(WLCG) tokens integration and support in EOS

Elvin Sindrilaru

on behalf of the **EOS team**

07.03.2022

Outline

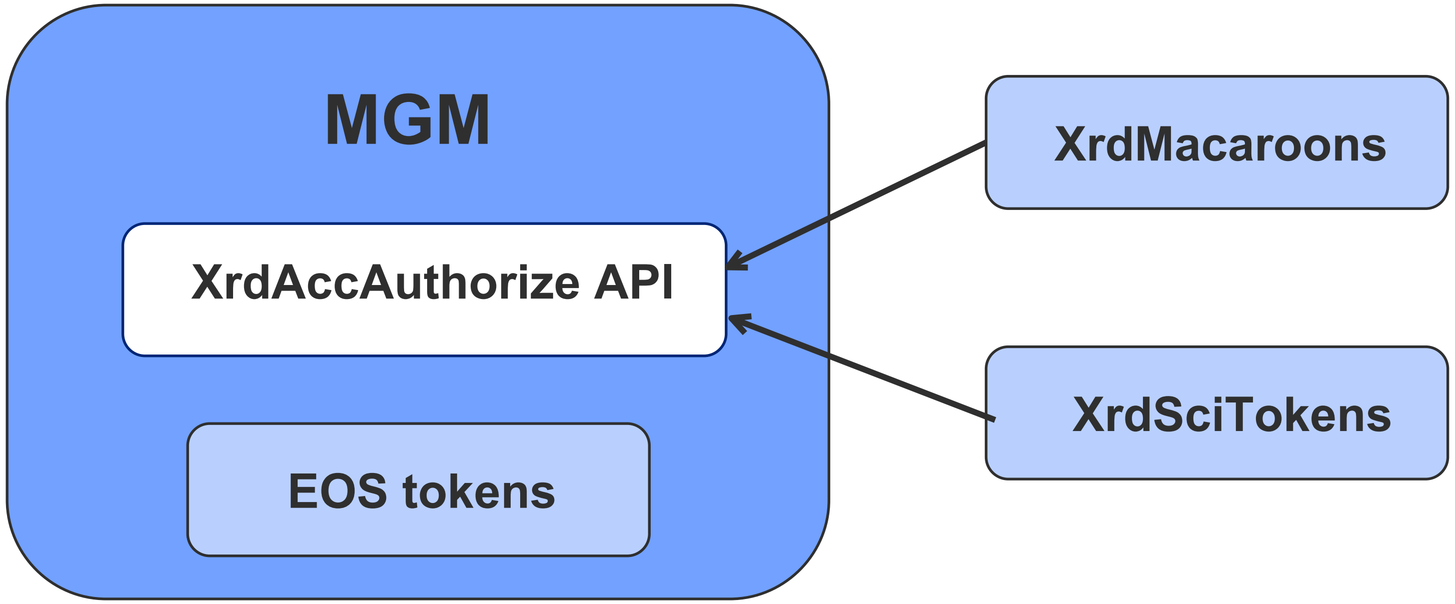
- What are tokens? Why use them?
- EOS design and support for tokens
  - Storage Element tokens
    - EOS tokens
    - Macaroons
  - SciTokens tokens (WLCG JWT tokens)
- WLCG JWT compliance test-suite
- Plans for the future

What are tokens? Why use them?

- ***Bearer Token:*** "*a string representing an access authorization issued to the client*"
- **Types of (bearer) tokens:**
  - **ID tokens** (Open ID Connect)
    - Contain info **who someone is**
    - ***Must not*** be used to make requests to the resource server
  - **Access tokens** (OAuth 2.0)
    - Contain info about **what someone is allowed to do**
    - **Should** be used only to make requests to the resource server
    - Different formats from simple hex string to JSON Web Tokens (JWT)
    - **JWT** - way to encode claims in a JSON document that is then signed
- **Advantages**
  - Simple to use for API requests
  - Don't require cryptographic signing of each request
- **Disadvantages**
  - Communication channel needs to be encrypted
  - Anyone getting access to a token can use it

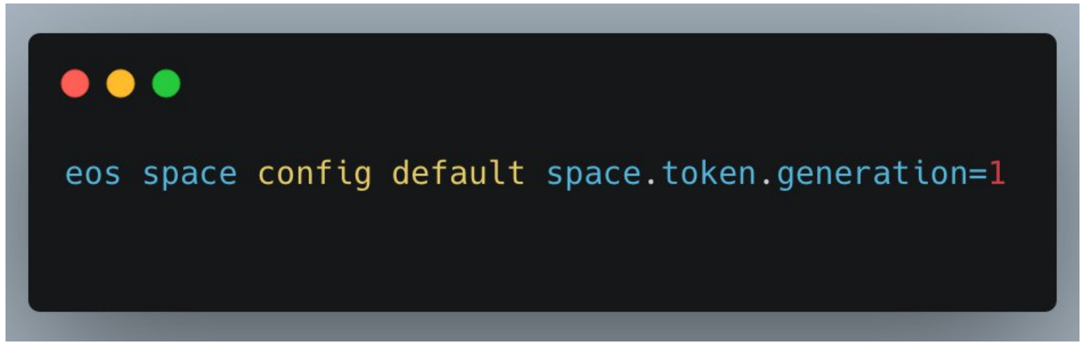
EOS tokens plug-in support

- In EOS **everything is a plug-in** including tokens support
- **Configuration** changes are only required **at the MGM level**
- All new authz plug-ins must implement the **XrdAccAuthorize** interface



