How to obtain an access token via oidc-agent

Corso di formazione “Panoramica su OAuth2/OpenID Connect e sue applicazioni tramite il servizio INDIGO IAM”, 12-14 Maggio 2025, LNF

Roberta Miccoli, INFN CNAF

oidc-agent

- oidc-agent is a set of tools to manage OpenID Connect tokens and make them easily usable from the command line
  - It follows the ssh-agent design
- Source code (developed by the KIT team)
- Documentation

Installation

- oidc-agent is directly available for some distributions
  - the newest packages are available for a wide range of different distribution at https://repo.data.kit.edu/
- Download the repository key to a location writable only by root

```
$ curl https://repo.data.kit.edu/repo-data-kit-edu-key.asc | sudo tee
/etc/apt/trusted.gpg.d/kitrepo-archive.asc
```

Debian/Ubuntu assumed here!

- Add one of the supported repos to your `/etc/apt/sources.list`

```
$ echo "deb [signed-by=/etc/apt/trusted.gpg.d/kit-data-repo.asc] https://repo.data.kit.edu/debian/ stable
./" | sudo tee /etc/apt/sources.list.d/kit-data-repo.list
```

- Update and install oidc-agent

```
$ sudo apt update
$ sudo apt install oidc-agent
```

Check the oidc-agent version

```
$ oidc-agent -V
oidc-agent 5.2.3
```

What oidc-agent does during client registration

In order to request tokens, you firstly have to register a Client.
With oidc-agent you have to run

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
```

The OAuth authorization flow used here is **device**. `oidc-agent` allows you to specify one of code, device, password and refresh. Default to code.
The `oidc-gen -w device` command basically replaces the curl version reported in the *Device code flow* slides

Device code flow is **required** when commands are typed in a UI

What oidc-agent does during client registration

In order to request tokens, you firstly have to register a Client.
With oidc-agent you have to run

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/

Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 2
```

Select the AS where you want to register the Client.
I choose IAM dev here

What oidc-agent does during client registration

In order to request tokens, you firstly have to register a Client.
With oidc-agent you have to run

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 1
The following scopes are supported: openid profile email offline_access wlcg wlcg.groups storage.read:/ storage.create:/
compute.read compute.modify compute.create compute.cancel storage.modify:/ eduperson_scoped_affiliation eduperson_entitlement
eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: storage.read:/ storage.create:/ compute.read compute.modify
```

List of supported scopes is taken from the `/.well-known/openid-configuration` endpoint of the AS. Insert the necessary scopes only

What oidc-agent does during client registration

In order to request tokens, you firstly have to register a Client.
With oidc-agent you have to run

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 1
The following scopes are supported: openid profile email offline_access wlcg wlcg.groups storage.read:/ storage.create:/
compute.read compute.modify compute.create compute.cancel storage.modify:/ eduperson_scoped_affiliation eduperson_entitlement
eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: storage.read:/ storage.create:/ compute.read compute.modify
Registering Client ...
Generating account configuration ...
accepted
```

The `oidc-agent` Client has been registered in the IAM dev. The configuration details have been saved locally

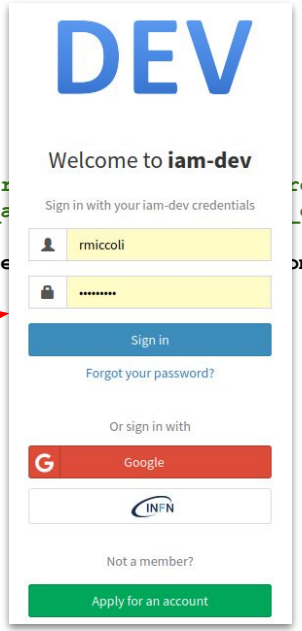
What oidc-agent does during client registration

In order to request tokens even when the AT is expired, you have to obtain a refresh token

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 2
The following scopes are supported: openid profile email offline_access wlcg wlcg.groups stor                    reate:/
compute.read compute.modify compute.create compute.cancel storage.modify:/ eduperson_scoped_a                    _entitlement
eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: storage.read:/ storage.cre                    ompute.modify
Registering Client ...
Generating account configuration ...
accepted

