

# Регистрация и конфигурация клиента (client)

Сервер Keycloak поддерживает для регистрации приложений OpenID Connect протокол и стандарт SAML (Security Assertion Markup Language).

[OpenID Connect](#) (OIDC) - это протокол аутентификации, который является расширением OAuth 2.0. (построенный поверх фреймворка авторизации OAuth 2.0)

Для регистрации приложений OpenID Connect является предпочтительным протоколом для защиты приложений.

В то время как [OAuth 2.0](#) является только основой для построения протоколов авторизации и, в основном, является неполным, OIDC является полноценным протоколом аутентификации и авторизации.

OIDC также активно использует набор стандартов [JSON Web Token](#) (JWT). Эти стандарты определяют формат JSON-токена идентификации и способы цифровой подписи и шифрования этих данных.

## Требования по заполнению базовых конфигурационных параметров у клиента

Сервер аутентификации Keycloak предоставляет конфигурацию клиентских приложений через встроенную панель администрирования KeyCloak, либо по REST- API.

Наиболее значимые параметры более подробно рассматриваются ниже.

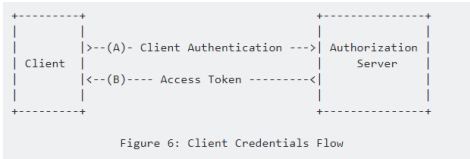
Для того чтобы клиентское приложение выполнило OpenID Connect запрос, при регистрации и конфигурации в сервере требуется задать индивидуальные параметры.

Наименование в UI Keycloak	Наименования по API в документации Keycloak	Описание	Допустимые символы	Уникальность
Client ID	clientId	Уникальный идентификатор, выданный клиенту, чтобы идентифицировать себя на сервере при запросах (OIDC).	/^[a-z0-9-]+\$ /	Y
Name	name	Отображаемое имя клиента на экране пользовательского интерфейса Keycloak.	/^[a-z0-9-]+\$ /	N
Description	description	Описание клиента	/^[a-z0-9-]+\$ /	N
Enabled	enabled	Если этот параметр отключен, то клиенту не будет разрешено запрашивать проверку подлинности.	-	N

Consent Required	consentRequired	<p>Запрос согласия.</p> <p>Если параметр активирован, то пользователям будет отражен экран с запросом на согласие обработки персональных данных. Он также будет отображать метаданные, которые запрашиваются для обработки приложением, чтобы пользователь точно знал, к какой информации приложение получает доступ.</p>	-	N
Login Theme		Тип темы UI, используемого при логине	-	N
Client Protocol	protocol	Протокол, используемый для защиты.	-	N

Access Type	access  bearerOnly	<p>Определяет тип доступа по OIDC.</p> <p><i>confidential</i></p> <p>"Конфиденциальный" тип доступа предназначен для клиент-серверных приложений, которые авторизуют своих пользователей через браузер, а также используют client_id и client_secret (секрет) клиента для генерации <a href="#">токена</a> доступа.</p> <p><i>public</i></p> <p>Тип общего доступа, клиентское приложение не может предоставить секрет. Вместо этого очень важно ограничить доступ, настроив <b>правильные URI перенаправления</b> для клиента.</p> <p><i>bearer-only</i></p> <p>Тип доступа "Только на предъявителя". Используется для аутентификации Web-сервисов. Если <b>используется <i>bearer-only</i> тип доступа, то приложение не может участвовать в авторизации с использованием браузера</b> (не используются учетные записи пользователей).</p> <p>При вызове REST-сервиса, к примеру, в заголовке указывается Authorization со значением Bearer \${значение JWT токена}.</p>	-	N
-------------	--------------------------	--	---	---

