

# DOMA TPC: Token-based AuthZ Testbed

Andrea Ceccanti  
INFN CNAF

May, 20th 2020



# Objectives of the testbed

Demonstrate that token-based authN/Z **using the WLCG JWT profile** can support LHC experiments data management

In more detail

- Clarify OAuth/OIDC protocol exchanges among the various components
  - UI → RUCIO → FTS → SEs
- Understand the JWT profile support in the various components
  - Scoped-based authZ
  - Group-based authZ
- Provide guidelines to sites on how token-based authn/z can be enabled
  - Baseline versions & configuration guidelines
- Provide guidelines to users
  - Docs that answer questions like how do I get a token? etc...

# The testbed documentation

Testbed documentation will live at

<https://wlcg-authz-wg.github.io/wlcg-authz-docs/>

- mkdocs site, documentation in plain markdown
- GH actions flow to deploy the site at each change

Currently is just a skeleton, will need to be filled with:

- Testbed endpoints, known issues in implementations, ...
- Guidelines, hints for configuration etc...
- ...

Collaborative effort!

# WLCG JWT profile conformance

A JWT profile compliance test-suite would help in assessing conformance with the profile

- Check that issuer checks, signature checks, temporal validity, audience restrictions, path constraints, ... are honoured by the implementations

I would use Robot Framework to write this test suite

- Easy to write tests using text files and python code
- Good reporting

The test suite could then be used to implement a periodic smoke test against the testbed endpoints

# The OAuth/OIDC flows

The scope-based authZ scenario has been already defined at the end of last year

- I've included slides from the January hackathon in this deck

TODO: the group-based AuthZ flows

- I can define a proposal and then we discuss it at one of the future meetings

**Backup slides**

# What does it mean supporting the WLCG profile?

As an **OAuth resource server** (RS):

- Ability to extract an access token from an incoming HTTP request
- Ability to parse and validate the incoming access token
  - identify if it has been issued by a trusted and recognized authorization server
  - verify temporal validity
  - verify signature, following OAuth/OIDC conventions
- Ability to honour access token audience restrictions
  - the RS needs the ability to identify itself with (one or multiple) audience labels and honour audience restrictions in access tokens
- Ability to map defined scopes to local authZ
  - e.g., storage.read:/cms grants read access to the /cms namespace (and any subdirectory)
- Ability to map group-based to local authZ
  - e.g., /cms group membership as stated grants read access to the /cms namespace

# What does it mean supporting the WLCG profile?

As an **OAuth resource server** (RS):

- Ability to extract an access token from an incoming HTTP request
- Ability to parse and validate the incoming access token
  - identify if it has been issued by a trusted and recognized authorization server
  - verify temporal validity
  - verify signature, following OAuth/OIDC conventions
- Ability to honour access token audience restrictions
  - the RS needs the ability to identify itself with (one or multiple) audience labels and honour audience restrictions in access tokens
- Ability to map defined scopes to local authZ
  - e.g., storage.read:/cms grants read access to the /cms namespace (and any subdirectory)
- Ability to map group-based to local authZ
  - e.g., /cms group membership as stated grants read access to the /cms namespace

**This is typically sorted out by OAuth/OIDC libraries**



# What does it mean supporting the WLCG profile?

This is typically sorted out by

As an **OAuth/OpenID Connect client**:

**OAuth/OIDC libraries**

- Ability to store client credentials securely
- Ability to start and manage an OAuth/OpenID Connect flow to obtain tokens from the Authorization Server (i.e., IAM)
  - Authorization code flow, for most use cases
  - Refresh token flow, to refresh access tokens about the expire
  - Client credentials flow, to obtain tokens linked not linked to user identities, but to the service itself
- Ability to parse and validate ID tokens resulting from OpenID Connect authentication flows in compliance with the OpenID connect spec
- Ability to honour audience restrictions
  - the ability to identity itself with (one or multiple) audience labels and honour audience restrictions in ID tokens
- (Optional) Ability to implement Level Of Assurance (LoA) policies