EOS token™

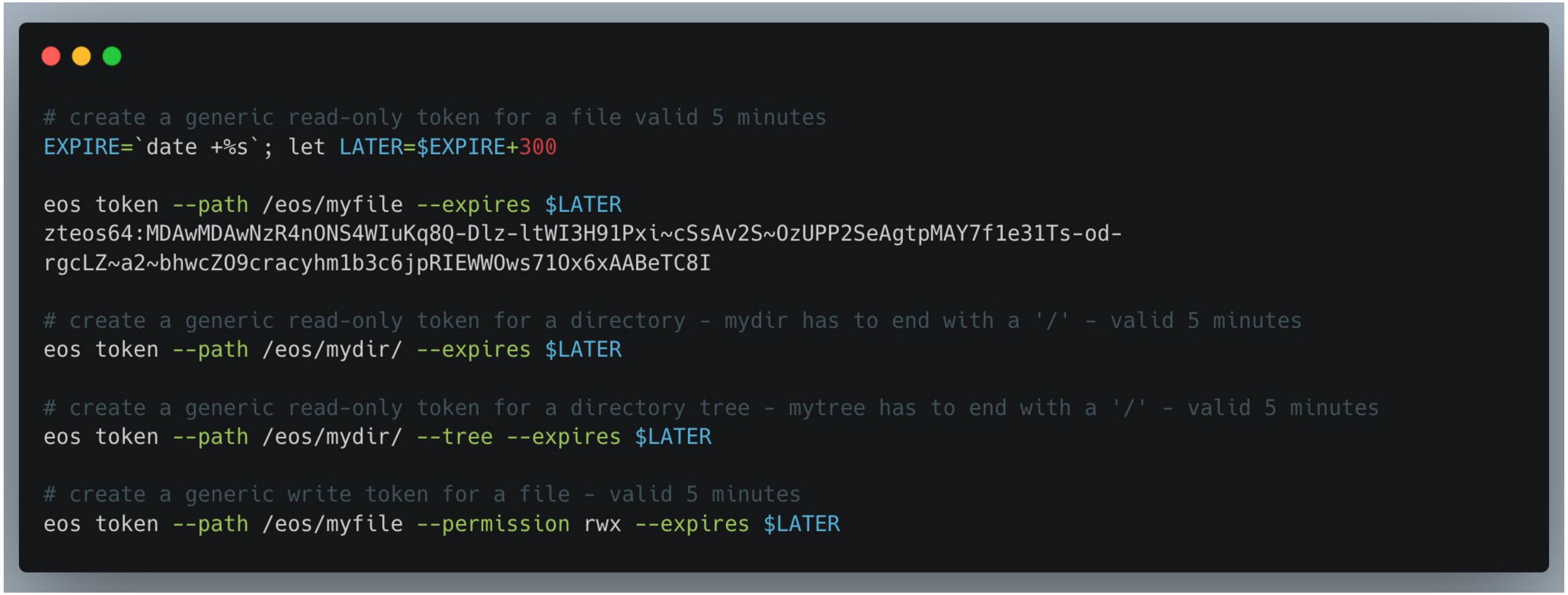
- **Generic EOS specific mechanism** to delegate permissions to bearer token
- Support needs to be enabled at instance level:



```
eos space config default space.token.generation=1
```

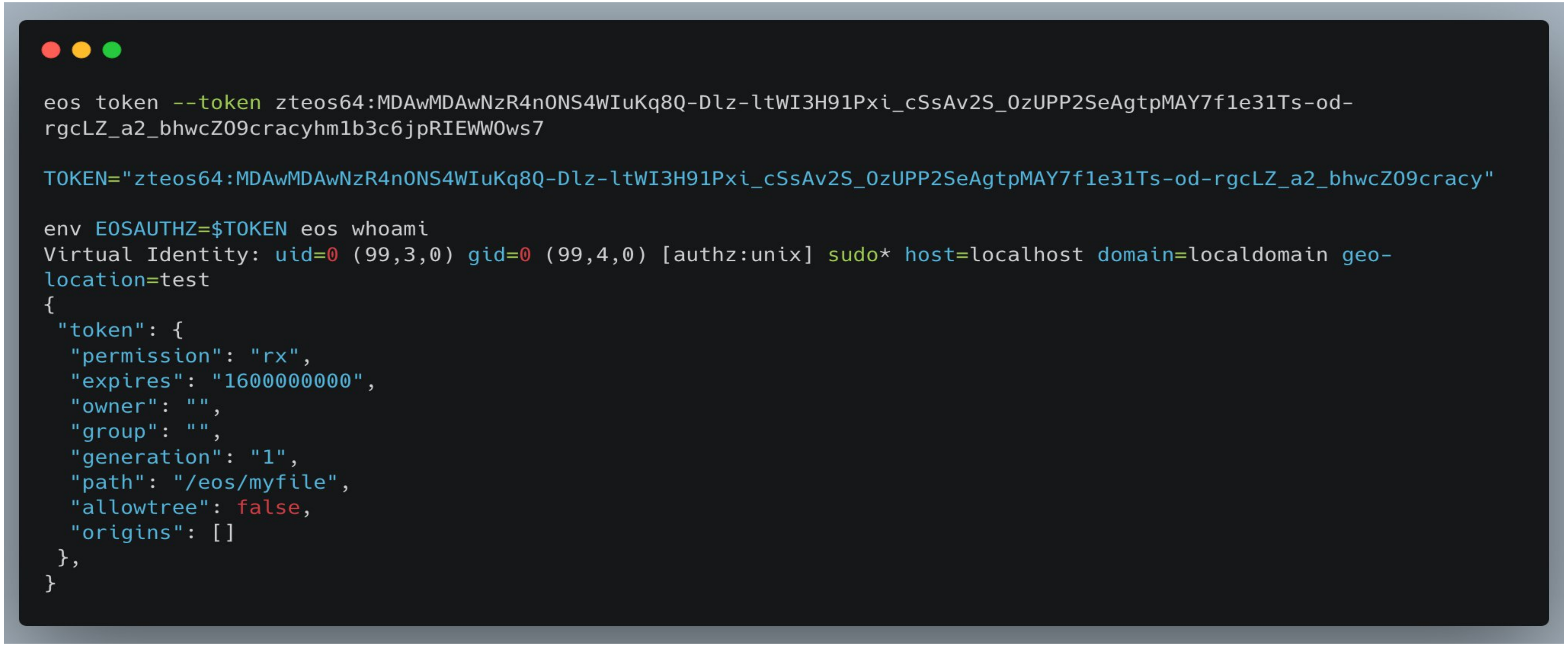
- Tokens are **signed, zlib compressed and base64url encoded**
- The token can include owner and group info or not, in which case vid mapping rules apply
- All operations are done through the ***eos token*** CLI