Using a browser on any device, visit:
https://iam-dev.cloud.cnaf.infn.it/device

And enter the code: LYE0TT
```



The AS asks the user to authenticate before entering the code

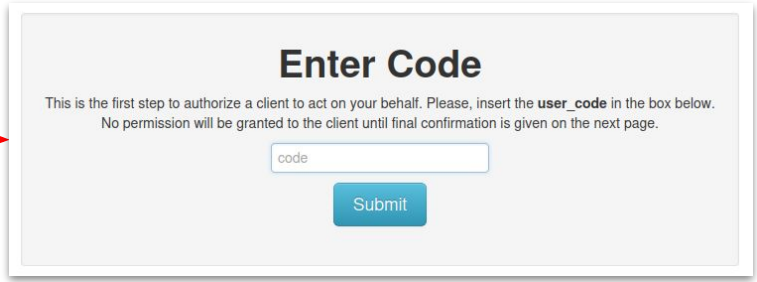
What oidc-agent does during client registration

In order to request tokens even when the AT is expired, you have to obtain a refresh token

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 2
The following scopes are supported: openid profile email offline_access wlcg wlcg.groups storage.read:/ storage.create:/
compute.read compute.modify compute.create compute.cancel storage.modify:/ eduperson_scoped_affiliation eduperson_entitlement
eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: storage.read:/ storage.create:/ compute.read compute.modify
Registering Client ...
Generating account configuration ...
accepted

Using a browser on any device, visit:
https://iam-dev.cloud.cnaf.infn.it/device

And enter the code: LYE0TT
```



What oidc-agent does during client registration

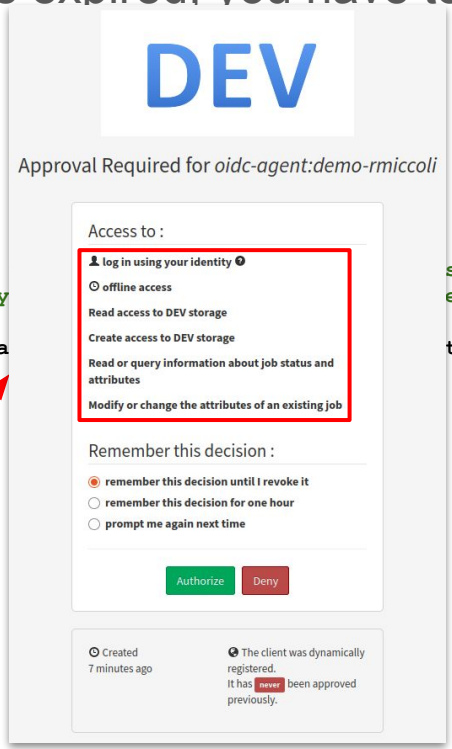
In order to request tokens even when the AT is expired, you have to obtain a refresh token

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 2
The following scopes are supported: openid profile email offline_access                                    storage.create:/
compute.read compute.modify compute.create compute.cancel storage.modify                                   eduperson_entitlement
eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: stora                                    te.read compute.modify
Registering Client ...
Generating account configuration ...
accepted

Using a browser on any device, visit:
https://iam-dev.cloud.cnaf.infn.it/device

And enter the code: LYE0TT
```

The AS asks for the consent of the user to access the information linked to the requested scopes



What oidc-agent does during client registration

In order to request tokens even when the AT is expired, you have to obtain a refresh token

```
$ eval $(oidc-agent)
$ oidc-gen -w device demo
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: 2
The following scopes are supported: openid profile email offline_access wlcg wlcg.groups storage.read:/ storage.create:/
compute.read compute.modify compute.create compute.cancel storage.modify:/ eduperson_scoped_affiliation eduperson_entitlement
eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: storage.read:/ storage.create:/ compute.read compute.modify
Registering Client ...
Generating account configuration ...
accepted

Using a browser on any device, visit:
https://iam-dev.cloud.cnaf.infn.it/device

And enter the code: LYE0TT

[ Polling the device code verification from the https://iam-dev.cloud.cnaf.infn.it/device/approve endpoint ]

Enter encryption password for account configuration 'demo': ***
Confirm encryption Password: ***
Everything setup correctly!
```

oidc-agent Client configuration

- Local oidc-agent Client configurations are saved in `~/.config/oidc-agent` or `~/.oidc-agent`
- In this example, `demo` is an alias for the oidc-agent Client
- The Client configuration is encrypted using the password set by the user during Client registration

