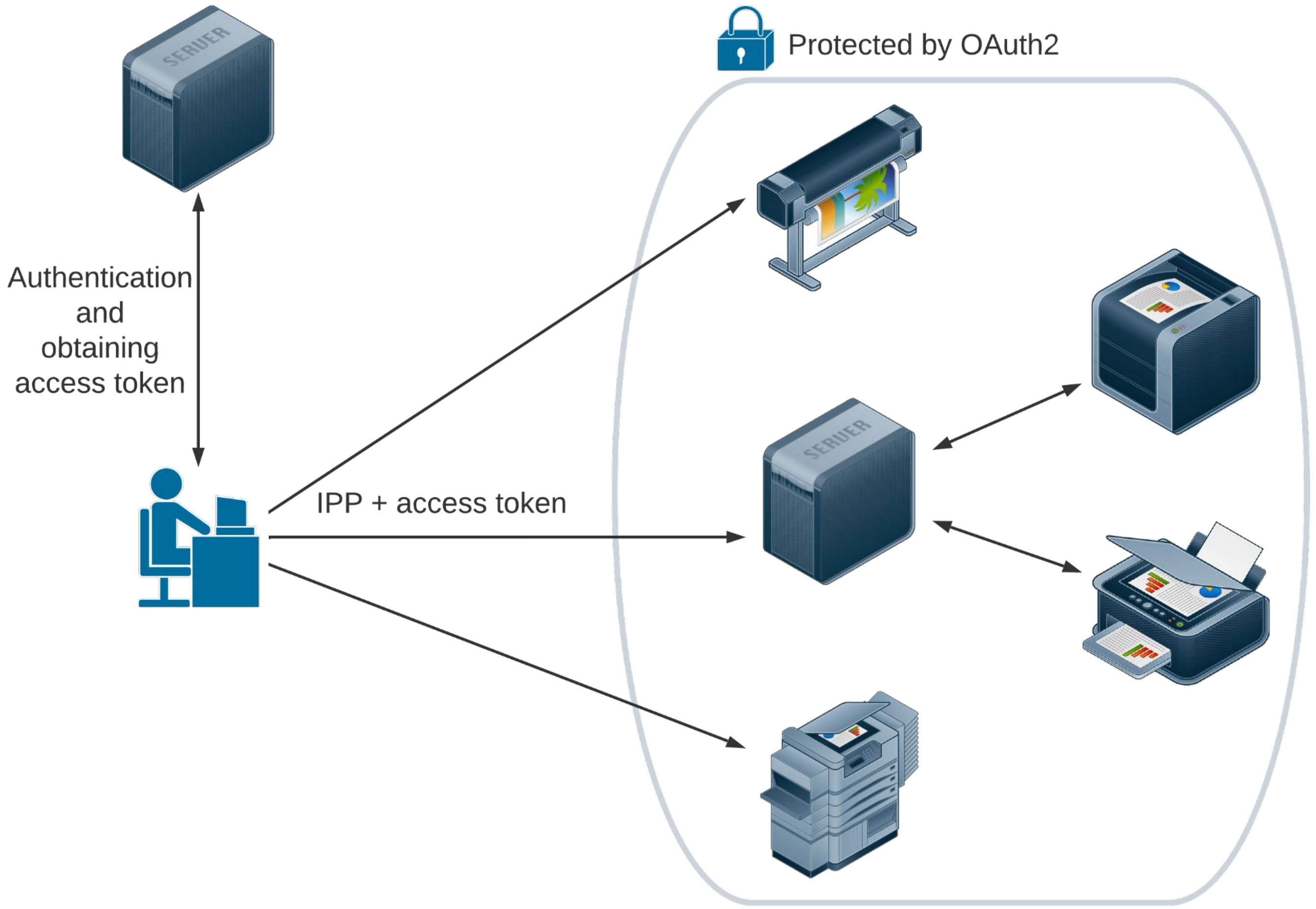
Google Open Source

Restricting access to IPP printers with OAuth 2 framework

Linux Plumbers Conference 2022

Piotr Pawliczek
(pawliczek@chromium.com)



