REPUBLIC OF ESTONIA
MINISTRY OF ECONOMIC AFFAIRS
AND COMMUNICATIONS

Nordic Council
of Ministers

eCMR Index Registry Adding Guide

Development funded by

DIGINNO-Proto

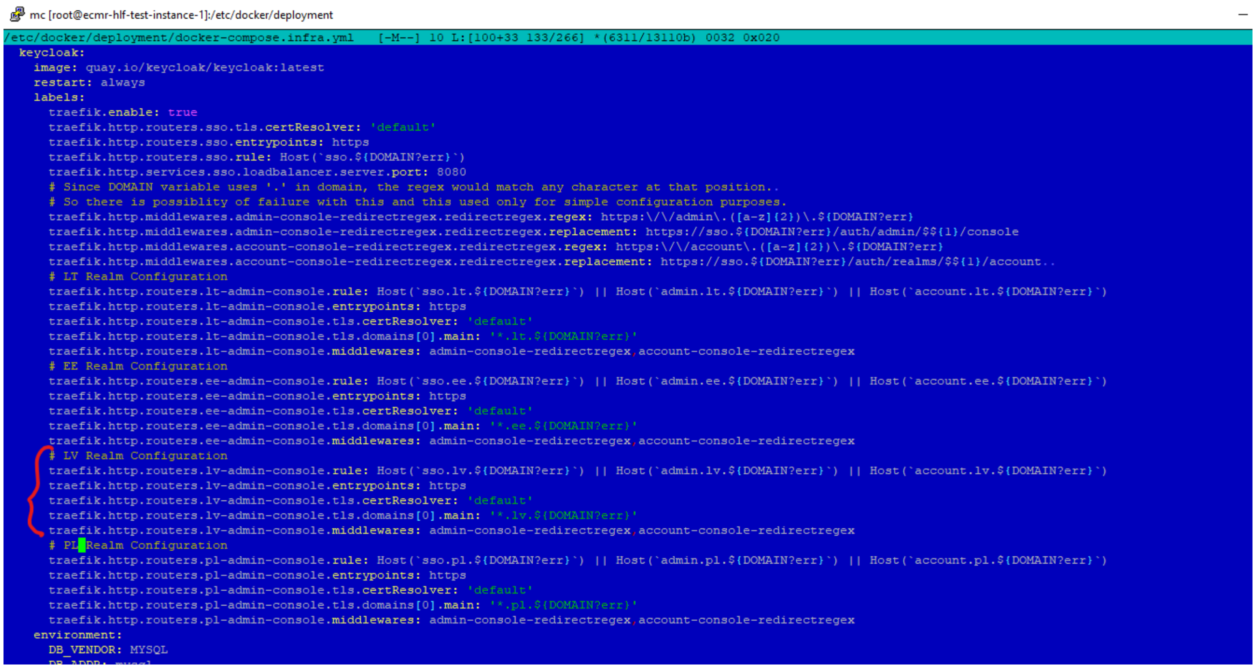
Version 1.00

Prerequisites/Notes

- Admin credentials of *Keycloak* instance *master* realm
- Credentials for connecting to the infrastructure of indexing servers

Steps

1. Go to "***/etc/docker/deployment/fabric-network/test-network/addCountry***"
2. Run "***addCountry.sh up -country <two_letter_country_code>***" ex.: "***addCountry.sh up -country ua***"
3. If everything goes successfully, go to "***/etc/docker/deployment/fabric-network/scripts/add-country-scripts***" and check if "***.tmp***" file contains all deployed countries (the file content should look like "*lt ee lv pl ua*", if the '*ua*' was added as a new indexing server).
4. Open browser and go to "***https://sso.playground.ecmr4.eu/auth/***"
5. Login with administrator account and create a realm with a name of two letter country code (lowercased)
6. Go to "***/etc/docker/deployment***", open "**docker-compose.infra.yml**" and update "**keycloak**" service with required Traefik variables for newly added realm (you must wait a minute or two until *keycloak* instance reacreates)

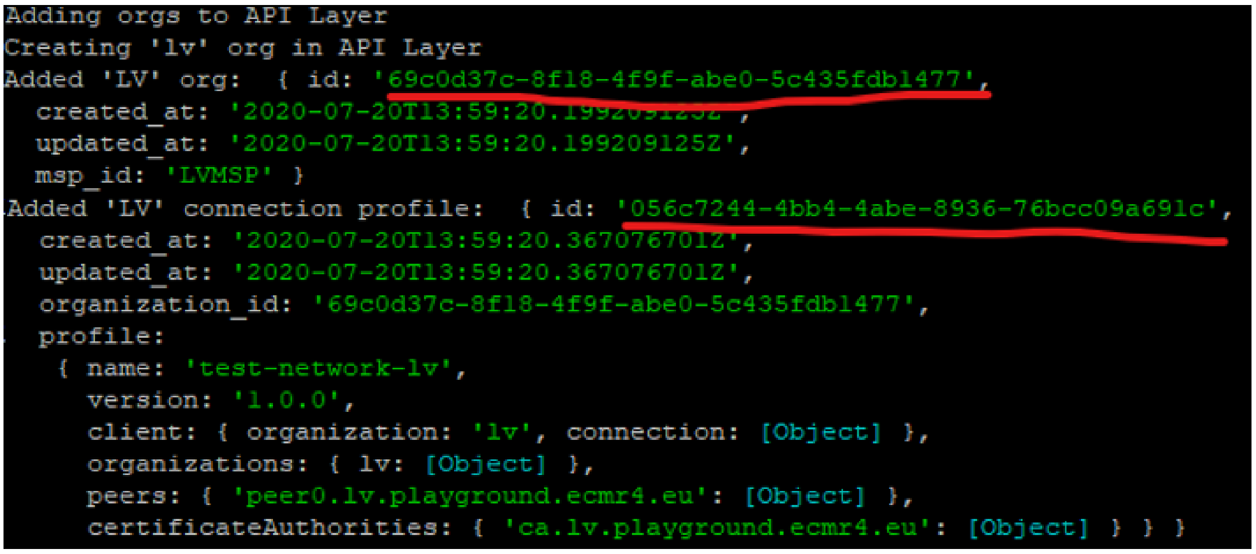


```
mc [root@ecmr-hlf-test-instance-1]:/etc/docker/deployment
/etc/docker/deployment/docker-compose.infra.yml    [-M--] 10 L:[100+33 133/266] *(6311/13110b) 0032 0x020
  keycloak:
    image: quay.io/keycloak/keycloak:latest
    restart: always
    labels:
      traefik.enable: true
      traefik.http.routers.sso.tls.certResolver: 'default'
      traefik.http.routers.sso.entrypoints: https
      traefik.http.routers.sso.rule: Host(`sso.${DOMAIN?err}`)
      traefik.http.services.sso.loadbalancer.server.port: 8080
      # Since DOMAIN variable uses '.' in domain, the regex would match any character at that position..
      # So there is possiblity of failure with this and this used only for simple configuration purposes.
      traefik.http.middlewares.admin-console-redirectregex.redirectregex.regex: https:\/\/admin\.([a-z]{2})\.${DOMAIN?err}
      traefik.http.middlewares.admin-console-redirectregex.redirectregex.replacement: https://sso.${DOMAIN?err}/auth/admin/$${1}/console
      traefik.http.middlewares.account-console-redirectregex.redirectregex.regex: https:\/\/account\.([a-z]{2})\.${DOMAIN?err}
      traefik.http.middlewares.account-console-redirectregex.redirectregex.replacement: https://sso.${DOMAIN?err}/auth/realms/$${1}/account..
      # LT Realm Configuration
      traefik.http.routers.lt-admin-console.rule: Host(`sso.lt.${DOMAIN?err}`) || Host(`admin.lt.${DOMAIN?err}`) || Host(`account.lt.${DOMAIN?err}`)
      traefik.http.routers.lt-admin-console.entrypoints: https
      traefik.http.routers.lt-admin-console.tls.certResolver: 'default'
      traefik.http.routers.lt-admin-console.tls.domains[0].main: '*.lt.${DOMAIN?err}'
      traefik.http.routers.lt-admin-console.middlewares: admin-console-redirectregex,account-console-redirectregex
      # EE Realm Configuration
      traefik.http.routers.ee-admin-console.rule: Host(`sso.ee.${DOMAIN?err}`) || Host(`admin.ee.${DOMAIN?err}`) || Host(`account.ee.${DOMAIN?err}`)
      traefik.http.routers.ee-admin-console.entrypoints: https
      traefik.http.routers.ee-admin-console.tls.certResolver: 'default'
      traefik.http.routers.ee-admin-console.tls.domains[0].main: '*.ee.${DOMAIN?err}'
      traefik.http.routers.ee-admin-console.middlewares: admin-console-redirectregex,account-console-redirectregex
      # LV Realm Configuration
      traefik.http.routers.lv-admin-console.rule: Host(`sso.lv.${DOMAIN?err}`) || Host(`admin.lv.${DOMAIN?err}`) || Host(`account.lv.${DOMAIN?err}`)
      traefik.http.routers.lv-admin-console.entrypoints: https
      traefik.http.routers.lv-admin-console.tls.certResolver: 'default'
      traefik.http.routers.lv-admin-console.tls.domains[0].main: '*.lv.${DOMAIN?err}'
      traefik.http.routers.lv-admin-console.middlewares: admin-console-redirectregex,account-console-redirectregex
      # PL Realm Configuration
      traefik.http.routers.pl-admin-console.rule: Host(`sso.pl.${DOMAIN?err}`) || Host(`admin.pl.${DOMAIN?err}`) || Host(`account.pl.${DOMAIN?err}`)
      traefik.http.routers.pl-admin-console.entrypoints: https
      traefik.http.routers.pl-admin-console.tls.certResolver: 'default'
      traefik.http.routers.pl-admin-console.tls.domains[0].main: '*.pl.${DOMAIN?err}'
      traefik.http.routers.pl-admin-console.middlewares: admin-console-redirectregex,account-console-redirectregex
    environment:
      DB_VENDOR: MYSQL
```

