактивную. Для подключения, использующего простую привязку, указывается true. Для подключения, использующего интерактивную привязку, указывается FALSE
use_tls	bool	TRUE, если необходимо установить TLS или SSL соединение
use_sasl	bool	TRUE, если используется протокол SASL
use_anon	bool	TRUE, если выполняется анонимное подключение («анонимная привязка»)
timeout	int	Время ожидания (тайм-аут) операции. Как только достигается указанное предельное значение, операция завершается неудачно
cacertfile	char*	Определяет полный путь к сертификату СА, который используется для проверки представленного сертификата сервера
certfile	char*	Путь к файлу сертификата клиента
keyfile	char*	Файл закрытого ключа, связанный с сертификатом клиента

5.3.4. Структура ld_entry_s

Структура ld_entry_s содержит информацию о записях LDAP.

Публичные переменные ld_entry_s представлены в Таблице 5.

Таблица 5. Публичные переменные ld_entry_s

Имя	Тип	Описание
dn	char*	Отличительное имя записи LDAP
attributes	GHashTable*	Хэш-таблица с атрибутами записи

5.3.5. 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 6.

Таблица 6. 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
bind_type	int	Тип подключения BIND (1 – интерактивное, 2 – простое)
sasl_options	struct ldap_sasl_options_t*	Параметры SASL аутентификации
ssl_mode	int	Устанавливает уровень защиты (режимы подключения по SSL)
tls_ca_cert_file	const char*	Файл, содержащий сертификаты всех удостоверяющих центров
tls_ca_cert_path	const char*	Каталог, содержащий файлы сертификатов удостоверяющих центров (путь к доверенному хранилищу ключей)
tls_cert_file	const char*	Файл, содержащий сертификат клиента
tls_key_file	const char*	Файл, содержащий закрытый ключ клиента (путь к файлу, в котором содержится закрытый ключ, ассоциированный с сертификатом, указанным в tls_cert_file)
tls_require_cert	bool*	Принудительно требовать от сервера и клиента наличие сертификатов
tls_min_protocol_version	int	Определяет минимальную версию протокола TLS
search_timelimit	int	Определяет время ожидания результатов поиска (в миллисекундах)
network_timeout	int	Определяет время сетевого ожидания (в миллисекундах)

5.3.6. Структура ldap_connection_ctx_t

Структура ldap_connection_ctx_t содержит данные сконфигурированного соединения.

Публичные переменные ldap_connection_ctx_t представлены в Таблице 7.

Таблица 7. Публичные переменные ldap_connection_ctx_t

Имя	Тип	Описание
handle	LDHandle*	Указатель на дескриптор сеанса libdomain
ldap	LDAP*	Контекст LDAP
next	ldap_connection_ctx_t*	Следующее соединение в очереди соединений
prev	ldap_connection_ctx_t*	Предыдущее соединение в очереди соединений
fd	int	Файловый дескриптор
handlers_installed	bool	Были ли установлены обработчики сообщений
base	verto_ctx*	Контекст события
read_event	verto_ev*	Событие чтения
write_event	verto_ev*	Событие записи
on_error_operation	operation_callback_fn	Обратный вызов, выполняющийся во время операции привязки.
bind_type	int	Тип подключения BIND (1 – интерактивное, 2 – простое)
directory_type	int	Тип каталога LDAP (1 – AD, 2 – OpenLDAP, 3 – FreeIPA)
msgid	int	Идентификатор сообщения
rmech	char*	Механизм используемый gssapi
callqueue	request_queue*	Очередь сообщений, служит для обработки LDAP сообщений
read_request	ldap_request_t	Запросы на чтение, например, результат операции поиска
write_request	ldap_request_t	Запросы на запись (в текущий момент не

		используется)
search_request	ldap_search_request_t	Поисковые запросы, содержит список обработчиков для операции поиска
n_read_requests	int	Количество активных запросов на чтение
n_write_requests	int	Количество активных запросов на запись
n_search_requests	int	Количество текущих поисковых операций
n_reconnect_attempts	int	Количество попыток переподключения (максимум 10)
state_machine	state_machine_ctx_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
config	ldap_connection_config_t*	Параметры для SASL/TLS подключения

5.3.7. Структура ldap_global_context_t

Структура ldap_global_context_t содержит глобальный контекст, связанный с LDHandle.

Публичные переменные ldap_global_context_t представлены в Таблице 8.

Таблица 8. Публичные переменные ldap_global_context_t

Имя	Тип	Описание
global_ldap	LDAP*	Глобальный контекст ldap для совместного использования между соединениями
talloc_ctx	TALLOC_CTX*	Указатель на действительный TALLOC_CTX. Используется при работе с записями ldap

5.3.8. Структура ldap_request_t

Структура ldap_request_t содержит запрос.

Публичные переменные ldap_request_t представлены в Таблице 9.

Таблица 9. Публичные переменные *ldap_global_context_t*

Имя	Тип	Описание
msgid	int	Номер сообщения
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
node	Queue_Node_s	Указатель на узел в очереди сообщений

5.3.9. Структура *ldap_sasl_default_t*

Структура *ldap_sasl_default_t* содержит параметры по умолчанию SASL.

Публичные переменные *ldap_sasl_default_t* представлены в Таблице 10.

Таблица 10. 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

5.3.10. Структура *ldap_sasl_options_t*

Структура *ldap_sasl_options_t* содержит параметры SASL.

Публичные переменные *ldap_sasl_options_t* представлены в Таблице 11.

Таблица 11. Публичные переменные *ldap_sasl_options_t*

Имя	Тип	Описание
mechanism	char*	Механизм SASL
passwd	char*	Пароль пользователя
sasl_nocanon	bool	Не 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

5.3.11. 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 12.

Таблица 12. 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

5.3.12. Структура *ldap_schema_t*

Структура *ldap_schema_t* предоставляет схему LDAP.

Публичные переменные *ldap_schema_t* представлены в Таблице 13.

Таблица 13. Публичные переменные *ldap_schema_t*

Имя	Тип	Описание
object_classes	LDAPObjectClass**	Указатель на список классов объектов. Список должен заканчиваться NULL
attribute_types	LDAPAttributeType**	Указатель на список типов атрибутов. Список должен заканчиваться NULL

object_classes_size	int	Текущее количество классов объектов в очереди
object_classes_capacity	int	Максимально допустимое количество классов объектов в очереди
attribute_types_size	int	Текущее количество типов атрибутов в очереди
attribute_types_capacity	int	Максимально допустимое количество типов атрибутов в очереди

5.3.13. Структура ldap_search_request_t

Структура ldap_search_request_t содержит поисковый запрос.

Публичные переменные ldap_search_request_t представлены в Таблице 14.

Таблица 14. Публичные переменные ldap_search_request_t

Имя	Тип	Описание
msgid	int	Номер сообщения
on_search_operation	search_callback_fn	Операция которая будет вызвана для обработки результатов поиска

5.3.14. Структура LDAPAttribute_s

Структура LDAPAttribute_s представляет атрибут LDAP.

Публичные переменные LDAPAttribute_s представлены в Таблице 15.

Таблица 15. Публичные переменные LDAPAttribute_s

Имя	Тип	Описание
name	char*	Имя атрибута
values	char**	Список значений атрибутов, завершающийся NULL

5.3.15. Структура ldhandle

Структура ldhandle содержит указатель на дескриптор сеанса libdomain.

Публичные переменные ldhandle представлены в Таблице 16.

Таблица 16. Публичные переменные *ldhandle*

Имя	Тип	Описание
<code>talloc_ctx</code>	<code>TALLOC_CTX*</code>	Глобальный контекст библиотеки Talloc
<code>global_ctx</code>	<code>struct ldap_global_context_t*</code>	Глобальный контекст библиотеки
<code>connection_ctx</code>	<code>struct ldap_connection_ctx_t*</code>	Контекст соединения
<code>config_ctx</code>	<code>struct ldap_connection_config_t*</code>	Конфигурация подключения
<code>global_config</code>	<code>ld_config_t*</code>	Глобальная конфигурация библиотеки

5.3.16. Структура `option_value_t`

Структура `option_value_t` содержит пару параметр-значение.

Публичные переменные `option_value_t` представлены в Таблице 17.

Таблица 17. Публичные переменные *option_value_t*

Имя	Тип	Описание
<code>option</code>	<code>int</code>	Параметр
<code>value</code>	<code>char*</code>	Значение

5.3.17. Структура `Queue_Node_s`

Структура `Queue_Node_s` содержит указатель на узел очереди.

Публичные переменные `Queue_Node_s` представлены в Таблице 18.

Таблица 18. Публичные переменные *Queue_Node_s*

Имя	Тип	Описание
<code>prev</code>	<code>struct Queue_Node_s*</code>	Указатель на предыдущий узел в очереди

5.3.18. Структура `request_queue`

Структура `request_queue` содержит указатель на очередь.

Публичные переменные `request_queue` представлены в Таблице 19.

Таблица 19. Публичные переменные *request_queue*

Имя	Тип	Описание
head	struct Queue_Node_s*	Указатель на первый узел в очереди
tail	struct Queue_Node_s*	Указатель на последний узел в очереди
size	int	Размер очереди
capacity	int	Максимальный размер очереди

5.3.19. Структура *state_machine_ctx_t*

Структура *state_machine_ctx_t* представляет собой конечный автомат соединения.

Публичные переменные *state_machine_ctx_t* представлены в Таблице 20.

Таблица 20. Публичные переменные *state_machine_ctx_t*

Имя	Тип	Описание
state	enum LdapConnectionState	Состояние соединения
ctx	struct ldap_connection_ctx_t	Контекст соединения

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

- LDAP_CONNECTION_STATE_INIT – начальное состояние;
- LDAP_CONNECTION_STATE_TLS_NEGOTIATION – соединение находится в процессе согласования шифрования TLS;
- 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_STATE_BOUND – состояние в которое переключается соединение при завершении операции интерактивного подсоединения;

- LDAP_CONNECTION_STATE_DETECT_DIRECTORY – состояние в которое переключается соединение в процессе определения типа службы каталогов LDAP;
- LDAP_CONNECTION_STATE_RUN – рабочее состояние соединения из этого состояния нет прямых переходов;
- LDAP_CONNECTION_STATE_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. 8.