```
$ cat ~/.config/oidc-agent/demo
1038
J37J7SrGe8cawpuaLqSky1Ny6EBqQd+S
NvFOQhh4GDmH4b4oECLJfg==
24:16:16:32:1:2:67108864:2
qEysXtC37u5b4hxdEStDnfCX5mMs6xrXCtdkXkjCkAvflEyr3RRakTjqxnUv5T6kly9v7mgRfhg/XMPkwY0XSwOO6KyM7vbe2id6spD6sFU+hsZQtGgs/9zBb+d5hfLTis
xI5/CvFJ8w4t71rigpfaZPl5B5j0+9NCE4xPTZc9C+G5UrgtX7pa92hMCwQnsn/WgCrlxA6Z9AO/Npt9aTUr84KTyHzg+1j3PcajGICgrJB+ov6F937cW1j67DKUidoG32
xqO4C9v4rVJiukKZ7kmdRiFnvp9IerrgIPaiYYWhpeLsF2NdjzDfCs13ArC86Ye1pzYeSSf/QbCaghHYcjQnWL7Iea4QLNS6pQ0oLoDefv7uIY0JDmQk+l7IYo9wwphFls
S7uV9AW4rGhOeDmhHlTE7G8MnRp1NaOWA2g3p9p4aweR3tSBhefLOwbw9WyeHGsMWUpf2Z7umoqiY8/wDgF2L1188iBKYouYkqobckLeKAVRuz0Qk6aSsVuTFcQDw0Bv5n
NpMbzQdzqNnHy5O66f2xTfmQDqi27n3toMnLu/RF6aWn4XPvSbblNapeeCfDtNwVNoXEMuokN0H0SXvVG7JCHj/YunprQ0Mbk4ZKZCZ2KpU9Lia4uAyWDmdr3qnKN6VVqi
PejDzBWuajnFqRW40zwAZSilMQ5HT/WydbONnlMD5XhoxdgNoes83Htn9nZa3dJ4XV5CfJjrUcNswD5P9g60kT6XdiD1/mVOV8A4CR12BCyl+Y9JA0HPweTzKzkiTyplDs
cMjhY3w9F1Cl5I7TwvXzN60XJmVOBr3/1RoMHUzlraZfAMEPULLGjUUqybTYcTxNKq3niINr4/6IDgvoybmIHiQHLaEcPAbf4E/zmINQuhcraLK5jAbFNQqUOIfizifD5N
dW/pCPWx5WxD6l6WiKR5gHRvCsDoZK8ouB03dyseGLZYvlgzEeQ3DkI7S25cT2hmCVSMCHaWeH4QSb5PxeVTYxzGMJd3Ugb7iY280JmvHd9lqzgd/Jhn6M38uL811p26F1
WH8XMvL6t4mego0qlUyMURAw0IdcZoQXzFvqQ/ATYH+zc5xJO72476o6ByjHz3dJNK5pVgGmUB9d0TDa8OCrB/AT0baBv9E2TeJyn9NwyhONTvv1VABhkp/Hl7nh2XUG2+
5Kp2MdXXfe1amTR4pBPkO2e3mjC4+7QH737olFoIngLUyzqKzay1atnAvNRaPs5evsuu9UxtZWdrhqzXSRaX1oW9B3Hgv0QPZRmOd3u6dbxOEuXwnUCNB4MUmskuo6oFcK
HChGBynNgLblz48w9FXaS0hYtOI8krk8A1+NsllIZg2i+pwkHeSS5gkm1WiRYaKEiuCWxHXYO4hI29wCvDgF
QOiU4pGrhcyK4K6olsK9tJs70H+ZmGeLE9Gp4w58WMc=
Generated using version: 5.2.3
```

oidc-agent Client configuration

- The decrypted oidc-agent configuration can be checked with `oidc-add -p` command
- The encryption pwd is required. It can be saved in an env variable or a file and passed to the agent using one of the `pw-cmd`/`pw-env`/`pw-file` option (useful for automated environment)

```
$ oidc-add -p demo
Enter decryption password for account config 'demo': ***
{
        "name":  "demo",
        "client_name":    "oidc-agent:demo-rmiccoli",
        "issuer_url":     "https://iam-dev.cloud.cnaf.infn.it/",
        "mytoken_url":    "",
        "config_endpoint":"https://iam-dev.cloud.cnaf.infn.it/.well-known/openid-configuration",
        "device_authorization_endpoint":    "https://iam-dev.cloud.cnaf.infn.it/devicecode",
        "daeSetByUser":   0,
        "client_id":      "08353bd6-72a9-4ba6-9de2-7583ad326702",
        "client_secret":  "XXX",
        "refresh_token":  "eyJhbGciOiJub251In0.eyJleHAiOjE3NDkwMzI3MjYsImp0aS…4M2Q5LWMxNDNhMDUwZmNmZSJ9.",
        "cert_path":      "",
        "auth_scope":     "storage.read:/ storage.create:/ compute.read compute.modify openid offline_access",
        "refresh_scope":  "compute.read storage.read:/ compute.modify storage.create:/ openid offline_access",
        "audience":       "",
        "oauth": 0,
        "uses_pub_client":0,
        "mytoken_profile":{
        },
        "redirect_uris":  ["edu.kit.data.oidc-agent:/redirect", "http://localhost:8080", "http://localhost:27491", "http://localhost:4242"],
        "username":       "",
        "password":       ""
}
```

oidc-agent Client configuration