# Scope-based AuthZ scenario

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

In this scenario, RUCIO delegates its identity to FTS to manage a third-party data transfer between SE 1 and SE 2

rucio.example



se1.example



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example



se1.example



RUCIO gets a token from IAM using the OAuth **client\_credentials** grant type. The token needs to provide the minimum privileges need to interact with FTS



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example

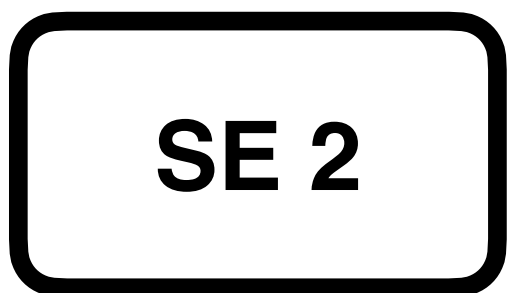
se1.example



Token  
request

```
POST /token HTTP/2
Host: iam.example
Authorization: Basic ZG...B
Accept: */*
Content-Length: ...
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&scope=fts:submit-transfer
&audience=fts.example
```



iam.example

fts.example

se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example

se1.example



Token  
request

```
POST /token HTTP/2
Host: iam.example
Authorization: Basic ZG...B
Accept: */*
Content-Length: ...
Content-Type: application/x-www-form-urlencoded
```