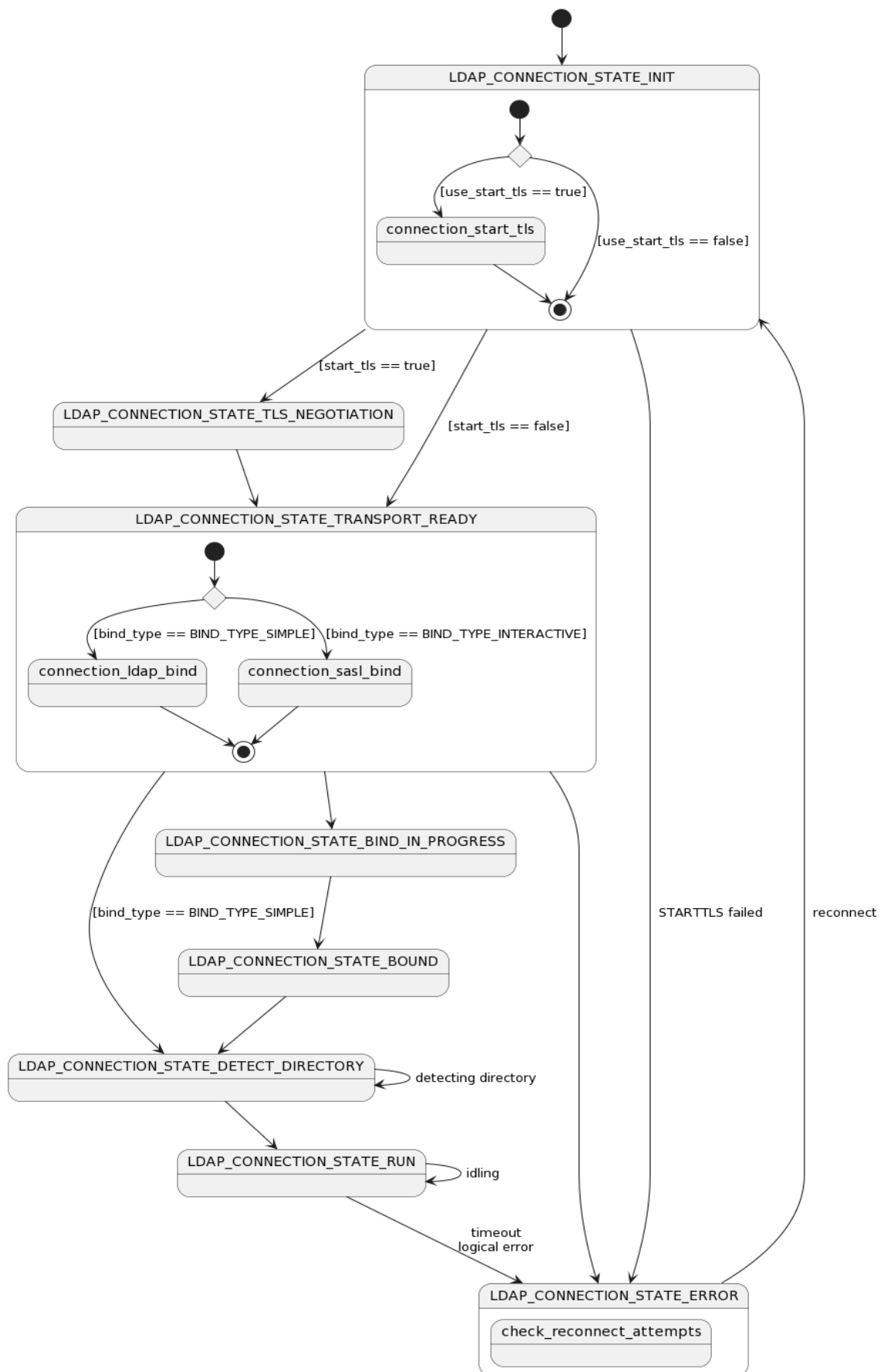


Рисунок 8. Диаграмма машины состояний управлением соединением

5.4. Файлы

5.4.1. Файл attribute.h

Функции attribute.h приведены в Таблице 21.

Таблица 21. Функции attribute.h

Имя	Описание
ld_add_attributes	Добавляет атрибуты
ld_del_attributes	Удаляет атрибуты

5.4.1.1. Функция ld_add_attributes

Функция ld_add_attributes

Синтаксис функции ld_add_attributes:

```
enum OperationReturnCode ld_add_attributes(LDHandle* handle, const char* cn, struct LDAPAttribute_s** attrs)
```

Параметры функции ld_add_attributes приведены в Таблице 22.

Таблица 22. Параметры функции ld_add_attributes

Имя	Тип	Описание
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
cn [in]	char	Имя записи
attrs [in]	LDAPAttribute_s	Список добавляемых атрибутов

Возвращаемое значение:

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

5.4.1.2. Функция ld_del_attributes

Функция ld_del_attributes

Синтаксис функции ld_del_attributes:

```
enum OperationReturnCode ld_del_attributes(LDHandle* handle, const char* cn, struct LDAPAttribute_s** attrs)
```

Параметры функции ld_del_attributes приведены в Таблице 23.

Таблица 23. Параметры функции *ld_del_attributes*

Имя	Тип	Описание
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
cn [in]	char	Имя записи
attrs [in]	LDAPAttribute_s	Список удаляемых атрибутов

Возвращаемое значение:

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

5.4.2. Файл common.h

Классы

struct ldap_global_context_t

Перечисления

enum OperationReturnCode

```
{
    RETURN_CODE_SUCCESS                = 1,
    RETURN_CODE_FAILURE                 = 2,
    RETURN_CODE_MISSING_ATTRIBUTE       = 3,
    RETURN_CODE_OPERATION_IN_PROGRESS  = 4,
    RETURN_CODE_REPEAT_LAST_OPERATION  = 5,
}
```

Значения LdapConnectionState приведены в Таблице 24.

Таблица 24. Значения *LdapConnectionState*

Значение	Описание
RETURN_CODE_SUCCESS	Успешное завершение операции
RETURN_CODE_FAILURE	Операция завершилась с ошибкой
RETURN_CODE_MISSING_ATTRIBUTE	Не указан обязательный атрибут для записи LDAP
RETURN_CODE_OPERATION_IN_PROGRESS	Текущая операция продолжается
RETURN_CODE_REPEAT_LAST_OPERATION	Необходимо повторить вызов функции

Типы

`using ldap_global_context_t = struct ldap_global_context_t`

`ldap_global_context_t` – один из контекстов, связанных с `LDHandle`.

Функции `common.h` приведены в Таблице 25.

Таблица 25. Функции `common.h`

Имя	Описание
<code>error</code>	Записывает ошибку в <code>stderr</code>
<code>warning</code>	Записывает предупреждение в <code>stderr</code>
<code>info</code>	Записывает информацию в <code>stderr</code>

5.4.2.1. Функция `error`

Функция `error` записывает ошибку в `stderr`.

Синтаксис функции `error`:

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

Параметры функции `error` приведены в Таблице 26.

Таблица 26. Параметры функции `error`

Имя	Тип	Описание
<code>format</code>	<code>char</code>	Формат, используемый в функции <code>printf</code>

5.4.2.2. Функция `warning`

Функция `warning` записывает предупреждение в `stderr`.

Синтаксис функции `warning`:

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

Параметры функции `warning` приведены в Таблице 27.

Таблица 27. Параметры функции `warning`

Имя	Тип	Описание
<code>format</code>	<code>char</code>	Формат, используемый в функции <code>printf</code>

5.4.2.3. Функция `info`

Функция `info` записывает информацию в `stderr`.

Синтаксис функции info:

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

Параметры функции info приведены в Таблице 28.

Таблица 28. Параметры функции info

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

5.4.3. Файл computer.h

Функции computer.h приведены в Таблице 29.

Таблица 29. Функции computer.h

Имя	Описание
ld_add_computer	Добавляет учётную запись компьютера
ld_del_computer	Удаляет учётную запись компьютера
ld_mod_computer	Изменяет учётную запись компьютера
ld_rename_computer	Переименовывает учётную запись компьютера

5.4.3.1. Функция ld_add_computer

Функция ld_add_computer добавляет учётную запись компьютера.

Синтаксис функции ld_add_computer:

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

Параметры функции ld_add_computer приведены в Таблице 30.

Таблица 30. Параметры функции ld_add_computer

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

Возвращаемое значение:

- RETURN_CODE_SUCCESS, если функция завершается успешно;

- RETURN_CODE_FAILURE, если функция выполняется неудачно.

5.4.3.2. Функция ld_del_computer

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 31.

Таблица 31. 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.

5.4.3.3. Функция ld_mod_computer

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 32.

Таблица 32. 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.

5.4.3.4. Функция ld_rename_computer

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 33.

Таблица 33. 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.

5.4.4. Файл connection.h

Классы

```
struct ldap_sasl_options_t
struct ldap_sasl_defaults_t
struct ldap_sasl_params_t
struct ldap_connection_config_t
struct ldap_search_request_t
struct ldap_request_t
struct ldap_connection_ctx_t
```

Перечисления

```
enum BindType
{
    BIND_TYPE_INTERACTIVE = 1,
    BIND_TYPE_SIMPLE      = 2,
```

};

Значения привязки (BindType) приведены в Таблице 34.

Таблица 34. Значения BindType

Значение	Описание
BIND_TYPE_INTERACTIVE	Интерактивная привязка
BIND_TYPE_SIMPLE	Простая привязка

Типы

```
using ldap_sasl_options_t = struct ldap_sasl_options_t
using ldap_sasl_defaults_t = struct ldap_defaults_t
using ldap_sasl_params_t = struct ldap_params_t
using ldap_connection_config_t = struct ldap_connection_config_t
using ld_entry_t = struct ld_entry_s
using operation_callback_fn = enum OperationReturnCode(*) (int,
LDAPMessage*, struct ldap_connection_ctx_t*)
using search_callback_fn = enum OperationReturnCode(*) (struct
ldap_connection_ctx_t*connection, ld_entry_t**entries)
using LDHandle = struct ldhandle
using ldap_search_request_t = struct ldap_search_request_t
using ldap_request_t = struct ldap_request_t
using ldap_connection_ctx_t = struct ldap_connection_ctx_t
```

Определения

```
#define MAX_REQUESTS
```

Функции connection.h приведены в Таблице 35.

Таблица 35. Функции connection.h

Имя	Описание
connection_configure	Настраивает соединение (подключение)
connection_start_tls	Настраивает TLS транспорт
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.
connection_start_tls_on_read	Обратный вызов, выполняющийся во время инициации соединения TLS.

5.4.4.1. Функция connection_configure

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 36.

Таблица 36. 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.

5.4.4.2. Функция connection_start_tls

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 37.

Таблица 37. Параметры функции connection_start_tls

Имя	Тип	Описание
connection [in]	ldap_connection_ctx_t	Сконфигурированное соединение

5.4.4.3. Функция connection_sasl_bind

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 38.

Таблица 38. 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.

5.4.4.4. Функция connection_ldap_bind

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 39.

Таблица 39. Параметры функции `connection_ldap_bind`

Имя	Тип	Описание
<code>connection</code> [in]	<code>ldap_connection_ctx_t</code>	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.

5.4.4.5. Функция `connection_close`

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 40.

Таблица 40. Параметры функции `connection_close`

Имя	Тип	Описание
<code>connection</code> [in]	<code>ldap_connection_ctx_t</code>	Сконфигурированное соединение

Возвращаемое значение:

- `RETURN_CODE_SUCCESS`, если функция завершается успешно.

5.4.4.6. Функция `connection_on_read`

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 41.

Таблица 41. Параметры функции `connection_on_read`

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

5.4.4.7. Функция `connection_on_write`

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 42.

Таблица 42. Параметры функции `connection_on_write`

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

5.4.4.8. Функция `connection_bind_on_read`

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 43.

Таблица 43. 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.

5.4.4.9. Функция connection_start_tls_on_read

Этот обратный вызов выполняется во время инициации соединения tls.

Синтаксис функции connection_start_tls_on_read:

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

Параметры функции connection_start_tls_on_read приведены в Таблице 44.

Таблица 44. Параметры функции connection_start_tls_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.