Google Open Source

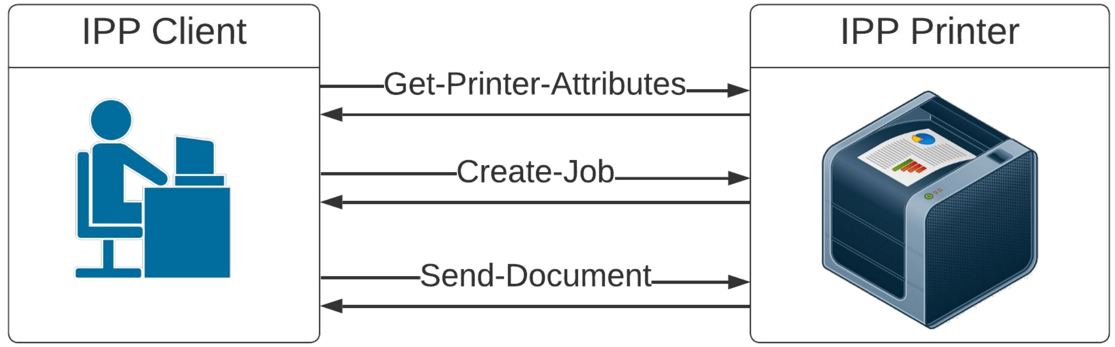
Agenda

1. How to fit OAuth 2 into IPP?
2. Security implications
3. Proposed protocol
4. Current status and Discussion

How to fit OAuth 2 into IPP?