Standard Flow Enabled	standardFlowEnabled	<p>В соответствии с <a href="#">OAuth 2.0</a>.</p>	-	N
<a href="#">Implicit Flow Enabled</a>	implicitFlowEnabled	<p>Аналогичен потоку Standard Flow Enabled, но не используются токены обновления. <b>Не рекомендуется использовать данный поток.</b></p>	-	N
Direct Access Grants Enabled  (Resource Owner Password Credentials Grant)	directAccessGrantsEnabled	<p>При включенном значении доступна функциональность <a href="#">Resource Owner Password Credentials Grant</a>.</p> <p>Figure 5: Resource Owner Password Credentials Flow</p> <p>В потоке передается пользовательский пароль.</p> <p>Включение параметра возможно совместно с используемым <a href="#">Standard Flow Enabled</a>.</p>	-	N

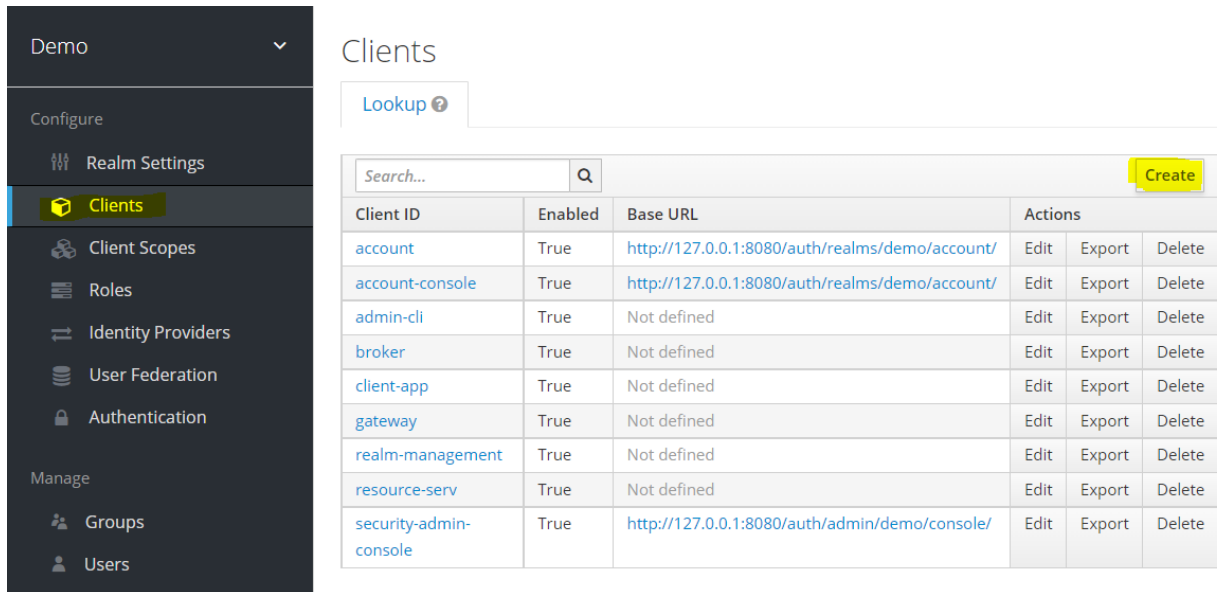
Service Accounts Enabled  (Client Credentials Grant)	serviceAccountsEnabled	<p>Каждый клиент OIDC имеет встроенную учетную запись, которая позволяет ему получить токен доступа. Чтобы использовать эту функцию, необходимо установить тип доступа клиента как <i>confidential</i>. При этом появится переключатель 'Service Accounts Enabled'. При включенном значении будет доступна возможность получения токена авторизации. <b>Рекомендуется использовать данную функциональность для защиты Rest-сервисов для внешних регистрируемых клиентов.</b></p>  <p>Figure 6: Client Credentials Flow</p> <p>В потоке client credentials flow требуется передавать только client_id и client_secret.</p> <p>Этот поток также используется REST - сервисами, но вместо получения токена, который работает от имени внешнего пользователя, токен создается на основе метаданных и разрешений учетной записи службы, связанной с клиентом.</p>	-	N
Authorization Enabled	authorizationServicesEnabled  authorizationSettings			
Root URL	rootUrl	Если Keycloak использует любые относительные URL-адреса, то это значение добавляется к ним.	/^(https?:\V)?([\da-z\.-]+\.[a-z\]{2,6})([\\w\.-]*)*\V?\$	Y