**requested scopes & audience**

```
grant type=client credentials
&scope=fts:submit-transfer
&audience=fts.example
```

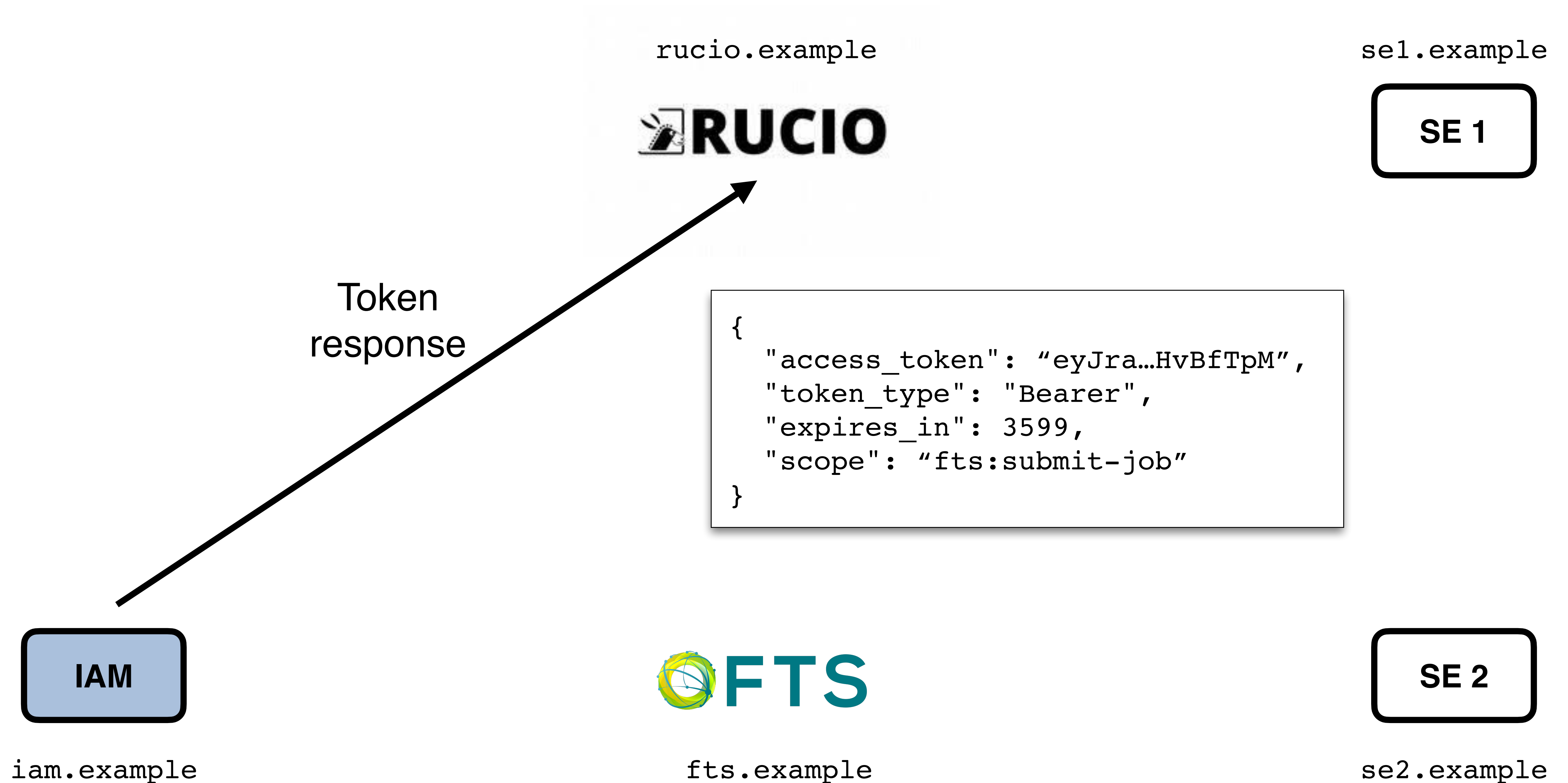


iam.example

fts.example

se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity



# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example



se1.example



Token  
response

```
{  
  "access_token": "eyJra...HvBfTpM",  
  "token_type": "Bearer",  
  "expires_in": 3599,  
  "scope": "fts:submit-job"  
}
```



iam.example



fts.example



se2.example



# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

Rucio extracts the access token from the response, and stores it in local memory.

rucio.example




se1.example



access token body:

```
{
  "sub": "rucio.example",
  "aud": "fts.example",
  "nbf": 1572840340,
  "scope": "fts:submit-transfer",
  "iss": "https://iam.example/",
  "exp": 1572843940,
  "iat": 1572840340,
  "jti": "be48f2ab-8dd9-4df2-ae0b-bcb1fdafafaa6"
}
```

```
{
  "access_token": "eyJra...HvBfTpM",
  "token_type": "Bearer",
  "expires_in": 3599,
  "scope": "fts:submit-job"
}
```

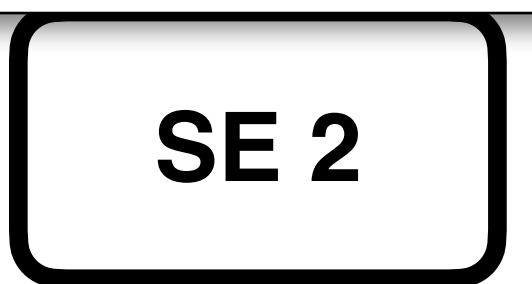
 **parse  
&  
validate  
JWT**



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

The token audience is limited to FTS, and the requested scope has been granted.

rucio.example



se1.example



access token body:

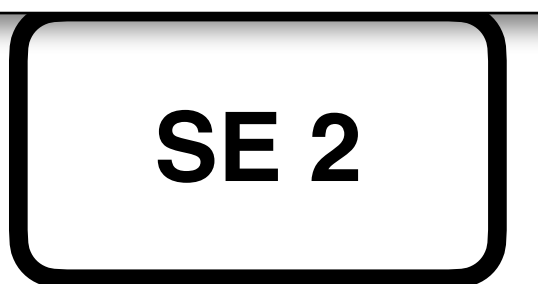
```
{
  "sub": "rucio.example",
  "aud": "fts.example",
  "nbf": 1572840340,
  "scope": "fts:submit-transfer",
  "iss": "https://iam.example/",
  "exp": 1572843940,
  "iat": 1572840340,
  "jti": "be48f2ab-8dd9-4df2-ae0b-bcb1fdafaa6"
}
```



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

RUCIO submits a transfer job to FTS, including the token obtained from IAM in the request

rucio.example



Submit  
transfer  
job



fts.example

se1.example



se2.example





iam.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

FTS validates the token extracted from the request and accepts the transfer, assuming the token is valid and provides the necessary rights

rucio.example



Submit transfer job  



fts.example

se1.example



se2.example



iam.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example



se1.example



FTS now needs a token that will be used for AuthN/Z at the storage elements. In this scenario, FTS impersonates RUCIO.