Figure 1. Update example of 'docker-compose.infra.yml" file

7. Save the updated file and run "***docker-compose -f ./docker-compose.infra.yml -p infra up -d***" command
8. Go to "***/etc/docker/deployment/fabric-network/scripts***" and run command "***network.sh deployCC -l go -v 16 -verbose***" (number '16' is chaincode version which should be incremented by one every time you update chaincode).
9. Go to "***/etc/docker/deployment***" and edit "***docker-compose.api.yml***" with service for added country index api (take ***lt-index-api*** as example).
10. Then run "***docker-compose -f ./docker-compose.api.yml -p api up -d***"
11. Go to "***/etc/docker/deployment/fabric-network/test-network/api-scripts***"

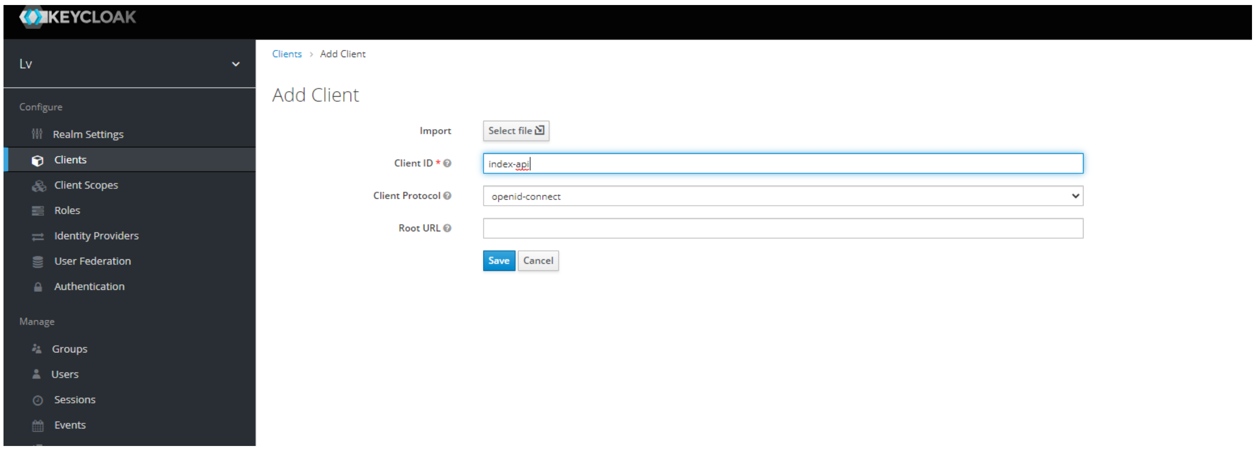
12. Run *“**node enrollAdmin.js <two_letter_deployed_country_code>**”* (ex.: *”**node enrollAdmin.js pl**”*)
13. Replace data in ***countries*** variable in ***addOrganizationsToAPI.js*** file with two letter country code you just newly added.
14. Run “**node addOrganizationsToAPI.js**” script and save output ID’s of added organizations and connection profiles. As an example, from the output bellow, we need to extract these ID’s as they will be used for user management configuration:
    - LV DLT organization ID: **69c0d37c-8f18-4f9f-abe0-5c435fdb1477**
    - LV DLT connection profile ID: **056c7244-4bb4-4abe-8936-76bcc09a691c**



```
Adding orgs to API Layer
Creating 'lv' org in API Layer
Added 'LV' org:  { id: '69c0d37c-8f18-4f9f-abe0-5c435fdb1477',
  created_at: '2020-07-20T13:59:20.199209125Z',
  updated_at: '2020-07-20T13:59:20.199209125Z',
  msp_id: 'LVMSP' }
Added 'LV' connection profile:  { id: '056c7244-4bb4-4abe-8936-76bcc09a691c',
  created_at: '2020-07-20T13:59:20.367076701Z',
  updated_at: '2020-07-20T13:59:20.367076701Z',
  organization_id: '69c0d37c-8f18-4f9f-abe0-5c435fdb1477',
  profile:
   { name: 'test-network-lv',
     version: '1.0.0',
     client: { organization: 'lv', connection: [Object] },
     organizations: { lv: [Object] },
     peers: { 'peer0.lv.playground.ecmr4.eu': [Object] },
     certificateAuthorities: { 'ca.lv.playground.ecmr4.eu': [Object] } } }
```

Figure 2. Example output of running 'node addOrganizationsToAPI.js' script

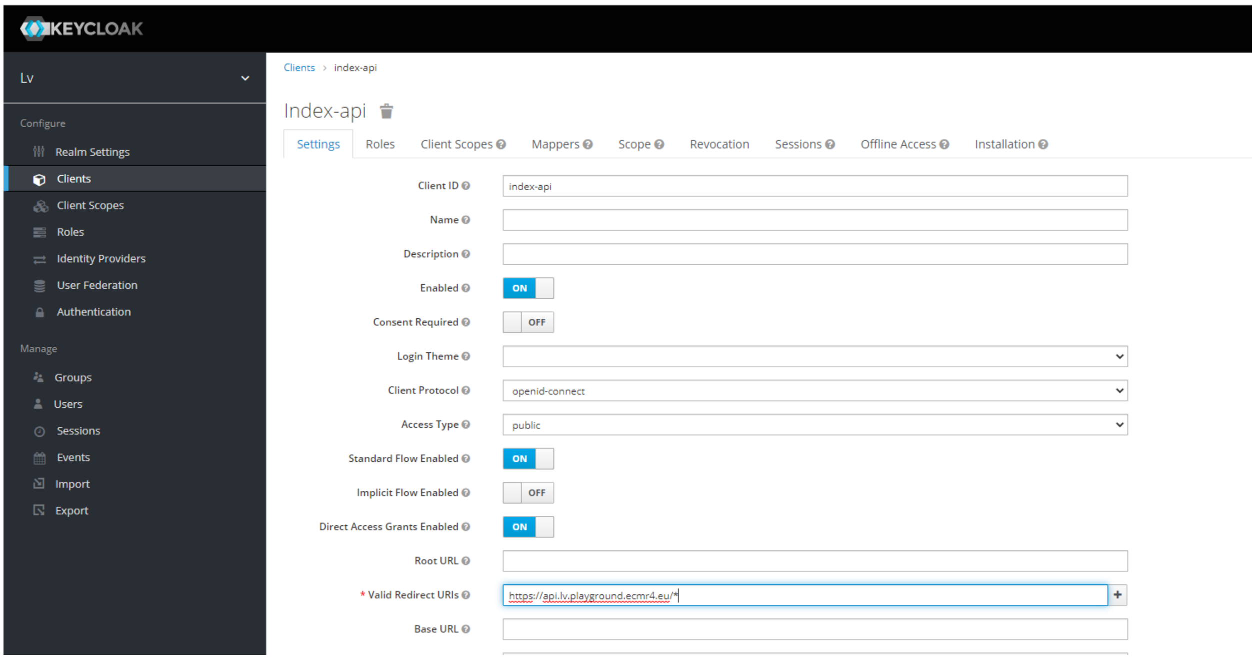
15. Go to https://sso.playground.ecmr4.eu/auth and login with admin credentials.
16. Configure realms you created previously:
    16.1. Create a client of with a name of ‘index-api’ and access type set as ‘public’