5.4.5. Файл connection_state_machine.h

Классы

```
struct state_machine_ctx_t
```

Перечисления

```
enum LdapConnectionState
```

```
{
    LDAP_CONNECTION_STATE_INIT                = 1,
    LDAP_CONNECTION_STATE_TLS_NEGOTIATION     = 2,
    LDAP_CONNECTION_STATE_TRANSPORT_READY     = 3,
    LDAP_CONNECTION_STATE_BIND_IN_PROGRESS    = 4,
    LDAP_CONNECTION_STATE_BOUND                = 5,
    LDAP_CONNECTION_STATE_DETECT_DIRECTORY    = 6,
    LDAP_CONNECTION_STATE_RUN                  = 7,
    LDAP_CONNECTION_STATE_ERROR                = 8,
}
```

Значения состояния соединения(LdapConnectionState) приведены в Таблице 45.

Таблица 45. Значения LdapConnectionState

Значение	Описание
LDAP_CONNECTION_STATE_INIT	Исходное состояние соединения LDAP
LDAP_CONNECTION_STATE_TLS_NEGOTIATION	Соединение находится в процессе согласования шифрования TLS
LDAP_CONNECTION_STATE_TRANSPORT_READY	Базовый транспортный уровень готов к обмену данными
LDAP_CONNECTION_STATE_BIND_IN_PROGRESS	Соединение находится в процессе привязки
LDAP_CONNECTION_STATE_BOUND	Соединение успешно выполнило привязку к LDAP
LDAP_CONNECTION_STATE_DETECT_DIRECTORY	Соединение находится в процессе определения типа службы каталогов LDAP
LDAP_CONNECTION_STATE_RUN	Соединение LDAP активно и готово к работе
LDAP_CONNECTION_STATE_ERROR	Соединение LDAP находится в состоянии ошибки

Типы

```
using state_machine_ctx_t = struct state_machine_ctx_t
```

Функции connection_state_machine.h приведены в Таблице 46.

Таблица 46. Функции connection_state_machine.h

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.
csm_is_in_state	Проверяет, находится ли конечный автомат в желаемом состоянии.

5.4.5.1. Функция csm_init

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 47.

Таблица 47. 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.

5.4.5.2. Функция csm_next_state

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 48.

Таблица 48. 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.

5.4.5.3. Функция csm_set_state

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 49.

Таблица 49. 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.

5.4.5.4. Функция csm_is_in_state

Функция csm_is_in_state проверяет, находится ли конечный автомат в желаемом состоянии.

Синтаксис функции csm_is_in_state:

```
bool csm_is_in_state(struct state_machine_ctx_t* ctx, enum  
LdapConnectionState state)
```

Параметры функции csm_is_in_state приведены в Таблице 50.

Таблица 50. Параметры функции csm_is_in_state

Имя	Тип	Описание
ctx [in]	state_machine_ctx_t	Текущее состояние машины
state [in]	LdapConnectionState	Запрашиваемое состояние машины

Возвращаемое значение:

- TRUE, если текущее состояние машины соответствует запрашиваемому состоянию (state);
- FALSE, если текущее состояние машины не соответствует запрашиваемому состоянию (state).

5.4.6. Файл directory.h

Перечисления

```
enum LdapDirectoryType
{
    LDAP_TYPE_UNINITIALIZED    = -1,
    LDAP_TYPE_UNKNOWN          = 0,
    LDAP_TYPE_ACTIVE_DIRECTORY = 1,
    LDAP_TYPE_OPENLDAP         = 2,
    LDAP_TYPE_FREE_IPA         = 3
};
```

Значения LdapDirectoryType приведены в Таблице 51.

Таблица 51. Значения LdapDirectoryType

Значение	Описание
LDAP_TYPE_UNINITIALIZED	Тип каталога не был инициализирован с допустимым значением. Соединение изначально инициализируется с использованием этого типа каталога
LDAP_TYPE_UNKNOWN	Не удалось определить тип службы каталогов
LDAP_TYPE_ACTIVE_DIRECTORY	Каталог Samba/Microsoft Active Directory
LDAP_TYPE_OPENLDAP	Каталог OpenLDAP
LDAP_TYPE_FREE_IPA	Каталог FreeIPA

Функции directory.h приведены в Таблице 52.

Таблица 52. 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.

5.4.6.1. Функция directory_get_type

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 53.

Таблица 53. 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.

5.4.6.2. Функция `directory_parse_result`

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(int rc, LDAPmessage*  
message, struct ldap_connection_ctx_t* connection)
```

Параметры функции `directory_parse_result` приведены в Таблице 54.

Таблица 54. Параметры функции `directory_parse_result`

Имя	Тип	Описание
rc [in]	int	Код возврата <code>ldap_result</code>
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. Файл `domain.h`

Классы

```
struct LDAPAttribute_s
```

Типы

```
using LDHandle = struct ldhandle
```

LDHandle — структура, представляющая дескриптор сеанса libdomain.

```
using ld_config_t = struct ld_config_s
```

ld_config_t – структура, содержащая конфигурацию, которая будет использоваться во время соединения.

```
using LDAPAttribute_t = struct LDAPAttribute_s
```

Структура LDAPAttribute_t представляет атрибут LDAP.

```
using error_callback_fn = enum OperationReturnCode(*) (int, void*, void*)
```

```
typedef enum OperationReturnCode(*error_callback_fn) (int, void*, void*)
```

Тип определяет обратный вызов ошибки. Этот обратный вызов будет запущен, когда соединение перейдет в состояние LDAP_CONNECTION_STATE_ERROR.

Функции domain.h приведены в Таблице 55.

Таблица 55. Функции domain.h

Имя	Описание
ld_load_config	Заполняет поля структуры конфигурации из файла (загружает настройки подключения из файла)
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 ошибок для приложения
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 циклически перебирает список событий. Может заблокировать цикл событий (event loop)

ld_free	Освобождает дескриптор библиотеки и связанные с ней ресурсы. После освобождения дескриптора будет невозможно выполнять какие-либо операции.
---------	---

5.4.7.1. Функция ld_load_config

Функция ld_load_config заполняет поля структуры конфигурации из файла.

Синтаксис функции ld_load_config:

```
ld_config_t* ld_load_config(TALLOC_CTX* talloc_ctx, const char* filename)
```

Параметры функции ld_load_config приведены в Таблице 56.

Таблица 56. Параметры функции ld_load_config

Имя	Тип	Описание
talloc_ctx	TALLOC_CTX*	Контекст библиотеки talloc
filename [in]	char*	Путь к файлу конфигурации

Возвращаемое значение:

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

Пример содержимого файла конфигурации (config.ini):

```
host = "ldap://dc0.domain.alt"
base_dn = "dc=domain,dc=alt"
username = "admin"
password = "password"
timeout = 1000
protocol_version = 3
ca_cert_file = "CA.cert"
cert_file = "dc0.domain.alt.cert"
key_file = "dc0.domain.alt.key"
simple_bind = false
use_tls = true
use_sasl = true
use_anon = false
```

Примечание. У пользователя должны быть права на чтение и доступ к каталогу в котором лежит файл конфигурации.

5.4.7.2. Функция ld_create_config

Функция ld_create_config заполняет поля структуры конфигурации.

Синтаксис функции ld_create_config:

```
ld_config_t *ld_create_config(TALLOC_CTX* talloc_ctx,  
                               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 57.

Таблица 57. Параметры функции ld_create_config

Имя	Тип	Описание
talloc_ctx	TALLOC_CTX	Контекст библиотеки talloc
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.

5.4.7.3. Функция ld_init

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 58.

Таблица 58. Параметры функции ld_init

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

5.4.7.4. Функция ld_install_default_handlers

Функция ld_install_default_handlers устанавливает обработчики по умолчанию для управления соединением. Этот метод необходимо вызывать перед выполнением каких-либо операций.

Синтаксис функции ld_install_default_handlers:

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

Параметры функции ld_install_default_handlers приведены в Таблице 59.

Таблица 59. Параметры функции `ld_install_default_handlers`

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

5.4.7.5. Функция `ld_install_handler`

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 60.

Таблица 60. 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

5.4.7.6. Функция `ld_install_error_handler`

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 61.

Таблица 61. Параметры функции `ld_install_handler`

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

5.4.7.7. Функция `ld_exec`

Функция `ld_exec` позволяет начать основной цикл событий. Если цикл событий уже существует, например, внутри приложения Qt, эту функцию вызывать не нужно.

Синтаксис функции ld_exec:

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

Параметры функции ld_exec приведены в Таблице 62.

Таблица 62. Параметры функции ld_exec

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

5.4.7.8. Функция ld_exec_once

Функция ld_exec_once один раз перебирает список событий. Может заблокировать цикл событий (event loop).

Синтаксис функции ld_exec_once:

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

Параметры функции ld_exec_once приведены в Таблице 63.

Таблица 63. Параметры функции ld_exec_once

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

5.4.7.9. Функция ld_free

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 64.

Таблица 64. Параметры функции ld_free

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

5.4.8. Файл entry.h

Типы

```
using LDAPAttribute_t = struct LDAPAttribute_s
```

```
using ld_entry_t = struct ld_entry_s
```

Функции entry.h приведены в Таблице 65.

Таблица 65. Функции *entry.h*

Имя	Описание
add	Оборачивает функцию <code>ldap_add_ext</code> , связывая ее с подключением
add_on_read	Обратный вызов, вызываемый при завершении операции добавления <code>ldap</code>
search	Оборачивает операцию <code>ldap_search</code> , связывая ее с подключением
search_on_read	Обратный вызов, вызываемый после завершения операции поиска <code>ldap</code> .
modify	Оборачивает операцию <code>ldap_modify_ext</code> , связывая ее с подключением
modify_on_read	Обратный вызов, вызываемый после завершения операции изменения <code>ldap</code> .
delete	Функция удаления оборачивает <code>ldap_delete_ext</code> , связывая ее с подключением
delete_on_read	Обратный вызов, определяющий результат операции удаления.
id_rename	Оборачивает функцию <code>ldap_rename</code> , 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 <code>whoami</code> .
ld_entry_new	Создает новую запись (<code>ld_entry_t</code>)
ld_entry_get_dn	Возвращает DN записи
ld_entry_add_attribute	Добавляет атрибут к записи
ld_entry_get_attribute	Возвращает значение атрибута из записи
ld_entry_get_attributes	Возвращает значения всех атрибутов записи
connection_remove_search_request	Удаляет поисковый запрос из соединения по индексу

5.4.8.1. Функция `add`

Функция `add` оборачивает (декорирует) функцию `ldap_add_ext`, связывая ее с подключением. Функция `ldap_add_ext` иницирует 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 66.

Таблица 66. 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.

5.4.8.2. Функция add_on_read

Этот обратный вызов выполняется после завершения операции добавления ldap.

Синтаксис функции add_on_read:

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

Параметры функции add_on_read приведены в Таблице 67.

Таблица 67. 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.

5.4.8.3. Функция search

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, search_callback_fn search_callback)
```

Параметры функции search приведены в Таблице 68.

Таблица 68. 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.
search_callback	search_callback_fn	Функция которая будет вызвана для обработки результатов поиска

Возвращаемое значение:

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

5.4.8.4. Функция search_on_read

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 69.

Таблица 69. *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.

5.4.8.5. Функция modify

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, LDAPMod** attrs)
```

Параметры функции modify приведены в Таблице 70.