Google Open Source

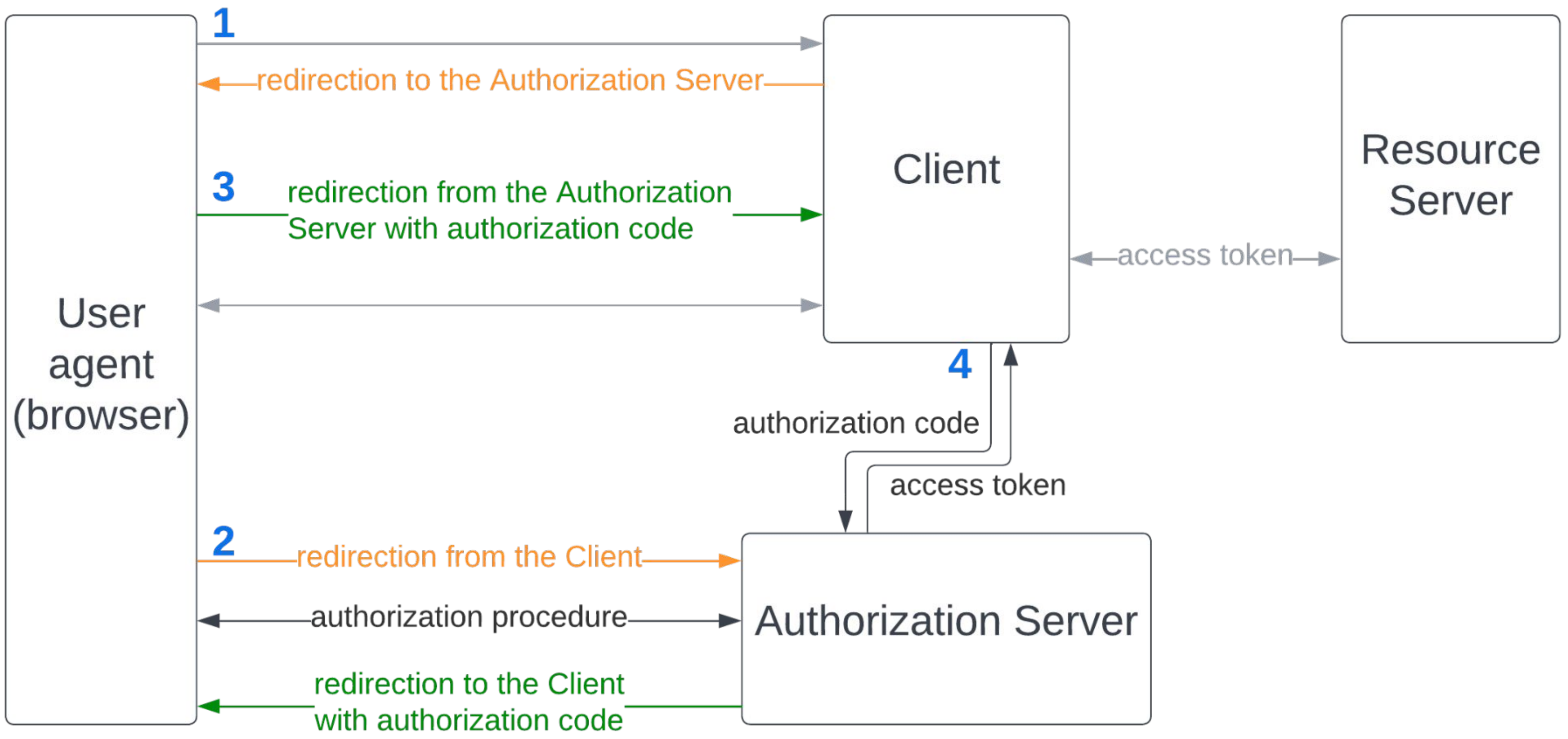
IPP = Internet Printing Protocol



IPP Printer may be:

- a real printer
- an interface exposed by a print server
- an interface exposed by a cloud print (infrastructure printer)