Valid Redirect URIs	redirectUris	<p>Обязательное поле для заполнения.</p> <p>Используется для идентификации тех точек приложения, куда допускается отправка результата после аутентификации (JWT-токеном). Также можно задать маску: /____/ *. Это дает право Keycloak делать редирект на любой ресурс, соответствующий данной маске.</p> <p>Подстановочные знаки ( \ * ) допускаются только в конце URI, т. е. <a href="http://host.com/">http://host.com/</a>.</p>	/^(https?:\V)?([\da-z\.-]+\.[a-z\]{2,6})([\Vw\.-]*)*\V?\$	Y
Base URL (Root URL)	baseUrl	Базовый URL используется сервером аутентификации Keycloak для редиректа на клиента (как точка обратного вызова).	/^(https?:\V)?([\da-z\.-]+\.[a-z\]{2,6})([\Vw\.-]*)*\V?\$	Y
Admin URL	adminUrl	Используется для обратных вызовов при служебных операциях.	/^(https?:\V)?([\da-z\.-]+\.[a-z\]{2,6})([\Vw\.-]*)*\V?\$	Y
Web Origins	webOrigins	<p>Этот параметр предоставляет функциональность CORS.</p> <p>Cross-origin resource sharing — «совместное использование ресурсов между разными источниками» — технология современных браузеров, которая позволяет предоставить веб-странице доступ к ресурсам другого домена.</p> <p>Если в значении указывать плюс: «+», то CORS будет работать только для значений, указанных в redirectUris, baseUrl, adminUrl.</p>	-	-

## Регистрация клиентского приложения через панель администрирования Keycloak

Ниже приводится описание шагов, необходимых для регистрации клиентских приложений через панель администрирования.

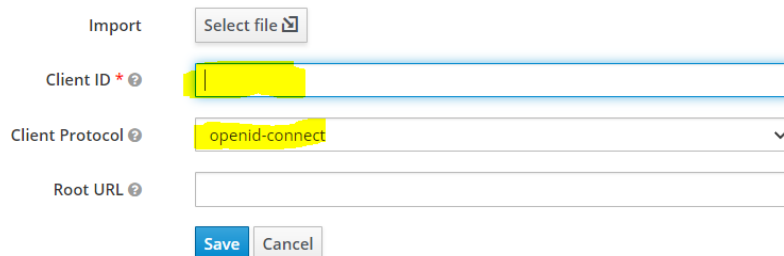
1. Для регистрации клиентского приложения необходимо в области 1. Clients выполнить "2. Create" .



Client ID	Enabled	Base URL	Actions
account	True	http://127.0.0.1:8080/auth/realms/demo/account/	Edit Export Delete
account-console	True	http://127.0.0.1:8080/auth/realms/demo/account/	Edit Export Delete
admin-cli	True	Not defined	Edit Export Delete
broker	True	Not defined	Edit Export Delete
client-app	True	Not defined	Edit Export Delete
gateway	True	Not defined	Edit Export Delete
realm-management	True	Not defined	Edit Export Delete
resource-serv	True	Not defined	Edit Export Delete
security-admin-console	True	http://127.0.0.1:8080/auth/admin/demo/console/	Edit Export Delete

2. При регистрации нового клиентского приложения задается обязательно Client ID. Идентификатор будет использоваться в запросах и в базе данных Keycloak для идентификации клиента. Затем выберите openid-connect в раскрывающемся списке протоколов. Введите базовый URL-адрес приложения в поле "Root URL" и нажмите кнопку Сохранить. Клиентское приложение будет зарегистрировано.