Add Client

Import: Select file
Client ID *: index-api
Client Protocol: openid-connect
Root URL:

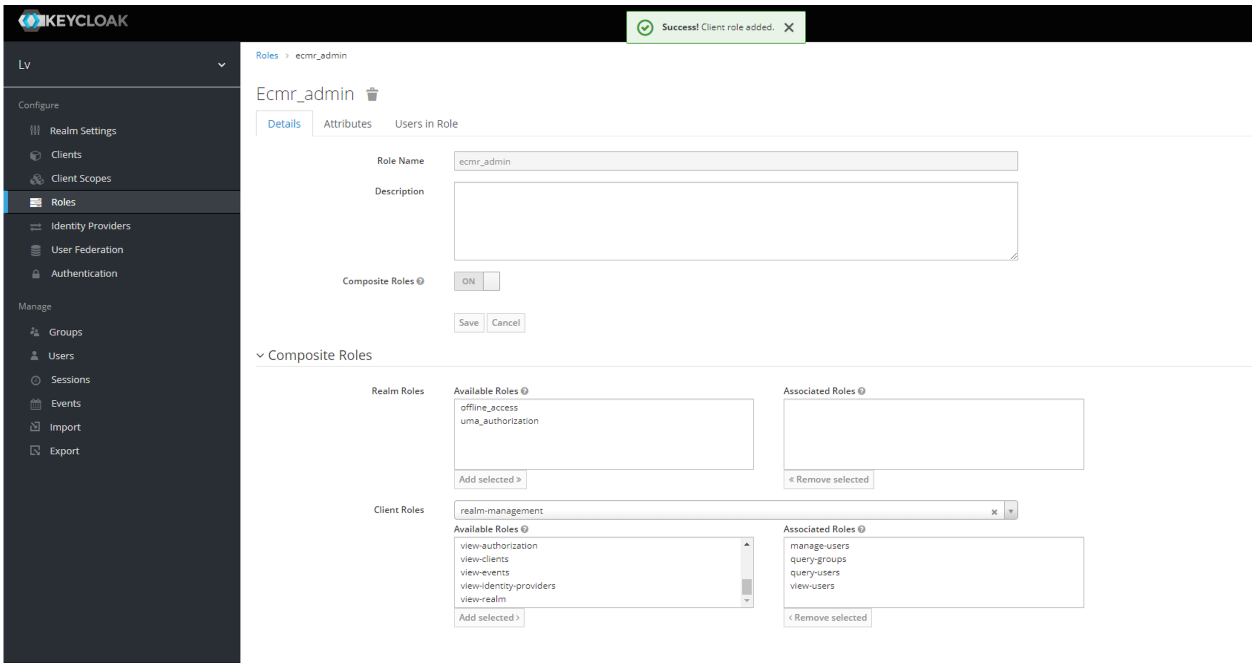
Figure 3. Client configuration for API layer (1)



Client ID	index-api
Name	
Description	
Enabled	ON
Consent Required	OFF
Login Theme	
Client Protocol	openid-connect
Access Type	public
Standard Flow Enabled	ON
Implicit Flow Enabled	OFF
Direct Access Grants Enabled	ON
Root URL	
* Valid Redirect URIs	https://api.lv.playground.ecmr4.eu/*
Base URL	

Figure 4. Client configuration for API layer (2)

16.2. Create an “***ecmr_admin***” role for country administrators with configuration as shown below (user with this role will be able to login to administration console and manage users/groups)



Success! Client role added.

Role Name	ecmr_admin
Description	
Composite Roles	ON

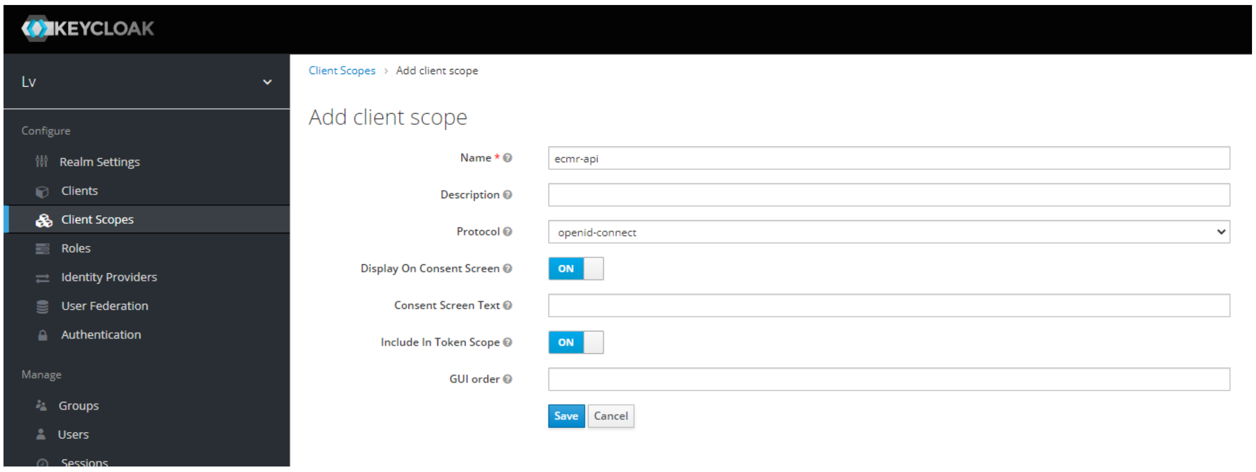
Composite Roles

Realm Roles
Available Roles: offline_access, uma_authorization
Associated Roles:

Client Roles: realm-management
Available Roles: view-authorization, view-clients, view-events, view-identity-providers, view-realm
Associated Roles: manage-users, query-groups, query-users, view-users