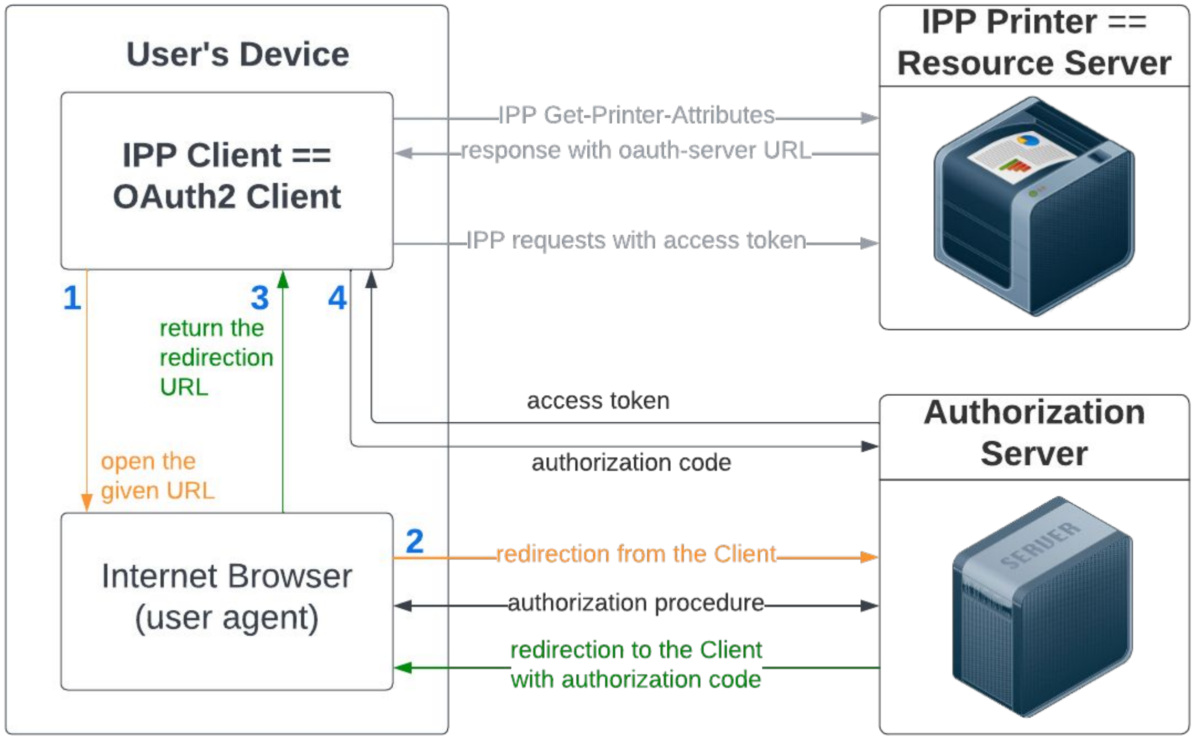
OAuth 2 with Authorization Code



1
redirection to the Authorization Server
3
redirection from the Authorization Server with authorization code
Client
Resource Server
access token
User agent (browser)
4
authorization code
access token
2
redirection from the Client
authorization procedure
Authorization Server
redirection to the Client with authorization code

Google Open Source

OAuth 2 for IPP

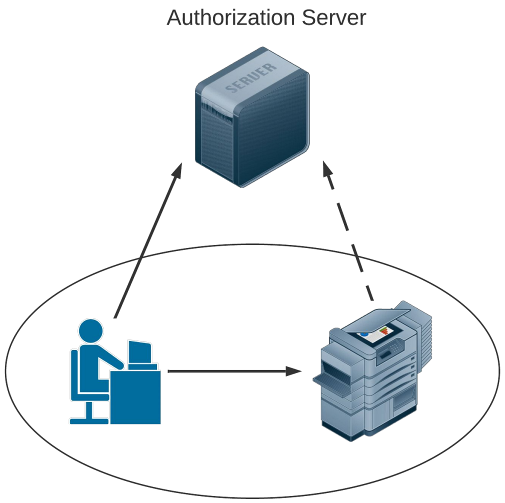


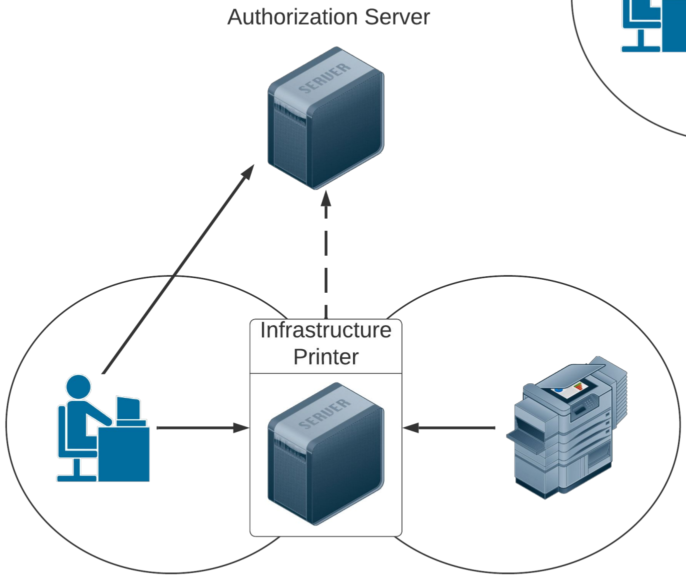
User's Device
IPP Client == OAuth2 Client
Internet Browser (user agent)
IPP Printer == Resource Server
Authorization Server
IPP Get-Printer-Attributes
response with oauth-server URL
IPP requests with access token
1
open the given URL
2
redirection from the Client
authorization procedure
redirection to the Client with authorization code
3
return the redirection URL
4
authorization code
access token