EOS token™ creation



```
# create a generic read-only token for a file valid 5 minutes
EXPIRE=`date +%s`; let LATER=$EXPIRE+300

eos token --path /eos/myfile --expires $LATER
zteos64:MDAwMDAwNzR4nONS4WIuKq8Q-Dlz-ltWI3H91Pxi~cSsAv2S~OzUPP2SeAgtpMAY7f1e31Ts-od-
rgcLZ~a2~bhwcZO9cracyhm1b3c6jpRIEWWOws71Ox6xAABeTC8I

# create a generic read-only token for a directory - mydir has to end with a '/' - valid 5 minutes
eos token --path /eos/mydir/ --expires $LATER

# create a generic read-only token for a directory tree - mytree has to end with a '/' - valid 5 minutes
eos token --path /eos/mydir/ --tree --expires $LATER

# create a generic write token for a file - valid 5 minutes
eos token --path /eos/myfile --permission rwx --expires $LATER
```

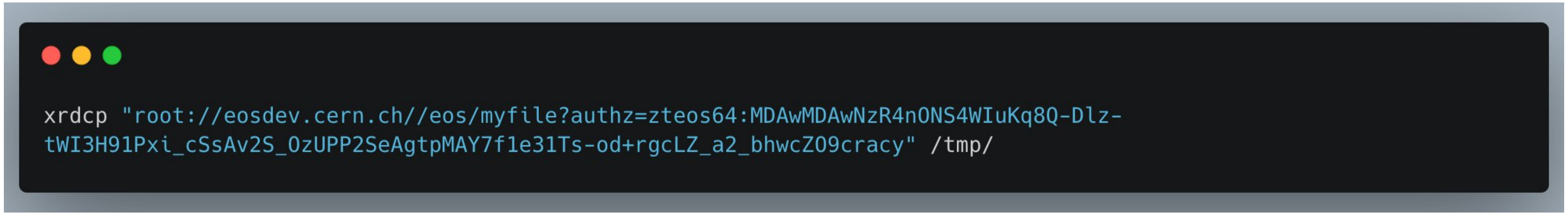
EOS token™ inspection



```
eos token --token zteos64:MDAwMDAwNzR4nONS4WIuKq8Q-Dlz-ltWI3H91Pxi_cSsAv2S_OzUPP2SeAgtpMAY7f1e31Ts-od-
rgcLZ_a2_bhwcZ09cracyhm1b3c6jpRIEWWOws7

TOKEN="zteos64:MDAwMDAwNzR4nONS4WIuKq8Q-Dlz-ltWI3H91Pxi_cSsAv2S_OzUPP2SeAgtpMAY7f1e31Ts-od-rgcLZ_a2_bhwcZ09cracy"

env EOSAUTHZ=$TOKEN eos whoami
Virtual Identity: uid=0 (99,3,0) gid=0 (99,4,0) [authz:unix] sudo* host=localhost domain=localdomain geo-
location=test
{
 "token": {
  "permission": "rx",
  "expires": "1600000000",
  "owner": "",
  "group": "",
  "generation": "1",
  "path": "/eos/myfile",
  "allowtree": false,
  "origins": []
 },
}
```

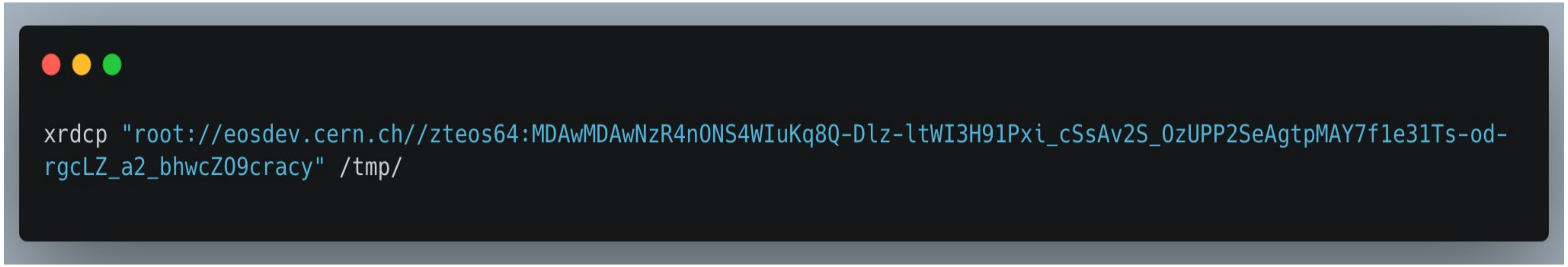
EOS token™ usage

- Direct usage **embedded as opaque (CGI)** info for transfers



```
xrdcp "root://eosdev.cern.ch//eos/myfile?authz=zteos64:MDAwMDAwNzR4nONS4WIuKq8Q-Dlz-
tWI3H91Pxi_cSsAv2S_OzUPP2SeAgtpMAY7f1e31Ts-od+rgcLZ_a2_bhwcZO9cracy" /tmp/
```

