OpenID vs OAuth 2.0

https://openid.net/connect/

OpenID is defined on top of the OAuth 2.0 protocol. It adds the following requirements to the general OAuth 2.0 HTTP exchanges, like:

- Location of the server's metadata:
  - `/.well-known/openid-configuration` instead of `/.well-known/oauth-authorization-server`
  - deals with an optional path component differently (RFC8414-3 vs OpenID spec)
- Client must add `openid` to `scope` in Authentication/Authorization Request
- Token Endpoint returns additional token called `id_token` (along with `access_token`)
  - `id_token` is a JWT containing at least the following fields (see OpenID-token):
    - `iss` (issuer identifier) - Authorization Server URI
    - `sub` (subject identifier) - unique user ID
    - `aud` (audience) - must contain `client_id`
    - `exp` (expiration time)
    - `iat` (time when JWT was issued)

`id_token` may contain different claims about the user as it is specified in OpenID-claims (like name, email, gender etc). Particular claims can be requested by adding additional values to the `scope` parameter (OpenID-scope-claims). Claims can be also queried from UserInfo Endpoint (OpenID-UserInfo) with obtained `access_token`.

In general, `id_token` is supposed to contain the user's identity while `access_token` contains scopes (access rights). Moreover, `id_token` is issued rather for a client and the client is supposed to validate it (using the values `iss`, `aud`, `exp`).

Although, according to RFC7662-2.2, an `access_token` may also contain the following fields (they can be queried from the server via Introspection Request or coded inside the token):

- `iss`
- `sub`
- `aud`
- `exp`
- `iat`
- `client_id`
- `username`

User identity in a printer

PWG_GetUserPrinterAttributes (section 4.1)
RFC8011-9.3
PWG_Job and Printer Extensions – Set 3 (section 5.1.6)
PWG_IPP System (section 7.1.7)

The Client can define the user identity by setting the following IPP attributes (in the Operation Attributes group):

- `requesting-user-name (name(MAX))`
- `requesting-user-uri (uri)`
- `requesting-user-vcard (1setOf text(MAX))`

BUT, the printer cannot use these attributes and must instead retrieve the user's identity from the `access_token`. Otherwise, the Client would be able to easily impersonate other users. So, I see two possible solutions here:

- `access_token` must contain user's identity and the printer is able to get it in one of the following ways:
  - `access_token` is JWT and user ID is inside, OR
  - user ID can be obtained via Token Introspection
- OpenID is used and the obtained `token_id` is sent to the printer (how?)
  - `token_id` must be decoded and verified by the printer - the printer must get `client_id` from the `access_token` to do that

Problems

- We do not know if the printer needs user identity
- We do not know what kind of user info the printer needs

Proposed solution

- If a printer needs user identity, it must acquire it from obtained `access_token`
- User ID required by the printer is stored in `access_token` as a `sub` value
- OpenId is not required to have user ID included in the `access_token`

RAR-based protocol

https://datatracker.ietf.org/doc/draft-ietf-oauth-rar/
https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml

This protocol is based on the “Example 2” described in the previous document I sent. I have introduced the following changes:

- `access_level` is removed from the `authorization_details`
- endpoint access token contains additional fields: `aud`, `iss`, `sub`
- the IPP attribute `oauth-groups` can be skipped

1. Client sends Get-Printer-Attributes to the printer

The client queries the printer "https://my.printer.intranet/ipp/print" with Get-Printer-attributes. The printer returns the following IPP attributes:

```
oauth-authorization-server-uri: “https://auth.server.intranet”
oauth-types-supported: [ "https://pwg.org/ipp/oauth2/basic" ]
oauth-groups: [ "printersA", “printersB” ]  // groups of printers/users the printer belongs to
```

The last parameter (groups) is optional, if skipped then any OAuth2 session with the given server can be used.

2. Client checks existing OAuth sessions with AS

The client chooses "https://pwg.org/ipp/oauth2/basic" as the type of authorization_details (the printer does not support anything else so there are no other choices).

First, the client checks if it already has any access token from AS that matches the following criteria:

- Issued by the AS “https://auth.server.intranet”.
- Has in groups any of the following: “printersA”, “printersB”.

(The second condition is not checked if the printer does not have the `oauth-groups` attribute.)

If the existing OAuth session matching these criteria is found, the Client reuses the existing access token to get an endpoint access token via Token Exchange Request (go to step 4). Otherwise, the Client initiates an authorization procedure with the server to obtain a new access token (go to step 3).

3. Client goes through an authorization procedure and gets new access token

The client sends to the AS an Authorization Request with the following parameter:

```
authorization_details = {
```

```
  "type": "https://pwg.org/ipp/oauth2/basic",
  "groups": [ "printersA", "printersB" ],
}
```

The parameter `groups` is skipped if the printer does not have the `oauth-groups` attribute. After completing the authorization procedure the client sends Token Request and receives a response with the following parameter:

```
authorization_details = {
  "type": "https://pwg.org/ipp/oauth2/basic",
  "groups": [ "printersA" ],
}
```

4. Client sends Token Exchange Request and gets a new endpoint access token

The Client sends a Token Exchange Request with the current access token and a resource_id equals "https://my.printer.intranet/ipp/print" (the printer's URL must be verified by its certificate). The AS returns an endpoint access token that must be used in communication with the printer.

5. The printer verifies the (endpoint) access token

The printer sends a Token Introspection Request with the endpoint access token OR decode the token (if it is a JWT). The AS's response or decoded content contains the following parameters:

```
authorization_details = {
  "type": "https://pwg.org/ipp/oauth2/basic",
  "groups": [ "printersA" ],
}
aud = https://my.printer.intranet/ipp/print
iss = https://auth.server.intranet
sub = user.id@as.email
```

The printer must verify `authorization_details`, `aud` and `iss`.

Other solution: code the user identity inside `authorization_details` (not only in `sub`)