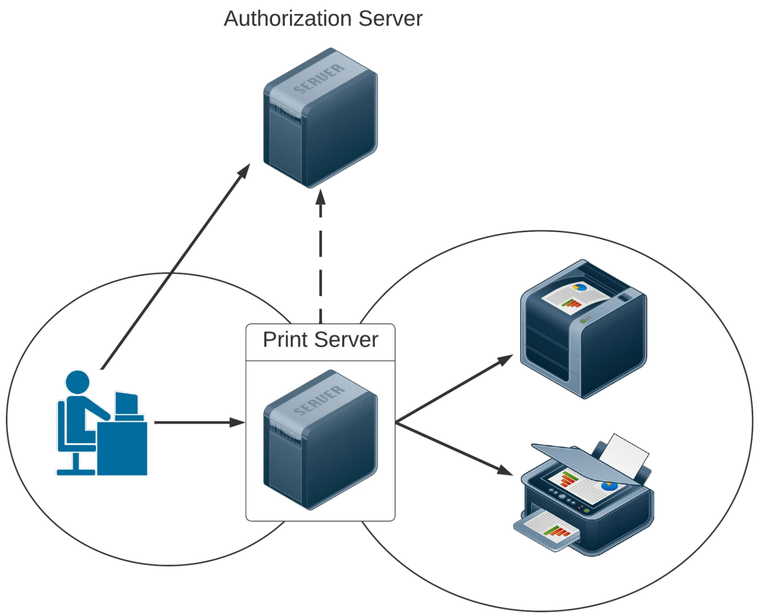
Main Assumptions

- **IPP Printer** can be managed by only one **Authorization Server**
- **IPP Printer** knows the URL of its **Authorization Server**
- **IPP Client** does not need any prior knowledge about the implementation of **IPP Printer** or **Authorization Server**
- **IPP Printer** does not need any prior knowledge about the implementation of **IPP Client**
- All communication between **IPP Client** and **IPP Printer** and between **IPP Client** and **Authorization Server** relies on https protocol
- **IPP Client** is a public OAuth 2 client and uses only the OAuth 2 flow with Authorization Code and PKCE (RFC 7636)

Possible configurations







Security implications

Google Open Source

Why do we need OAuth 2?

1. Communication between **IPP Client** and **IPP Printer** cannot be intercepted by any third party.
   The immediate goal: to protect user data.

2. Access to **IPP Printer** can be restricted to a limited set of authorized users.
   The immediate goal: to protect printer resources (e.g., paper, ink, printing time, etc.).

The second condition may be achieved only if the first requirement is fulfilled. Otherwise, attackers would be able to intercept credentials/access tokens and impersonate authorized users.

IPP Client - initial configuration / discovery

- **IPP Client** has no a priori knowledge about the Printing System
- Possible sources of **IPP Printers** addresses (URLs):
  - Discovered via mDNS (Bonjour/zeroconf)
  - Provided by a user
  - Queried from a print server
- Possible sources of **Authorization Server** URL:
  - Preconfigured / provided by a user
  - Queried from the **IPP Printer**
- **IPP Printers** and **Authorization Server** MUST use https and have valid certificates

What do certificates and TLS give us?

- Encryption of the whole point-to-point communication
- Guarantee that we communicate directly with the host with given name (domain)
  - Man-in-the-middle attack is not possible on this level

What do we not get?

- Knowledge what the host with given name (domain) really do
  - (Internet) Everyone can buy a domain and a certificate for it
  - (Local network) Potential attackers may take control over one or two hosts

Google Open Source

Authorization Server - the initial point of trust

- The URL of the **Authorization Server** MUST be verified
- Possible solutions:
    - Preconfigured
        - FQDN of well-known public service
        - Provided by the administrator of the system/local network
    - Entered by a user
        - Copied from some manual or instruction
        - Provided by the **IPP Printer** - very risky !!!
            - The user must acknowledge that the obtained URL is trusted

How to verify **IPP Printer**?

How do we now if the given host is really an **IPP Printer**?

