Introduktion til CiQ API

Senest opdateret 2018-04-16

GraphQL

CiQ API giver adgang til CiQ's PEP-data, der består af konsoliderede, udvalgte data fra forskellige datakilder, herunder flere relations-databaser, søgeindeks og grafdata.

Data udveksles primært via HTTP POST i GraphQL-formatet, der bl.a. til forskel fra de fleste REST- og Hypermedia-baserede formater har type-check og giver API-brugere mulighed for at vælge præcis det subset i retur-data, man har brug for, omdøbe navne på returværdier efter ønske, mv.

GraphQL er udviklet af Facebook, og anvendes af stadig flere større virksomheder, der udover Facebook selv bl.a. tæller Github, Pinterest, Coursera, m.fl.

Der findes masser af dokumentation, tutorials og 3.-parts biblioteker i forskellige programmeringssprog tilgængelig online, f.eks. http://graphql.org/ og https://github.com/chentsulin/awesome-graphql.

API Browser

GraphQL giver en "smart" klient mulighed for at læse API'ens fulde schema og f.eks. auto-generere funktionalitet, validering eller lignende, men der findes også en api-browser i web-applikationen, https://data.ciq.dk/#/data/ciq-browser, der kan hjælpe med at udforme de ønskede query-payloads korrekt; når man har valgt de ønskede attributter og parametre kan man vælge "Formatteret JSON request" i dropdown i øverste højre hjørne, kopiere og anvende direkte i GraphQL-queries.

Sikkerhed

Til authentication og authorization anvender CiQ API JSON Web Token (JWT), https://jwt.io/ - en simpel men sikker metode, der er illustreret under eksempler.

Eksempler

VIGTIGT!

Kun i det første GraphQL-eksempel er query-payload helt korrekt formatteret; i de efterfølgende er det re-formatteret med linieskift og indrykning for overskuelighed og læselighed, men man kan *ikke* formattere queries således i live-queries - det returnerer ganske enkelt en fejl. Hvis den korrekte formattering med korrekt whitespace og escaping volder problemer findes der en del GraphQL-værktøjer, der kan hjælpe med at udforme korrekte query-payloads, herunder CiQ's data-browser https://data.ciq.dk/#/data/ciq-browser, der tillige illustrerer en unik, indbygget feature ved GraphQL, schema introspection.

I eksemplerne herunder er eksempel/dummy-værdier indlejret i kursiv; de skal naturligvis udskiftes med korrekte bruger- eller retur-data.

Authentication

For at anvende CiQ API skal man først hente et JWT-token - her eksemplificeret med curl, men princippet er det samme for alle HTTP-klienter:

```
curl -X POST -d email="eksempel\@domaene.dk" -d password="<INDSÆT PASSWORD>" \
https://data.ciq.dk/api/v1/auth_user
```

return →

```
{
    "auth_token":"megetMegetL4ngtToken",
    "user": {
        "id":1,
        "email":"eksempel\@domaene.dk"
    }
}
```

Hvis brugernavn og password begge er korrekte finder man i det returnerede response-payload attributten auth_token, der herefter skal anvendes i headeren i samtlige GraphQL-queries.

Test: ping

Man kan herefter teste sit token, hvis man ønsker:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer megetMegetL4ngtToken" \
https://data.ciq.dk/api/v1/ping
```

return →

```
{
    "data": {
        "message":"Pong, eksempel@domaene.dk"
    }
}
```

GraphQL Query uden parametre

Hvis alt går som forventet efter test af JWT, kan man nu kalde api'en.

Her et simpelt eksempel på et query på `current_user`, først i råt, fungerende format:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer megetMegetL4ngtToken" \
```

```
-d '{
  "query": "query UserQuery { current_user { email } }",
  "variables": "",
  "operationName": "UserQuery"
}' \
https://data.ciq.dk/api/v1/
```

Og formatteret, men altså ikke anvendeligt i live-queries:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer megetMegetL4ngtToken" \
-d '{
    "query": "query UserQuery {
        current_user {
            email
        }
    }",
    "variables": "",
    "operationName": "UserQuery"
}' \
https://data.ciq.dk/api/v1/
```

return →

