N. Aillery
C. Marais
Orange
April 4, 2017

Versatile, Synchronous and Asynchronous, JSON Web Token (JWT) Assertion Profile for OAuth 2.0 Authorization Grants
draft-oauth-versatile-jwt-profile-03

Abstract

This specification defines the use of a JSON Web Token (JWT) Bearer Token as a mean for requesting an OAuth 2.0 access token in a versatile way, synchronous and asynchronous. This specification is a profile of [RFC6749] and an extension to [RFC7523].

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 RFC 2119 [RFC2119].

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

1.2. Terminology

All terms are as defined in the following specifications: "The OAuth 2.0 Authorization Framework" [RFC6749], the OAuth Assertion Framework [RFC7521], [RFC7523], and "JSON Web Token (JWT)" [JWT].

1.3. Overview

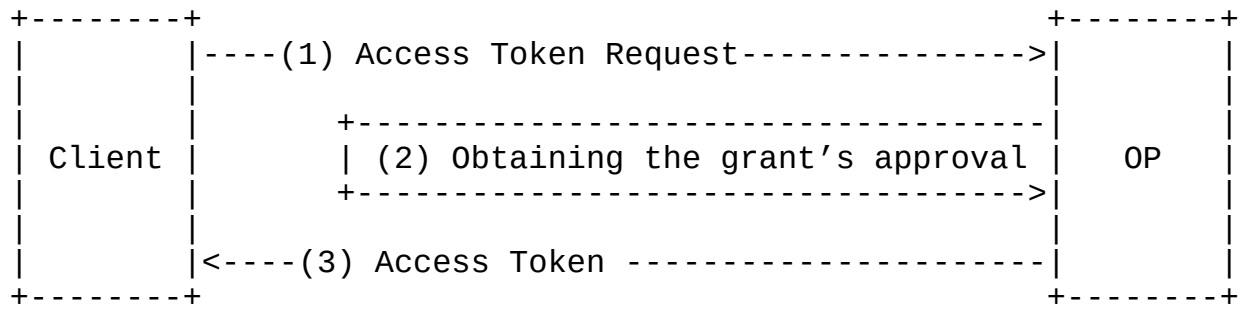
The versatile JWT profile enables a Client to get an Access Token on the token endpoint. The versatile JWT profile is a profile to [RFC6749] that extends the synchronous JWT profile, defined in [RFC7523]. The versatile JWT profile enables the OP to respond either in a synchronous or asyncronous way, depending on the grant approval process.

The versatile JWT profile, in abstract, follows the following steps.

1. The Client sends a Access Token Request to the OpenID Provider (OP).
2. The OP approves the grant (potentially involving the Resource Owner in an out of band mode).

3. The OP responds to the Client with an Access Token.

These steps are illustrated in the following diagram:



In order to obtain the grant's approval, the OP may reuse a previously approved grant. It may also have some interactions with other parties. This specification does not define the way a grant is approved.

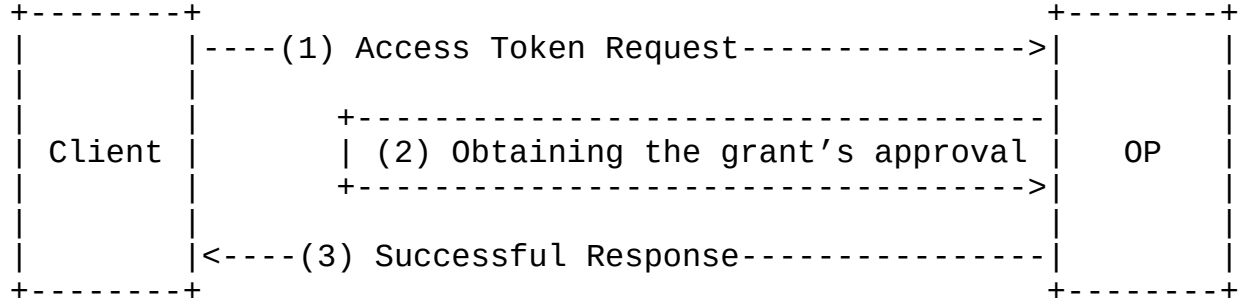
In order to retrieve the Access Token, three flows are possible:

Synchronous flow: The Client gets the Access Token in the initial response. Refer to Section 1.3.1 for more details.
Asynchronous Poll flow: The Client regularly calls the "Token Endpoint". Refer to Section 1.3.2 for more details.
Asynchronous Push flow: The Client is notified on the "Client Notification Endpoint". Refer to Section 1.3.3 for more details.

In this specification, the Client requests an asynchronous flow, but both asynchronous flows can become a synchronous flow if no asynchronism is necessary.

