KOMBIT
OAuth Token Request Profile
Version 0.9

Status: Draft
Date: 12.04.2021

1 Introduction

This document contains a ‘sub-profile’ prepared by KOMBIT of the ‘OIO OIDC Profiles’ [OIO-OIDC] document from the Danish Agency for Digitisation. Definitions, descriptions and requirements from the base profile apply to this sub-profile as well and are not repeated here.

The goal of the sub-profile is to add support for a scenario where a system client in its own context requests a security token from an Authorization Server using OAuth 2.0, retrieves a JWT token, and subsequently uses the token to invoke a REST API offered by a Service Provider. Scenarios with 'pure' system clients are missing from the OIO OIDC profiles document from DIGST which only covers clients acting on behalf of human users.

The sub-profile aims at achieving equivalent functionality and security to the already established system user scenarios in KOMBIT based on OIO WS-Trust, OIO SAML, Liberty Basic SOAP Binding and OIO BPP – but with more modern technology. With the new profile it will be easier to provide and consume REST services with JWT tokens in KOMBIT's infrastructure as well as supporting new client types in the future.

This profile has a companion profile [KOMBIT-JWT] specifying the JWT token format including the required attributes / claims for a system user scenario.

2 Scenario overview

The profile enables the scenario illustrated below:

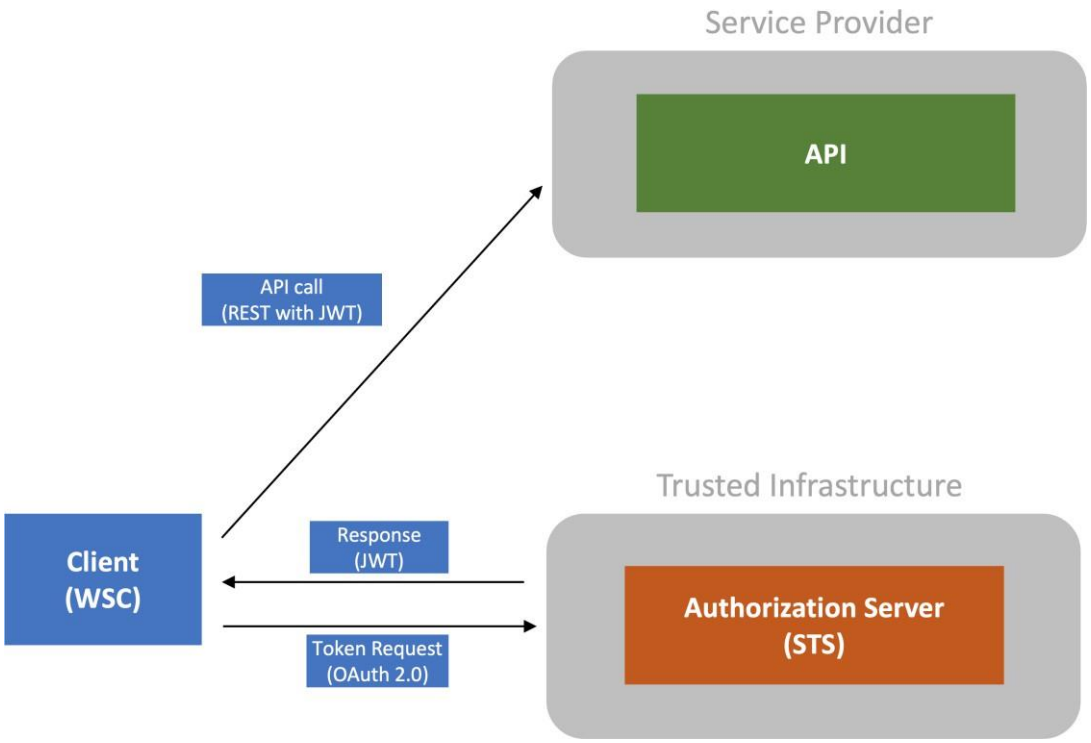

Figure 1: System user scenario overview

In the scenario a system client requests a token from an Authorization Server (aka STS) to be used at a particular REST API offered by a Service Provider. The main underpinnings of the scenario are as follows:

- The client system (WSC) is already registered with a credential (X.509 certificate) at the Authorization Server which can be used to authenticate requests for security tokens.
- The client acts in its own context (as a system user) and not in context of a human user.
- The client has been authorized using an out-of-band procedure - e.g. it could be agreement based[1]. The authorization for the Service Provider is represented in the security token issued by the Authorization Server using roles and data restrictions following the OIO BPP model (but with slightly different syntax suitable for JWT tokens).
- The Service Provider trusts tokens issued and signed by the Authorization Server, and trust anchors have been established in advance such that the Service Provider can verify incoming tokens.