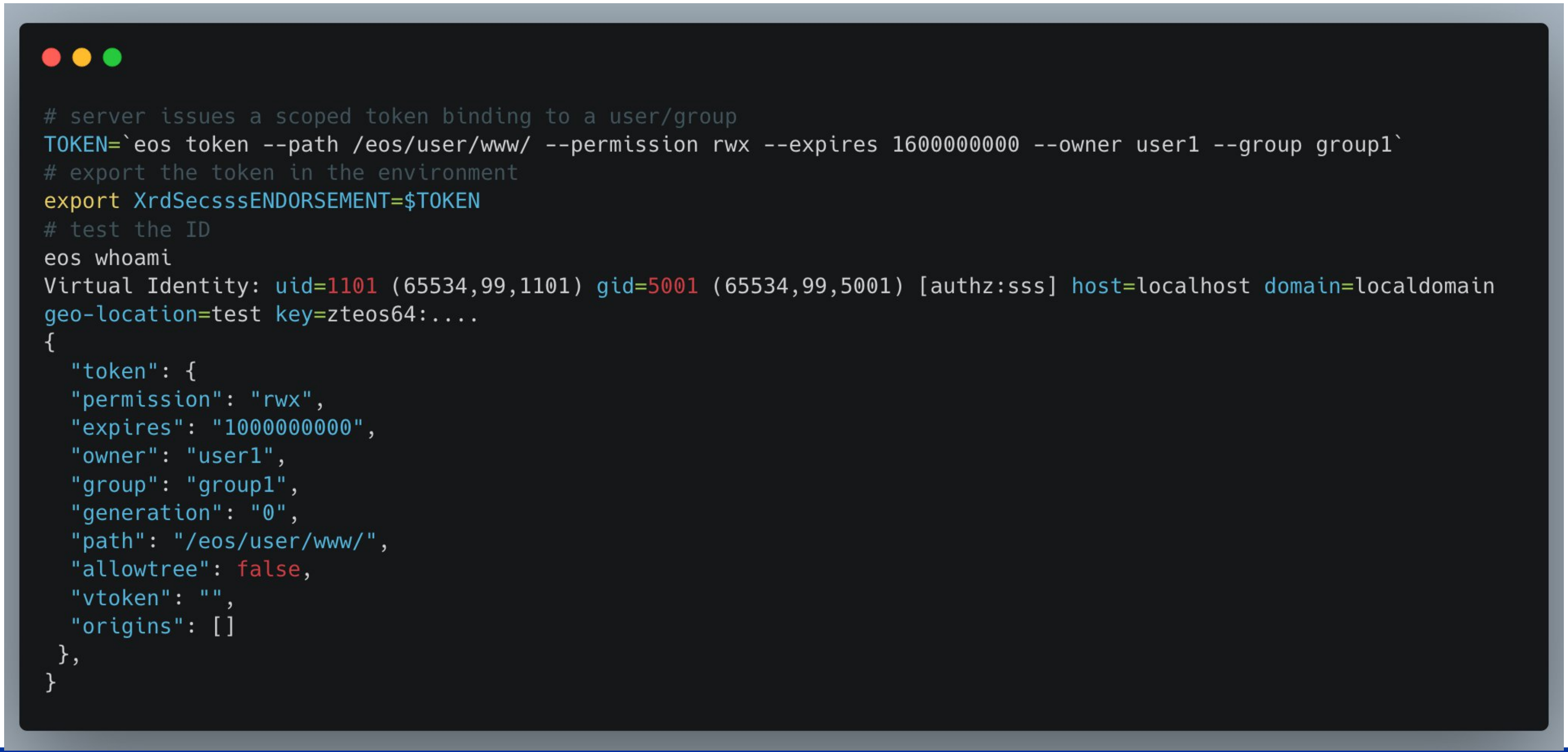
- Direct usage of the token itself **as a filename**



```
xrdcp "root://eosdev.cern.ch//zteos64:MDAwMDAwNzR4nONS4WIuKq8Q-Dlz-ltWI3H91Pxi_cSsAv2S_OzUPP2SeAgtpMAY7f1e31Ts-od-
rgcLZ_a2_bhwcZO9cracy" /tmp/
```

EOS token™ usage with SSS

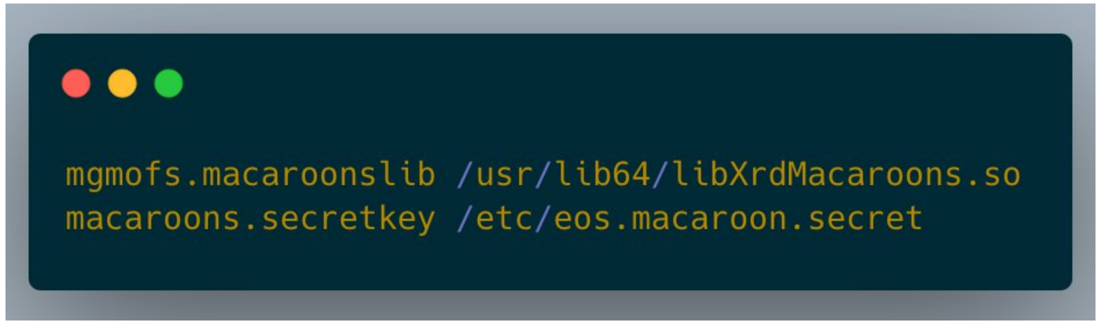
- Use **SSS (Simple Shared Secret) endorsement field** to forward tokens in a secure way
- Client and server need to **share an SSS key**
  - **not authorized** to use the instance
  - acts as a secure **carrier for the token**



```
# server issues a scoped token binding to a user/group
TOKEN=`eos token --path /eos/user/www/ --permission rwx --expires 1600000000 --owner user1 --group group1`
# export the token in the environment
export XrdSecsssENDORSEMENT=$TOKEN
# test the ID
eos whoami
Virtual Identity: uid=1101 (65534,99,1101) gid=5001 (65534,99,5001) [authz:sss] host=localhost domain=localdomain
geo-location=test key=zteos64:....
{
  "token": {
  "permission": "rwx",
  "expires": "1000000000",
  "owner": "user1",
  "group": "group1",
  "generation": "0",
  "path": "/eos/user/www/",
  "allowtree": false,
  "vtoken": "",
  "origins": []
 },
}
```

