

OAuth  
Internet-Draft  
Intended status: Standards Track  
Expires: January 19, 2017

W. Denniss  
Google  
S. Myrseth  
ForgeRock  
J. Bradley  
Ping Identity  
M. Jones  
Microsoft  
H. Tschofenig  
ARM Limited  
July 18, 2016

**OAuth 2.0 Device Flow**  
**draft-ietf-oauth-device-flow-03**

Abstract

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

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 19, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- [1.](#) Introduction . . . . . [2](#)
- [2.](#) Terminology . . . . . [4](#)
- [3.](#) Specification . . . . . [4](#)
  - [3.1.](#) Device Authorization Request . . . . . [4](#)
  - [3.2.](#) Device Authorization Response . . . . . [5](#)
  - [3.3.](#) User Instruction . . . . . [6](#)
  - [3.4.](#) Device Token Request . . . . . [6](#)
  - [3.5.](#) Device Token Response . . . . . [7](#)
- [4.](#) IANA Considerations . . . . . [8](#)
  - [4.1.](#) OAuth URI Registration . . . . . [8](#)
    - [4.1.1.](#) Registry Contents . . . . . [8](#)
  - [4.2.](#) OAuth Extensions Error Registration . . . . . [8](#)
    - [4.2.1.](#) Registry Contents . . . . . [8](#)
- [5.](#) Security Considerations . . . . . [9](#)
- [6.](#) Normative References . . . . . [9](#)
- [Appendix A.](#) Acknowledgements . . . . . [9](#)
- [Appendix B.](#) Document History . . . . . [9](#)
- Authors' Addresses . . . . . [10](#)

**1. Introduction**

The device flow is suitable for clients executing on devices that do not have an easy data-entry method and where 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.

Note that this device flow does not utilize the client secret.



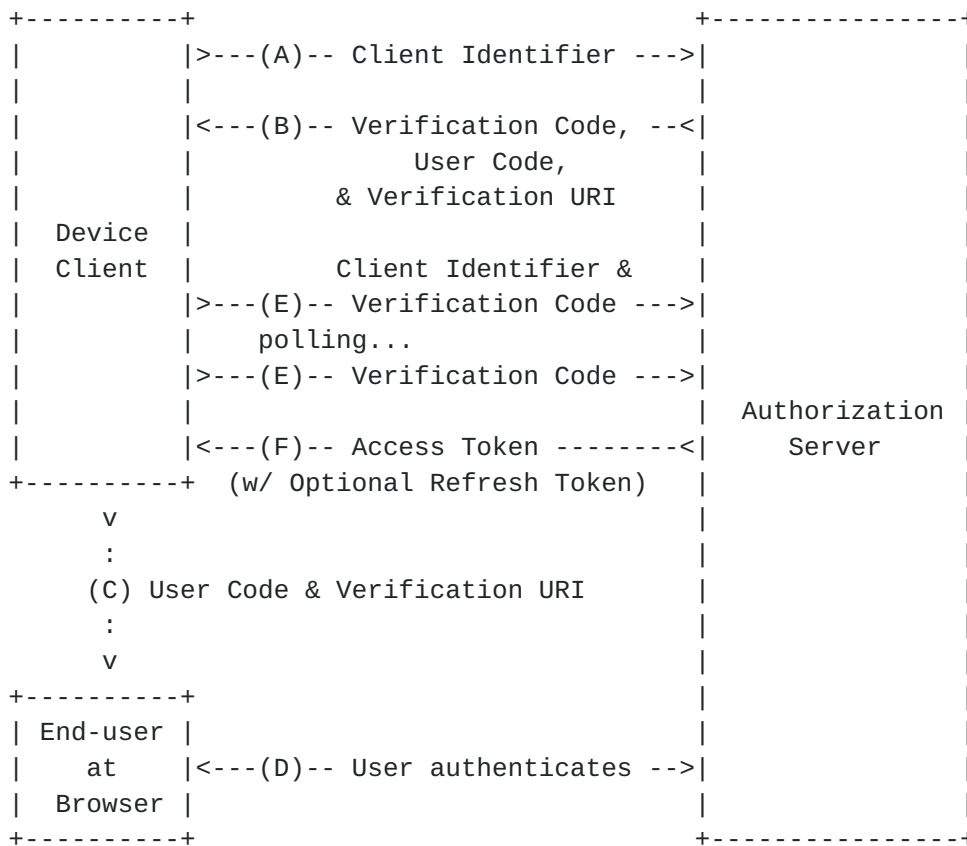


Figure 1: Device Flow.

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.

## 2. Terminology

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

Device Endpoint:

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

Device Verification Code:

A short-lived token representing an authorization session.

End-User Verification Code:

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

## 3. Specification

### 3.1. Device Authorization Request

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 2.2 of \[RFC6749\]](#).

scope:

OPTIONAL. The scope of the access request as described by [Section 3.3 of \[RFC6749\]](#).

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
```

### **3.2. Device Authorization Response**

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
}
```

### **3.3. User Instruction**

After receiving a successful Authorization Response, 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.

### **3.4. Device Token Request**

As the user is authorizing the request on secondary device which may not have a way to communicate to the original device, the client polls the token endpoint until the end-user grants or denies the request, or the device code expires.

The client polls at reasonable interval which MUST NOT exceed the minimum interval provided by the authorization server via the "interval" parameter (if provided).