Таблица 70. 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]	LDAPMod	Массив изменений, завершающихся значением NULL, указывающий, какие изменения следует внести в запись.

Возвращаемое значение:

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

5.4.8.6. Функция modify_on_read

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 71.

Таблица 71. 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.

5.4.8.7. Функция delete

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 72.

Таблица 72. Параметры функции delete

Имя	Тип	Описание
connection [in]	ldap_connection_ctx_t	Используемое соединение
dn [in]	char	Имя записи (DN), 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.

5.4.8.8. Функция delete_on_read

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 73.

Таблица 73. 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.

5.4.8.9. Функция ld_rename

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 74.

Таблица 74. 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.

5.4.8.10. Функция `rename_on_read`

Этот обратный вызов выполняется после завершения операции переименования `ldap`.

Синтаксис функции `rename_on_read`:

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

Параметры функции `rename_on_read` приведены в Таблице 75.

Таблица 75. Параметры функции `rename_on_read`

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

Возвращаемое значение:

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

5.4.8.11. Функция `whoami`

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 76.

Таблица 76. Параметры функции `whoami`

Имя	Тип	Описание
<code>connection</code> [in]	<code>ldap_connection_ctx_t</code>	Используемое соединение

5.4.8.12. Функция `whoami_on_read`

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 77.

Таблица 77. *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.

5.4.8.13. Функция `ld_entry_new`

Функция `ld_entry_new` создает новую запись (`ld_entry_t`).

Синтаксис функции `ld_entry_new`:

```
ld_entry_t* ld_entry_new(TALLOC_CTX* ctx, const char* dn)
```

Параметры функции `ld_entry_new` приведены в Таблице 78.

Таблица 78. *Параметры функции ld_entry_new*

Имя	Тип	Описание
ctx [in]	TALLOC_CTX	Контекст библиотеки talloc
dn [in]	char	Относительное различающееся имя для записи

Возвращаемое значение:

- указатель на `ld_entry_t`, если функция завершается успешно;
- `NULL`, если функция выполняется неудачно.

5.4.8.14. Функция `ld_entry_get_dn`

Функция `ld_entry_get_dn` возвращает DN записи.

Синтаксис функции `ld_entry_get_dn`:

```
const char* ld_entry_get_dn(ld_entry_t* entry)
```

Параметры функции `ld_entry_get_dn` приведены в Таблице 79.

Таблица 79. *Параметры функции ld_entry_get_dn*

Имя	Тип	Описание
entry [in]	ld_entry_t	Запись

Возвращаемое значение:

- DN, если функция завершается успешно;
- NULL, если функция выполняется неудачно.

5.4.8.15. Функция ld_entry_add_attribute

Функция ld_entry_add_attribute добавляет новый атрибут к записи.

Синтаксис функции ld_entry_add_attribute:

```
enum OperationReturnCode ld_entry_add_attribute(ld_entry_t* entry,
const LDAPAttribute_t* attr)
```

Параметры функции ld_entry_add_attribute приведены в Таблице 80.

Таблица 80. *Параметры функции ld_entry_add_attribute*

Имя	Тип	Описание
entry [in]	ld_entry_t	Запись
attr [in]	LDAPAttribute_t	Добавляемый атрибут

Возвращаемое значение:

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

5.4.8.16. Функция ld_entry_get_attribute

Функция ld_entry_get_attribute возвращает атрибут записи.

Синтаксис функции ld_entry_get_attribute:

```
LDAPAttribute_t* ld_entry_get_attribute(ld_entry_t* entry, const char*
name_or_oid)
```

Параметры функции ld_entry_get_attribute приведены в Таблице 81.

Таблица 81. *Параметры функции ld_entry_get_attribute*

Имя	Тип	Описание
entry [in]	ld_entry_t	Запись
name_or_oid[in]	char	Имя атрибута

Возвращаемое значение:

- Указатель на LDAPAttribute_t, если атрибут найден;
- NULL, если атрибут не найден.

5.4.8.17. Функция ld_entry_get_attributes

Функция ld_entry_get_attributes возвращает все атрибуты записи.

Синтаксис функции ld_entry_get_attributes:

```
LDAPAttribute_t** ld_entry_get_attributes(ld_entry_t* entry)
```

Параметры функции ld_entry_get_attributes приведены в Таблице 82.

Таблица 82. *Параметры функции ld_entry_add_attribute*

Имя	Тип	Описание
entry [in]	ld_entry_t	Запись

Возвращаемое значение:

- список атрибутов, завершающийся NULL, если функция завершается успешно;
- NULL в случае ошибки.

Примечание. По завершении работы с атрибутами необходимо вызвать Talloc_free().

5.4.9. Файл group.h

Перечисления

```
enum GroupScope
{
    GROUP_SCOPE_DOMAIN_LOCAL    = 0,
    GROUP_SCOPE_GLOBAL           = 1,
    GROUP_SCOPE_UNIVERSAL       = 2
};
```

```
enum GroupCategory
{
    GROUP_CATEGORY_DISTRIBUTION    = 0,
    GROUP_CATEGORY_SECURITY        = 1
};
```

Возможные области действия группы (GroupScope) приведены в Таблице 83.

Возможные категории группы (GroupCategory) приведены в Таблице 84.

Таблица 83. Значения GroupScope

Значение	Описание
GROUP_SCOPE_DOMAIN_LOCAL	Домен локальная группа
GROUP_SCOPE_GLOBAL	Глобальная группа
GROUP_SCOPE_UNIVERSAL	Универсальная

Таблица 84. Значения GroupCategory

Значение	Описание
GROUP_CATEGORY_DISTRIBUTION	Рассылка
GROUP_CATEGORY_SECURITY	Безопасность

Функции group.h приведены в Таблице 85.

Таблица 85. Функции group.h

Имя	Описание
ld_add_group	Добавляет группу
ld_del_group	Удаляет группу
ld_mod_group	Изменяет группу
ld_rename_group	Переименовывает группу
ld_group_add_user	Добавляет пользователя в группу
ld_group_remove_user	Удаляет пользователя из группы

5.4.9.1. Функция ld_add_group

Функция ld_add_group добавляет группу.

Синтаксис функции ld_add_group:

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

Параметры функции ld_add_group приведены в Таблице 86.

Таблица 86. Параметры функции ld_add_group

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

Возвращаемое значение:

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

5.4.9.2. Функция ld_del_group

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 87.

Таблица 87. 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.

5.4.9.3. Функция ld_mod_group

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 88.

Таблица 88. 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.

5.4.9.4. Функция ld_rename_group

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 89.

Таблица 89. Параметры функции ld_rename_group

Имя	Тип	Описание
handle [in]	LDHandle	Указатель на дескриптор сеанса libdomain
old_name [in]	char	Текущее 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.

5.4.9.5. Функция ld_group_add_user

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 90.

Таблица 90. 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.

5.4.9.6. Функция ld_group_remove_user

Функция ld_group_remove_user удаляет пользователя из группы.

Синтаксис функции ld_group_remove_user:

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

Параметры функции ld_group_remove_user приведены в Таблице 91.

Таблица 91. 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.

5.4.10. Файл ldap_syntaxes.h

Функции ldap_syntaxes.h приведены в Таблице 92.

Таблица 92. Функции ldap_syntaxes.h

Имя	Описание
validate_boolean	Проверяет, является ли значение логическим типом
validate_integer	Проверяет, является ли значение целым числом
validate_octet_string	Проверяет, является ли значение строкой октетов
validate_oid	Проверяет, является ли значение строкой OID
validate_numeric_string	Проверяет, является ли значение числовой строкой (числом в строковом формате)
validate_printable_string	Проверяет, является ли значение строкой, содержащая символы из набора символов для печати
validate_case_ignore_string	Проверяет, является ли значение строкой строкой Unicode (UTF-8) без учёта регистра символов
validate_ia5_string	Проверяет, является ли значение строкой IA5
validate_utc_time	Проверяет, является ли значение временем в UTC формате
validate_generalized_time	Проверяет, является ли значение временем LDAP в формате GeneralizedTime
validate_case_sensitive_string	Проверяет, является ли значение строкой строкой Unicode (UTF-8) с учётом регистра символов
validate_directory_string	Проверяет, является ли значение строкой Unicode (UTF-8)
validate_large_integer	Проверяет, является ли значение большим целым числом
validate_object_security_descriptor	Строка октета, содержащая идентификатор безопасности (SID)
validate_dn	Проверяет, является ли значение уникальным именем (DN)
validate_dn_with_octet_string	Проверяет, является ли значение строкой UTF-8 в формате: B:char_count:binary_value:object_DN
validate_dn_with_string	Проверяет, является ли значение строкой UTF-8 в формате: S:byte_count:string_value:object_DN
validate_or_name	Проверяет, является ли значение строкой UTF-8 в формате: object_DN

validate_presentation_address	Не реализовано в текущей редакции. Синтаксис определен в RFC 1278
validate_access_point	Не реализовано в текущей редакции. Синтаксическая проверка, которая проверяет, является ли значение точкой доступа

5.4.10.1. Функция validate_boolean

Функция validate_boolean принимает строку и проверяет, соответствует ли она формату логического значения, определенному в RFC 4517. Логический тип LDAP имеет OID=1.3.6.1.4.1.1466.115.121.1.7.

Синтаксис функций:

```
bool validate_boolean(const char* value);
```

Параметры функции validate_boolean приведены в Таблице 93.

Таблица 93. Параметры функции validate_boolean

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является выражением логического типа;
- FALSE, если выражение не является выражением логического типа.

5.4.10.2. Функция validate_integer

Функция validate_integer принимает строку и проверяет, соответствует ли она формату целочисленного значения, как определено в RFC 4517.

Целое число имеет OID=1.3.6.1.4.1.1466.115.121.1.27. Целое число — это число в диапазоне от $2^{31}-1$ (2147483647) до -2^{31} (2147483648).

Синтаксис функции:

```
bool validate_integer(const char* value);
```

Параметры функции validate_integer приведены в Таблице 94.

Таблица 94. Параметры функции validate_integer

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если строка является целым числом;
- FALSE, если строка не является целым числом.

5.4.10.3. Функция `validate_octet_string`

Функция `validate_octet_string` принимает строку и проверяет, соответствует ли она формату значения строки октетов, определенному в RFC 4517. (OID=1.3.6.1.4.1.1466.115.121.1.40). Этот синтаксис используется для хранения двоичных данных. Из RFC 4517: OctetString = *ОКТЕТ ОКТЕТ = x00-FF; Любой октет (8-битная единица данных)

Синтаксис функции:

```
bool validate_octet_string(const char* value);
```

Параметры функции `validate_octet_string` приведены в Таблице 95.

Таблица 95. Параметры функции `validate_octet_string`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка представляет собой строковое значение октета;
- FALSE, если входная строка не является строковым значением октета .

5.4.10.4. Функция `validate_oid`

Функция `validate_oid` принимает строку и проверяет, соответствует ли она формату значения OID, как определено в RFC 4517 (OID=1.3.6.1.4.1.1466.115.121.1.38). Строка OID представляет собой строку, содержащую цифры (0–9) и десятичные точки (.), например:

```
1.3.6.1.4.1.1466.109.114.2  
2.5.13.5
```

Синтаксис функции:

```
bool validate_oid(const char* value);
```

Параметры функции `validate_oid` приведены в Таблице 96.