- **IPP Client** does not know how to verify printer’s address - can only verify its certificate
- Any host with a valid certificate can claim to be a printer

The **Authorization Server** must verify the identity of the **IPP Printer** before the **IPP Client** sends any sensitive data to it

- **IPP Client** must send to the **Authorization Server** the URL of the **IPP Printer** (with hostname matching the **IPP Printer**’s certificate)
- The **Authorization Server** must be able to check if the printer belongs to its zone

Verification of **IPP Printer** - proposed solution:

Extend the protocol by additional request Token Exchange (RFC 8693)

- User must authorize to the **Authorization Server** to obtain the *access token*
- Client sends Token Exchange to the **Authorization Server**
    - The request contains:
        - *access token*
        - the **IPP Printer**'s URL (as a value of the parameter *resource*)
    - The **Authorization Server** returns:
        - *endpoint access token* (for this particular **IPP Printer**), OR
        - error *invalid_target* (means "Not my printer")

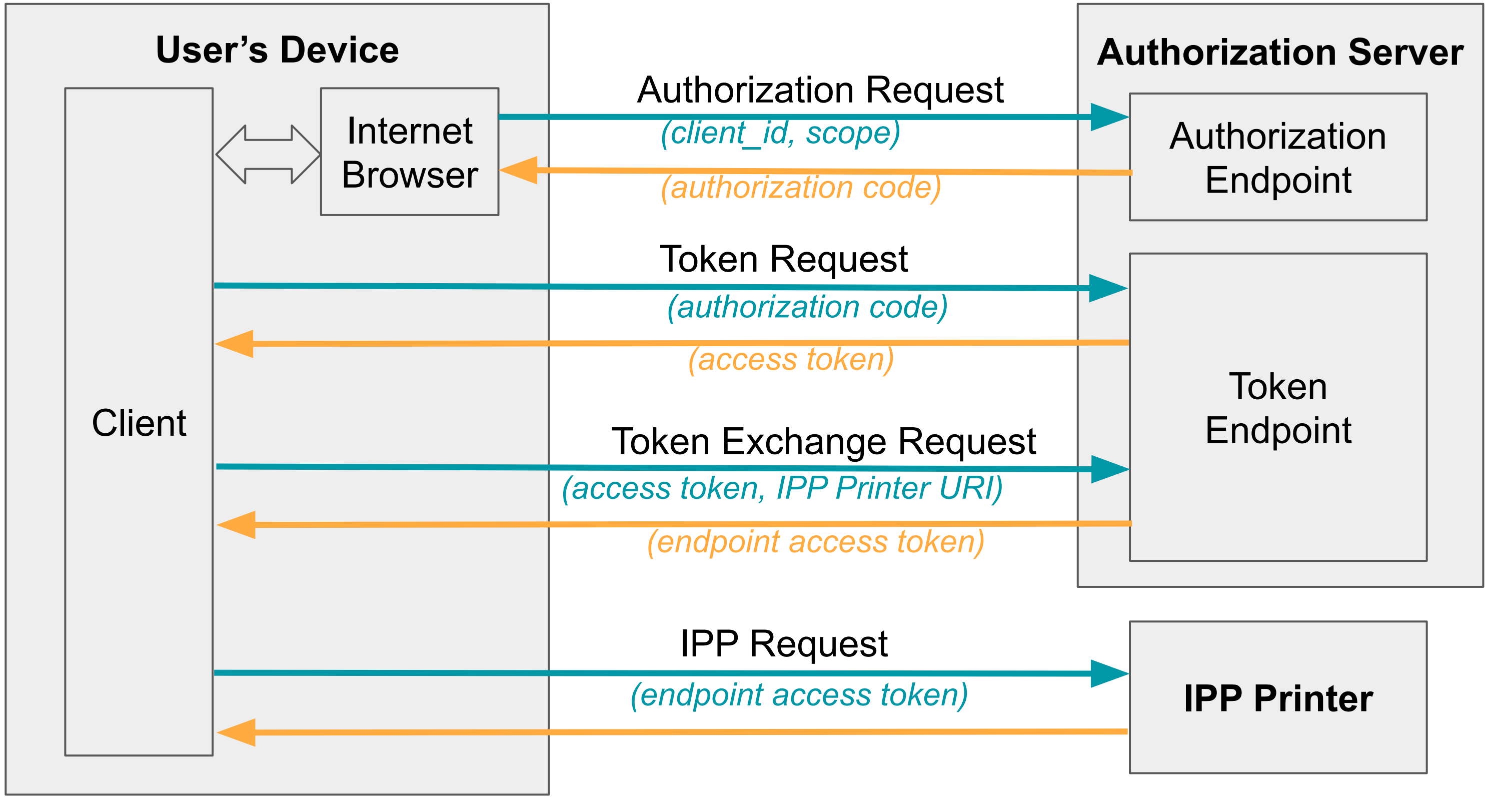
Token Exchange - pros and cons

- Cons
  - **Authorization Server** must support Token Exchange request
  - More complicated implementation of **IPP Client**
- Pros
  - More secure:
    - **IPP Printers** do not know the main *access token*
    - *endpoint access token* can be different for every **IPP Printer**
  - More convenient for users:
    - As long as the main *access token* is valid, Token Exchange is performed in the background and needs no user's interactions

Are alternative solutions possible?

- Obtaining URLs of **IPP Printers** from trusted source only
  - Discovery via mDNS and user’s input not possible
  - URLs are queried from an endpoint provided by the **Authorization Server**
- Different rules for different **IPP Printers**
  - More complicated protocol and implementation of **IPP Client**
  - Some **IPP Printers** will need verification anyway
- Others ?
  - We are open to ideas!

Proposed protocol



Google Open Source

Proposed protocol