The client makes a request to the token endpoint by sending the following parameters using the "application/x-www-form-urlencoded" format per [Appendix B](#) with a character encoding of UTF-8 in the HTTP request entity-body:

grant\_type

REQUIRED. Value MUST be set to "urn:ietf:params:oauth:grant-type:device\_code".



#### device\_code

REQUIRED. The device verification code, "device\_code" from the Device Authorization Response, defined in [Section 3.2](#).

#### client\_id

REQUIRED, if the client is not authenticating with the authorization server as described in [Section 3.2.1. of \[RFC6749\]](#)

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
```

```
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-
type%3Adevice_code&device_code=pxDoJ3Bt9WVMTXfDATLkxJ9u
&client_id=459691054427
```

Note that unlike the Access Token Request for the authorization\_code grant type defined in [Section 4.1.3 of \[RFC6749\]](#) the "redirect\_uri" parameter is NOT REQUIRED as part of this request.

If the client was issued client credentials (or assigned other authentication requirements), the client MUST authenticate with the authorization server as described in [Section 3.2.1 of \[RFC6749\]](#).

### **[3.5.](#) Device Token Response**

If the user has approved the grant, the token endpoint responds with a success response defined in [Section 5.1 of \[RFC6749\]](#) otherwise, it responds with an error, as defined in [Section 5.2 of \[RFC6749\]](#).

In addition to the error codes defined in [Section 5.2 of \[RFC6749\]](#), the following error codes are specific for the device flow:

#### 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.

expired\_token

The device\_code has expired. The client will need to make a new Device Authorization Request.

The error code "authorization\_pending" and "slow\_down" are considered soft errors. The the client should continue to poll when receiving "authorization\_pending" errors, reducing the interval if a "slow\_down" error is received. Other error codes are considered hard errors, the client should stop polling and react accordingly, for example, by displaying an error to the user.

## **4. IANA Considerations**

### **4.1. OAuth URI Registration**

This specification registers the following values in the IANA "OAuth URI" registry [[IANA.OAuth.Parameters](#)] established by [[RFC6755](#)].

#### **4.1.1. Registry Contents**

- o URN: urn:ietf:params:oauth:grant-type:device\_code
- o Common Name: Device flow grant type for OAuth 2.0
- o Change controller: IESG
- o Specification Document: [Section 3.1](#) of [[ this specification ]]

### **4.2. OAuth Extensions Error Registration**

This specification registers the following values in the IANA "OAuth Extensions Error Registry" registry [[IANA.OAuth.Parameters](#)] established by [[RFC6749](#)].

#### **4.2.1. Registry Contents**

- o Error name: authorization\_pending
- o Error usage location: Token endpoint response
- o Related protocol extension: [[ this specification ]]
- o Change controller: IETF
- o Specification Document: [Section 3.5](#) of [[ this specification ]]
  
- o Error name: slow\_down
- o Error usage location: Token endpoint response
- o Related protocol extension: [[ this specification ]]
- o Change controller: IETF
- o Specification Document: [Section 3.5](#) of [[ this specification ]]



- o Error name: expired\_token
- o Error usage location: Token endpoint response
- o Related protocol extension: [[ this specification ]]
- o Change controller: IETF
- o Specification Document: [Section 3.5](#) of [[ this specification ]]

## 5. Security Considerations

TBD

## 6. Normative References

[IANA.OAuth.Parameters]

IANA, "OAuth Parameters",  
<<http://www.iana.org/assignments/oauth-parameters>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997,  
<<http://www.rfc-editor.org/info/rfc2119>>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), DOI 10.17487/RFC6749, October 2012,  
<<http://www.rfc-editor.org/info/rfc6749>>.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", [RFC 6755](#), DOI 10.17487/RFC6755, October 2012,  
<<http://www.rfc-editor.org/info/rfc6755>>.

## [Appendix A](#). Acknowledgements

The -00 version of this document was based on [draft-recordon-oauth-v2-device](#) edited by David Recordon and Brent Goldman. The content of that document was initially part of the OAuth 2.0 protocol specification but was later removed due to the lack of sufficient deployment expertise at that time. We would therefore also like to thank the OAuth working group for their work on the initial content of this specification through 2010.

## [Appendix B](#). Document History

[[ to be removed by the RFC Editor before publication as an RFC ]]

-01

- o Applied spelling and grammar corrections and added the Document History appendix.





-00

- o Initial working group draft based on [draft-recordon-oauth-v2-device](#).

#### Authors' Addresses

William Denniss  
Google  
1600 Amphitheatre Pkwy  
Mountain View, CA 94043  
USA

Phone: +1 650-253-0000  
Email: [wdenniss@google.com](mailto:wdenniss@google.com)  
URI: <http://google.com/>

Stein Myrseth  
ForgeRock  
Lysaker torg 2  
Lysaker 1366  
NORWAY

Email: [stein.myrseth@forgerock.com](mailto:stein.myrseth@forgerock.com)

John Bradley  
Ping Identity

Email: [ve7jtb@ve7jtb.com](mailto:ve7jtb@ve7jtb.com)  
URI: <http://www.thread-safe.com/>

Michael B. Jones  
Microsoft

Email: [mbj@microsoft.com](mailto:mbj@microsoft.com)  
URI: <http://self-issued.info/>

Hannes Tschofenig  
ARM Limited  
Austria

Email: [Hannes.Tschofenig@gmx.net](mailto:Hannes.Tschofenig@gmx.net)  
URI: <http://www.tschofenig.priv.at>