Figure 5. Country administrator role configuration

16.3. Create an ‘***ecmr-api***’ scope

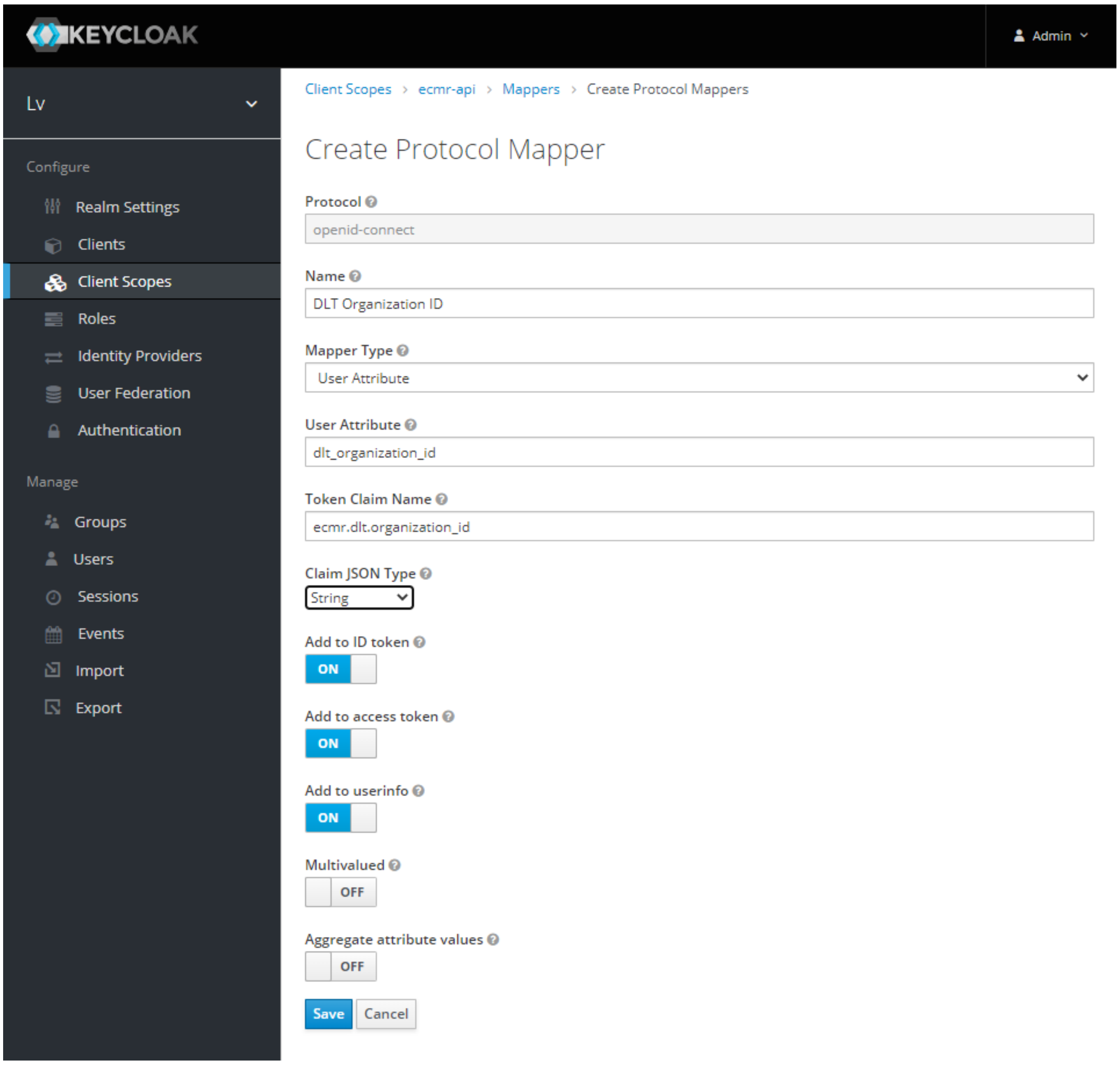


Add client scope

Name *	ecmr-api
Description	
Protocol	openid-connect
Display On Consent Screen	ON
Consent Screen Text	
Include In Token Scope	ON
GUI order	