[1] In KOMBIT's implementation it would be based on service agreements.

3 Token Request Profile

This chapter describes a token request profile based on the OAuth 2.0 client credentials grant flow. In the profile, the client requests an Access Token (for a specific SP API and with scopes covering this API) from an Authorization Server and authenticates using a client certificate. If multiple Service Provider APIs are to be accessed by the client, multiple tokens have to be requested.

The Authorization Server verifies that the presented certificate is valid and registered with a known and approved client, and subsequently issues an JWT Access Token [KOMBIT-JWT] which contains the relevant scopes encoded as privileges in a JSON structure.

3.1.1 Step 1: Client sends Token Request

[TRP-1]

The token request MUST use the [OAuth] 2.0 client credentials grant type as defined in section 4.4 of [OAuth]. Unless otherwise stated explicitly, the requirements from the [OAuth] specification apply directly.

[TRP-2]

The request parameters in the token request MUST fulfill the requirements specified in the table below:

Parameter	Man-da-tory	Usage
grant_type	Y	Value MUST be `client_credentials`
scope	Y	The scope parameter contains a comma-separated list of 'objects' specifying what the tokens should provide access to. In KOMBIT's implementation the scope MUST be exactly one EntityID specifying the Service Provider and one anvenderkontekst being a CVR-number or a short-hand[2] for a group of CVR-numbers. The scope should have the following syntax: scope=entityid:<value>,anvenderkontekst:<value> and subsequently be URL encoded for transmission.

[2] The list of allowed short-hand / wildcard strings representing multiple CVR numbers is handled outside this profile document.

A sample Token Request is shown below:

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type:  application/x-www-form-urlencoded

client_id=https%3A%2F%2Fclient.example.org%2Fcb
&grant_type=client_credentials
&scope=entityid%3Ahttp%3A%2F%2Fbeskedfordeler.kombit.dk%2Canvenderkontekst%3AK98
```

[TRP-3]

The token request MUST be authenticated using a TLS client certificate that has been registered out-of-band with the Authorization Server.

3.1.2 Step 2: Authorization Server validates request and returns token

[TRP-4]

The Authorization Server MUST verify that the TLS client certificate is valid.

[TRP-5]

The Authorization Server MUST verify the request, including that requested `scope` values have been previously been authorized to the client. If the request contains `scope` values which the client is not authorized to (e.g. an EntityID of Service Provider), the request MUST be rejected and no token issued. Any error responses MUST follow section 5.2 of RFC 6749 and both include an `error` parameter and an `error_description` with the detailed reason for the error.

[TRP-6]

If the request is successful, the Authorization Server MUST issue and sign an Access Token complying with the [KOMBIT-JWT] Profile corresponding to the requested and granted access.

[TRP-7]

The Access Token MUST be a holder-of-key token where a SHA-256 thumbprint of the client certificate is included as a claim in the token (the `x5t#S256` claim). This means that the token can only be presented by clients in possession of the corresponding private key (assuming the Service Provider validates the certificate and token as required - see requirement AAP-4).

[TRP-8]

The Access Token SHOULD have a validity period of no more than 8 hours.

A sample Token Response is shown below - please consult [KOMBIT-JWT] for a description of the token format:

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "token_type": "Holder-of-key",
   "expires_in": 3600,
   "access_token":
   ”eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
     yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
     NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
     fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
     AKfQ.gga8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
     Jp6IcmD1HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
     NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
     QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
     K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
     XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
  }
```

Note: Since Access Tokens are long-lived and new tokens can simply be requested, no provisions for Refresh Tokens have been made.

4 API Access Profile

This chapter describes how a client can invoke an external SP API using an Access Token obtained via the mechanisms described in the Token Request profile in chapter 3.

[AAP-1]

The client MUST use TLS towards the Service Provider using the same client certificate that was used to retrieve the Access Token from the Authorization Server.