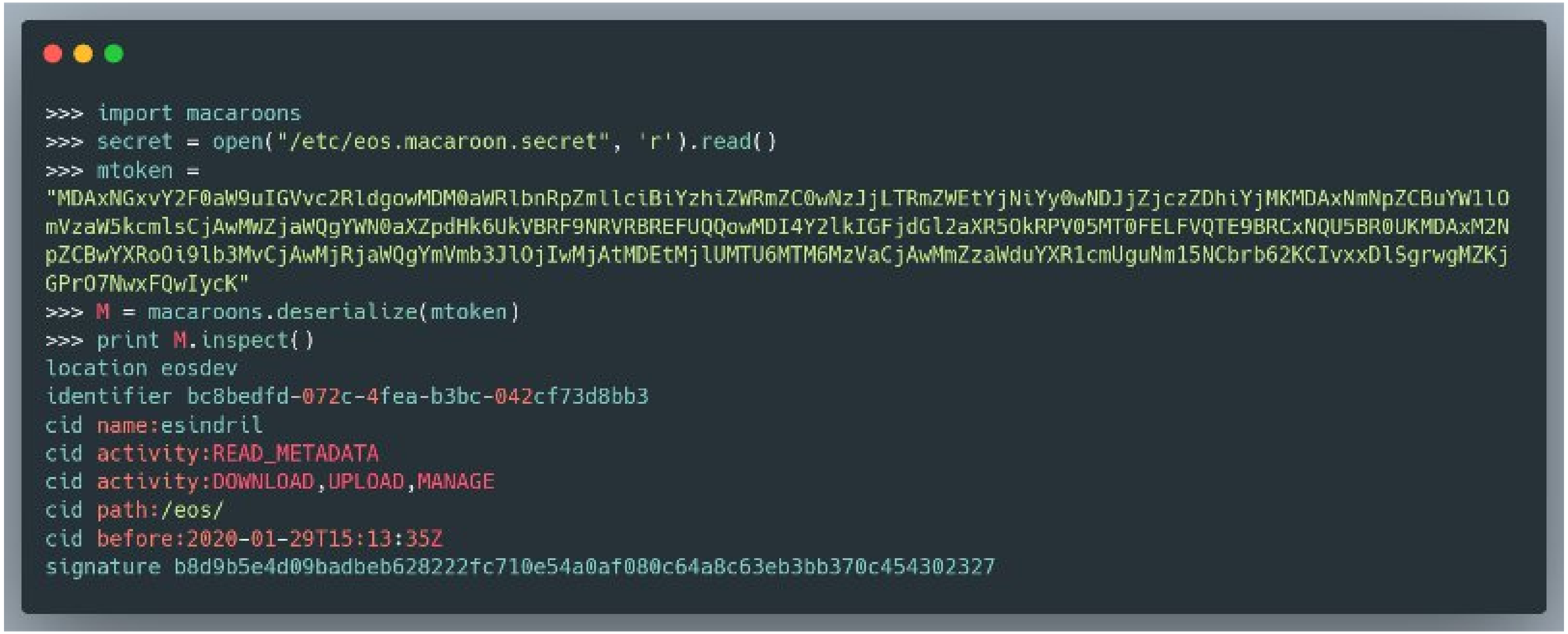
XrdMacaroons configuration and use

- **Macaroon** tokens supported by **libXrdMacaroons.so** that comes by default with **XRootD**
- Type of Storage Element token
  - **issued** to the client **by the MGM**
  - **used** by the client **at the MGM** to gain access to the data
  - no dependency on any external service
- **Configuration** directives at the MGM
- Required packages
  - **x509-scitokens-issuer**
  - **x509-scitokens-issuer-client** - include macaroon-init tool for obtaining macaroons using X509
  - **python2-macaroons** – for inspecting the contents of macaroons



```
mgmofs.macaroonslib /usr/lib64/libXrdMacaroons.so
macaroons.secretkey /etc/eos.macaroon.secret
```