```
$ oidc-add -p demo
Enter decryption password for account config 'demo': ***
{
        "name":"demo",
        "client_name":"oidc-agent:demo-rmiccoli",
        "issuer_url": "https://iam-dev.cloud.cnaf.infn.it/",
        "mytoken_url":"",
        "config_endpoint":    "https://iam-dev.cloud.cnaf.infn.it/.well-known/openid-configuration",
        "device_authorization_endpoint":    "https://iam-dev.cloud.cnaf.infn.it/devicecode",
        "daeSetByUser":       0,
        "client_id":   "08353bd6-72a9-4ba6-9de2-7583ad326702",
        "client_secret":      "XXX",
        "refresh_token":      "eyJhbGciOiJub25lIn0.eyJleHAiOjE3NDkwMzI3MjYsImp0aS…4M2Q5LWMxNDNhMDUwZmNmZSJ9.",
        "cert_path":   "",
        "auth_scope": "storage.read:/ storage.create:/ compute.read compute.modify openid offline_access",
        "refresh_scope":      "compute.read storage.read:/ compute.modify storage.create:/ openid offline_access",
        "audience":    "",
        "oauth":       0,
        "uses_pub_client":    0,
        "mytoken_profile":    {
        },
        "redirect_uris":      ["edu.kit.data.oidc-agent:/redirect", "http://localhost:8080", "http://localhost:27491",
        "http://localhost:4242"],
        "username":    "",
        "password":    ""
}
```

Even if not requested by the user, oidc-agent adds the `openid` and `offline_access` scopes during the token request. This triggers the AS to issue an ID token and a refresh token; the latter is stored by oidc-agent

What oidc-agent does when requesting a token

- When a user wants to obtain an AT with the `oidc-token` command, the RT stored during the Client registration is used
  - oidc-agent triggers an OAuth **refresh token flow**
- No need to re-run `oidc-gen` before. Just start the agent (`eval $(oidc-agent)`) and load the Client configuration (`oidc-add <client-alias>`) in case a new session is started
- Limit the scopes requested for your token as much as possible. The `oidc-token` command without arguments will request all the scopes allowed by your Client

```
$ oidc-token -s storage.read:/myPath demo | cut -d. -f2 | base64 -d 2>/dev/null | jq
{
  "wlcg.ver": "1.0",
  "sub": "8b7b42fd-0e42-43c5-8254-729aa8f6a12d",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746441308,
  "scope": "storage.read:/myPath",
  "iss": "https://iam-dev.cloud.cnaf.infn.it/",
  "exp": 1746442508,
  "iat": 1746441308,
  "jti": "d6f1cc78-8ec7-48ce-aed6-3eabf1cc2022",
  "client_id": "08353bd6-72a9-4ba6-9de2-7583ad326702"
}
```

My uuid on the IAM dev instance

client_id of the `demo` client

Token request with curl command

The same token request can be performed via `curl` using the command learnt:

```
$ curl -s -L -u ${CLIENT_ID}:${CLIENT_SECRET} -d grant_type=refresh_token -d
scope=storage.read:/myPath -d refresh_token=${REFRESH_TOKEN}
https://iam-dev.cloud.cnaf.infn.it/token | jq .access_token | tr -d '"' | cut -d. -f2 | base64
-d 2>/dev/null | jq
{
  "wlcg.ver": "1.0",
  "sub": "8b7b42fd-0e42-43c5-8254-729aa8f6a12d",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746441308,
  "scope": "storage.read:/myPath",
  "iss": "https://iam-dev.cloud.cnaf.infn.it/",
  "exp": 1746442508,
  "iat": 1746441308,
  "jti": "d6f1cc78-8ec7-48ce-aed6-3eabf1cc2022",
  "client_id": "08353bd6-72a9-4ba6-9de2-7583ad326702"
}
```

My uuid on the IAM dev instance

client_id of the `demo` client

Requesting a new refresh token

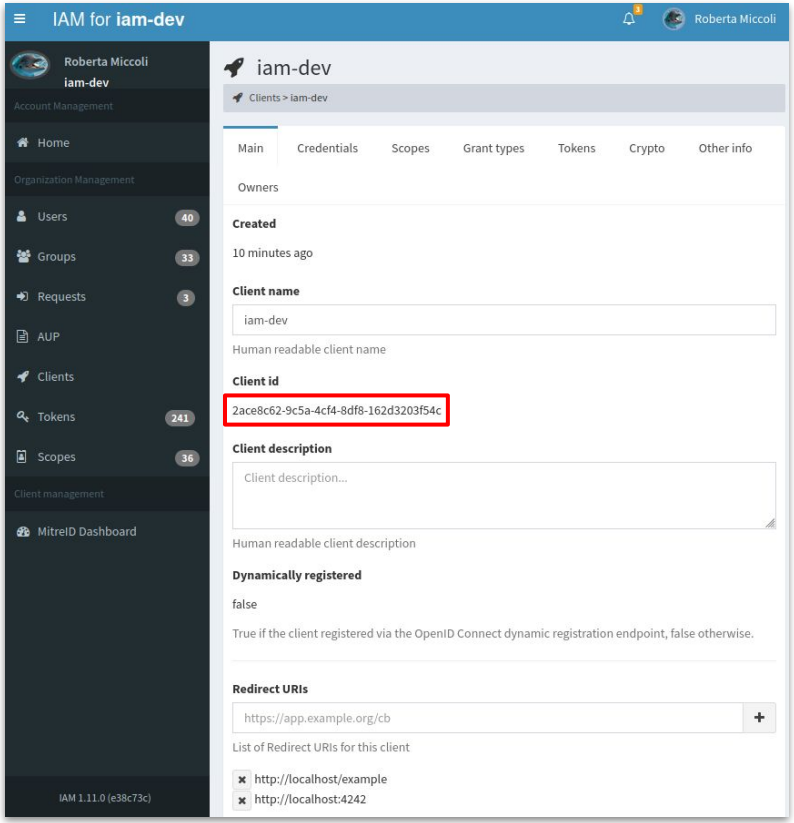
- In INDIGO IAM, the default validity period of a refresh token is **30 days**
  - after this period, the refresh token expires and must be renewed to continue obtaining access tokens
- To generate a new refresh token and update the local `oidc-agent` configuration, run