### Add Client



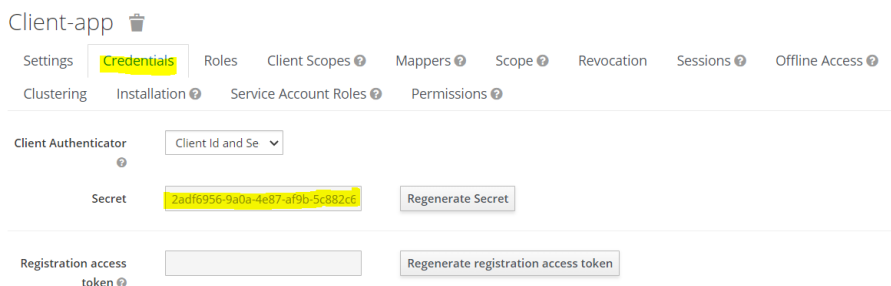
Import

Client ID \*

Client Protocol

Root URL

3. Далее будет предложено настроить клиента.
4. После регистрации confidential клиента Keycloak автоматически создает Secret, на основе которого и Client ID возможна генерация токена доступа.



Client-app

Settings **Credentials** Roles Client Scopes Mappers Scope Revocation Sessions Offline Access

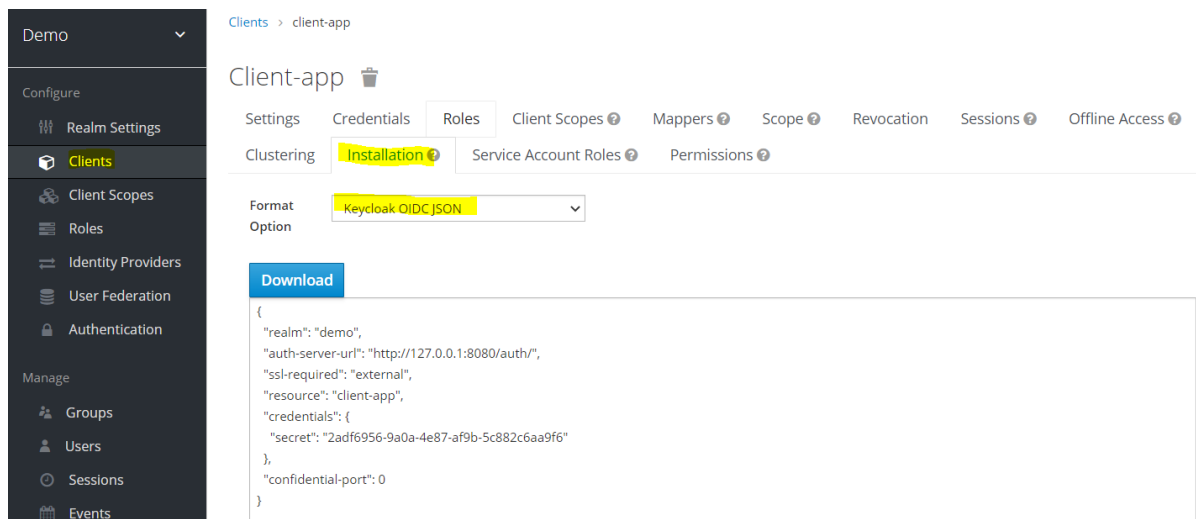
Clustering Installation Service Account Roles Permissions