```
{
    "data": {
        "current_user": {
            "email":"eksempel@domaene.dk",
            "first_name": "Eksempel",
            "last_name":"Efternavn",
            "organizations":[
                {
                    "name":"Min Organisation"
                }
            ]
        }
    }
}
```

Som det fremgår skal de formatterede JSON-request, man kan udforme i api-browseren, https://data.ciq.dk/#/data/ciq-browser altså indsættes i request-body i det HTTP-request, man sender til api'en.

*Bemærk, at man **skal** vælge attributter i både overordnede og indlejrede entiteter; i ovenstående request vælges f.eks. email, first_name og last_name som direkte attributter på current_user, mens der for den indlejrede liste af organizations er valgt name.*

GraphQL query med parametre

De fleste queries kræver parametre, hvilket ikke ændrer på den overordnede struktur i det JSON-payload, man sender til endpoint; bemærk dog de escapede quotes i variabel-delen.

Her et entitets-query, der generelt anvendes til at hente detaljerede, omfattende data om en entitet udfra en global UUID:

```
curl -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer megetMegetL4ngtToken" \
-d '{
    "query": "query EntitetQuery($uuid: ID!) {
        entitet(uuid: $uuid) {
            by
            foedselsdato
            navn
            pep_kategorier {
                kode
            }
            postnr
            relations {
                navn
                pep_kategorier {
                    kode
                }
                relation_direction
                relation_type
                uuid
            }
            vejnavn
            verificeret
        }
    }",
    "variables": "{\"uuid\":\"55CFB52E-2D27-4349-A192-19AE476FFE68\"}",
    "operationName": "EntitetQuery"
}' \
https://data.ciq.dk/api/v1/
```

return →


```
{
    "data": {
        "entitet": {
            "by": "København K",
            "foedselsdato": "1964-05-15",
            "navn": "Lars Løkke Rasmussen",
            "pep_kategorier": [
```

```
        {
            "kode": "A"
        },
        {
            "kode": "B"
        },
        {
            "kode": "C"
        },
        {
            "kode": "G"
        }
    ],
    "postnr": 1056,
    "relations": [
        {
            "navn": "Statsministeriet, Departementet",
            "pep_kategorier": null,
            "relation_direction": "outgoing",
            "relation_type": "boarddirector",
            "uuid": "B118092B-D5C0-49AA-A115-361A0FC9C67A"
        },
        {
            "navn": "VENSTRES LANDSORGANISATION",
            "pep_kategorier": null,
            "relation_direction": "outgoing",
            "relation_type": "boardmember",
            "uuid": "0BCE9EC6-071A-4D30-92DE-B9DBFBFD615C"
        },
        {
            "navn": "Regeringen Lars Løkke Rasmussen III",
            "pep_kategorier": null,
            "relation_direction": "outgoing",
            "relation_type": "member",
            "uuid": "df925b51-dcc5-4bfa-8ed2-0e449cc4a059"
        },
        {
            "navn": "Folketinget",
            "pep_kategorier": null,
            "relation_direction": "outgoing",
            "relation_type": "member",
            "uuid": "11BE8DEF-8F5E-4670-AE07-5929ED700D94"
        },
        {
            "navn": "Bergur Løkke Rasmussen",
            "pep_kategorier": [
                {
                    "kode": "J"
```

```
            }
        ],
        "relation_direction": "incoming",
        "relation_type": "family",
        "uuid": "49977A2B-6554-4CCE-B910-5343F46FB54F"
    },
    {
        "navn": "Lisa Løkke Rasmussen",
        "pep_kategorier": [
            {
                "kode": "J"
            }
        ],
        "relation_direction": "incoming",
        "relation_type": "family",
        "uuid": "be42b69a-86a9-4d67-a158-c2a58de79004"
    },
    {
        "navn": "Símun Larsson Løkke Rasmussen",
        "pep_kategorier": [
            {
                "kode": "J"
            }
        ],
        "relation_direction": "incoming",
        "relation_type": "family",
        "uuid": "2837a64f-8e97-472d-913c-bc2bd0b138fa"
    },
    {
        "navn": "Sólrun Jákupsdóttir Rasmussen",
        "pep_kategorier": [
            {
                "kode": "J"
            }
        ],
        "relation_direction": "incoming",
        "relation_type": "family",
        "uuid": "5c173d33-e50d-401e-a1b1-5b0aa8b2415d"
    },
    {
        "navn": "Knud Løkke Rasmussen",
        "pep_kategorier": [
            {
                "kode": "J"
            }
        ],
        "relation_direction": "outgoing",
        "relation_type": "family",
```