Figure 6. 'ecmr-api' client scope creation

16.4. Configure token mappers for ‘ecmr-api’ scope



Client Scopes > ecmr-api > Mappers > Create Protocol Mappers

Create Protocol Mapper

Protocol: openid-connect

Name: DLT Organization ID

Mapper Type: User Attribute

User Attribute: dlt_organization_id

Token Claim Name: ecmr.dlt.organization_id

Claim JSON Type: String

Add to ID token: ON

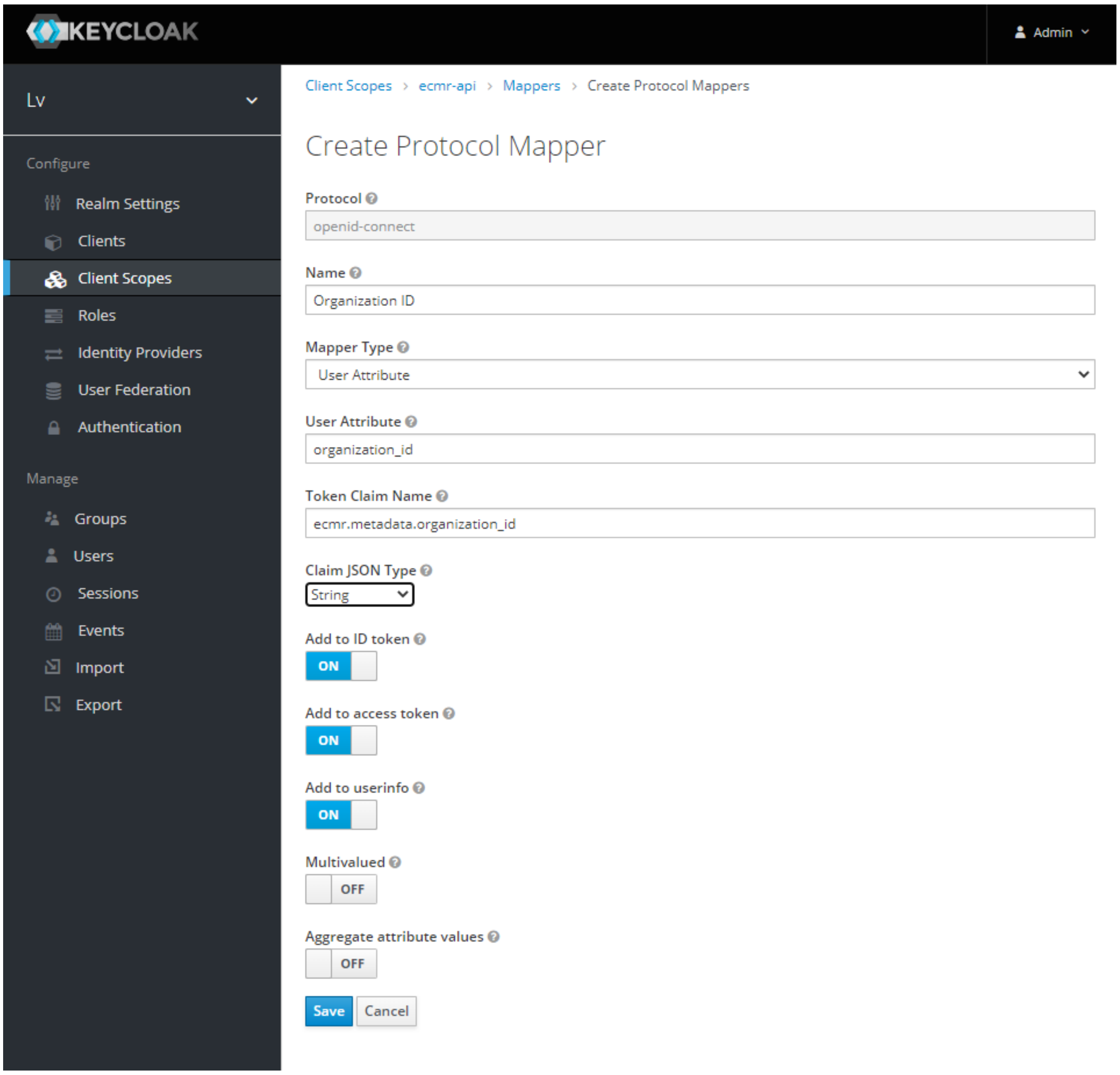
Add to access token: ON

Add to userinfo: ON

Multivalued: OFF

Aggregate attribute values: OFF

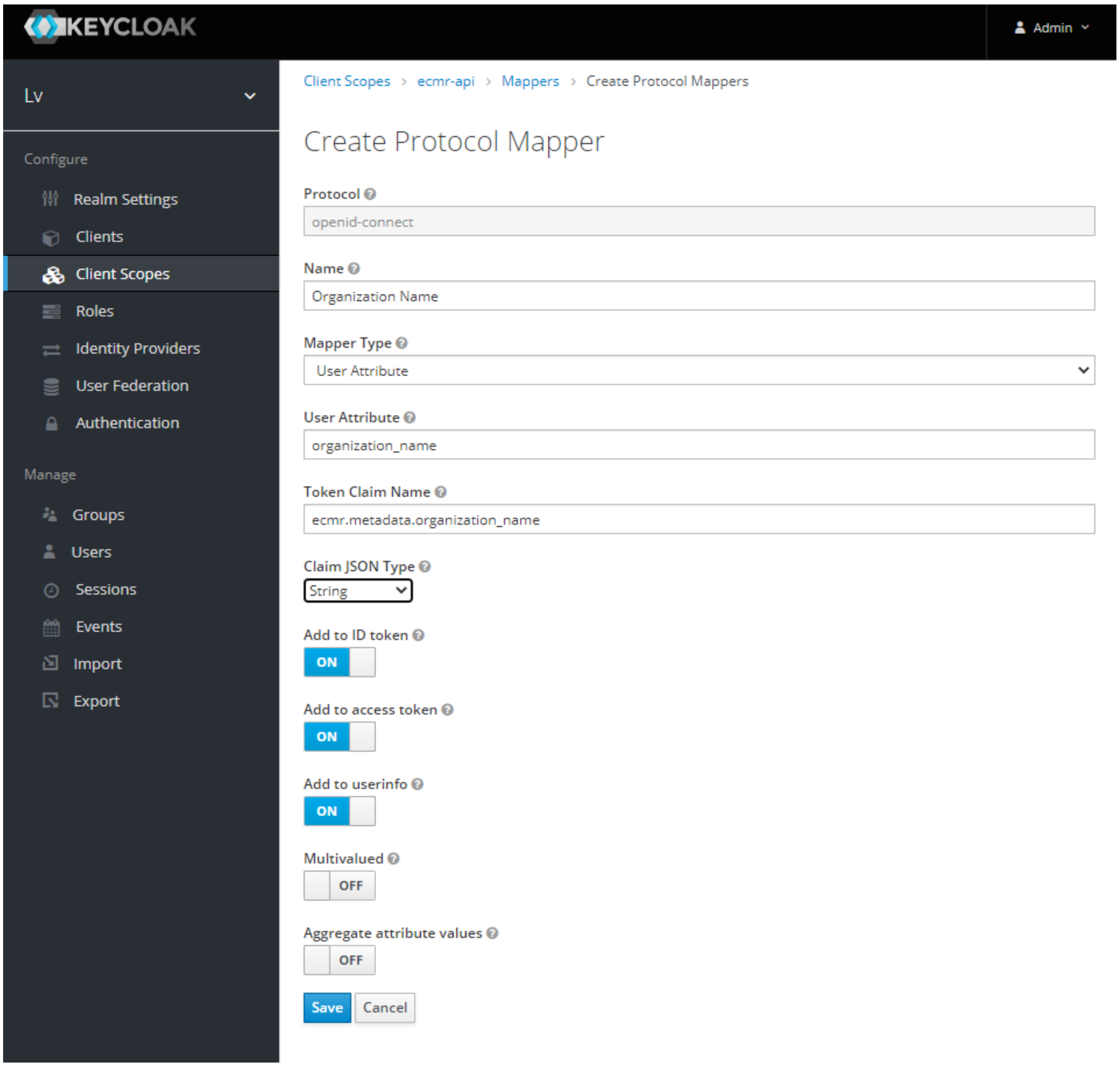
Figure 7. 'ecmr-api' client scope token mapping configuration



Create Protocol Mapper

Protocol	openid-connect
Name	Organization ID
Mapper Type	User Attribute
User Attribute	organization_id
Token Claim Name	ecmr.metadata.organization_id
Claim JSON Type	String
Add to ID token	ON
Add to access token	ON
Add to userinfo	ON
Multivalued	OFF
Aggregate attribute values	OFF