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example



se1.example



The token it already has cannot be used for the transfer:  
it's scoped to fts.example and does not provide the necessary rights to read and store files at storage elements



iam.example



fts.example



se2.example

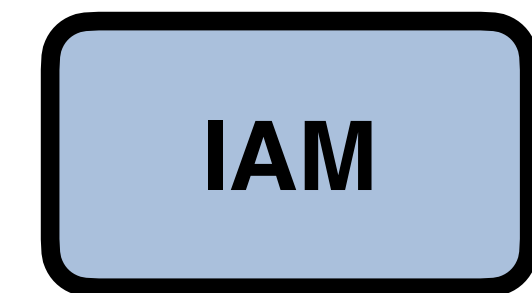
# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

FTS then **exchanges the obtained token** with a couple of tokens, an access token and refresh token, that will be used to manage the transfer

rucio.example



se1.example



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example

se1.example



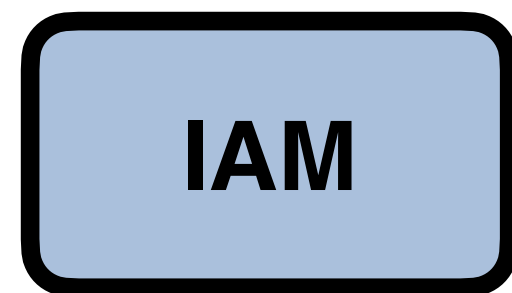
SE 1

FTS requests the following scopes:  
storage.read:/  
storage.create:/  
offline\_access

```
POST /token HTTP/2
Host: iam.example
Authorization: Basic u89...
Accept: */*
Content-Length: ...
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:token-exchange
&subject_token=eyJra...HvBfTpM
&audience=se1.example%20se2.example
&scope=storage.read%3A%2F%20storage.create%3A%2F%20offline_access
```

Token  
exchange  
request



SE 2

iam.example

fts.example

se2.example





# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example

se1.example



SE 1

The audience of the token is limited to only apply to the storage elements involved in the transfer

```
POST /token HTTP/2
Host: iam.example
Authorization: Basic u89...
Accept: */*
Content-Length: ...
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:token-exchange
&subject token=eyJra...HvBfTpM
&audience=se1.example%20se2.example
&scope=storage.read%3A%2F%20storage.create%3A%2F%20offline_access
```

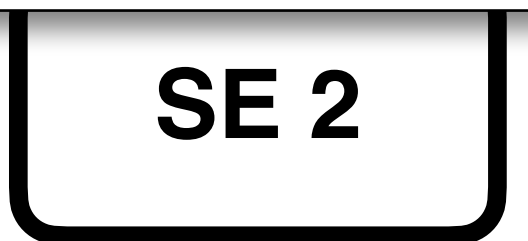
Token exchange request



iam.example



fts.example



se2.example



# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

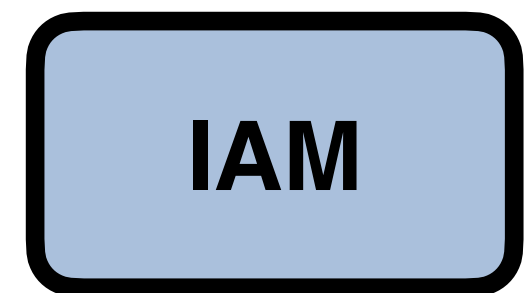
rucio.example



se1.example



IAM validates the token exchange request, and assuming there's a policy that authorizes the exchange, issues the requested tokens



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example

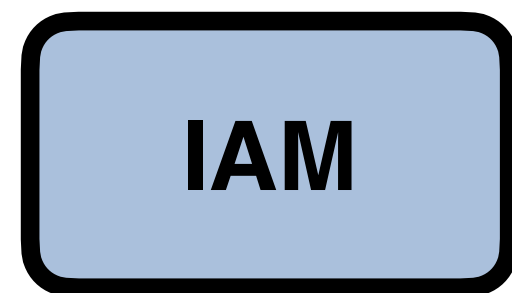


se1.example



```
{  
  "access_token": "e7nd...HvBfTpM",  
  "refresh_token": "9njuk...",  
  "token_type": "Bearer",  
  "expires_in": 3599,  
  "scope": "storage.read:/ storage.create:/ offline_access"  
}
```