```
                    "uuid": "5A53B591-A3DC-4909-B14C-38590D1C1DA5"
                }
            ],
            "vejnavn": "Heibergsgade",
            "verificeret": true
        }
    }
}
```

PEP search

PEP-søgning er en "grovkornet" fritekst-søgning blandt PEP- og Risiko-entiteter, hvor der søges i både navn, cvr-nummer og adresse - også i kombination. En søgning på "Løkke Humlebæk" kan eksempelvis se sådan ud:

```
{
    "query": "query BiqEntitetListQuery($text: String!) {
        pep_search(text: $text) {
            biq_entiteter {
                by
                foedselsdato
                navn
                pep_kategorier {
                    kode
                }
                postnr
                risikokategorier {
                    kode
                }
                uuid
                verificeret
            }
        }
    }",
    "variables": "{\"text\":\"Løkke Humlebæk\"}",
    "operationName": "BiqEntitetListQuery"
}
```

return →

```
{
    "data": {
        "pep_search": {
            "biq_entiteter": [
                {
                    "by": "Humlebæk",
                    "foedselsdato": "1968-10-08",
                    "navn": "Knud Løkke Rasmussen",
```

```
                    "pep_kategorier": [
                        {
                            "kode": "J"
                        }
                    ],
                    "postnr": 3050,
                    "risikokategorier": [],
                    "uuid": "5A53B591-A3DC-4909-B14C-38590D1C1DA5",
                    "verificeret": false
                }
            ]
        }
    }
}
```


PEP-søgning returnerer som nævnt både entiteter i PEP- og Risiko-kategorierne.

Ønsker man mere omfattende data end der er tilgængeligt i søgeresultatet hentes det via UUID i et entitets-query som ovenstående.

Søgning på ultimative ejere

Hvor PEP-søgningen kun søger i entiteter, der er markeret som PEP eller Risiko, søger et `owner_search` query i hele BiQ's data-grundlag og returnerer derfor potentielt endnu flere resultater:

```
{
    "query": "query HalEntitetListQuery($text: String!) {
        owner_search(text: $text) {
            hal_entiteter {
                name
                uuid
            }
        }
    }",
    "variables": "{\"text\":\"Løkke Humlebæk\"}",
    "operationName": "HalEntitetListQuery"
}
```

return →


```
{
    "data": {
        "owner_search": {
            "hal_entiteter": [
                {
                    "name": "Carolina Løkke Christina Pedersen",
                    "uuid": "69C1B397-76B1-4FAB-93A0-8A7375A41628"
                },
```

```
                {
                    "name": "Knud Løkke Rasmussen",
                    "uuid": "5A53B591-A3DC-4909-B14C-38590D1C1DA5"
                }
            ]
        }
    }
}
```

I sammenhæng med søgning på PEP- og Risiko-data for personer giver en PEP- eller Match-søgning, der er beskrevet herunder derfor ofte umiddelbart bedre resultater, men hvis man eksempelvis ønsker at undersøge ejerkredsen på en virksomhed er en søgning i BiQ's fulde data-grundlag særdeles anvendelig. En søgning på CVR-nummer giver altid et resultat på 0..1:

```
{
    "query": "query HalEntitetListQuery($text: String!) {
        owner_search(text: $text) {
            hal_entiteter {
                excerpt
                name
                type
                uuid
            }
        }
    }",
  "variables": "{\"text\":\"25729013\"}",
  "operationName": "HalEntitetListQuery"
}
```

return →

```
{
    "data": {
        "owner_search": {
            "hal_entiteter": [
                {
                    "excerpt": "CVR: 25729013, direktør: Rasmus Philip Rask",
                    "name": "BIQ ApS",
                    "type": "companies",
                    "uuid": "68DE6B51-2A4B-416A-8442-B25BF76DC9EA"
                }
            ]
        }
    }
}
```

Også her gælder det, at ønsker man mere omfattende data end der er tilgængeligt i søgeresultatet hentes det via UUID i et entitets-query som ovenstående.

Match search

Match-søgning er en mere avanceret, finkornet søgning end den pep-søgning, der kun anvender en enkelt søgestreng. Søgningen prioriterer de forskellig parametre i en forudbestemt "væsentligheds-rækkefølge" og returnerer eventuelle resultater, der hver er påhæftet en vægtning, samt hvilken type søgning, der er anvendt. I den nuværende version, pr. januar 2018, kan søges på en kombination af

- navn
- vejnavn
- husnr
- postnr
- foedselsdato

hvor navn *altid* er påkrævet, og *skal* kombineres med enten `fodselsdato` eller fuld adresse.

Eksempelvis kan et payload til søgning på "Lars Løkke Rasmussen" som navn og "1964-05-15" som foedselsdato inklusive udvalgte felter se sådan ud:

