Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2011

D. Recordon, Ed.
B. Goldman
Facebook

Jul 2010

OAuth 2.0 Device Profile
draft-recordon-oauth-v2-device-00

Abstract

The device profile is suitable for OAuth 2.0 clients executing on devices which do not have an easy data-entry method (e.g. game consoles or media hubs), but where the end-user has separate access to a user-agent on another computer or device (e.g. home computer, a laptop, or a smart phone).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 2, 2011.

Copyright Notice

Table of Contents

1. Introduction

1.1. Notational Conventions

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in [RFC2119].

1.2. Terminology

device endpoint
: The authorization server's HTTP endpoint capable of issuing verification codes, user codes, and verification URLs.

device verification code
: A short-lived token representing an authorization session.

end-user verificition code
: A short-lived token which the device displays to the end user, is entered by the end-user on the authorization sever, and is thus used to bind the device to the end-user.

1.3. Overview

The device profile is suitable for clients executing on devices which do not have an easy data-entry method (e.g. game consoles or media hubs), but where the end-user has separate access to a user-agent on another computer or device (e.g. home computer, a laptop, or a smart phone). The client is incapable of receiving incoming requests from the authorization server (incapable of acting as an HTTP server).

Instead of interacting with the end-user's user-agent, the client instructs the end-user to use another computer or device and connect to the authorization server to approve the access request. Since the client cannot receive incoming requests, it polls the authorization server repeatedly until the end-user completes the approval process.

This device flow does not utilize the client secret since the client executables reside on a local device which makes the client secret accessible and exploitable.

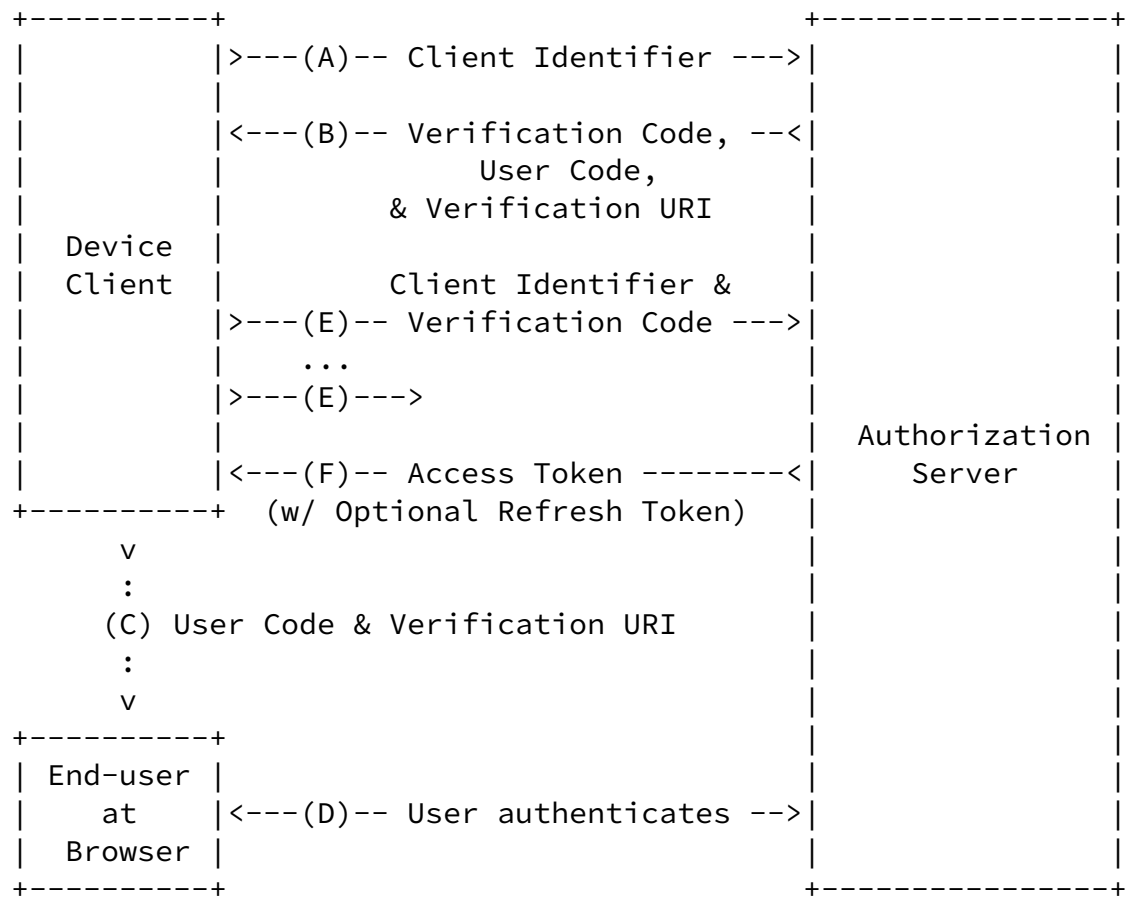

Figure 1: Device Flow

The device flow illustrated in Figure 1 includes the following steps:

(A) The client requests access from the authorization server and includes its client identifier in the request.