1. **IPP Printer** managed by **Authorization Server** MUST return attributes:
    a. *oauth-authorization-server-uri* (always)
    b. *oauth-authorization-scope* (if needed).
2. **IPP Client** MUST:
    a. check that *oauth-authorization-server-uri* is on the list of trusted servers
    b. query metadata from the **Authorization Server** as described in RFC 8414
    c. try to register as a new client as described in RFC 7591 when:
        i. *client_id* is not known, AND
        ii. the **Authorization Server** allows for dynamic registration of new clients.

Proposed protocol

1. **IPP Client** MUST open session with **Authorization Server** as described in RFC 6749:
    a. the **IPP Client** uses an internet browser to open authorization link from **Authorization Server** and enables the user to complete authentication procedure provided by the server;
    b. the **IPP Client** obtains *access token* (and, if provided, *refresh token*) from the **Authorization Server**
2. The **IPP Client** uses *access token* to obtain *endpoint access token* for specific **IPP Printer** as described in RFC 8693
    a. the **IPP Client** sends to the **Authorization Server** the URL of the **IPP Printer**

Current status and Discussion

Google Open Source

Current status

- The PWG group (https://www.pwg.org/ipp/) works on the protocol
- Some OAuth2 requests are already implemented in CUPS 2.4
  - more functionality is coming in CUPS 2.5/3.0
- Work on a prototype **IPP Client** in ChromeOS
  - experimental feature
  - activated by enabling a feature flag
- Gathering feedback and opinions

Known issues - any input/feedback is appreciated

- Possible alternatives for extending the protocol with Token Exchange request?
- How to verify identity of **IPP Printers** with local addresses?
  - Add a fingerprint of the certificate to the **IPP Printer's** URL
  - Use mDNS name in the **IPP Printer's** URL
    - Is it secure?
- How to use OAuth 2 scopes?
  - Value of *scope* can be provided to **IPP Client** by **IPP Printer**
  - **IPP Client** does not have to understand *scope*

Questions and Discussion

Google Open Source