Таблица 96. Параметры функции *validate_oid*

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой OID;
- FALSE, если входная строка не является строкой OID.

5.4.10.5. Функция *validate_numeric_string*

Функция *validate_numeric_string* проверяет, является ли значение значение числовой строкой (OID=1.3.6.1.4.1.1466.115.121.1.36). Это строковый тип с ограниченным набором символов (0-9) и пробел, например:

15 079 672 281
199412160532

Синтаксис функции:

```
bool validate_numeric_string(const char* value);
```

Параметры функции *validate_numeric_string* приведены в Таблице 97.

Таблица 97. Параметры функции *validate_numeric_string*

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является числовой строкой;
- FALSE, если входная строка не является числовой строкой.

5.4.10.6. Функция *validate_printable_string*

Функция *validate_printable_string* проверяет, является ли значение строкой с учётом регистра, содержащей символы из набора символов для печати (OID=1.3.6.1.4.1.1466.115.121.1.44). Это строковый тип с ограниченным набором символов. a-z, A-Z, 0-9, '()+,-.= /:?' и пробел.

Синтаксис функции:

```
bool validate_printable_string(const char* value);
```

Параметры функции *validate_printable_string* приведены в Таблице 98.

Таблица 98. Параметры функции *validate_printable_string*

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой, содержащей символы из набора символов для печати;
- FALSE, если входная строка не является строкой, содержащей символы из набора символов для печати.

5.4.10.7. Функция *validate_case_ignore_string*

Функция *validate_case_ignore_string* проверяет, является ли значение строкой строкой Unicode (UTF-8) без учёта регистра символов. Тоже что и *validate_directory_string*. Значение с таким синтаксисом является строкой UTF-8 без учёта регистра, но сервер не требует, чтобы значение этого синтаксиса было допустимой строкой UTF-8.

Синтаксис функции:

```
bool validate_case_ignore_string(const char* value);
```

Параметры функции *validate_case_ignore_string* приведены в Таблице 99.

Таблица 99. Параметры функции *validate_case_ignore_string*

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой Unicode (UTF-8) без учёта регистра символов;
- FALSE, если входная строка не является строкой Unicode (UTF-8) без учёта регистра символов.

5.4.10.8. Функция *validate_ia5_string*

Функция *validate_ia5_string* проверяет, является ли значение строкой IA5 (OID=1.3.6.1.4.1.1466.115.121.1.26). Строка IA5 – это строковый тип с ограниченным набором символов: a-z, A-Z, 0-9, '()+,=-./?: и пробел.

Синтаксис функции:

```
bool validate_ia5_string(const char* value);
```

Параметры функции `validate_ia5_string` приведены в Таблице 100.

Таблица 100. Параметры функции `validate_ia5_string`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой, содержащей символы из набора символов IA5;
- FALSE, если входная строка не является строкой, содержащей символы из набора символов IA5.

5.4.10.9. Функция `validate_utc_time`

Функция `validate_utc_time` проверяет, является ли значение временем в UTC формате. UTC-Time – это строковый формат времени, определенный в стандартах ASN.1. Формат синтаксиса мирового времени (Universal Time):

ГГММДДЧЧММ[сс][(+ | -)ЧЧММ)|Z]

Этот формат включает по 2 цифры для обозначения года, месяца, дня, часов и минут, а также необязательное обозначение долей секунды. Можно указать смещение относительно мирового времени, например:

120412123000Z

120412123000+1230

Синтаксис функции:

```
bool validate_utc_time(const char* value);
```

Параметры функции `validate_utc_time` приведены в Таблице 101.

Таблица 101. Параметры функции `validate_utc_time`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой, содержащей время в UTC-формате;
- FALSE, если входная строка не является строкой, содержащей время в UTC-формате .

5.4.10.10. Функция `validate_generalized_time`

Функция `validate_generalized_time` проверяет, является ли значение временем LDAP в формате `GeneralizedTime` (OID=1.3.6.1.4.1.1466.115.121.1.24). Формат синтаксиса `Generalized-Time`:

`ГГГГММДДЧЧММСС.0Z`

Значение «Z» указывает на отсутствие разностного времени. Если время указано в часовом поясе, отличном от GMT, разница между часовыми поясами и GMT добавляется к строке вместо «Z» в формате

`ГГГГММДДННММСС.0 [+ / -] ННММ`

Примеры допустимых значений:

`19941216103200Z`

`199412160532-0500`

Синтаксис функции:

```
bool validate_generalized_time(const char* value);
```

Параметры функции `validate_generalized_time` приведены в Таблице 102.

Таблица 102. Параметры функции `validate_generalized_time`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой, содержащей время в формате `GeneralizedTime`;
- FALSE, если входная строка не является строкой, содержащей время в формате `GeneralizedTime`.

5.4.10.11. Функция `validate_case_sensitive_string`

Функция `validate_case_sensitive_string` проверяет, является ли значение строкой строкой Unicode (UTF-8) с учётом регистра символов. То же что и `validate_directory_string`. Значение с таким синтаксисом является строкой UTF-8 с учётом регистра, но сервер не требует, чтобы значение этого синтаксиса было допустимой строкой UTF-8.

Синтаксис функции:

```
bool validate_case_sensitive_string(const char* value);
```

Параметры функции `validate_case_sensitive_string` приведены в Таблице 103.

Таблица 103. Параметры функции `validate_case_sensitive_string`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если входная строка является строкой Unicode (UTF-8) с учётом регистра символов;
- FALSE, если входная строка не является строкой Unicode (UTF-8) с учётом регистра символов.

5.4.10.12. Функция `validate_directory_string`

Функция `validate_directory_string` проверяет, является ли значение строкой Unicode (OID=1.3.6.1.4.1.1466.115.121.1.15). (Unicode) кодировка UTF-8 – кодировочная система с нефиксированным количеством бит на символ. Включает в себя IA5/ASCII в качестве подмножества, поддерживает расширенные символы.

Синтаксис функции:

```
bool validate_directory_string(const char* value);
```

Параметры функции `validate_directory_string` приведены в Таблице 104.

Таблица 104. Параметры функции `validate_directory_string`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является строкой Unicode (UTF-8);
- FALSE, если выражение не является строкой Unicode (UTF-8).

5.4.10.13. Функция `validate_large_integer`

Функция `validate_large_integer` принимает строку и проверяет, соответствует ли она формату большого целочисленного значения, как определено в RFC 4517 (64-разрядным целым числом со знаком). Это число в диапазоне от $2^{63}-1$ (9223372036854775807) до -2^{63} (-9223372036854775808).

Синтаксис функции:

```
bool validate_large_integer(const char* value);
```

Параметры функции `validate_large_integer` приведены в Таблице 105.

Таблица 105. Параметры функции `validate_large_integer`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является большим целым числом;
- FALSE, если выражение не является большим целым числом.

5.4.10.14. Функция `validate_object_security_descriptor`

Функция `validate_object_security_descriptor` проверяет, является ли значение строкой октета, содержащей идентификатор безопасности (SID).

Синтаксис функции:

```
bool validate_object_security_descriptor(const char* value);
```

Параметры функции `validate_object_security_descriptor` приведены в Таблице 106.

Таблица 106. Параметры функции `validate_object_security_descriptor`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является строкой октета, содержащей идентификатор безопасности ;

- FALSE, если выражение не является строкой октета, содержащей идентификатор безопасности.

5.4.10.15. Функция validate_dn

Функция validate_dn проверяет, является ли значение уникальным именем (DN, Distinguished Name) (OID=1.3.6.1.4.1.1466.115.121.1.12). DN указывается как строка, состоящая из последовательности пар атрибут/значение атрибута, разделенных запятой:

<атрибут>=<значение> [, <атрибут>=<значение>] *

Должен быть указан хотя бы один атрибут. Пары атрибутов могут повторяться. <атрибут> — это либо поддерживаемое короткое имя, либо десятичная кодировка идентификатора объекта ASN.1, например:

UID=jsmith,DC=example,DC=net

1.3.6.1.4.1.1466.0=#04024869,DC=example,DC=com

Синтаксис функции:

```
bool validate_dn(const char* value);
```

Параметры функции validate_dn приведены в Таблице 107.

Таблица 107. Параметры функции validate_dn

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является строкой, содержащей уникальное имя (DN);
- FALSE, если выражение не является строкой, содержащей уникальное имя (DN).

5.4.10.16. Функция validate_dn_with_octet_string

Функция validate_dn_with_octet_string проверяет, является ли значение строкой UTF-8 в следующем формате:

B:char_count:binary_value:object_DN

где:

- char_count – это число (в десятичной системе счисления) шестнадцатеричных цифр в binary_value;

- `binary_value` – шестнадцатеричное представление двоичного значения;
- `object_DN` – это DN в форме объекта (DS-DN)
- все остальные символы — строковые литералы.

Каждый байт представлен парой шестнадцатеричных символов в `binary_value`, причем первый символ каждой пары соответствует наиболее значимому фрагменту байта. Первая пара в `binary_value` соответствует первому байту двоичного значения, с последующими парами, соответствующими оставшимся байтам в последовательном порядке. `char_count` всегда является четным в синтаксически допустимом объектном (DN-двоичном) значении.

Синтаксис функции:

```
bool validate_dn_with_octet_string(const char* value);
```

Параметры функции `validate_dn_with_octet_string` приведены в Таблице 108.

Таблица 108. Параметры функции `validate_dn_with_octet_string`

Имя	Тип	Описание
<code>value [in]</code>	<code>char</code>	Массив символов, завершающийся NULL

Возвращаемое значение:

- `TRUE`, если выражение является строкой UTF-8 в нужном формате;
- `FALSE`, если выражение не является строкой UTF-8 в нужном формате.

5.4.10.17. Функция `validate_dn_with_string`

Функция `validate_dn_with_string` проверяет, является ли значение строкой UTF-8 в следующем формате:

```
S:byte_count:string_value:object_DN
```

где:

- `byte_count` – это число (в десятичной системе счисления) байтов в строке `string_value`;
- `string_value` – строка в формате UTF-8;
- `object_DN` – это DN в форме объекта (DS-DN);
- все остальные символы – строковые литералы.

Поскольку `string_value` это строка в формате UTF-8, для представления одного символа может потребоваться более одного байта.

Синтаксис функции:

```
bool validate_dn_with_string(const char* value);
```

Параметры функции `validate_dn_with_octet_string` приведены в Таблице 109.

Таблица 109. Параметры функции `validate_dn_with_string`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является строкой UTF-8 в нужном формате;
- FALSE, если выражение не является строкой UTF-8 в нужном формате.

5.4.10.18. Функция `validate_or_name`

Функция `validate_or_name` проверяет, является ли значение строкой UTF-8 в следующем формате (OID (2.6.6.1.2.5.11.29)):

`object_DN`

где `object_DN` – это DN в форме объекта (DS-DN).

Синтаксис функции:

```
bool validate_or_name(const char* value);
```

Параметры функции `validate_or_name` приведены в Таблице 110.