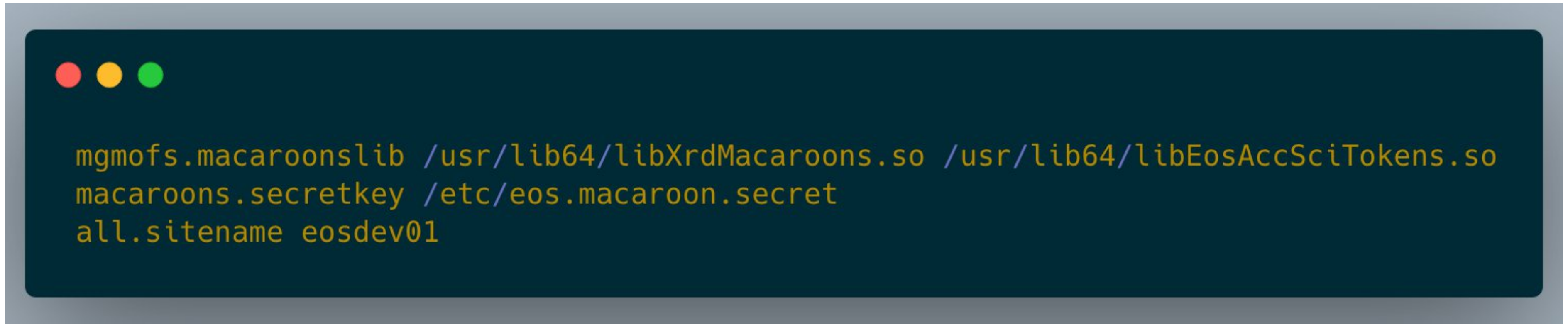
XrdMacaroons inspect token



```
>>> import macaroons
>>> secret = open("/etc/eos.macaroon.secret", 'r').read()
>>> mtoken =
"MDAxNGxvY2F0aW9uIGVvc2RldgowMDM0aWRlbnRpZmllciBiYzhiZWRmZC0wNzJjLTRmZWEtYjNiYy0wNDJjZjczZDhiYjMKMDAxNmNpZCBuYW1lO
mVzaW5kcmlsCjAwMWZjaWQgYWN0aXZpdHk6UkVBRF9NRVRBREFUQQowMDI4Y2lkIGFjdGl2aXR5OkRPV05MT0FELFVQTE9BRCxNQU5BR0UKMDAxM2N
pZCBwYXRoOi9lb3MvCjAwMjRjaWQgYmVmb3JlOjIwMjAtMDEtMjlUMTU6MTM6MzVaCjAwMmZzaWduYXR1cmUguNm15NCbrb62KCIvxxDlSgrwgMZKj
GPrO7NwxFQwIycK"
>>> M = macaroons.deserialize(mtoken)
>>> print M.inspect()
location eosdev
identifier bc8bedfd-072c-4fea-b3bc-042cf73d8bb3
cid name:esindril
cid activity:READ_METADATA
cid activity:DOWNLOAD,UPLOAD,MANAGE
cid path:/eos/
cid before:2020-01-29T15:13:35Z
signature b8d9b5e4d09badbeb628222fc710e54a0af080c64a8c63eb3bb370c454302327
```

XrdSciTokens configuration and use

- **SciTokens** supported by **libXrdSciTokens.so** that comes by default with **XRootD**
- **EOS** uses a repackaged version of the code but will soon switch to the default XRootD

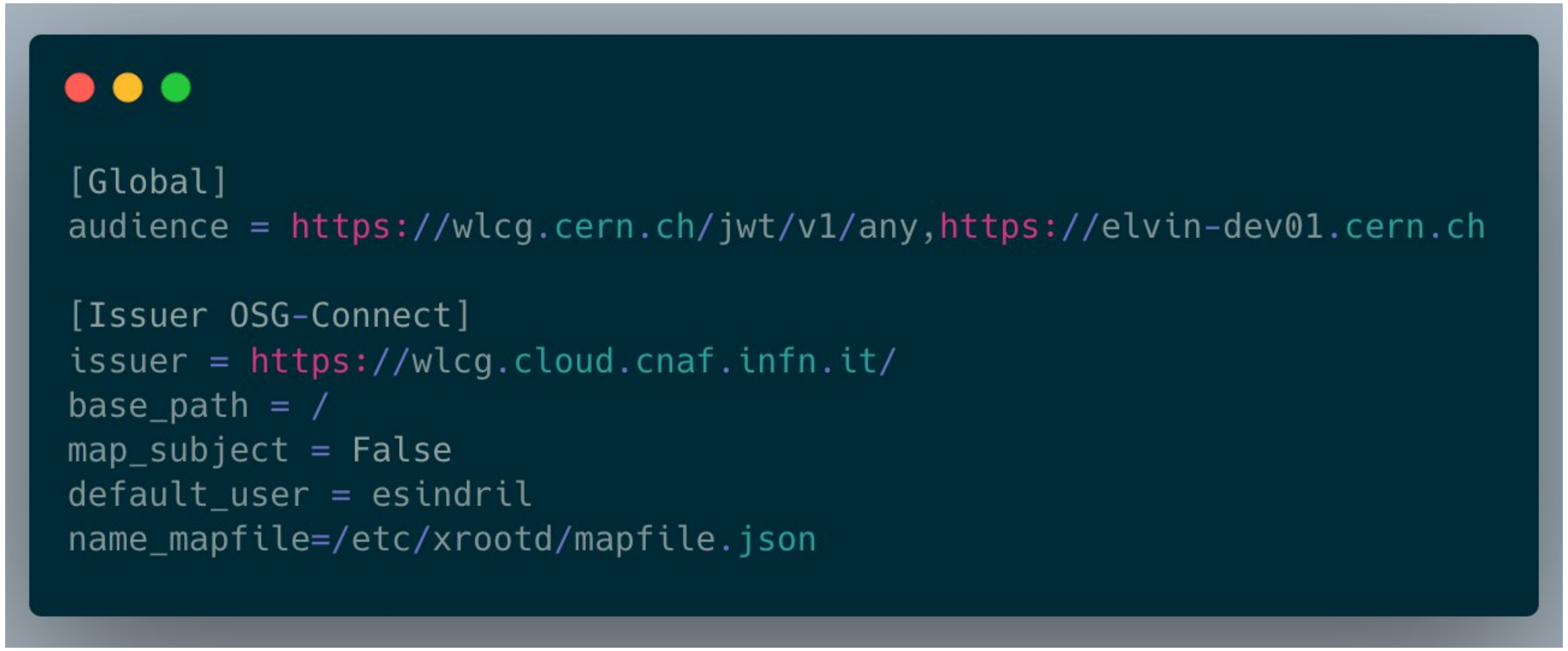


```
mgmofs.macaroonslib /usr/lib64/libXrdMacaroons.so /usr/lib64/libEosAccSciTokens.so
macaroons.secretkey /etc/eos.macaroon.secret
all.sitename eosdev01
```

