Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: March 1, 2020

T. Lodderstedt
yes.com

August 29, 2019

OAuth 2.0 Rich Authorization Requests
draft-lodderstedt-oauth-rar-00

Abstract

This document specifies a new parameter "authorization_details" that is used to carry fine grained authorization data into the OAuth authorization request.

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 https://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 March 1, 2020.

Table of Contents

1. Introduction

[RFC6749] defines the parameter "scope" that allows OAuth clients to specify the expected scope, i.e. the permission, of an access token. This mechanism is sufficient to implement static scenarios and course grain authorization requests, such as "give me read access to the resource owner's profile" but it's not sufficient to specify fine grained authorization requirements, such as "please let me make a payment with the amount of 45 Euros" or "please give me read access to folder A and write access to file X".

This draft introduces a new parameter "authorization_details" that allows clients to specify their fine grained authorization requirements using the expresivness of JSON data structures. For example, a request for payment authorization can use a JSON object like this:

```
{
   "instructedAmount":{
      "currency":"EUR",
      "amount":"123.50"
   },
   "debtorAccount":{
      "iban":"DE40100100103307118608"
   },
   "creditorName":"Merchant123",
   "creditorAccount":{
      "iban":"DE02100100109307118603"
   },
   "remittanceInformationUnstructured":"Ref Number Merchant"
}
```

For a comprehensive discussion of the challenges arising from new use cases in the open banking and eletronic signing spaces see [transaction-authorization].

1.1. Conventions and 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification uses the terms "access token", "refresh token", "authorization server", "resource server", "authorization endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", "access token response", and "client" defined by The OAuth 2.0 Authorization Framework [RFC6749].

2. Request parameter "authorization_details"

The request parameter "authorization_details" contains a JSON object. This JSON object is composed of one or more JSON objects on the second level, each of them containing the data to specify the authorization requirements for a certain type of resource. The type of resource or access requirement is determined by the name of the JSON object.

This example shows the specification of authorization details for a payment initiation transaction:

```
{
   "payment":{
      "instructedAmount":{
         "currency":"EUR",
         "amount":"123.50"
      },
      "debtorAccount":{
         "iban":"DE40100100103307118608"
      },
      "creditorName":"Merchant123",
      "creditorAccount":{
         "iban":"DE02100100109307118603"
      },
      "remittanceInformationUnstructured":"Ref Number Merchant"
   }
}
```

This example shows a combined request asking for access to account information and allowance to initiate a payment:

```
{
   "accounts":{
      "access":{
         "accounts":[

         ],
         "balances":[

         ],
         "transactions":[

         ]
      }
   },
   "payment":{
      "instructedAmount":{
         "currency":"EUR",
         "amount":"123.50"
      },
      "debtorAccount":{
         "iban":"DE40100100103307118608"
      },
      "creditorName":"Merchant123",
      "creditorAccount":{
         "iban":"DE02100100109307118603"
      },
      "remittanceInformationUnstructured":"Ref Number Merchant"
   }
}
```

The named JSON objects "account" and "payment" represent the different authorization data to be used by the AS to ask for consent and must subsequently also be made available to the respective RSs.

2.1. Structure of the authorization data elements

It is assumed that the sructure of each of the authorization data elements is tailored to the needs of a certain application, API, or resource type.

For example, the example structures shown above are based on certain kinds of APIs that can be found in the Open Banking space.

This draft therefore does only define a few requirements (see below) on the structure of authorization data elements.

2.2. Multiple instances of the same authorization data type

It's possible that the client asks for different kinds of access to different resources of the same type. There are two ways to cope with such a situation.

For some applications, e.g. file access, it might be reasonable to build a way to allocated authorization data to certain resources into the application specific JSON structure.

