Token capabilities and HTC-CE configuration

BY S. DAL PRA

INFN

AuthZ/IAM WS, 2022, Oct. 11

Email: `dalpra@infn.it`

Foreword

The following considerations are from the point of view of a HTCondor-CE on top of a HTCondor Batch system administrator in a WLCG Site (INFN-T1).

The "What If" motivation

Given current status of things, assuming that VOMS proxy support is removed from HTCondor-CE and only tokens are accepted:

- can **all** of our current use cases still be handled with WLCG Tokens?
- Can new or foreseeable future use cases also be handled?

If yes: how?

Bonus question

Can/should JWT extend / integrate / enrich capablilities provided by HTCondor itself?

New or Foreseeable scenarios

Computing resources becoming more heterogenous

- **CPUs:** `x86_64`, `ppc64le`, `aarch64`
- **GPUs:** several models, fast evolution

Different QoS tipologies

- High/Low priority queues (hierarchical fairshare)
- pledged/opportunistic resources
- High/Low power devices

Note: These can potentially cohexist (example: High priority access to opportunistic low power resources)

HTCondor-CE and Tokens

Once the client token is validated (`"scope"` inspection included), **two more steps** are following:

1. Map token credentials to a local authorized batch user

- Only `"iss"` and `"sub"` claim are considered

Note1 with `X509` proxy the `FQAN` would also be considered, i.e. the `"wlcg.groups"` claim. This is available at the second steps, when the job owner has been set already.

Note2 OSG tokens use to set the VOName in the “path” part of the `"iss"` claim, ex:

```
"iss": "https://scicomp.jlab.org/scitokens/clas12"
```

Note3 extending the above idea `"iss"` could also specify a user group, such as:

```
"iss": "https://some.issuer.org/somevo/sgm"
```

2. Apply Job Transform rules

The JobRouter is where further classad attributes for the incoming job can be set

- This is where capabilities/requirements for the routed job are to be set
- This is where **how** the token brings information can make a significant difference.

Examples: “I’m entitled to run on high priority queue”, “I have access to GPU resources”,...

The following claims are currently available, as classad attributes

```
"jti", "iss", "sub", "scope", "wlcg.groups"
```

Examples

Note The following are proposed for discussion: none of these are currently adopted in production environments, and they assume a specific way to bring needed information within the token.

Hierarchical Fairshare (with X509)

This is a use case currently handled with `X509`. The commonly adopted strategy to configure H.F. is:

1. The `FQAN` specify the H.F. group (i.e. `/virgo/virgo` vs `/virgo/ligo`)
2. An external service (i.e. `Argus`, `LCMAPS`) maps to local users in **corresponding unix groups** (i.e. Unix Group ↔ H.F. Group)
3. a text file defines the map username → H.F. Group (`AcctGroup` in HTCondor terms). The actual mapping is performed by the `UserMap` classad function.

Hierarchical Fairshare (with tokens)

The previous method cannot work with tokens because

- the username is set before evaluating groups.
- We need to support individual submission (i.e. no pilot factory)

In such a scenario we need to allow different jobs from the same user to belong to different subshares of its main group (VO).

Two strategies can be devised here, depending on the subshare group name being specified in the `wlcg.groups` claim, or in the `scope` claim.

Hierarchical Fairshare (with tokens)/2

Using wlcg.groups The strategy in this case would be:

1. The *first* element of the `wlcg.groups` list specify the H.F. Group (i.e. `/virgo/virgo` or `/virgo/ligo`)
2. A JobRouter clause join the values (Owner,HFgroup) in a string and invokes the `UserMap` classad function to set the appropriate `AcctGroup`.
3. A text file defines the mapping (Owner,HFgroup) ⟶ `AcctGroup`

- **Complete example** is detailed in the CERN wiki.
- **Main idea**: get a value from a known position in a list, and use that value as a search key in a lookup table.
- This method can be used to set different attributes as well.

Hierarchical Fairshare (with tokens)/3

Using scope The strategy in this case would be:

1. One *random* element of the scope claim specify the H.F. Group (i.e. `hfgroup:/virgo/virgo` or `hgroup:/virgo/ligo`)
2. A JobRouter clause check the `scope` for presence of each known HFgroup and set the `AcctGroup` classad attribute trough a chain of `IF THEN ELSE` clauses

Comments

- If two or more H.F.Group are present, the first (unpredictable) match will be considered. Group names are hardcoded in the JobRouter conf; the `IF ELSE` chain can complicate the rules; each change requires editing rules instead of
- values from the `scope` claim are being used at “different stages”: auth*zn at Grid side, Job routing at Batch side. Using different claims for the two might be good.

Other use cases:

The following have been tested by direct manual submission to a HTC-CE

Non x86

```
# Test 1.1: Job aarch64 / ppc64le

JOB_ROUTER_ROUTE_cms_arm @=jrt
  REQUIREMENTS (WantRoute =?= "cms_arm" &&\
 (AuthTokenIssuer =?= "https://cms-auth.web.cern.ch/"))
UNIVERSE VANILLA
  SET Requirements (TARGET.Arch =?= "aarch64")\
@jrt


JOB_ROUTER_ROUTE_cms_m100ITB @=jrt
  REQUIREMENTS (WantRoute =?= "cms_m100ITB") && (AuthTokenIssuer =?= "https://cms-
auth.web.cern.ch/" && StringListMember(AuthTokenSubject ?: "", "78f275d5-bb1a-4b2d-9956-
f82316a8482e:9662c0b5-31a1-4478-963e-bdf3783232ed",":"))

  UNIVERSE VANILLA
  SET Requirements (TARGET.Arch =?= "ppc64le")
@jrt
```

GPU

```
# JOB_ROUTER_ROUTE_virgo_gpu_v100 @=jrt

 REQUIREMENTS (WantRoute =?= "gems_V100") && (((x509UserProxyVoName =?= "virgo")\
 && RegExp("John Smith|Mario Rossi",x509userproxysubject)) ||
StringListMember(AuthTokenSubject,"9662c0b5-31a1-4478-963e-bdf3783232ed",":"))
  UNIVERSE VANILLA
  SET Requirements (TARGET.CUDACapability >= 6) &&\
        (TARGET.CUDADeviceName =?= "Tesla V100-SXM2-32GB")
  SET WantGPU True

@jrt
```

Comments

The above examples expect a Custom Attribute (`WantRoute`) to be present in the submit file. This could be passed by the pilot factory (Pandas can do that). In that case a generic user could reach the resource by direct submission. Passing that information in the token would enforce proper access.

Expressing capabilities

- In The Hierarchical Fairshare use case, retrieving the share group name from `wlcg.groups` looks like a natural choice, as this claim act much like `FQAN` in the VOMS proxy.
- The method adopted is quite general and just assumes that a value is present in a list at a known position

Example

```
"wlcg.groups": [
  "/virgo/ligo",        # the fairshare subgroup
  "QoS/opportunistic",  # can run on opportunistic resources
  "WantRoute/ppc64le"   # payload for specific CPU arch.
],
```

- using `wlcg.groups` in such a way might look inappropriate or not even possible
- A dedicated claim could be introduced, similar to `scope` but respecting elements order. In this case HTCondor-CE should provide its value in a classad attribute.