- The various **authz libraries** are **chained**
  - Once one of the libraries can handle a token the rest are not invoked anymore
  - Configuration file for SciTokens: ***/etc/xrootd/scitokens.cfg***
- Requires direct interaction with a **IAM (Identity & Access Management) Provider**

XrdSciTokens example basic configuration

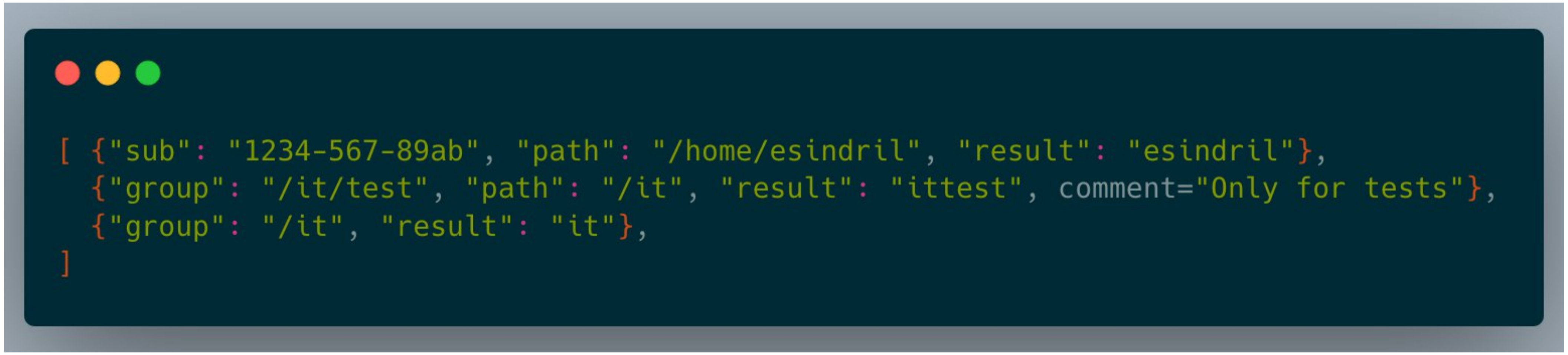
- Several ways of doing authorization:
  - **scope-based** - when a certain path is authorized
  - **group-based** - group info is copied to the XRootD internal credentials object (XrdSecEntity)



```
[Global]
audience = https://wlcg.cern.ch/jwt/v1/any,https://elvin-dev01.cern.ch

[Issuer OSG-Connect]
issuer = https://wlcg.cloud.cnaf.infn.it/
base_path = /
map_subject = False
default_user = esindril
name_mapfile=/etc/xrootd/mapfile.json
```