Here is an example:

```
{
   "files":{
      "permissions":[
         {
            "path":"/myfiles/A",
            "access":[
               "read"
            ]
         },
         {
            "path":"/myfiles/A/X",
            "access":[
               "read",
               "write"
            ]
         }
      ]
   }
}
```

Alternatively, a client MAY specify multiple instances of the same authorization data element type and distinguish those elements by adding a suffix "$"+"<instancename>". The following shows authorization details for requesting access to different folder on different IMAP servers (assuming the resource owner has access to both of them):

```
{
   "imap":{
      "server":"imap.example.com",
      "mailbox":"/users/<current>",
      "access":[
         "read",
         "write"
      ]
   },
   "imap$2":{
      "server":"imap.example.org",
      "mailbox":"/users/shared/folder3",
      "access":[
         "read"
      ]
   }
}
```

3. Using "authorization_details"

The request parameter can be used anywhere where the "scope" parameter is used, examples include:

- Authorization requests as specified in [RFC6749],
- Request objects as specified in [I-D.ietf-oauth-jwsreq],
- Device Authorization Request as specified in [RFC8628]

Parameter encoding is determined by the respective context.

In the context of an authorization request according to [RFC6749], the parameter is encoded using the "application/x-www-form-urlencoded" format as shown in the following example (JSON string trimmed for brevity):

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
    &code_challenge_method=S256,
    &code_challenge=5c305578f8f19b2dcdb6c3c955c0a...97e43917cd,
    &authorization_details=%7B%22payment%22%3A%7B%22instructedAmount
    %22%3A%7B%22currency%22%3A%22EUR%22%2C%22amount%22%3A%22123.50%2
    2%7D%2C%22debtorAccount%22%3A%7B%22iban%22%3A%22DE40100100103307
    118608%22%7D%2C%22creditorName%22%3A%22Merchant123%22%2C%22credi
    torAccount%22%3A%7B%22iban%22%3A%22DE02100100109307118603%22%7D%
    2C%22remittanceInformationUnstructured%22%3A%22Ref%20Number%20Me
    rchant%22%7D%7D HTTP/1.1
Host: server.example.com
```

In the context of a request object, "autorization_details" is added as another top level JSON element.

```
{
   "iss":"s6BhdRkqt3",
   "aud":"https://server.example.com",
   "response_type":"code",
   "client_id":"s6BhdRkqt3",
   "redirect_uri":"https://client.example.com/cb",
   "state":"af0ifjsldkj",
   "code_challenge_method":"S256",
   "code_challenge":"5c305578f8f19b2dcdb6c3c955c0a...97e43917cd",
   "authorization_details":{
      "payment":{
         "instructedAmount":{
            "currency":"EUR",
            "amount":"123.50"
         },
         "debtorAccount":{
            "iban":"DE40100100103307118608"
         },
         "creditorName":"Merchant123",
         "creditorAccount":{
            "iban":"DE02100100109307118603"
         },
         "remittanceInformationUnstructured":"Ref Number Merchant"
      }
   }
}
```

4. Processing

Based on the data provided in the "authorization_details" parameter the AS will ask the user for consent to the requested access permissions.

Note: the AS is supposed to merge the authorization requirements given in the "scope" parameter and the "authorization_details" parameter if both are present in the authorization request.

The AS MUST refuse to process any unknown authorization data type. If the "authorization_details" contains any unknown authorization data type, the AS MUST abort processing and respond with an error "invalid_scope" to the client.

If the resource owner grants the client the request access, the AS will issue tokens to the client that are associated with the respective "authorization_details".

The AS MUST make the "authorization_details" available to the respective resource servers. The AS MAY add the "authorization_details" element to access tokens in JWT format and to Token Introspection responses.

The AS MUST take into consideration the privacy implications when sharing authorization details with the resource servers. The AS SHOULD share this data with the resource servers on a "need to know" basis.

5. Relationship to "resource" parameter

[I-D.ietf-oauth-resource-indicators] defines the request parameter "resource" indicating to the AS the resource(s) where the client intends to use the access tokens issued based on a certain grant.

This mechanism is a way to audience restrict access tokens and to allow the AS to create resource specific access tokens.

This draft can be used in conjunction with [I-D.ietf-oauth-resource-indicators] in the same way as the "scope" parameter. The AS is supposed to narrow down the authorization details and respective permissions to the needs of the particular resource when minting an access token.

While this depends on the AS to know what authorization details are relevant for what RS, this draft can also be combined with the concept of resource indicators to make this relationsship explicit and to narrow the privileges of an access token down to certain

permissions given on a certain resource down to the individual operation (see [I-D.ietf-oauth-security-topics], section-3.3).

As an example, it is possible to specify that the client will get "read" access to "file X" stored at the resource "https://store.example.com" [1]. To achieve this the example given above for access to an IMAP server is slightly modfied to use the "resource" element in as part of the top level claims within the authorization data element.

