Using EGI Check-in tokens

Nicolas Liampotis
Valeria Ardizzone
Andrei Tsaregorodtsev

June 21, 2023
WLCG Community Meeting | EGI Conference 2023

TLP: GREEN Limited disclosure
Outline

• Introduction to Check-in
• Check-in token profile
• Scope-based VO/group Membership & Role Selection
• Scope policies
• Managing tokens from command line
• Token-based Access to DIRAC, ARC & HTCondor
  • Token validation plugin
• Open Issues
Using EGI Check-in tokens

Introduction to Check-in
Check-in

In a nutshell

• Identity and Access Management solution that makes it easy to secure access to services and resources

• Single sign-on to services using existing credentials:
  • Academic (e.g. eduGAIN, ORCID)
  • Social media (e.g. Google, Facebook, LinkedIn)
  • Community-managed identities (e.g. Research Infrastructures)

• Federated access to multiple heterogeneous (web and non-web) services using different technologies (SAML, OpenID Connect/OAuth 2.0, X.509)

• Identity linking for accessing resources using different login credentials (institutional/social)

• Aggregation and harmonisation of authorisation information (groups, roles, assurance) from multiple sources

• Implementation of AARC Blueprint Architecture for interoperability with other IAM services for research collaboration
Implementation of the AARC Blueprint Architecture
Using EGI Check-in tokens

Token Profile
**EG Check-in Token Profile**

### Claims

<table>
<thead>
<tr>
<th>CLAIM</th>
<th>ORIGIN</th>
<th>USAGE NOTES</th>
<th>REQUIRED</th>
</tr>
</thead>
</table>
| voperson_id| voPersonID & AARC-G026 | String representation of the subject’s identifier that is globally unique; For instance, this can be achieved by combining an identifier locally unique to the issuing system with a unique property of the issuing system, such as a domain. The identifier meets the following requirements:  
- It MUST be assigned so that no two values created by distinct identity systems could collide when identifying different subjects.
- Once assigned, MUST NOT be reassigned to another subject
- It SHOULD be permanent
- It MUST be persistent
- It MUST be shared; if there are privacy and regulatory requirements that need to be met, the issuing system may not release this identifier to specific relying parties; for instance to prevent them from using the identifier as a basis for correlation
- It MUST contain ASCII characters
  The voperson_id value is a case sensitive string.
  When the token is issued to an end-user managed by EGI Check-in the voperson_id value has the following format: <USER_ID>@egi.eu | OPT      |
| sub        | RFC7519 & OpenID Connect Core | Subject identifier. Contains the same value as the voperson_id | REQ      |
## EG Check-in Token Profile

### Claims

<table>
<thead>
<tr>
<th>CLAIM</th>
<th>ORIGIN</th>
<th>USAGE NOTES</th>
<th>REQUIRED</th>
</tr>
</thead>
</table>
| iss   | RFC7519 & OpenID Connect Core | Issuer of the token:  
- https://aai.egi.eu/auth/realms/egi (Production)  
- https://aai-demo.egi.eu/auth/realms/egi (Demo)  
- https://aai-dev.egi.eu/auth/realms/egi (Development) | REQ |
| scope | RFC9068 | Scope of the access token issued | REQ |
| exp   | RFC7519 & OpenID Connect Core | Expiration time on or after which the Token MUST NOT be accepted for processing | REQ |
| iat   | RFC7519 | Time at which the token was issued | REQ |
| aud   | RFC7519 & OpenID Connect Core | Audience(s) the token is intended for. It can be requested either by using the "resource" parameter in the authorization request to the authorization endpoint, as per Resource Indicators for OAuth 2.0 (RFC8707), or using the "audience" request parameter during token exchange (RFC8693). | OPT |
# EG Check-in Token Validation

## Claims

<table>
<thead>
<tr>
<th>CLAIM</th>
<th>ORIGIN</th>
<th>USAGE NOTES</th>
<th>REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>jti</td>
<td>RFC7519</td>
<td>Unique identifier for the JWT</td>
<td>REQ</td>
</tr>
<tr>
<td>authenticating_authority</td>
<td>EGI</td>
<td>Unique identifier of the authenticating authority of the subject that the token was issued to</td>
<td>OPT</td>
</tr>
<tr>
<td>eduperson_entitlement</td>
<td>AARC-G002, AARC-G027</td>
<td>VO/group memberships and roles &amp; capabilities of the subject this token was issued to. The presence of the eduperson_entitlement Claim in the access token can be signalled by using the experimental eduperson_entitlement_jwt scope in its simple or parameterised form.</td>
<td>OPT</td>
</tr>
</tbody>
</table>
Using EGI Check-in tokens