Figure 8. 'ecmr-api' client scope token mapping configuration



Create Protocol Mapper

Protocol: openid-connect

Name: Organization Name

Mapper Type: User Attribute

User Attribute: organization_name

Token Claim Name: ecmr.metadata.organization_name

Claim JSON Type: String

Add to ID token: ON

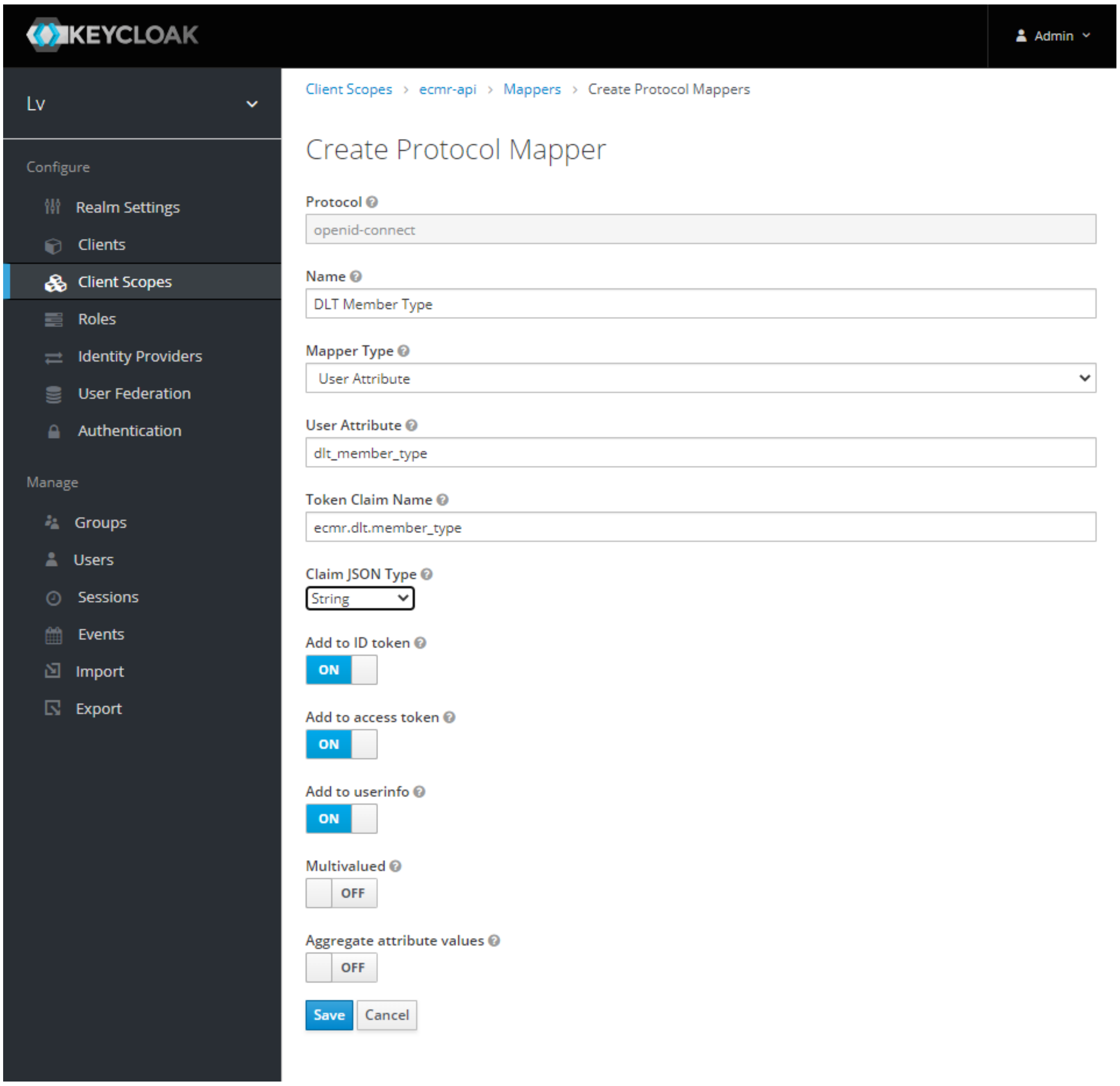
Add to access token: ON

Add to userinfo: ON

Multivalued: OFF

Aggregate attribute values: OFF

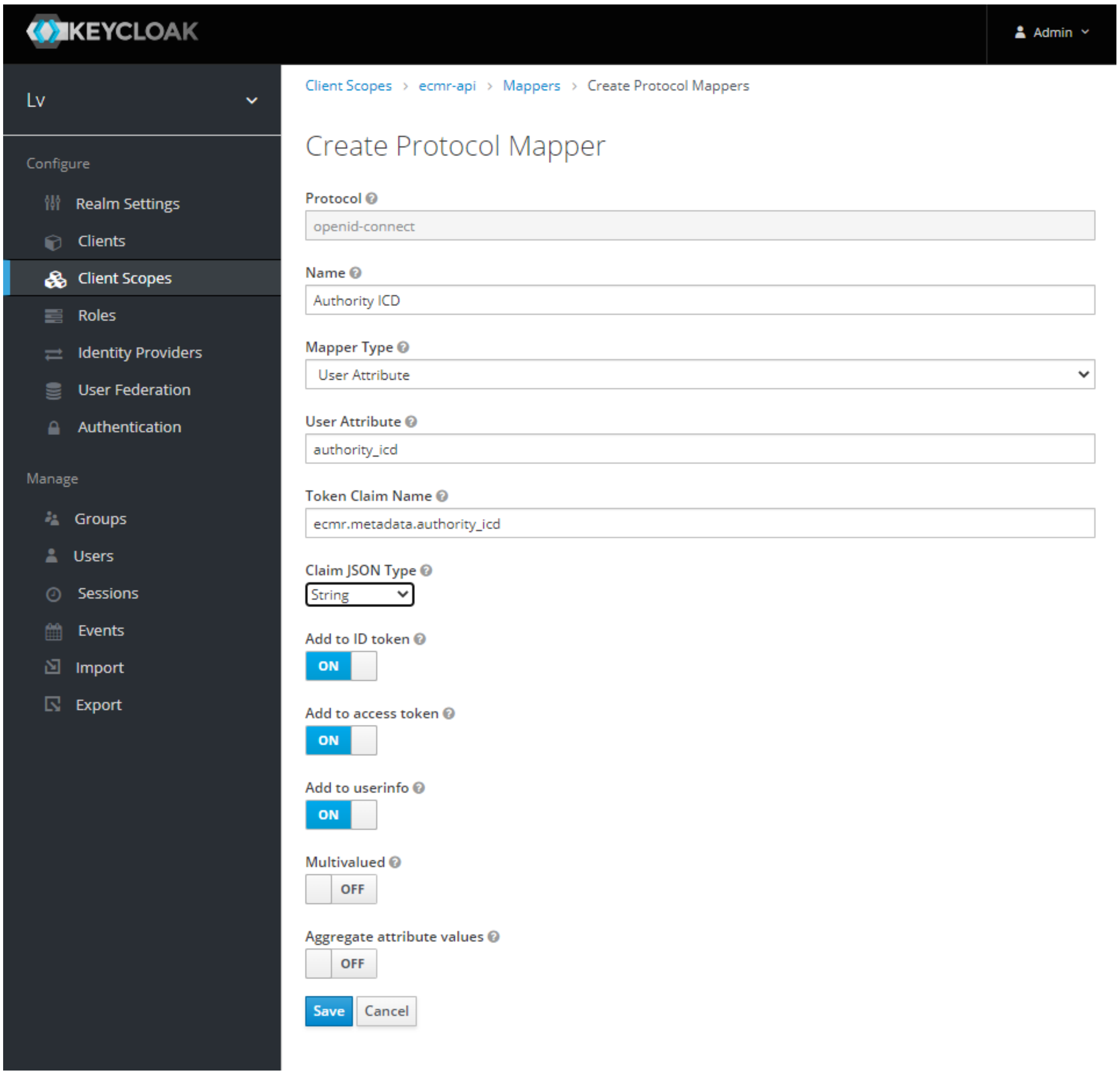
Figure 9. 'ecmr-api' client scope token mapping configuration



Create Protocol Mapper

Protocol: openid-connect
Name: DLT Member Type
Mapper Type: User Attribute
User Attribute: dlt_member_type
Token Claim Name: ecmr.dlt.member_type
Claim JSON Type: String
Add to ID token: ON
Add to access token: ON
Add to userinfo: ON
Multivalued: OFF
Aggregate attribute values: OFF