```
{
   "imap":{
      "resource":"imap.example.com",
      "mailbox":"/users/<current>",
      "access":[
         "read",
         "write"
      ]
   },
   "imap$2":{
      "resource":"imap.example.org",
      "mailbox":"/users/shared/folder3",
      "access":[
         "read"
      ]
   }
}
```

The AS MUST respect the value of the "resource" element when deciding whether a certain element is placed into a (structured) access token or token introspection response.

6. Metadata

The AS advertises support for "authorization_details" using the metadata parameter "authorization_details_supported" of type boolean.

The authorization data types supported can be determined using the metadata parameter "authorization_data_types_supported", which is an JSON array.

Note: different applications MUST make sure their authorization data types do not collide. This is either achieved by using a namespace under the control of the entity defining the type name or by registering the type with the new "OAuth Authorization Data Type Registry".

Clients annonce the authorization data types the use in the new dynamic client registration parameter "authorization_data_types".

The registration of new authorization data types with the AS is out of scope of this draft.

7. Further Examples

TBD

- self contained (account information, claims, signing)
- external reference (payment)
- multiple payments
- access to e-mail
- access to files/directories

8. Implementation Considerations

The scheme and processing will significantly vary among different authorization data types. Any implementation of this draft is therefore supposed to allow the customization of the user consent and the handling of access token data.

One option would be to have a mechanism allowing the registration of extension modules, each of them responsible for rendering the respective user consent and any transformation needed to provide the data needed to the resource server by way of structured access tokens or token introspection responses.

9. Acknowledgements

I would would like to thank Brian Campbell, Daniel Fett, Sebastian Ebling, Dave Tonge, Mike Jones, Nat Sakimura, Rob Otto, and Justin Richer for their valuable feedback during the preparation of this draft.

10. IANA Considerations

- "authorization_details" as JWT claim
- "authorization_details_supported" and "authorization_data_types_supported" as metadata parameters

- "authorization_data_types" as dynamic client registration parameter

- establish authorization data type registry

11. Security Considerations

TBD

12. References

12.1. Normative References

[transaction-authorization]
Lodderstedt, T., "Transaction Authorization or why we need to re-think OAuth scopes", Apr 2019, <https://medium.com/oauth-2/transaction-authorization-or-why-we-need-to-re-think-oauth-scopes-2326e2038948>.

12.2. Informative References

[I-D.ietf-oauth-jwsreq]
Sakimura, N. and J. Bradley, "The OAuth 2.0 Authorization Framework: JWT Secured Authorization Request (JAR)", draft-ietf-oauth-jwsreq-19 (work in progress), June 2019.

[I-D.ietf-oauth-resource-indicators]
Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", draft-ietf-oauth-resource-indicators-05 (work in progress), July 2019.

[I-D.ietf-oauth-security-topics]
Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "OAuth 2.0 Security Best Current Practice", draft-ietf-oauth-security-topics-13 (work in progress), July 2019.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <https://www.rfc-editor.org/info/rfc8628>.

12.3. URIs

[1] https://store.example.com"

Appendix A. Document History

[[To be removed from the final specification]]

-00

o first draft

Author's Address

Torsten Lodderstedt
yes.com

Email: torsten@lodderstedt.net