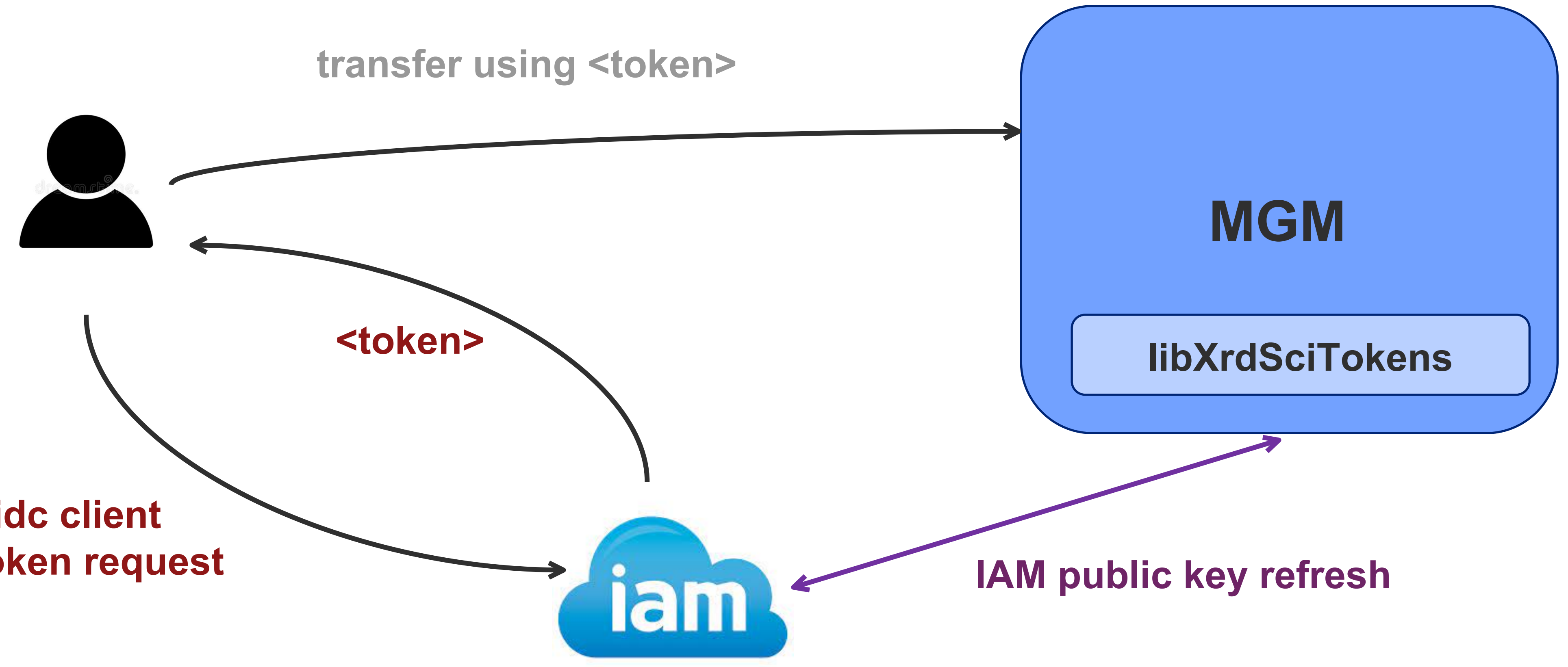
XrdSciTokens name-map functionality

- **Storage systems** need to associate a **local username** with any incoming request
- **SciTokens** provide a **"gridmap-like" functionality** to perform identity mapping
  - can be enabled by specifying the **namemap-file** directive
  - **allows fine-grained control** over the identity mapping



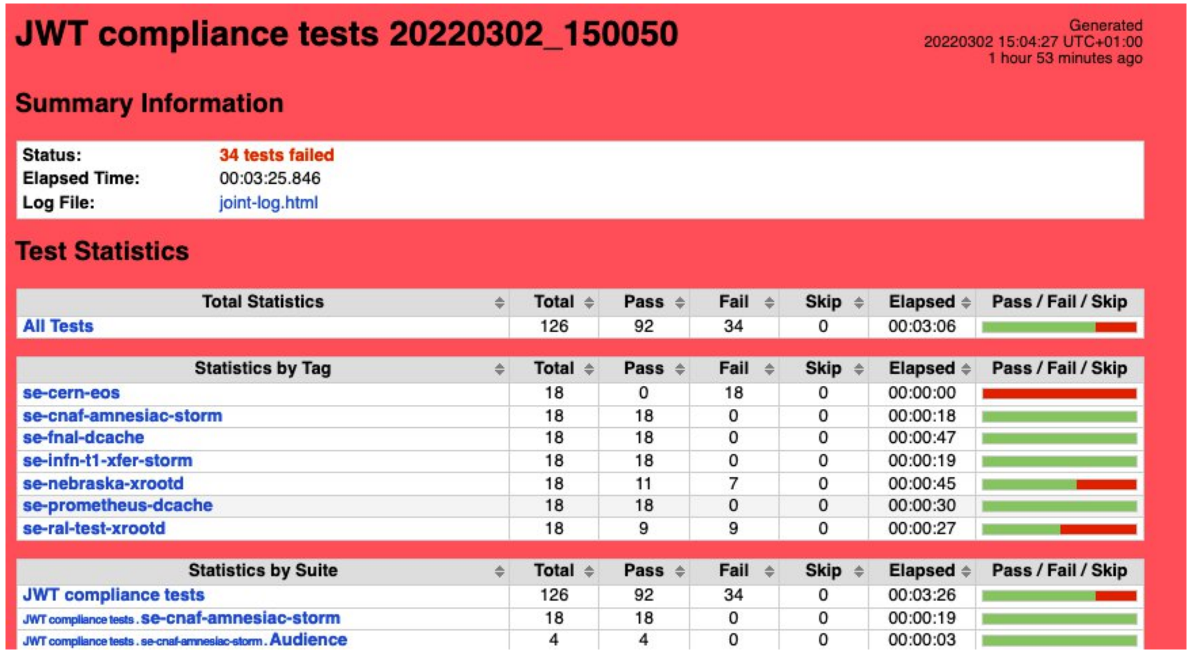
```
[ {"sub": "1234-567-89ab", "path": "/home/esindril", "result": "esindril"},
  {"group": "/it/test", "path": "/it", "result": "ittest", comment="Only for tests"},
  {"group": "/it", "result": "it"},
]
```

XrdSciTokens interactions



WLCG JWT compliance test-suite

- Effort within the **WLCG DOMA Bulk Data Transfers** meeting
- Testing framework to ensure compliance and interoperability of different Storage Systems
- **Ongoing** activities to standardize and adjust the tests



JWT compliance tests 20220302_150050

Generated
20220302 15:04:27 UTC+01:00
1 hour 53 minutes ago

Summary Information

Status:	34 tests failed
Elapsed Time:	00:03:25.846
Log File:	joint-log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	126	92	34	0	00:03:06	🟩🟥

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
se-cern-eos	18	0	18	0	00:00:00	🟥
se-cnaf-amnesiac-storm	18	18	0	0	00:00:18	🟩
se-fnal-dcache	18	18	0	0	00:00:47	🟩
se-infn-t1-xfer-storm	18	18	0	0	00:00:19	🟩
se-nebraska-xrootd	18	11	7	0	00:00:45	🟩🟥
se-prometheus-dcache	18	18	0	0	00:00:30	🟩
se-ral-test-xrootd	18	9	9	0	00:00:27	🟩🟥

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
JWT compliance tests	126	92	34	0	00:03:26	🟩🟥
JWT compliance tests . se-cnaf-amnesiac-storm	18	18	0	0	00:00:19	🟩
JWT compliance tests . se-cnaf-amnesiac-storm . Audience	4	4	0	0	00:00:03	🟩

Plans for the future

- Code restructuring and simplification
  - Drop eos repackaging of the XrdSciTokens library
  - Rely on the XrdSciTokens library provided by the XRootD 5 framework

- Have EOS comply with all the WLGC JWT tests

- Decide on the operational procedures and configuration requirements for providing local user mapping

- Gain operational experience with the newly introduced failure scenarios and ways to mitigate them

CERN

home.cern