  ```
  $ oidc-gen --reauthenticate -w device demo
  ```

  This command initiates a reauthentication flow and replaces the expired refresh token in the local configuration

Importing a registered Client into oidc-agent

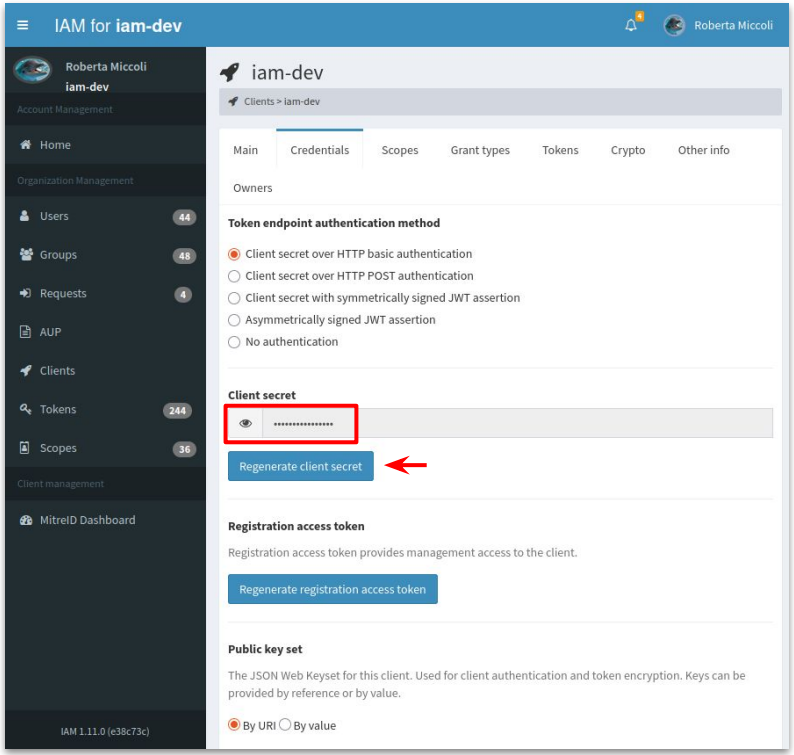
```
$ oidc-gen --manual -w device iam-dev
No account exists with this short name. Creating new
configuration …
[1] http://localhost:8080/
[2] https://iam-dev.cloud.cnaf.infn.it/
Issuer [https://iam-dev.cloud.cnaf.infn.it/]:
Client id: 2ace8c62-9c5a-4cf4-8df8-162d3203f54c
```



Importing a registered Client into oidc-agent

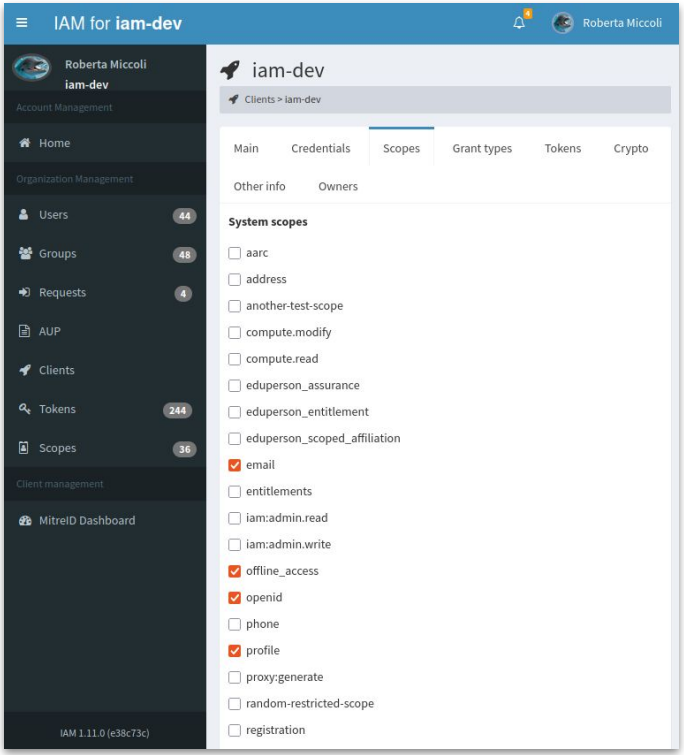
```
$ oidc-gen --manual -w device iam-dev
No account exists with this short name. Creating new
configuration …
[1] http://localhost:8080/
[2] https://iam-dev.cloud.cnaf.infn.it/
Issuer [https://iam-dev.cloud.cnaf.infn.it/]:
Client id: 2ace8c62-9c5a-4cf4-8df8-162d3203f54c
Client secret: <insert the client secret>
```

In upcoming IAM releases, the client secret will no longer be visible in the dashboard. It will only be shown at registration time or when regenerated via the dedicated button.



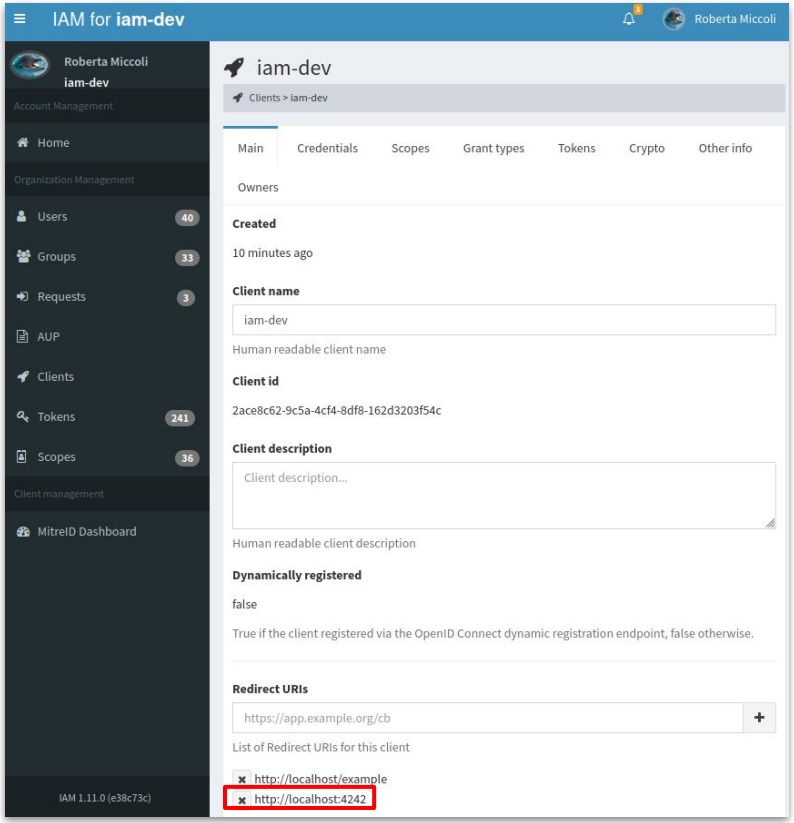
Importing a registered Client into oidc-agent