Scope-based VO/group Membership & Role Selection
Scope-based VO/group Membership and Role Selection

Parametric scopes: What are they and why we need them

- **Scopes**: “Used to request that specific sets of information be made available as claim values” [OIDC-Core]

- **Parametric scopes**: Used to selectively filter the claim values that will be made available when requesting specific sets of information in access tokens, ID tokens and results for userinfo endpoint and token introspection requests

- **Syntax**: 
  ```
  <scope>[::<claim_value>]
  ```

- **Use case**: Obtain Access Tokens that can filter and narrow down the values of the `eduperson_entitlement` claim that represent the subject’s VO/group membership and role information
Scope-based VO/group Membership and Role Selection

Parametric scopes: How to use them

• Syntax:

\texttt{eduperson\_entitlement[[:<entitlement\_value>]]?}

• If the scope is parametric, the AS will return the requested entitlement as a value in the \texttt{eduperson\_entitlement} claim, \textit{only} if the user has been assigned the given entitlement

• To request multiple \texttt{eduperson\_entitlement} values, multiple parametric \texttt{eduperson\_entitlement:<entitlement\_value>} scopes are included in the authorisation request:

\begin{verbatim}
https://aai.egi.eu/auth/realms/egi/protocol/openid-connect/auth?
&client_id=...  
&scope=openid%20profile%20eduperson\_entitlement%3A<entitlement\_value1>%20eduperson\_entitlement%3A<entitlement\_value2>
\end{verbatim}
Scope-based VO/group Membership and Role Selection

Parametric scopes: How to use them

• If the scopes specified in the authorization request contain any parametric forms of the scope then the non-parametric form of the scope, i.e. `eduperson_entitlement` will be ignored.

• The returned `eduperson_entitlement` claim will not contain duplicate entitlements.

• If the user has not been assigned a requested entitlement, the parametric scope will be ignored by the AS and the corresponding token will:
  • not include the given entitlement in the returned `eduperson_entitlement` claim
  • not include the parametric `eduperson_entitlement:<entitlement_value>` scope in the `scope` claim of the returned access token
Scope-based VO/group Membership and Role Selection

Parametric scopes: How to use them

- Alternative syntax for including the VO/group entitlements values in the Access Token:

```
eduperson_entitlement_jwt[:<entitlement_value>]]?
```

```plaintext
https://aai-dev.egi.eu/auth/realms/egi/protocol/openid-connect/auth?
&client_id=...
&scope=openid%20profile%20eduperson_entitlement_jwt%3A<entitlement_value1>%20eduperson_entitlement_jwt%3A<entitlement_value2>
```

```
{
    "exp": 1687310046,
    "iat": 1687306446,
    "sub": "abc...@egi.eu",
    "scope": "openid profile
eduperson_entitlement_jwt:urn:mace:egi.eu:group:<entitlement_value1>
eduperson_entitlement_jwt:urn:mace:egi.eu:group:<entitlement_value2>"
'   "voperson_id": "abc...@egi.eu",
   "eduperson_entitlement": [
   "<entitlement_value1",
   "<entitlement_value2"
   ],
```

EXPERIMENTAL
Scope-based VO/group Membership and Role Selection

Parametric scopes: Could it be done better?

- **Ideally** claim value filtering should be done using the "claims" Request Parameter as per **Sec 5.5, OIDC-Core**

- **BUT**
  - Limited support in OAuth2 client libraries
  - More complex for clients to configure

```json
{
  "userinfo": {
    "eduperson_entitlement": {"values": ["<entitlement_value1>"] }
  }
}
```
Using EGI Check-in tokens

Scope-based Capability Selection
Scope-based VO/group Membership and Role Selection

Scope-based Capability Selection

- **Capabilities** such as `compute.*` need to be granted based on the user’s profile attributes, e.g. identifier or VO/group memberships and roles.

- **Scope policies** enable control over whether the scope should be included in the access token requested by the client.