[AAP-2]

The client MUST pass the Access Token in an `Authorization` HTTP header with token type `Holder-of-key` as shown below:

Example:

```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Holder-of-key
7Fjfp0ZBr1H8JgaJs97Jb.8shJgaJs97Jb.asd&DSasdaJs97Jb
```

Note: In the above example, the token is very small. In practice, tokens can be much bigger and limitations on HTTP header sizes (usually around 8 KB) should be observed.

[AAP-3]

The Service Provider MUST validate the received Access Token including (as a minimum) that:

- The token is not expired.
- The token is signed by a trusted Authorization Server using an allowed algorithm.
- The Service Provider itself is the intended audience of the token (`aud` field).
- The required privileges for the request are included (`priv` field). See the [KOMBIT-JWT] profile for details.

[AAP-4]

The Service Provider MUST validate the Holder-of-key relation by comparing client certificate used for TLS establishment to the SHA-256 thumbprint included in the token in the `x5t#S256` field[3]. The digest value MUST match exactly or the request MUST be rejected.

[3] The `x5t#S256` (X.509 certificate SHA-256 thumbprint) parameter is a base64url-encoded SHA-256 thumbprint (a.k.a. digest) of the DER encoding of the X.509 certificate [RFC5280] used as client certificate.

5 Security Requirements

The security requirements below apply to sections of this document.

[SR-1]

All transport communication MUST use TLS 1.2 or higher and SHOULD only use cipher suites supporting perfect forward secrecy. Servers MUST reject negotiation of insecure TLS connections. The document [NIST 800-52] (section "Minimum Requirements for TLS Servers") or subsequent revision may serve as reference for an acceptable level of transport security.

[SR-2]

Access Tokens MUST comply to the KOMBIT JWT Profile [KOMBIT-JWT].

[SR-3]

The private key used to sign the JWT Tokens by the Authorization Server MUST be protected by strict access control and SHOULD be placed in a hardware security module.

6 References

- [OIO-OIDC] "OIO OIDC Profiles 0.3", Danish Digitisation Agency.
- [KOMBIT-JWT] "KOMBIT JWT Profile".
- [JWA] Jones, M., "JSON Web Algorithms (JWA)," draft-ietf-jose-json-web-algorithms (work in progress), July 2014.
- [JWE] Jones, M., Rescorla, E., and J. Hildebrand, "JSON Web Encryption (JWE)," draft-ietf-jose-json-web-encryption (work in progress).
- [JWK] Jones, M., "JSON Web Key (JWK)," draft-ietf-jose-json-web-key (work in progress), July 2014.
- [JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)," draft-ietf-jose-json-web-signature (work in progress), July 2014.
- [JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)," draft-ietf-oauth-json-web-token (work in progress), July 2014.
- [BBA] Parecki, Waite: "OAuth 2.0 for Browser-Based Apps", IETF draft.
- [NSIS] "National Standard for Identiteters Sikringsniveauer 2.0.1". https://digst.dk/it-loesninger/nemlog-in/det-kommende-nemlog-in/vejledninger-og-standarder/nsis-standarden/
- [OIOSAML] "OIOSAML Web SSO Profile 3.0". https://digst.dk/it-loesninger/nemlog-in/det-kommende-nemlog-in/vejledninger-og-standarder/oiosaml-30/
- [OIO-BPP] "OIO Basic Privilege Profile 1.1". https://www.digitaliser.dk/resource/2377872
- [RFC6819] "OAuth 2.0 Threat Model and Security Considerations", IETF. https://tools.ietf.org/html/rfc6819
- [RFC8252] "OAuth 2.0 for Native apps", IETF.
- [RFC6750] "The OAuth 2.0 Authorization Framework: Bearer Token Usage", IETF, https://tools.ietf.org/html/rfc6750
- [RFC7009] "OAuth 2.0 Token Revocation", IETF.
- [OIDC] "OpenID Connect Core 1.0 incorporating errata set 1, November 2014", OpenID.Net.
- [OAuth] "The OAuth 2.0 Authorization Framework", RFC6749, IETF, October 2012.
- [OIO JWT] "OIO JWT Token Profile", Danish Agency for Digitisation.
- [NSIS] "National Standard for Identiteters Sikringsniveauer 2.0.1", Digitaliseringsstyrelsen. https://digst.dk/it-loesninger/nemlog-in/det-kommende-nemlog-in/vejledninger-og-standarder/nsis-standarden/

- [OSBP] ”OAuth 2.0 Security Best Current Practice”, IEFT. https://data-tracker.ietf.org/doc/draft-ietf-oauth-security-topics/