Таблица 110. Параметры функции `validate_or_name`

Имя	Тип	Описание
value [in]	char	Массив символов, завершающийся NULL

Возвращаемое значение:

- TRUE, если выражение является строкой UTF-8 в формате `object_DN`;
- FALSE, если выражение не является строкой UTF-8 в формате `object_DN`.

5.4.11. Файл organization_unit.h

Функции organization_unit.h приведены в Таблице 111.

Таблица 111. Функции organization_unit.h

Имя	Описание
ld_add_ou	Создает подразделение
ld_del_ou	Удаляет подразделение
ld_mod_ou	Изменяет подразделение
ld_rename_ou	Переименовывает подразделение

5.4.11.1. Функция ld_add_ou

Функция ld_add_ou создает подразделение.

Синтаксис функции ld_add_ou:

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

Параметры функции ld_add_ou приведены в Таблице 112.

Таблица 112. Параметры функции ld_add_ou

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

Возвращаемое значение:

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

5.4.11.2. Функция ld_del_ou

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 113.

Таблица 113. Параметры функции `ld_del_ou`

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

Возвращаемое значение:

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

5.4.11.3. Функция `ld_mod_ou`

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 114.

Таблица 114. Параметры функции `ld_mod_ou`

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

Возвращаемое значение:

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

5.4.11.4. Функция `ld_rename_ou`

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 115.

Таблица 115. Параметры функции `ld_rename_ou`

Имя	Тип	Описание
<code>handle [in]</code>	<code>LDHandle</code>	Указатель на дескриптор сеанса <code>libdomain</code>
<code>old_name [in]</code>	<code>char</code>	Текущее название подразделения
<code>new_name [in]</code>	<code>char</code>	Новое название подразделения
<code>parent [in]</code>	<code>char</code>	Родительский элемент для подразделения

Возвращаемое значение:

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

5.4.12. Файл `request_queue.h`

Примечание. Этот функционал нужен при модификации объектов LDAP, например, при добавлении и модификации записей и атрибутов и следит за тем чтобы для сообщений вызывались корректные обработчики сообщений.

Классы

```
struct Queue_Node_s
```

Перечисления

```
enum RequestQueueErrorCode
```

```
{  
    OPERATION_SUCCESS                = 0,  
    OPERATION_ERROR_FULL              = 1,  
    OPERATION_ERROR_INVALID_PARAMETER = 2,  
    OPERATION_ERROR_FAULT             = 3,  
}
```

Значения состояния соединения(`LdapConnectionState`) приведены в Таблице 116.

Таблица 116. Значения `LdapConnectionState`

Значение	Описание
<code>OPERATION_SUCCESS</code>	Операция с очередью прошла успешно
<code>OPERATION_ERROR_FULL</code>	Не удалось добавить узел в очередь, т. к.

	базовое хранилище заполнено
OPERATION_ERROR_INVALID_PARAMETER	В функцию передан недопустимый параметр
OPERATION_ERROR_FAULT	Внутренняя логическая ошибка очереди

Типы

```
using request_queue = struct request_queue
```

Функции `request_queue.h` приведены в Таблице 117.

Таблица 117. Функции `request_queue.h`

Имя	Описание
<code>request_queue_new</code>	Создает новый <code>request_queue</code>
<code>request_queue_push</code>	Добавляет узел в начало очереди
<code>request_queue_pop</code>	Получает узел из начала очереди и удаляет его
<code>request_queue_peek</code>	Получает указатель на начало очереди
<code>request_queue_empty</code>	Возвращает <code>true</code> , если очередь пуста.
<code>request_queue_move</code>	Перемещает узел из одной очереди в другую

5.4.12.1. Функция `request_queue_new`

Функция `request_queue_new` создает новый `request_queue`.

Синтаксис функции `request_queue_new`:

```
request_queue* request_queue_new(TALLOC_CTX* ctx, unsigned int
capacity);
```

Параметры функции `request_queue_new` приведены в Таблице 118.

Таблица 118. Параметры функции `request_queue_new`

Имя	Тип	Описание
<code>ctx [in]</code>	<code>TALLOC_CTX</code>	Контекст памяти, с которым нужно работать.
<code>capacity [in]</code>	<code>int</code>	Максимальный размер очереди

Возвращаемое значение:

- указатель на очередь, если функция завершается успешно;
- `NULL`, если функция выполняется неудачно.

5.4.12.2. Функция request_queue_push

Функция request_queue_push добавляет узел в начало очереди.

Синтаксис функции request_queue_push:

```
enum RequestQueueErrorCode request_queue_push(request_queue* queue,  
struct Queue_Node_s *node);
```

Параметры функции request_queue_push приведены в Таблице 119.

Таблица 119. Параметры функции request_queue_push

Имя	Тип	Описание
queue [in]	request_queue	Текущая очередь, в которую требуется добавить узел.
node [in]	Queue_Node_s	Узел

Возвращаемое значение:

- OPERATION_SUCCESS, если функция завершается успешно;
- OPERATION_ERROR_FULL, если функция выполняется неудачно.

5.4.12.3. Функция request_queue_pop

Функция request_queue_pop получает узел из начала очереди и удаляет его.

Синтаксис функции request_queue_pop:

```
struct Queue_Node_s* request_queue_pop(request_queue* queue);
```

Параметры функции request_queue_pop приведены в Таблице 120.

Таблица 120. Параметры функции request_queue_pop

Имя	Тип	Описание
queue [in]	request_queue	Текущая очередь, из которой требуется получить элемент.

Возвращаемое значение:

- указатель на элемент, если функция завершается успешно;
- NULL при переполнении.

5.4.12.4. Функция request_queue_peek

Функция request_queue_peek получает указатель на начало очереди.

Синтаксис функции request_queue_peek:

```
struct Queue_Node_s* request_queue_peek(request_queue* queue);
```

Параметры функции `request_queue_peek` приведены в Таблице 121.

Таблица 121. Параметры функции `request_queue_peek`

Имя	Тип	Описание
<code>queue [in]</code>	<code>request_queue</code>	Текущая очередь, из которой требуется получить элемент.

Возвращаемое значение:

- указатель на элемент, если функция завершается успешно;
- `NULL` при переполнении.

5.4.12.5. Функция `request_queue_empty`

Функция `request_queue_empty` определяет пуста ли очередь.

Синтаксис функции `request_queue_empty`:

```
bool request_queue_empty(request_queue* queue);
```

Параметры функции `request_queue_empty` приведены в Таблице 122.

Таблица 122. Параметры функции `request_queue_empty`

Имя	Тип	Описание
<code>queue [in]</code>	<code>request_queue</code>	Текущая очередь

Возвращаемое значение:

- `TRUE`, если очередь пуста;
- `FALSE`, если в очереди есть элементы.

5.4.12.6. Функция `request_queue_move`

Функция `request_queue_move` перемещает узел из одной очереди в другую.

Синтаксис функции `request_queue_move`:

```
enum RequestQueueErrorCode request_queue_move(request_queue* from,  
request_queue* to);
```

Параметры функции `request_queue_move` приведены в Таблице 123.

Таблица 123. Параметры функции `request_queue_move`

Имя	Тип	Описание
<code>from [in]</code>	<code>request_queue</code>	Текущая очередь, из которой требуется переместить узел

to [in]	request_queue	Очередь, в которую требуется переместить узел
---------	---------------	---

Возвращаемое значение:

- OPERATION_SUCCESS, если функция завершается успешно;
- OPERATION_ERROR_FULL, если функция выполняется неудачно.

5.4.13. Файл schema.h

Типы

```
using ldap_schema_t = struct ldap_schema_t
```

Функции schema.h приведены в Таблице 124.

Таблица 124. Функции schema.h

Имя	Описание
ldap_schema_new	Выделяет ldap_schema_t и проверяет его достоверность
ldap_schema_object_classes	Возвращает список структур LDAPObjectClass
ldap_schema_append_attributetype	Добавляет в список типов атрибутов схемы новый тип атрибута
ldap_schema_attribute_types	Возвращает список структур LDAPAttributeType
ldap_schema_append_objectclass	Добавляет в список классов объектов схемы новый объектный класс

5.4.13.1. Функция ldap_schema_new

Функция ldap_schema_new выделяет ldap_schema_t и проверяет его достоверность.

Синтаксис функции ldap_schema_new:

```
ldap_schema_t* ldap_schema_new(TALLOC_CTX* ctx);
```

Параметры функции ldap_schema_new приведены в Таблице 125.

Таблица 125. Параметры функции ldap_schema_new

Имя	Тип	Описание
ctx [in]	TALLOC_CTX	Контекст памяти, с которым нужно работать.

Возвращаемое значение:

- указатель на очередь, если функция завершается успешно;
- NULL, если функция выполняется неудачно.

5.4.13.2. Функция ldap_schema_object_classes

Функция `ldap_schema_object_classes` возвращает список структур `LDAPObjectClass`.

Синтаксис функции `ldap_schema_object_classes`:

```
LDAPObjectClass** ldap_schema_object_classes(const ldap_schema_t*  
schema);
```

Параметры функции `ldap_schema_object_classes` приведены в Таблице 126.

Таблица 126. Параметры функции `ldap_schema_object_classes`

Имя	Тип	Описание
schema [in]	ldap_schema_t	Схема, с которой нужно работать

Возвращаемое значение:

- список классов объектов из схемы;
- NULL, если схема имеет значение NULL.

5.4.13.3. Функция ldap_schema_append_attributetype (schema.h)

Функция `ldap_schema_append_attributetype` добавляет в список типов атрибутов схемы новый тип атрибута.

Синтаксис функции `ldap_schema_append_attributetype`:

```
bool ldap_schema_append_attributetype(ldap_schema_t* schema,  
LDAPAttributeType* attributetype);
```

Параметры функции `ldap_schema_append_attributetype` приведены в Таблице 127.

Таблица 127. Параметры функции `ldap_schema_append_attributetype`

Имя	Тип	Описание
schema [in]	ldap_schema_t	Схема, с которой нужно работать.
attributetype	LDAPAttributeType	Тип атрибута который нужно добавить в схему

Возвращаемое значение:

- TRUE, если функция завершается успешно;
- FALSE, если функция выполняется неудачно.

5.4.13.4. Функция ldap_schema_attribute_types

Функция `ldap_schema_attribute_types` возвращает список структур `LDAPAttributeType`.

Синтаксис функции `ldap_schema_attribute_types`:

```
LDAPAttributeType** ldap_schema_attribute_types(const ldap_schema_t  
*schema);
```

Параметры функции `ldap_schema_attribute_types` приведены в Таблице 128.

Таблица 128. Параметры функции `ldap_schema_attribute_types`

Имя	Тип	Описание
schema [in]	ldap_schema_t	Схема, с которой нужно работать.

Возвращаемое значение:

- список типов атрибутов из схемы;
- NULL, если схема имеет значение NULL.

5.4.13.5. Функция ldap_schema_append_objectclass

Функция `ldap_schema_append_objectclass` добавляет в список классов объектов схемы новый объектный класс.

Синтаксис функции `ldap_schema_append_objectclass`:

```
bool ldap_schema_append_objectclass(ldap_schema_t* schema,  
LDAPObjectClass* objectclass);
```

Параметры функции `ldap_schema_append_objectclass` приведены в Таблице 129.