- If any of the requested scopes are not permitted for the client → The authorisation request will **fail**.
Managing tokens with oidc-agent

Using EGI Check-in tokens
Managing tokens

Introduction to oidc-agent

- oidc-agent is a set of tools developed by KIT to simplify the management of tokens from the command line
- Documentation: https://indigo-dc.gitbook.io/oidc-agent & Check-in docs
- Compatible with multiple operating systems → Linux, Windows, and MacOS
  - Releases available at GitHub or http://repo.data.kit.edu/
- Use cases → Non-web-browser flows: REST APIs, batch jobs
- Benefits of using oidc-agent:
  - Simplified token management: oidc-agent abstracts the complexity of OpenID Connect/OAuth2 flows making it easier to work with Access and Refresh Tokens.
  - Enhanced security: oidc-agent uses encryption mechanisms to securely store tokens
Managing tokens
Using oidc-agent from the command line

• Generate new account configuration with `oidc-gen`
  • `$ oidc-gen --pub --issuer https://aai.egi.eu/auth/realms/egi` → Production
  • `$ oidc-gen --pub --issuer https://aai-demo.egi.eu/auth/realms/egi` → Demo
  • `$ oidc-gen --pub --issuer https://aai-dev.egi.eu/auth/realms/egi` → Devel

• Enter the following information
  o Short name for the account configuration
  o Scopes that the Access Token should contain
    • Include `offline_scope` to obtain Refresh Token
  • Log in with Check-in using your credentials from your preferred identity provider
  • Provide an encryption password to protect the obtained Access/Refresh Tokens
Managing tokens
Using oidc-agent from the command line

• Example: Creating oidc-agent account for Check-in development:

$ oidc-gen --pub --issuer https://aai-dev.egi.eu/auth/realms/egi
   --scope="openid voperson_id compute.read compute.create compute.modify
   compute.cancel eduperson_entitlement offline_access"
Managing tokens
Using oidc-agent from the command line

• Example: Creating oidc-agent account for Check-in development:

```bash
$ oidc-gen --pub --issuer https://aai-dev.egi.eu/auth/realms/egi --scope="openid voperson_id compute.read compute.create compute.modify compute.cancel eduperson_entitlement offline_access"
```
Managing tokens
Using oidc-agent from the command line

• Example: Creating oidc-agent account for Check-in development:

```
$ oidc-gen --pub --issuer https://aai-dev.egi.eu/auth/realms/egi
   --scope="openid voperson_id compute.read compute.create compute.modify compute.cancel eduperson_entitlement offline_access"
```
Managing tokens
Using oidc-agent from the command line

• Example: Obtaining Access Token:

```bash
$ oidc-token egi-dev
```

```
{
  "exp": 1687306617,
  "iat": 1687303017,
  "auth_time": 1687303013,
  "jti": "4cf9c838-5336-4f12-9195-8375402c6066",
  "iss": "https://aai-dev.egi.eu/auth/realms/egi",
  "sub": "jane.doe@egi.eu",
  "typ": "Bearer",
  "azp": "oidc-agent",
  "session_state": "8be81645-6a15-4278-a348-5432a39c8092",
  "scope": "openid compute.modify compute.create offline_access compute.read eduperson_entitlement compute.cancel voperson_id",
  "sid": "8be81645-6a15-4278-a348-5432a39c8092",
  "voperson_id": "jane.doe@egi.eu",
  "authenticating_authority": "https://idp.admin.grnet.gr/idp/shibboleth"
}
```
Managing tokens

Using oidc-agent from the command line

- Example: Obtaining Access Token using parametric scope for specific VO/group membership:

  ```
  $ oidc-token egi-dev --scope="voperson_id compute.create compute.read compute.modify compute.cancel eduperson_entitlement:urn:mace:egi.eu:group:vo.token-integration.egi.eu:role=member#aai.egi.eu"
  ```

  ```
  
  
  
  
  
  
  
  
  
  
  
  
  ```

June 2023 | WLCG Community Meeting | EGI Conference 2023

www.egi.eu | 25
Managing tokens

Using oidc-agent from the command line

• Example Response from UserInfo endpoint:

```bash
$ curl -H "Authorization: Bearer $ACCESS_TOKEN"
https://aai-dev.egi.eu/auth/realms/egi/protocol/openid-connect/userinfo
```

```json
{
  "sub": "auser@egi.eu",
  "voperson_id": "auser@egi.eu",
  "eduperson_entitlement": [
    "urn:mace:egi.eu:group:vo.token-integration.egi.eu:role=member#aai.egi.eu"
  ]
}
```
Managing tokens