```
{
    "query": "query MatchListQuery($navn: String!, $foedselsdato: String) {
        match_search(navn: $navn, foedselsdato: $foedselsdato) {
            matches {
                match {
                    navn
                    pep_kategorier {
                        beskrivelse
                        kode
                    }
                    postnr
                    uuid
                }
                match_type
                weight
            }
        }
    }",
    "variables": "{
        \"navn\":\"Lars Løkke Rasmussen\",
        \"foedselsdato\":\"1964-05-15\"
    }",
    "operationName": "MatchListQuery"
}
```

Hvilket vil returnere følgende (rå) data:

```
{
    "data": {
```

```
        "match_search": {
            "matches": [
                {
                    "match": {
                        "navn": "Lars Løkke Rasmussen",
                        "pep_kategorier": [
                            {
                                "beskrivelse": "Statschef, regeringschef, minister
og viceminister eller assisterende minister",
                                "kode": "A"
                            },
                            {
                                "beskrivelse": "Parlamentsmedlem eller medlem af
tilsvarende lovgivende organer",
                                "kode": "B"
                            },
                            {
                                "beskrivelse": "Medlem af politisk partis
styrelsesorgan",
                                "kode": "C"
                            },
                            {
                                "beskrivelse": "Medlem af statsejet virksomheds
administrative, ledende eller kontrollerende organ",
                                "kode": "G"
                            }
                        ],
                        "postnr": 1056,
                        "uuid": "55CFB52E-2D27-4349-A192-19AE476FFE68"
                    },
                    "match_type": "d_exact_name_birthday",
                    "weight": 1.0
                }
            ]
        }
    }
}
```


Som ved de øvrige former for søgning gælder også her, at mere omfattende data end der er tilgængeligt i søgeresultatet hentes det via UUID i et efterfølgende entitets-query som ovenstående.

Total-dump

Det er muligt at hente total-dumps af CiQ's PEP- og Risiko-entiteter i tab-separeret format, og selv stå for matching op imod egne data.

Man kan vælge mellem tre forskellige "modes" i dumps, samt to forskellige versioner af retur-kolonner:

Modes

- **pep**, der kun returnerer entiteter, der har en eller flere PEP-kategorier - det er den der er valgt i eksemplet nedenfor
- **risiko**, der kun returnerer entiteter, der har en eller flere Risiko-kategorier
- **full**, der vælger begge ovenstående og desuden er default hvis man ikke vælger mode

Versioner

- **v1**, der indeholder kolonnerne `id uuid navn cvr aws_id vejnavn husnr etage side_doer_nr postnr by land sha created_at updated_at pep_kategorier risikokategorier` - af "legacy"-årsager uden `foedselsdato` og `verificeret`. Det er nødvendigt, specifikt at angive `v=v1` som URL-parameter for at få det begrænsede sæt - det er eksemplificeret nedenfor.
- **v2**, der indeholder det fulde kolonne-dump: `id uuid navn cvr aws_id vejnavn husnr etage side_doer_nr postnr by land sha created_at updated_at foedselsdato verificeret pep_kategorier risikokategorier`. **v2** er default og behøves ikke angivet i URL-parameter.

Selve API-kaldet til total-dumps ser således ud:

```
curl -O -X GET \
-H "Authorization: Bearer megetMegetL4angtToken" \
https://data.ciq.dk/api/v1/dump?v=v1&mode=pep
```

→ (tab-separet pep-data)

Bemærk, at som ved kald til GraphQL-endpointet kræves også her kræves authorization og authentication via et JSON Web Token i en HTTP-header.

Værktøjer til hjælp i forbindelse med test og udvikling

CiQ's udviklingsafdeling anvender selv pt. en desktop HTTP-klient, der hedder Insomnia til test af api-kald, https://insomnia.rest/, og har lavet en workspace-profil, der indeholder masser af eksempler på api-kald. Profilen skal blot importeres i Insomnia og tilrettes; først retter man email og password i det POST request, der hedder CiQ Login, sender det afsted og kopierer herefter auth_token fra response til Authorization-headeren - husk at det skal indeholde "Bearer" og mellemrum således, `Authorization: Bearer <auth_token>`, ellers fungerer det ikke.