Client Authenticator

Secret

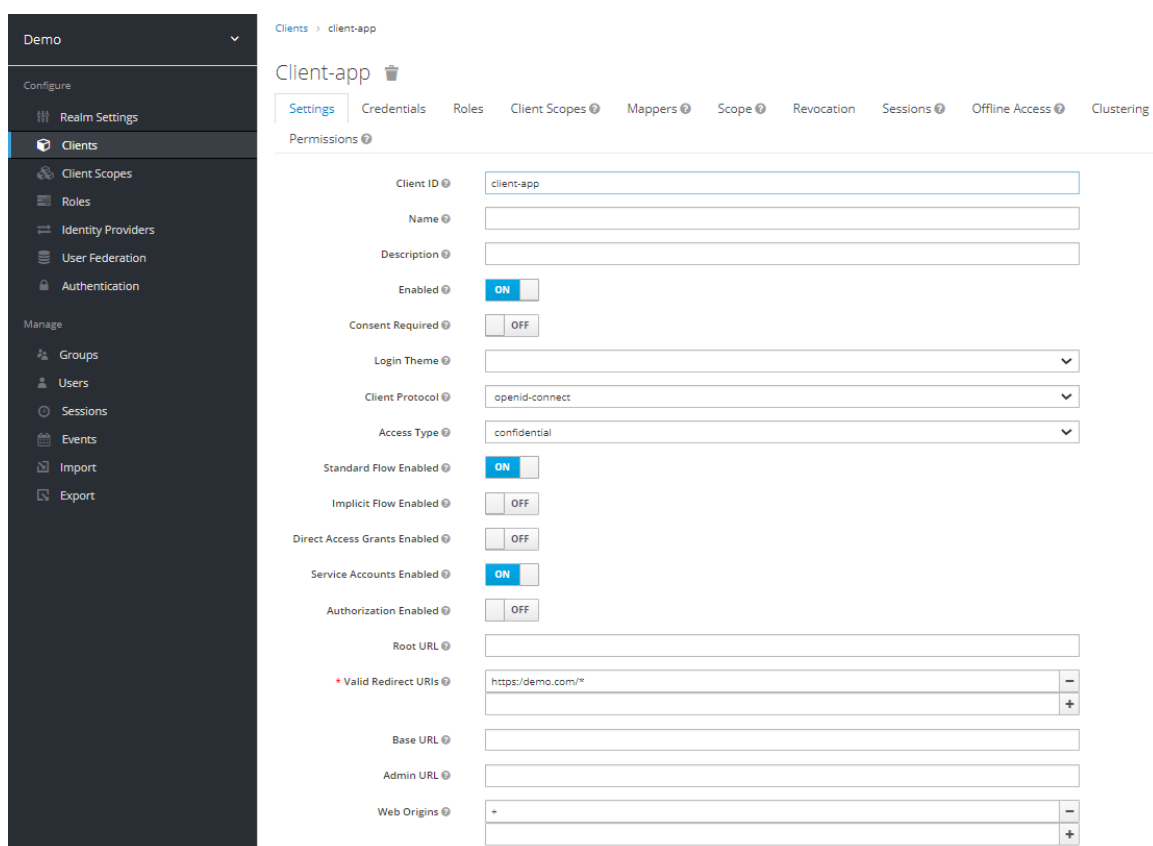
Registration access token

5. Клиентское приложение должно быть настроено на аутентификацию через Keycloak. Для этого следует воспользоваться настройкой из самого Keycloak для данного клиента. Ниже приведен пример из Keycloak.



## Конфигурация клиентского приложения через панель администрирования KeyCloak

- Конфигурация клиента выполняется в зависимости от необходимости аутентификации
- (с типом *"confidential"* + PKCE, *"public"*+ PKCE) с редиректом на окно аутентификации KeyCloak или для REST-сервисов с типом доступа *"bearer-only"*. Пример конфигурации клиентского приложения с типом доступа *"confidential"* с редиректом на окно авторизации KeyCloak приводится ниже.



- Пример конфигурации web-сервиса с типом доступа *"bearer-only"* приводится ниже.



Settings

Credentials

Roles

Revocation

Clustering

Installation ?

Permissions ?

Client ID ?

restservice

Name ?

rest-service-test

Description ?

тестирование функциональности KeyCloak при имитации работы rest-сервиса

Enabled ?

ON

Consent Required ?

OFF

Login Theme ?

Client Protocol ?

openid-connect

Access Type ?

bearer-only

Authorization Enabled ?

OFF

Admin URL ?

> Fine Grain OpenID Connect Configuration ?

> OpenID Connect Compatibility Modes ?

> Advanced Settings ?

> Authentication Flow Overrides ?

Save

Cancel