(B) The authorization server issues a verification code, an end-user code, and provides the end-user verification URI.

(C) The client instructs the end-user to use its user-agent (elsewhere) and visit the provided end-user verification URI. The client provides the end-user with the end-user code to enter in order to grant access.

(D) The authorization server authenticates the end-user (via the user-agent) and prompts the end-user to grant the client's access request. If the end-user agrees to the client's access request, the end-user enters the end-user code provided by the client. The authorization server validates the end-user code provided by the end-user.

(E) While the end-user authorizes (or denies) the client's request (D), the client repeatedly polls the authorization server to find out if the end-user completed the end-user authorization step. The client includes the verification code and its client identifier.

(F) Assuming the end-user granted access, the authorization server validates the verification code provided by the client and responds back with the access token.

1.4. Client Requests Authorization

The client initiates the flow by requesting a set of verification codes from the authorization server by making an HTTP "POST" request to the device endpoint. The client constructs a request URI by adding the following parameters to the request:

response_type
: REQUIRED. The parameter value MUST be set to "device_code".

client_id
: REQUIRED. The client identifier as described in Section 2 of

[I-D.ietf.oauth-v2].

scope
: OPTIONAL. The scope of the access request expressed as a list of space-delimited strings. The value of the "scope" parameter is defined by the authorization server. If the value contains multiple space-delimited strings, their order does not matter, and each string adds an additional access range to the requested scope.

For example, the client makes the following HTTPS request (line breaks are for display purposes only):

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

response_type=device_code&client_id=s6BhdRkqt3
```

In response, the authorization server generates a verification code and an end-user code and includes them in the HTTP response body using the "application/json" format with a 200 status code (OK). The response contains the following parameters:

device_code
: REQUIRED. The verification code.

user_code
: REQUIRED. The end-user verification code.

verification_uri
: REQUIRED. The end-user verification URI on the authorization server. The URI should be short and easy to remember as end-users will be asked to manually type it into their user-agent.

expires_in
: OPTIONAL. The duration in seconds of the verification code lifetime.

interval
      OPTIONAL. The minimum amount of time in seconds that the client SHOULD wait between polling requests to the token endpoint.

For example:

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

{
  "device_code":"74tq5miHKB",
  "user_code":"94248",
  "verification_uri":"http://www.example.com/device",
  "interval"=5
}
```

The client displays the end-user code and the end-user verification URI to the end-user, and instructs the end-user to visit the URI using a user-agent and enter the end-user code.

The end-user manually types the provided verification URI and authenticates with the authorization server. The authorization server prompts the end-user to authorize the client's request by entering the end-user code provided by the client. Once the end-user approves or denies the request, the authorization server informs the end-user to return to the device for further instructions.

1.5. Client Requests Access Token

Since the client is unable to receive incoming requests from the authorization server, it polls the authorization server repeatedly until the end-user grants or denies the request, or the verification code expires.

The client makes the following request at an arbitrary but reasonable

interval which MUST NOT exceed the minimum interval rate provided by the authorization server (if present via the "interval" parameter). Alternatively, the client MAY provide a user interface for the end-user to manually inform it when authorization was granted.

The client requests an access token by making an HTTP "POST" request to the token endpoint as described in Section 4.1.1 of [I-D.ietf.oauth-v2]. The "redirect_uri" parameter is NOT REQUIRED as part of this request.

1.6. Additional Error Responses

The following error responses are defined in addition to those within Section 4.3.1 of [I-D.ietf.oauth-v2].

authorization_pending
: The authorization request is still pending as the end-user hasn't yet visited the authorization server and entered their verification code.

slow_down
: The client is polling too quickly and should back off at a reasonable rate.

2. Security Considerations

Length of codes (Google has done some research here).

3. Normative References

[I-D.ietf.oauth-v2]
: Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, "The OAuth 2.0 Protocol", Jun 2010.

[RFC2119]
: Bradner, B., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119.

Authors' Addresses

David Recordon (editor)
Facebook

Email: davidrecordon@facebook.com

Brent Goldman
Facebook

Email: brent@facebook.com