

Creating and Configuring Clients in Keycloak

A **client** in Keycloak represents an application or service that uses Keycloak to authenticate users. Clients can be web apps, REST APIs, mobile apps, or even CLI tools. This guide explains how to create and configure clients through the Admin Console, REST API, and CLI (Docker), and also includes roles, best practices, and common troubleshooting steps.

Creating Clients via Keycloak Admin Console

This is the simplest way to register and configure a client visually.

Access the Admin Console

Log in to:

`http://<your-keycloak-domain>/admin/`

Choose the realm where the client should be added.

Add a New Client

1. Go to **Clients > Create**
2. Fill in the fields:
 - **Client ID:** A unique name, e.g., frontend-app or api-service
 - **Client Type:** Choose between OpenID Connect (default) or SAML
 - **Root URL:** The application base URL (e.g., http://localhost:3000)

3. Click **Next**, then **Save**

Clients > Create client

Create client

Clients are applications and services that can request authentication of a user.

- 1 General settings
- 2 Capability config
- 3 Login settings

Client type ⓘ OpenID Connect

Client ID * ⓘ

Name ⓘ

Description ⓘ

Always display in UI ⓘ ☐ Off

Back Next Cancel

Configure Client Settings

- Go to the **Settings** tab for the client:
 - **Access Type**: Choose public, confidential, or bearer-only
 - **Valid Redirect URIs**: Add allowed redirect URLs (e.g., `http://localhost:3000/*`)
 - **Web Origins**: Add * or specific origins allowed to call this client
 - **Standard Flow Enabled**: Enable for browser-based login
 - **Direct Access Grants**: Enable if using password grant from API
- Save the changes

Creating Clients via Keycloak REST API

Get Access Token

```
curl -X POST "https://<keycloak-domain>/realms/master/protocol/openid-connect/token" \  
-H "Content-Type: application/x-www-form-urlencoded" \  
-d "grant_type=password&username=admin@localhost.com&password=admin" \  
-d "client_id=admin-cli"
```

```
-d "username=admin" \  
-d "password=admin-password" \  
-d "grant_type=password" \  
-d "client_id=admin-cli"
```

Save the access_token.

Create a Client

```
curl -X POST "https://<keycloak-domain>/admin/realms/<realm>/clients" \  
-H "Authorization: Bearer <access_token>" \  
-H "Content-Type: application/json" \  
-d '{  
  "clientId": "my-app",  
  "enabled": true,  
  "publicClient": false,  
  "redirectUris": ["http://localhost:3000/*"],  
  "webOrigins": ["http://localhost:3000"],  
  "protocol": "openid-connect"  
'
```

This creates a confidential client named my-app.

Creating Clients via Docker CLI

Step into the Container

```
docker exec -it keycloak bash
```

Authenticate and Create Client

```
/opt/keycloak/bin/kcadm.sh config credentials \  
--server http://localhost:8080 \  
--realm master --user admin --password admin  
  
/opt/keycloak/bin/kcadm.sh create clients -r <realm> \  
-s clientId=my-cli-client \  
-s enabled=true \  
-s publicClient=false \  
-s redirectUris='["http://localhost:3000/*"]' \  

```

```
-s webOrigins='["http://localhost:3000"]'
```

Required Permissions for Client Management

- Requires manage-clients or admin role in the realm
- Token used via REST or CLI must be scoped to allow client creation

To grant roles via Admin Console:

Users > admin > Role Mappings > Realm Roles > Assign 'manage-clients'

Best Practices for Client Configuration

- **Use Confidential Clients for Backends:** Set `publicClient = false` and use `client_secret` for server-to-server communication.
- **Use Public Clients for SPAs:** Frontend apps using redirect flows should be marked as `publicClient = true`.
- **Set Narrow Redirect URIs:** Avoid using wildcards like `*` unless absolutely necessary. Use precise URIs for better security.
- **Limit Token Lifespans:** Go to **Realm Settings > Tokens** and configure access and refresh token lifetimes.
- **Rotate Client Secrets Regularly:** Manually rotate secrets or use automation for higher security compliance.
- **Use Roles and Mappers for RBAC:** Assign client roles and use **protocol mappers** to inject them into access tokens for authorization checks.

Common Issues and Troubleshooting

Issue	Possible Cause	Solution
Invalid redirect URI	Redirect URI doesn't match registered value	Ensure exact match in Valid Redirect URIs
Client not visible after creation	UI or API delay	Refresh or re-login to see updated clients
Access token doesn't include roles	Missing mappers	Add protocol mapper for client roles under Client > Mappers

Issue	Possible Cause	Solution
403 Forbidden when using client credentials	Client type is public or secret is wrong	Verify publicClient=false and check the client secret
Invalid client credentials error	Wrong client ID or secret	Verify spelling and match values from Admin Console

Revision #1
Created 17 June 2025 11:46:20 by kaiwalya
Updated 17 June 2025 11:58:06 by kaiwalya