Таблица 129. Параметры функции `ldap_schema_append_objectclass`

Имя	Тип	Описание
schema [in]	ldap_schema_t	Схема, с которой нужно работать
objectclass	LDAPObjectClass	Класс объекта который нужно добавить в схему

Возвращаемое значение:

- TRUE, если функция завершается успешно;
- FALSE, если функция выполняется неудачно.

5.4.14. Файл user.h

Функции user.h приведены в Таблице 130.

Таблица 130. Функции user.h

Имя	Описание
ld_add_user	Создает пользователя
ld_del_user	Удаляет пользователя
ld_mod_user	Изменяет пользователя
ld_rename_user	Переименовывает пользователя
ld_block_user	Блокирует пользователя
ld_unblock_user	Разблокирует пользователя

5.4.14.1. Функция ld_add_user

Функция ld_add_user создает пользователя.

Синтаксис функции ld_add_user:

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

Параметры функции ld_add_user приведены в Таблице 131.

Таблица 131. Параметры функции ld_add_user

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

5.4.14.2. Функция ld_del_user

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 132.

Таблица 132. 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.

5.4.14.3. Функция `ld_mod_user`

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 133.

Таблица 133. 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.

5.4.14.4. Функция `ld_rename_user`

Функция `ld_rename_user` переименовывает пользователя.

Синтаксис функции `ld_rename_user`:


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

Параметры функции `ld_rename_user` приведены в Таблице 134.

Таблица 134. 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.

5.4.14.5. Функция `ld_block_user`

Функция `ld_block_user` блокирует пользователя.

Синтаксис функции `ld_block_user`:

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

Параметры функции `ld_block_user` приведены в Таблице 135.

Таблица 135. Параметры функции `ld_block_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.

5.4.14.6. Функция `ld_unblock_user`

Функция `ld_unblock_user` разблокирует пользователя.

Синтаксис функции `ld_unblock_user`:

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

Параметры функции `ld_unblock_user` приведены в Таблице 136.

Таблица 136. Параметры функции `ld_unblock_user`

Имя	Тип	Описание
<code>handle [in]</code>	<code>LDHandle</code>	Указатель на дескриптор сеанса <code>libdomain</code>
<code>name [in]</code>	<code>char</code>	Имя пользователя
<code>parent [in]</code>	<code>char</code>	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;
- 5 (`RETURN_CODE_REPEAT_LAST_OPERATION`) – необходимо повторить вызов функции ещё раз.

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 137 приведены примеры фильтров поиска LDAP.

Таблица 137. 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

8. Инструкция по разворачиванию стенда

Схема стенда представлена на рис. 9. Состав технических и программных средств стенда приведён в табл. 138.

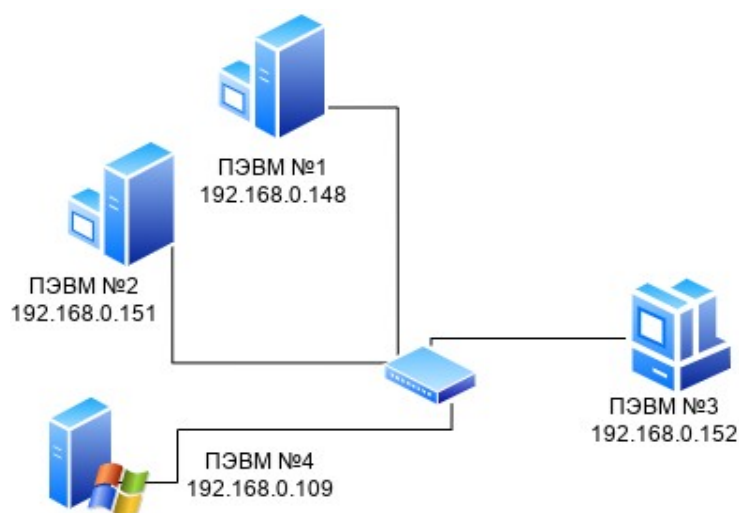


Рисунок 9. Схема стенда

Таблица 138. Состав технических и программных средств стенда

№ ПЭВМ	Программная среда	Описание
1	ОС Альт Сервер 10	Контроллер домена (DC)
2	ОС Альт Сервер 10	Сервер LDAP
3	ОС Альт (Рабочая станция 10, Рабочая станция К 10, Образование 10)	Машина с libdomain

№ ПЭВМ	Программная среда	Описание
5	ОС Microsoft Windows Server	Сервер Active Directory

Параметры домена:

- домен AD – domain.alt;
- сервер AD (ОС ALT) – dc0.domain.alt (192.168.0.148);
- сервер LDAP (ОС ALT) – dc1.domain.alt (192.168.0.151);
- рабочая станция 1 (ОС ALT) – host-01.domain.alt (192.168.0.152);
- рабочая станция 4 (ОС Windows) – PK1.domain.alt (192.168.0.109);
- имя пользователя-администратора – Administrator;
- пароль администратора – Pa\$\$word.

Примечание. На текущий момент библиотека поддерживает только протокол IPv4.

8.1. Контроллер домена (Samba AD DC)

8.1.1. Установка ОС «Альт Сервер» 10.x

Ссылка для скачивания ОС: <https://www.basealt.ru/alt-server/download>.

Инструкция по установке ОС:

<https://docs.altlinux.org/ru-RU/alt-server/10.1/html/alt-server/install-distro.html>

8.1.2. Разворачивание сервера Samba AD DC

Все действия выполняются на узле dc0.domain.alt (192.168.0.148).

Для установки Samba AD DC выполняются следующие шаги:

1. Установить пакеты task-samba-dc и libsasl2-plugin-gssapi (нужен для работы библиотеки libdomain):

```
# apt-get install task-samba-dc libsasl2-plugin-gssapi
```

2. Остановить конфликтующие службы krb5kdc и slapd, а также bind:

```
# for service in smb nmb krb5kdc slapd bind; do chkconfig $service  
off; service $service stop; done
```

3. Очистить базы и конфигурацию Samba:

```
# rm -f /etc/samba/smb.conf
# rm -rf /var/lib/samba
# rm -rf /var/cache/samba
# mkdir -p /var/lib/samba/sysvol
```

4. Установить имя домена. Имя домена, для разворачиваемого DC, должно состоять минимум из двух компонентов, разделённых точкой. При этом должно быть установлено правильное имя узла и домена для сервера. Для этого в файл `/etc/sysconfig/network` необходимо добавить строку:

```
HOSTNAME=dc1.domain.alt
```

И выполнить команды:

```
# hostnamectl set-hostname dc0.domain.alt
# domainname domain.alt
```

5. Для корректного функционирования домена в файле `/etc/resolv.conf` должна присутствовать строка:

```
nameserver 127.0.0.1
```

Если этой строки в файле `/etc/resolv.conf` нет, то в конец файла `/etc/resolvconf.conf` следует добавить строку:

```
name_servers='127.0.0.1'
```

и перезапустить сервис `resolvconf`:

```
# resolvconf -u
```

6. Создать домен `domain.alt` с паролем администратора `Pa$$word`:

```
# samba-tool domain provision --realm=domain.alt --domain=domain --
adminpass='Pa$$word' --dns-backend=SAMBA_INTERNAL --option="dns
forwarder=8.8.8.8" --server-role=dc
```

где

- `--realm` – область Kerberos (LDAP), и DNS имя домена;
- `--domain` – имя домена (имя рабочей группы);
- `--adminpass` – пароль основного администратора домена;
- `--server-role` – тип серверной роли.

Примечание. Пароль администратора должен быть не менее 7 символов и содержать символы как минимум трёх групп из четырёх возможных: латинских букв

в верхнем и нижнем регистрах, чисел и других небуквенно-цифровых символов. Пароль не полностью соответствующий требованиям это одна из причин завершения развертывания домена ошибкой.

7. Запустить службы:

```
# systemctl enable --now samba
```

8. Настроить Kerberos. В момент создания домена Samba конфигурирует шаблон файла krb5.conf для домена в каталоге /var/lib/samba/private/. Заменить этим файлом файл, находящийся в каталоге /etc/:

```
# cp /var/lib/samba/private/krb5.conf /etc/krb5.conf
```

9. Проверить работоспособность домена:

— просмотр общей информации о домене:

```
# samba-tool domain info 127.0.0.1
Forest                : domain.alt
Domain                : domain.alt
Netbios domain       : DOMAIN
DC name               : dc0.domain.alt
DC netbios name      : DC0
Server site           : Default-First-Site-Name
Client site           : Default-First-Site-Name
```

— убедиться в наличии nameserver 127.0.0.1 в /etc/resolv.conf:

```
# host domain.alt
domain.alt has address 192.168.0.148
```

— проверить имена хостов:

```
# host -t SRV _kerberos._udp.domain.alt.
_kerberos._udp.domain.alt has SRV record 0 0 88 dc0.domain.alt.
# host -t SRV _ldap._tcp.domain.alt.
_ldap._tcp.domain.alt has SRV record 0 100 389 dc0.domain.alt.
# host -t A dc0.domain.alt.
dc0.domain.alt has address 192.168.0.148
```

— проверка Kerberos (имя домена должно быть в верхнем регистре):

```
# kinit administrator@DOMAIN.ALT
```

Примечание. Если имена не находятся, необходимо проверить исключение службы named.

10. Создать и разблокировать пользователя ivanov в домене:

```
# samba-tool user create ivanov --given-name='Иван Иванов' --mail-address='ivanov@domain.alt'
# samba-tool user setexpiry ivanov --noexpiry
```

8.2. Настройка сервера LDAP

Все действия выполняются на узле (192.168.0.151).

Для установки и настройки сервера LDAP выполняются следующие шаги:

1. Установить пакеты openldap-servers openldap-clients:

```
# apt-get install openldap-servers openldap-clients
```

2. Создать скрипт генерации сертификатов generate_cert.sh со следующим содержанием:

```
#!/bin/bash
set -euxe pipefail

CERT_PATH="${1:-/var/lib/samba/private/tls}"

# Генерация ключа CA с использованием RSA, длина ключа 4096 бит
openssl genrsa -out "${CERT_PATH}/ca.key" 4096
# Генерация root сертификата, со сроком действия 1 год. Необходимо
указать CN, это должно быть полное доменное имя домена в верхнем
регистре.
openssl req -new -x509 -nodes -days 365 -key "${CERT_PATH}/ca.key" -
out "${CERT_PATH}/ca.pem" -subj "/O=Test Inc/OU=Samba CA
Cert/CN=domain.alt"
# Генерация ключа
openssl genrsa -out "${CERT_PATH}/dc0.domain.alt.key" 4096
# Запрос сертификата CSR
openssl req -new -sha256 -key "${CERT_PATH}/dc0.domain.alt.key" -subj
"/O=Test Inc/OU=Samba CA Cert/CN=dc0.domain.alt" -out
"${CERT_PATH}/dc0.domain.alt.csr"
```

```
# Подпись сертификата
openssl x509 -req -in "${CERT_PATH}/dc0.domain.alt.csr" -CA "${CERT_PATH}/ca.pem" -CAkey "${CERT_PATH}/ca.key" -CAcreateserial -out "${CERT_PATH}/dc0.domain.alt.pem" -days 365
# Проверка сертификата
openssl verify -CAfile "${CERT_PATH}/ca.pem" "${CERT_PATH}/dc0.domain.alt.pem"
```