Token  
exchange  
response



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

FTS extracts the tokens from the response and saves them locally

rucio.example



se1.example



```
{  
  "access_token": "e7nd...HvBfTpM",  
  "refresh_token": "9njuk...",  
  "token_type": "Bearer",  
  "expires_in": 3599,  
  "scope": "storage.read:/  
           storage.create:/  
           offline_access"  
}
```



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

The new access token can be refreshed from IAM with the **refresh\_token** flow.

Refresh tokens are typically much longer lived than access tokens and

rucio.example



se1.example



access token body:

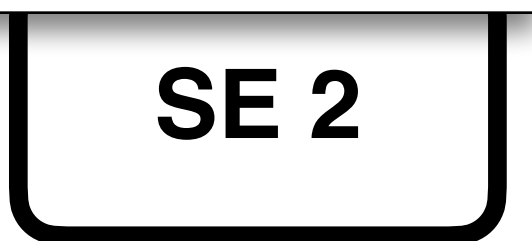
```
{
  "sub": "rucio.example",
  "aud": ["se1.example", "se2.example"],
  "nbf": 1572840345,
  "scope": "storage.read:/ storage.create:/
           offline_access",
  "iss": "https://iam.example/",
  "exp": 1572843945,
  "iat": 1572840345,
  "jti": "be48..."
}
```



iam.example



fts.example



se2.example

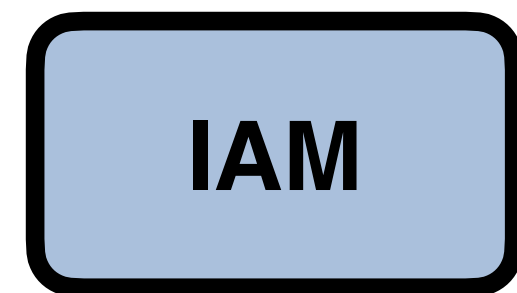
# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

FTS will enqueue the transfer job, and when the transfer is about to start can use the refresh token to get a fresh access token that will be used for the transfer.

rucio.example



se1.example



iam.example



fts.example



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

rucio.example



se1.example



FTS then submits the third-party transfer against SE 2, including the token in the request

```
COPY /example/file HTTP/2
Host: se2.example
Source: https://se1.example/example/file
Authorization: Bearer e7nd...
TransferHeaderAuthorization: Bearer e7nd...
```



iam.example



fts.example

Submit  
third-party transfer



se2.example

# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

The same token will be used for authn/z at se1 and se2.

It's also possible to have two separate tokens for each SE

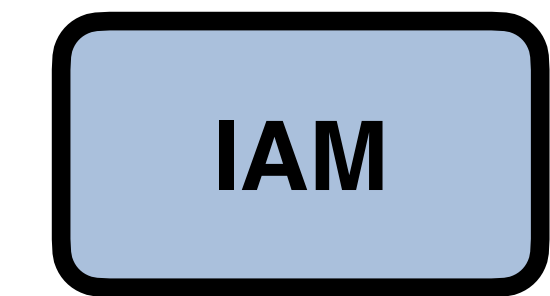
rucio.example



se1.example



```
COPY /example/file HTTP/2
Host: se2.example
Source: https://se1.example/example/file
Authorization: Bearer e7nd...
TransferHeaderAuthorization: Bearer e7nd...
```



iam.example



fts.example

Submit  
third-party transfer



se2.example



# Token-based AuthN/Z for DOMA xfers: RUCIO delegated identity

SE2 will then use the  
obtained token for authn/z  
against SE1

rucio.example



```
GET /example/file HTTP/2
Host: se1.example
Authorization: Bearer e7nd...
```

se1.example



Data  
Transfer 



se2.example



iam.example



fts.example



**Thanks for your attention.  
Questions?**