Access Token Verification

The PEP receives the request with the information of the user and the access token:

```
{
"didSP": "tangoUser",
"sar": {
"action": "GET",
"resource": "/temperature"
},
"didRequester": "did",
"accessToken":"eyJ..."
}
```

This access token presents this format:

Header

```
{
 "alg": "ES256",
 "kid": "Cwb9a33gUQ7Zl4jiIe27jY0sZ0yGtetUjljuAlotRZw",
 "typ": "JWT"
}
```

Payload

```
{
 "aud": [
  "portal.testing1.k8s-cluster.tango.rid-intrasoft.eu"
 ],
 "client_id": "did:web:ips.testing1.k8s-cluster.tango.rid-intrasoft.eu:did",
 "exp": 1725267658,
 "iss": "did:web:ips.testing1.k8s-cluster.tango.rid-intrasoft.eu:did",
 "kid": "Cwb9a33gUQ7Zl4jiIe27jY0sZ0yGtetUjljuAlotRZw",
 "sub": "did:my:wallet",
 "verifiableCredential": {
  "@context": [
   "https://www.w3.org/2018/credentials/v1",
   "https://w3id.org/security/suites/jws-2020/v1"
  ],
  "credentialSchema": {
   "id": "https://raw.githubusercontent.com/FIWARE-Ops/tech-x-
challenge/main/schema.json",
   "type": "FullJsonSchemaValidator2021"
  },
  "credentialSubject": {
   "email": "standard-employee@ips.org",
   "id": "eeb86f52-e6d0-4eb9-93bc-e3f10e22cae5",
      "lastName": "IPS",
      "roles": [
       {
        "names": [
         "GOLD_CUSTOMER"
        ],
        "target": "did:web:ips.k8s-cluster.tango.rid-intrasoft.eu:did"
       }
      ],
      "type": "gx:NaturalParticipant"
    },
    "id": "urn:uuid:d143edb2-688d-493f-a816-62148473e212",
    "issuanceDate": "2024-09-02T08:29:40Z",
    "issued": "2024-09-02T08:29:40Z",
    "issuer": "did:web:ips.k8s-cluster.tango.rid-intrasoft.eu:did",
    "proof": {
      "created": "2024-09-02T08:29:40Z",
      "jws":
"eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJQUzI1NiJ9..IBuOi35WgPsN3LA
9wFWYAn8LzSebU1c_D2wNrgy4gCE3NtDTDyLTvHoQo14xEi1cbMwZePn5rJ9vqB8Aq27
INW6dWe3T5j0M-
5BvDGK2OsGECX_DUBVx6Y5iFbGUvCSqNHgn1goHo5JvkFhzi6U5D1utVLlfM-
x9BEwCw1gvVNb6TsyvL2vEIUxenuMIpgN78dwFnqwCvUMBHtyzXUfnJdf-
jWMLdDXib1qJGEGSvm1nx_j7OrYwEUS_M8aZG7HZ6kps7I5odKGoBJs6SR9Vfn5k82xYc
GPwZZJ3TI738tEa5-ITeXZgDMWvLa4w6wu9KDclpmhbPaza7LVJNFcCkQ",
      "type": "JsonWebSignature2020",
      "verificationMethod": "did:web:ips.k8s-cluster.tango.rid-
intrasoft.eu:did#8164e7ea196a457f86b3f887ab69162b"
    },
    "type": [
      "EmployeeCredential"
    ],
    "validFrom": "2024-09-02T08:29:40Z"
  }
}
```

Signature

```
ECDSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  ,
Public key,
```

Private key
)

This Access Token will be verified using the public key that was used to issue its signature. This is set in an endpoint of the verifier: /.well-known/jwks
So, the PDP makes a call to this endpoint to get the JWKS:

http://ips-verifier.testing1.k8s-cluster.tango.rid-intrasoft.eu/.well-known/jwks

And this is the answer received:

```
{
"keys": [
{
"crv": "P-256",
"kid": "Cwb9a33gUQ7Zl4jiIe27jY0sZ0yGtetUjljuAlotRZw",
"kty": "EC",
"x": "VZ4w4GLQZPl1Ef86X-LmhmoYXZMNDmk9gVYCnnrG_vl",
"y": "-xfMDZzj8ChrVu0UXbicCeBDjfhYppMs5CXuWpHelQ"
}
]
}
```

There is a class in PEP/PDP system that interprets this response and processes this to finish the verification.

Policy Matching

To do this, we need to focus on the “verifiableCredential” field:

```
 "verifiableCredential": {
  "@context": [
   "https://www.w3.org/2018/credentials/v1",
   "https://w3id.org/security/suites/jws-2020/v1"
  ],
  "credentialSchema": {
   "id": "https://raw.githubusercontent.com/FIWARE-Ops/tech-x-
challenge/main/schema.json",
   "type": "FullJsonSchemaValidator2021"
  },
  "credentialSubject": {
   "email": "standard-employee@ips.org",
   "id": "eeb86f52-e6d0-4eb9-93bc-e3f10e22cae5",
   "lastName": "IPS",
   "roles": [
    {
     "names": [
      "GOLD_CUSTOMER"
     ],
     "target": "did:web:ips.k8s-cluster.tango.rid-intrasoft.eu:did"
    }
   ],
   "type": "gx:NaturalParticipant"
  }
```

The PDP uses the policies that are stored for the resource "temperature" and checks if the fields of the credential subject's path match with the statements of the policies that the requester needs to fit if he can access the resource.

An example of a policy could be:

1. The requester must reveal its id:

```
{
  "id": "<policyCounter_value>",
  "nombre": "Tango User Information",
  "purpose": "Reveal id of the user.",
  "serviceProvider": "did:serviceProvider:1",
  "accessRights": [
    {
      "resource": "/resource/temperature",
      "action": "GET"
    }
  ],
  "authTime": 123123123,
  "minTrustScore": 0.5,
  "constraints": {
    "fields": [
      {
        "path": ["$.id"]
      }
    ]
  }
}
```

2. The requester must reveal its id, and its last name must be "IPS":

```json
{
  "id": "<policyCounter_value>",
  "nombre": "Tango User Information",
  "purpose": "Reveal id and last name of the user.",
  "serviceProvider": "did:serviceProvider:1",
  "accessRights": [
    {
      "resource": "/resource/temperature",
      "action": "POST"
    }
  ],
  "authTime": 123123123,
  "minTrustScore": 0.5,
  "constraints": {
    "fields": [
      {
        "path": ["$.id"]
      },
      {
        "path": ["$.lastName"],
        "filter": {
          "type": "string",
          "pattern": "IPS"
        }
      }
    ]
  }
}
```