3. Запустить скрипт генерации сертификатов:

```
# chmod +x generate_cert.sh
# mkdir -p /certs && ./generate_cert.sh /certs
```

4. Создать файл конфигурации OpenLDAP slapd.conf со следующим содержимым:

```
#
# See slapd.conf(5) for details on configuration options.
#

# [ GLOBAL SETTINGS ]

# Default schemas
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/openldap.schema
include /etc/openldap/schema/nis.schema
# Password policy
include /etc/openldap/schema/ppolicy.schema

# ALT Domain schemas
include /etc/openldap/schema/samba4.schema

# Set pid file
pidfile /tmp/slapd.pid

#
```



```

# Loading MDB database and Sync Provider
# See slapo-syncprov(5) and slapd.backends(5) for more details.
#
moduleload back_mdb.la
moduleload syncprov.la
moduleload ppolicy.la

database mdb
suffix "dc=domain,dc=alt"
rootdn "cn=admin,dc=domain,dc=alt"
rootpw password

overlay ppolicy
ppolicy_default "cn=default,ou=policies,dc=domain,dc=alt"
ppolicy_use_lockout

directory /tmp/ldap

# This option configures one or more hashes to be used in generation
of user passwords
# {CLEARTEXT} indicates that the new password should be added to
userPassword as clear text.
password-hash {CLEARTEXT}

# SASL Auth

#
# SASL Users authenticate against the following
# meta DNS in the LDAP tree:
#
# With a SASL Realm:
# uid=<username>,cn=<realm>,cn=<mechanism>,cn=auth
#
# Without a SASL Realm:
# uid=<username>,cn=<mechanism>,cn=auth

```

```
#
# Map the meta DN to a real dn using authz-regexp.
#
# See slapauth(8) for more details on SASL authentication.
#
```

```
authz-regexp
```

```
uid=admin,cn=[^,]*,cn=auth
cn=admin,dc=domain,dc=alt
```

```
authz-regexp
```

```
uid=([^,]*) ,cn=[^,]*,cn=auth
uid=$1,ou=people,dc=domain,dc=alt
```

```
TLSCACertificateFile /certs/ca.pem
```

```
TLSCertificateFile /certs/dc0.domain.alt.pem
```

```
TLSCertificateKeyFile /certs/dc0.domain.alt.key
```

5. Создать файл тестовых данных domain.alt.ldif со следующим содержимым:

```
dn: dc=domain,dc=alt
```

```
objectClass: organization
```

```
objectClass: dcObject
```

```
dc: domain
```

```
o: alt
```

```
dn: ou=users,dc=domain,dc=alt
```

```
objectClass: top
```

```
objectClass: organizationalUnit
```

```
ou: users
```

```
description: Central location for users
```

```
dn: ou=groups,dc=domain,dc=alt
```

```
objectClass: top
```

```
objectClass: organizationalUnit
```

```
ou: groups
```

description: Central location for groups

dn: ou=equipment,dc=domain,dc=alt

objectClass: top

objectClass: organizationalUnit

ou: equipment

description: Central location for computers

dn: ou=policies,dc=domain,dc=alt

objectClass: top

objectClass: organizationalUnit

ou: policies

description: Central location for policies

dn: cn=default,ou=policies,dc=domain,dc=alt

cn: default

objectClass: organizationalRole

objectClass: pwdPolicy

pwdAttribute: userPassword

pwdMinLength: 12

pwdCheckQuality: 2

pwdMaxFailure: 10

pwdLockout: TRUE

pwdLockoutDuration: 600

pwdInHistory: 5

pwdMustChange: TRUE

dn: ou=test_delete_ou,dc=domain,dc=alt

objectClass: top

objectClass: organizationalUnit

ou: test_delete_ou

description: OU for deletion testing

dn: ou=test_rename_ou,dc=domain,dc=alt

objectClass: top

objectClass: organizationalUnit
ou: test_rename_ou
description: OU for rename testing

dn: ou=test_mod_ou,dc=domain,dc=alt
objectClass: top
objectClass: organizationalUnit
ou: test_mod_ou
description: OU for modification testing

dn: cn=test_delete_group,ou=groups,dc=domain,dc=alt
objectClass: top
objectClass: posixGroup
cn: test_delete_group
gidNumber: 0

dn: cn=test_rename_group,ou=groups,dc=domain,dc=alt
objectClass: top
objectClass: posixGroup
cn: test_rename_group
gidNumber: 1

dn: cn=test_mod_group,ou=groups,dc=domain,dc=alt
objectClass: top
objectClass: posixGroup
cn: test_mod_group
gidNumber: 1

dn: cn=test_delete_user,ou=users,dc=domain,dc=alt
uid: test_delete_user
gecos: test_delete_user
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount

userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/false
uidNumber: 801
gidNumber: 801
homeDirectory: /home/test_delete_user

dn: cn=test_mod_user,ou=users,dc=domain,dc=alt
uid: test_mod_user
gecos: test_mod_user
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/false
uidNumber: 801
gidNumber: 801
homeDirectory: /home/test_mod_user

dn: cn=test_rename_user,ou=users,dc=domain,dc=alt
uid: test_rename_user
gecos: test_rename_user
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa

```
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/false
uidNumber: 801
gidNumber: 801
homeDirectory: /home/test_rename_user

dn: cn=test_search_user,ou=users,dc=domain,dc=alt
uid: test_search_user
gecos: test_search_user
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}RsAMqOI3647qg1gAZF3x2BKBnp0sEVfa
shadowLastChange: 15140
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/false
uidNumber: 801
gidNumber: 801
homeDirectory: /home/test_search_user

dn: cn=test_block_user,ou=users,dc=domain,dc=alt
uid: test_block_user
gecos: test_block_user
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword: {SSHA}gVK8WC9YyFT1gMsQHTGCgT3sSv5zYWx0
shadowLastChange: 15140
```

shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/false
uidNumber: 801
gidNumber: 801
homeDirectory: /home/test_block_user

dn: cn=test_rename_computer,ou=equipment,dc=domain,dc=alt
objectClass: top
objectClass: device
cn: test_rename_computer
description: Some brand of computer
seeAlso: dc=domain,dc=alt
serialnumber: 1-77-23-13
l: Room 17
owner: cn=john smith,ou=people,dc=domain,dc=alt
ou: equipment

dn: cn=test_mod_computer,ou=equipment,dc=domain,dc=alt
objectClass: top
objectClass: device
cn: test_mod_computer
description: Some brand of computer
seeAlso: dc=domain,dc=alt
serialnumber: 1-77-23-17
l: Room 17
owner: cn=john smith,ou=people,dc=domain,dc=alt
ou: equipment

dn: cn=test_delete_computer,ou=equipment,dc=domain,dc=alt
objectClass: top
objectClass: device
cn: test_delete_computer
description: Some brand of computer

```
seeAlso: dc=domain,dc=alt
serialnumber: 1-77-23-18
l: Room 17
owner: cn=john smith,ou=people,dc=domain,dc=alt
ou: equipment
```

6. Создать скрипт заполнения OpenLDAP тестовыми данными start-ldap.sh со следующим содержимым:

```
#!/bin/bash

mkdir /tmp/ldap

slapd -d any -h "ldap://0.0.0.0:3890/ ldaps://0.0.0.0:6360" -f
./slapd.conf 2>&1 > /tmp/slapd.log &

i=0
while [ $i -le 15 ]
do
if ldapadd -x -f ./domain.alt.ldif -H ldap://127.0.0.1:3890 -D
"cn=admin,dc=domain,dc=alt" -w password ; then
break
else
sleep 2
i=$(( $i + 1 ))
fi
done

if [ $? -ne 0 ]; then
echo "Error while configuring slapd service!"
cat /tmp/slapd.log
exit 1
fi
```

7. Запустить скрипт:

```
# chmod +x start-ldap.sh
```



```
# ./start-ldap.sh
```

8.3. Настройка узла с libdomain

Все действия выполняются на узле (192.168.0.152).

Для установки libdomain на машину с ОС «Альт» следует установить пакеты libdomain libdomain-tests:

```
# apt-get install libdomain libdomain-tests
```

Примечание. Если libdomain устанавливается на машину с другим дистрибутивом, необходимо выполнить следующие команды:

```
$ git clone https://github.com/libdomain/libdomain.git
```

```
$ cd libdomain && mkdir build && cd build && cmake .. && make -j  
`nproc`
```

8.4. Тестирование libdomain

Для тестов samba/AD нужно указать переменные:

- LDAP_SERVER=ldap://dc0.domain.alt:389
- LDAPS_SERVER=ldaps://dc0.domain.alt:636
- LDAP_CA_CERT=/certs/ca.pem
- DIRECTORY_TYPE=AD
- VALID_CONFIG_FILE=/**<путь к файлу>**/config.ini

Для тестов OpenLDAP нужно указать переменные:

- LDAP_SERVER=ldap://**<ipv4 хоста>**:3890
- LDAPS_SERVER=ldaps://dc1.domain.alt:6360
- LDAP_CA_CERT=/certs/ca.pem
- DIRECTORY_TYPE=OpenLDAP
- VALID_CONFIG_FILE=/**<путь к файлу>**/config.ini

8.5. Использование библиотеки libdomain

Подключение библиотеки:

```
#include <ldap.h>
```

```
#include <malloc.h>

#include <libdomain/common.h>
#include <libdomain/domain.h>
#include <libdomain/domain_p.h>
#include <libdomain/directory.h>
#include <libdomain/entry.h>
#include <libdomain/connection_state_machine.h>
```

Инициализация соединения:

```
// Replace these values with your LDAP server details
const char *ldap_server = "ldap://example.com";
const char *ldap_username = "your_username";
const char *ldap_password = "your_password";
const char *ldap_bind_dn = "dc=example,dc=com";

// Initialize the LDAP connection
const int update_interval = 1000;

ld_config_t *config = NULL;
config = ld_create_config(talloc_ctx, ldap_server, 0, LDAP_VERSION3,
                        ldap_bind_dn, ldap_username, ldap_password,
                        false, false, true, false,
                        update_interval, "", "", "");

LDHandle *handle = NULL;
ld_init(&handle, config);

ld_install_default_handlers(handle);
ld_install_handler(handle, connection_on_update, update_interval);
```

Выполнение операций LDAP:

```
// Install search handler
ld_install_handler(handle, connection_on_update, update_interval);

// Start event loop.
ld_exec(handle);
// Example: Search for a user
```

```

static char* LDAP_DIRECTORY_ATTRS[] = { "objectClass", NULL };

static void connection_on_update(verto_ctx *ctx, verto_ev *ev)
{
    (void) (ctx);

    struct ldap_connection_ctx_t* connection = verto_get_private(ev);

    if (connection->state_machine->state == LDAP_CONNECTION_STATE_RUN)
    {
        verto_del(ev);

        search(connection, "ou=users,dc=example,dc=com",
                LDAP_SCOPE_SUBTREE,
                "(uid=johndoe)", LDAP_DIRECTORY_ATTRS, 0, NULL);
    }

    if (connection->state_machine->state ==
LDAP_CONNECTION_STATE_ERROR)
    {
        verto_break(ctx);

        error("Error encountered during bind!\n");
    }
}

```

Заккрытие соединения:

```

// Close the LDAP connection when done
ld_free(handle);

talloc_free(talloc_ctx);

```