Figure 10. 'ecmr-api' client scope token mapping configuration



Create Protocol Mapper

Protocol: openid-connect

Name: Authority ICD

Mapper Type: User Attribute

User Attribute: authority_icd

Token Claim Name: ecmr.metadata.authority_icd

Claim JSON Type: String

Add to ID token: ON

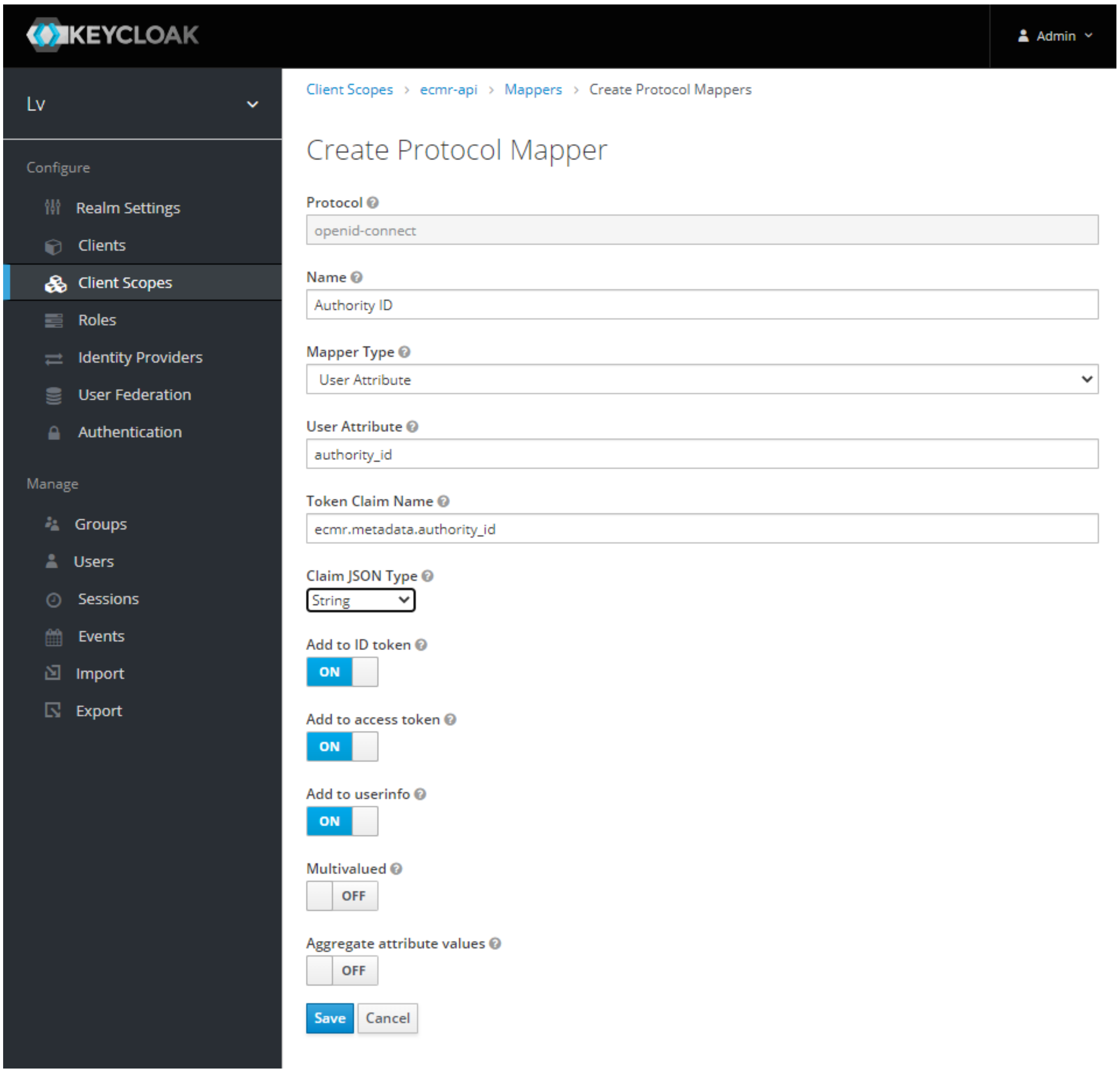
Add to access token: ON

Add to userinfo: ON

Multivalued: OFF

Aggregate attribute values: OFF

Figure 11. 'ecmr-api' client scope token mapping configuration



Create Protocol Mapper

Protocol: openid-connect

Name: Authority ID

Mapper Type: User Attribute

User Attribute: authority_id

Token Claim Name: ecmr.metadata.authority_id

Claim JSON Type: String

Add to ID token: ON

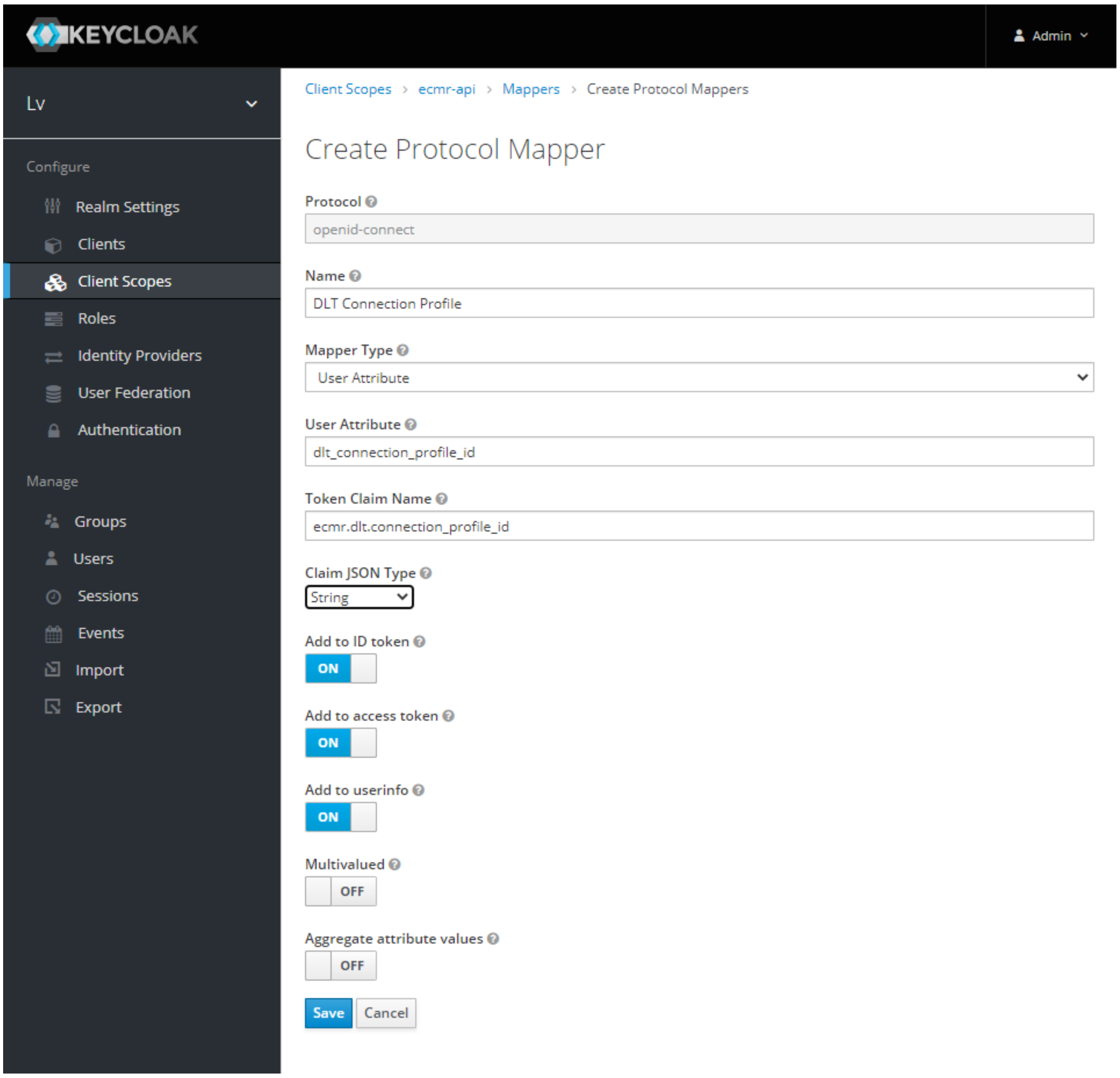
Add to access token: ON

Add to userinfo: ON

Multivalued: OFF

Aggregate attribute values: OFF

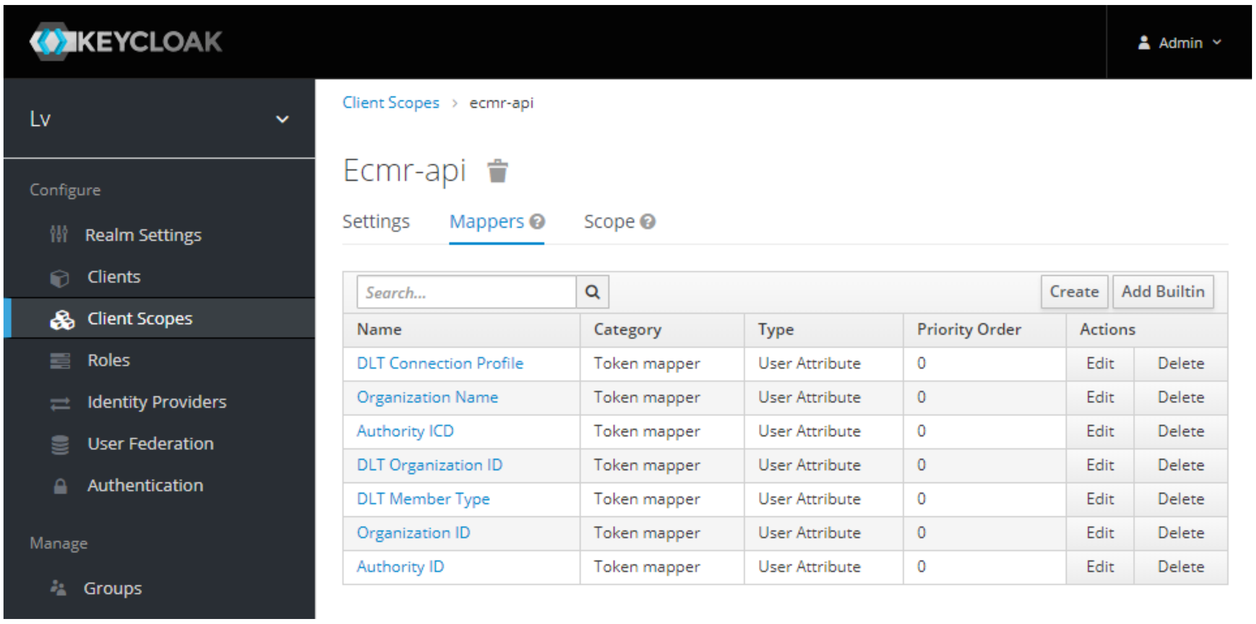
Figure 12. 'ecmr-api' client scope token mapping configuration



Create Protocol Mapper

Protocol: openid-connect
Name: DLT Connection Profile
Mapper Type: User Attribute
User Attribute: dlt_connection_profile_id
Token Claim Name: ecmr.dlt.connection_profile_id
Claim JSON Type: String
Add to ID token: ON
Add to access token: ON
Add to userinfo: ON
Multivalued: OFF
Aggregate attribute values: OFF