Using oidc-agent from the command line

- Example: Obtaining Access Token for specific audience:

```bash
```

```json
{
  "exp": 1687310046,
  "iat": 1687306446,
  "auth_time": 1687303013,
  "jti": "e6b103c7-b1f3-48b0-b5c2-491a2dcd7209",
  "iss": "https://aai-dev.egi.eu/auth/realms/egi",
  "aud": "https://rs.example.org:8080",
  "sub": "",
  "typ": "Bearer",
  "azp": "oidc-agent",
  "session_state": "8be81645-6a15-4278-a348-5432a39c8092",
  "scope": "compute.modify compute.create eduperson_entitlement:urn:mace:egi.eu:group:vo.token-int egration.egi.eu:role=member#aai.egi.eu compute.read compute.cancel voperson_id",
  "sid": "8be81645-6a15-4278-a348-5432a39c8092",
  "voperson_id": "f@egi.eu",
  "authenticating_authority": "https://idp.admin.grnet.gr/idp/shibboleth"
}
```
Token-based access to DIRAC, ARC & HTCondor
Token Validation Plugin

How it started

**Desired Outcome:** "Common mapping library or plugin for mapping tokens to local user accounts"

https://github.com/EGI-Federation/check-in-validator-plugin
Token Validation Plugin

Plugin configuration

- Local identity **mappings** are configured in 
  /etc/egi-check-in-validator/config/egi-check-in-validator.ini
- Syntax:

  MAPPING=UNIQUE_IDENTIFIER ISSUER AUDIENCE SCOPE GROUP

- **MAPPING**: single value (exact match), the identity that will be written to the plugin’s stdout if matched
- **UNIQUE_IDENTIFIER**: single value (exact match or wildcard (*)), the community identifier of the user
- **ISSUER**: single value (exact match), the issuer of the token
- **AUDIENCE**: single value (exact match or wildcard (*)), the audience that the token is intended for
- **SCOPE**: single value (exact match), the scope value that should be included in the scopes of the token
- **GROUP**: single value (exact match), the group/role entitlement that the user should be member of
Example:

```plaintext
foo=xyz@egi.eu https://aai-dev.egi.eu/auth/realms/egi * compute.create
urn:mace:egi.eu:group:vo.token-integration.egi.eu:role=member#aai.egi.eu

bar=* https://aai-dev.egi.eu/auth/realms/egi * compute.create
urn:mace:egi.eu:group:vo.token-integration.egi.eu:role=pilot#aai.egi.eu
```

- Mapping rules are evaluated in the order they are written.
- Values of VO/group entitlements are searched in the following token claims:
  - `eduperson_entitlement` claim values
  - `eduperson_entitlement:<entitlement>` scopes in the `scope` claim
  - `eduperson_entitlement_jwt:<entitlement>` in the `scope` claim
Token Validation Plugin

HTCondor configuration

• Add the following line to 
/etc/condor-ce/mapfiles.d/10-scitokens.conf:

    \n    \n    SCITOKENS /\^[https:\/\/]aai-dev.egi.eu\/[auth\/]realms\/[egi,.\]*/ PLUGIN:EGI

• Create file under /etc/condor-ce/config.d/:

    \n    \n    SEC_SCITOKENS_ALLOW_FOREIGN_TOKENS=true
    SEC_SCITOKENS_PLUGIN_NAMES=EGI
    SEC_SCITOKENS_PLUGIN_EGI_COMMAND=/usr/bin/egi-check-in-validator.py
• Users are not expected to manage tokens themselves to directly submit jobs to CEs
• Instead they use “dirac-login” or DIRAC web portal to submit jobs through DIRAC which hides the complexity of OIDC/OAuth2.0 and automatically refreshes tokens when they expire
• Refresh token obtained upon interactive user authentication is associated to all VO/group memberships of the user and is securely stored in DIRAC’s token manager service
• Refresh token is then used to obtain Access Tokens associated with specific VO using the refresh token grant combined with the parametric use of the eduperson_entitlement scope
• Pilot jobs are submitted with tokens using Client Credentials grant [RFC6749]
  • Use parametric `eduperson_entitlement` to obtain tokens scoped to specific VO/group & role
  • Use “resource” parameter (“Resource Indicators for OAuth 2.0” RFC8707) to audience-restrict the tokens
    • Currently, the token audience (aud claim) is set to the CE endpoint by default but can be configured

• DIRAC uses 1-to-1 mapping of VO/group entitlement values to DIRAC groups
• Tested successfully with HTCondor using the Check-in token validation plugin
• Testing with ARC7 still in progress
Using EGI Check-in tokens

Open Issues
Open Issues

• Handling tokens associated with multiple VOs
  • ...what about accounting?
  • Note: Issue specific to tokens directly managed from users – NOT through DIRAC
• Tokens combining capabilities with VO/group claims
• Traceability requirement & uses cases
Thank you

Nicolas Liampotis

✉ nliam@grnet.gr | check-in@egi.eu

www.egi.eu