```
$ oidc-gen --manual iam-dev
No account exists with this short name. Creating new
configuration …
[1] http://localhost:8080/
[2] https://iam-dev.cloud.cnaf.infn.it/
Issuer [https://iam-dev.cloud.cnaf.infn.it/]:
Client id: 2ace8c62-9c5a-4cf4-8df8-162d3203f54c
Client secret: <insert the client secret>
The following scopes are supported: openid profile email
address phone offline_access eduperson_scoped_affiliation
eduperson_entitlement wlcg wlcg.groups storage.read:/ aarc
eduperson_assurance entitlements storage.create:/
storage.modify:/ storage.stage:/ compute.read
compute.modify
Scopes or 'max' (space separated) [openid profile
offline_access]:
```



Importing a registered Client into oidc-agent

```
$ oidc-gen --manual iam-dev
No account exists with this short name. Creating new
configuration …
[1] http://localhost:8080/
[2] https://iam-dev.cloud.cnaf.infn.it/
Issuer [https://iam-dev.cloud.cnaf.infn.it/]:
Client id: 2ace8c62-9c5a-4cf4-8df8-162d3203f54c
Client secret: <insert the client secret>
The following scopes are supported: openid profile email
address phone offline_access eduperson_scoped_affiliation
eduperson_entitlement wlcg wlcg.groups storage.read:/ aarc
eduperson_assurance entitlements storage.create:/
storage.modify:/ storage.stage:/ compute.read
compute.modify
Scopes or 'max' (space separated) [openid profile
offline_access]:
Redirect_uris (space separated): http://localhost:4242
Generating account configuration …
accepted
```



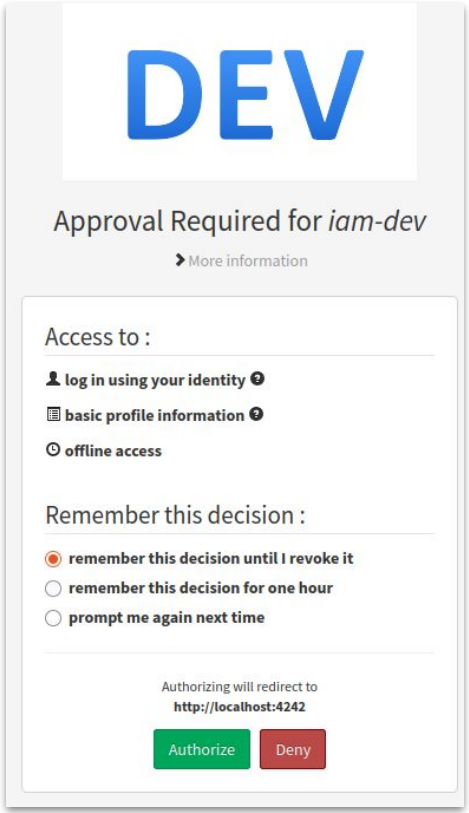
Importing a registered Client into oidc-agent

```
Using a browser on any device, visit:
https://iam-dev.cloud.cnaf.infn.it/device

And enter the code: T-CHV6

[ Polling the device code verification from the
https://iam-dev.cloud.cnaf.infn.it/device/approve endpoint ]

Enter encryption password for account configuration 'iam-dev': ***
Confirm encryption password: ***
Everything setup correctly!
```



Importing a registered Client into oidc-agent

Check that the IAM Client has been imported

```
$ oidc-add -p iam-dev
{
      "name":  "iam-dev",
      "client_name":  "oidc-agent:iam-dev-rmiccoli",
      "issuer_url":   "https://iam-dev.cloud.cnaf.infn.it/",
      "mytoken_url":  "",
      "config_endpoint":      "https://iam-dev.cloud.cnaf.infn.it/.well-known/openid-configuration",
      "device_authorization_endpoint":       "https://iam-dev.cloud.cnaf.infn.it/devicecode",
      "daeSetByUser": 0,
      "client_id":    "2ace8c62-9c5a-4cf4-8df8-162d3203f54c",
      "client_secret":"XXX",
      "refresh_token":"eyJhbGciOiJub25lIn0.eyJleHAiOjE3NDkwNTIzNDYsImp0aSI6ImZmZWQ4MztNXXX.",
      "cert_path":    "",
      "auth_scope":   "openid profile offline_access",
      "refresh_scope":"openid profile offline_access",
      "audience":     "",
      "oauth": 0,
      "uses_pub_client":      0,
      "mytoken_profile":      {
      },
      "redirect_uris":["http://localhost:4242"],
      "username":     "",
      "password":     ""