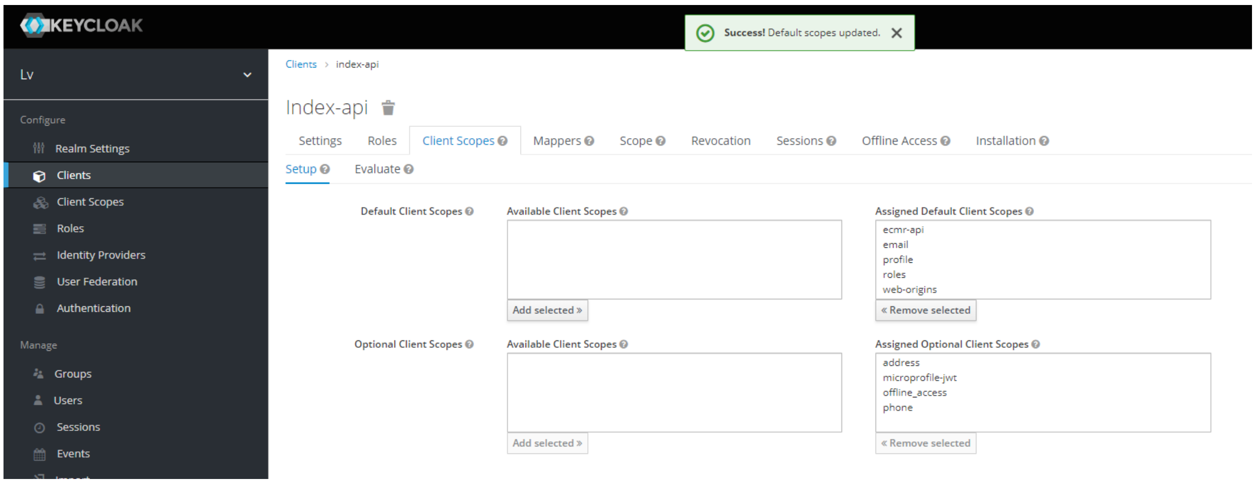
Figure 13. 'ecmr-api' client scope token mapping configuration



Name	Category	Type	Priority Order	Actions	
DLT Connection Profile	Token mapper	User Attribute	0	Edit	Delete
Organization Name	Token mapper	User Attribute	0	Edit	Delete
Authority ICD	Token mapper	User Attribute	0	Edit	Delete
DLT Organization ID	Token mapper	User Attribute	0	Edit	Delete
DLT Member Type	Token mapper	User Attribute	0	Edit	Delete
Organization ID	Token mapper	User Attribute	0	Edit	Delete
Authority ID	Token mapper	User Attribute	0	Edit	Delete

Figure 14. 'ecmr-api' client scope token mapping configuration

16.5. Set ‘ecmr-api’ scope as default client scope for created ‘index-api’ client



Success! Default scopes updated.

Default Client Scopes	Available Client Scopes	Assigned Default Client Scopes
		ecmr-api email profile roles web-origins
Optional Client Scopes	Available Client Scopes	Assigned Optional Client Scopes
		address microprofile-jwt offline_access phone

Figure 15. Setting 'ecmr-api' client scope as default for 'index-api' client

16.6. Create ‘Business’ and ‘Government’

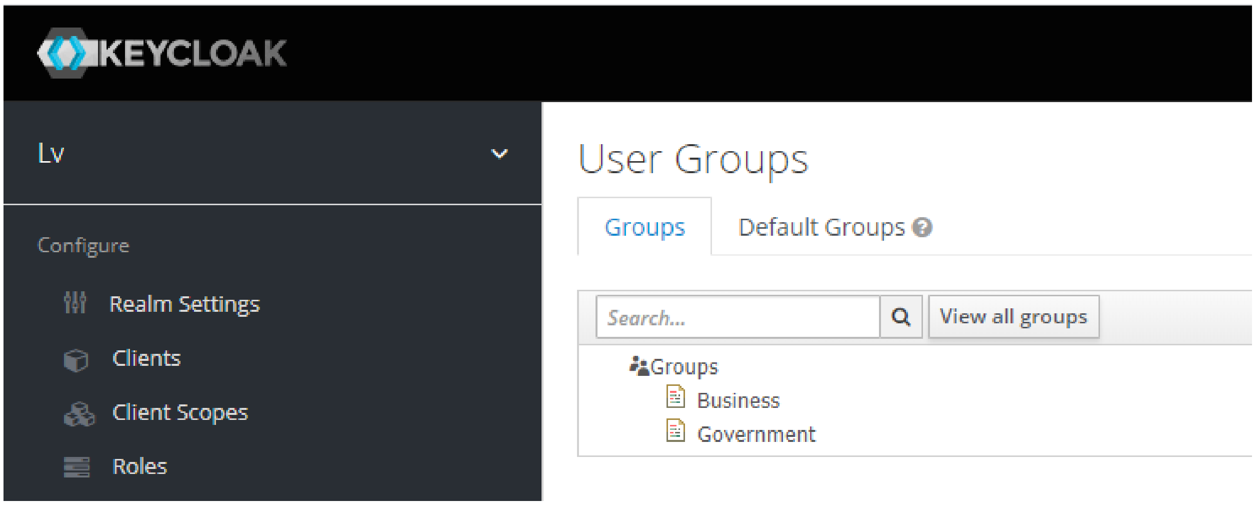
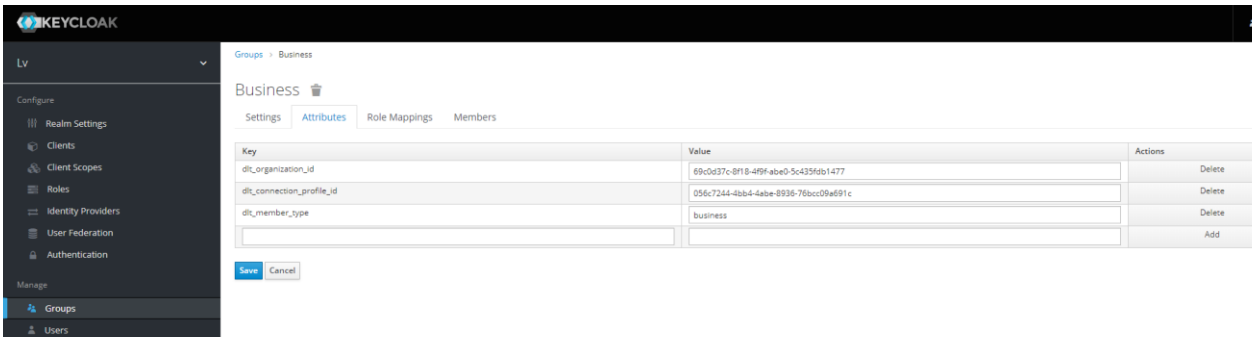


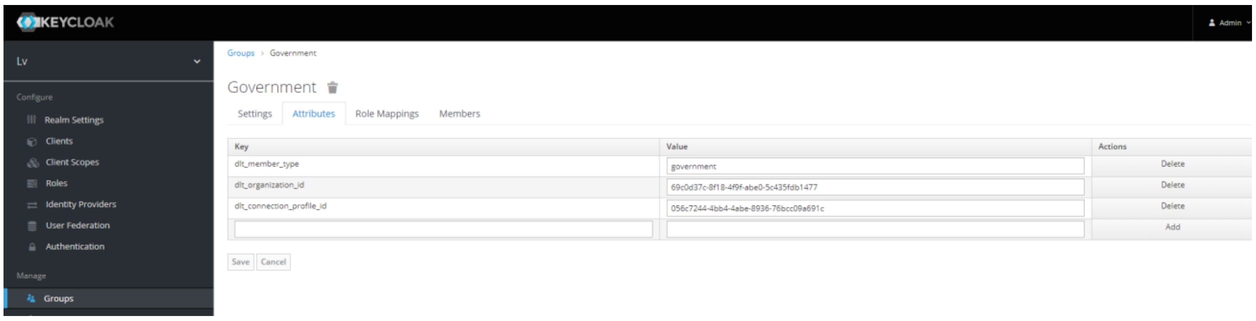
Figure 16. Creation of 'Business' and 'Government' groups

16.7. Set attributes for created groups with id's received from step '12'



Key	Value	Actions
dlt_organization_id	69c0d37c-8f18-4f9f-abe0-5c435fdb1477	Delete
dlt_connection_profile_id	056c7244-4bb4-4abe-8936-76bcc09a691c	Delete
dlt_member_type	business	Delete
		Add

Figure 17. Attribute configuration for 'Business' group



Key	Value	Actions
dlt_member_type	government	Delete
dlt_organization_id	69c0d37c-8f18-4f9f-abe0-5c435fdb1477	Delete
dlt_connection_profile_id	056c7244-4bb4-4abe-8936-76bcc09a691c	Delete
		Add

Figure 18. Attribute configuration for 'Government' group

17. After all steps we completed successfully – you can start using country API.