}
```

Importing a registered Client into oidc-agent

You can avoid the interactive interface of `--manual` (or `-m`) by directly using command line options with `oidc-gen`

```
$ oidc-gen -w device \
  --client-id=2ace8c62-9c5a-4cf4-8df8-162d3203f54c \
  --client-secret=XXX \
  --issuer=https://iam-dev.cloud.cnaf.infn.it/ \
  --redirect-uri=http://localhost:4242 \
  --scope="openid profile offline_access" \
  iam-dev
```

Updating oidc-agent Client configuration

If you request an access token with a scope that is allowed by the Client but was not granted during the OAuth authorization flow, you will receive an *up-scoping error*

```
$ oidc-token -s phone iam-dev
Error: invalid_scope: Up-scoping is not allowed.
We cannot get these scopes with the current configuration. To get these scopes you might need to adapt the
account configuration with
    $ oidc-gen -m iam-dev
but it also might be necessary to change the client configuration with the OpenID provider.
```

Updating oidc-agent Client configuration

```
$ oidc-gen -m -w device iam-dev
Enter decryption password for account config 'iam-dev':
Client id [2ace8c62-9c5a-4cf4-8df8-162d3203f54c]:
Client secret [***]:
The following scopes are supported: openid profile email address phone offline_access
eduperson_scoped_affiliation eduperson_entitlement wlcg wlcg.groups storage.read:/ aarc
eduperson_assurance entitlements storage.create:/ storage.modify:/ storage.stage:/ compute.read
compute.modify
Scopes or 'max' (space separated) [openid profile offline_access]: openid profile offline_access phone
Redirect_uris (space separated) [http://localhost:4242]:
Generating account configuration …
accepted
```

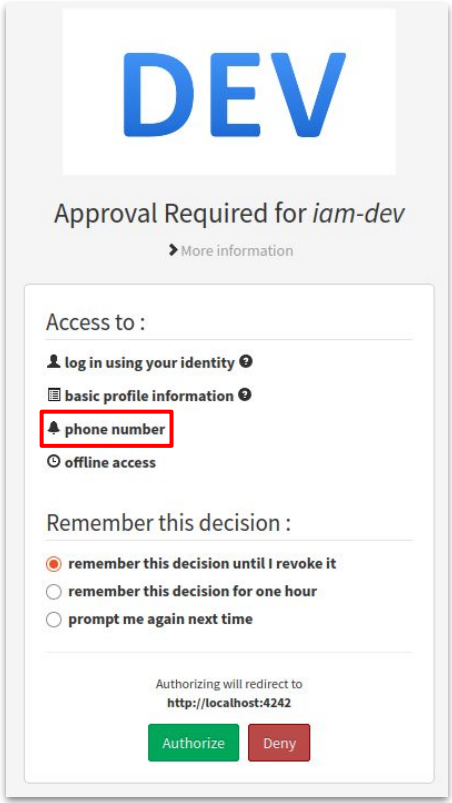
Updating oidc-agent Client configuration

```
Using a browser on any device, visit:
https://iam-dev.cloud.cnaf.infn.it/device

And enter the code: ABCDEF

[ Polling the device code verification from the
https://iam-dev.cloud.cnaf.infn.it/device/approve endpoint ]

Enter encryption password for account configuration 'iam-dev' [***]: ***
Confirm encryption password: ***
Everything setup correctly!
```



Updating oidc-agent Client configuration

```
$ oidc-token -s phone iam-dev | cut -d. -f2 | base64 -d 2>/dev/null | jq
{
  "wlcg.ver": "1.0",
  "sub": "8b7b42fd-0e42-43c5-8254-729aa8f6a12d",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746526113,
  "scope": "phone",
  "iss": "https://iam-dev.cloud.cnaf.infn.it/",
  "exp": 1746527313,
  "iat": 1746526113,
  "jti": "da699ee1-253b-4f17-b6b0-0ce74dbbddcb",
  "client_id": "2ace8c62-9c5a-4cf4-8df8-162d3203f54c"
}
```

More options

- Many other options can be used together with the `oidc-gen`/`oidc-add`/`oidc-token` commands to handle the Client configuration and token requests
- To know more about it, read the documentation!