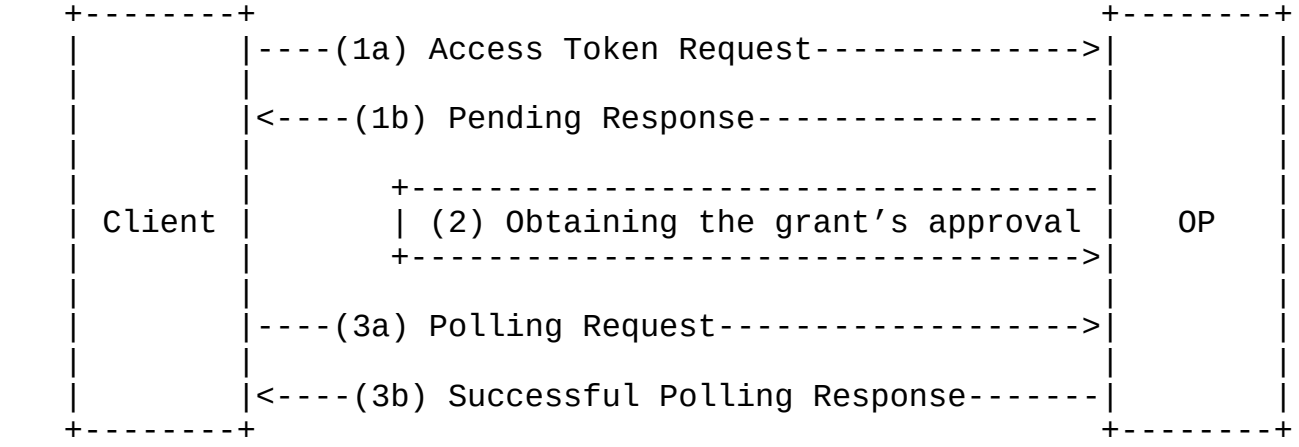
1.3.1. Synchronous flow

In this flow, the Client gets the Access Token in the initial response.



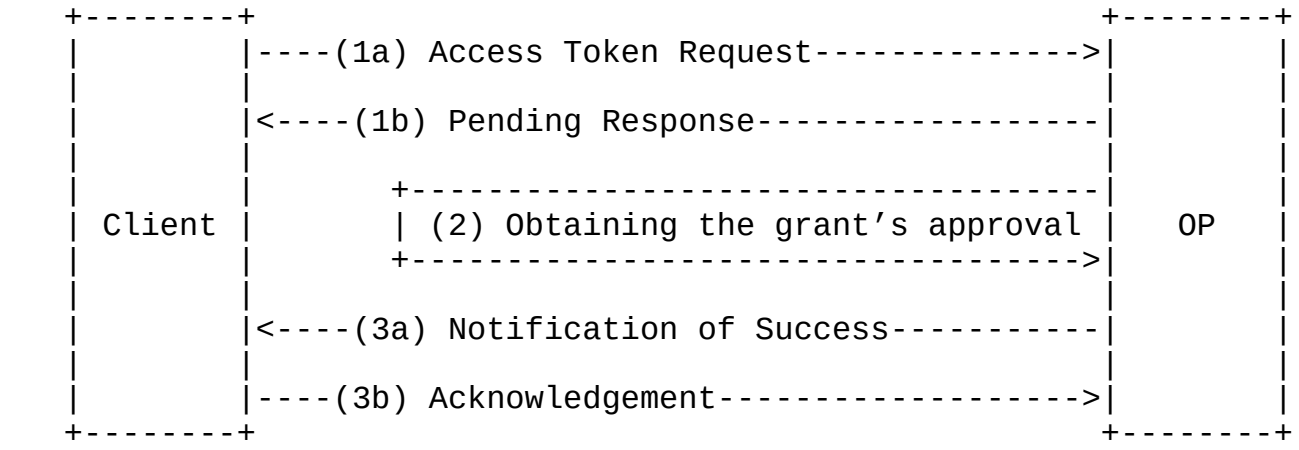
1.3.2. Asynchronous Poll flow

In this flow, the Client regularly calls the "Token Endpoint".



1.3.3. Asynchronous Push flow

In this flow, the Client is notified on the "Client Notification Endpoint".



2. Client registration

If the Client wants to use the Asynchrounous Push mode (Section 4), it MUST register its "client_notification_endpoint". The "client_notification_endpoint" is a callback URL on the Client. If the client does not use the Asynchrounous Push mode, it can obtain the result by using the Asynchrounous Poll mode (Section 3). Both mechanisms are not mutual exclusive, i.e. a given "client_id" can use both mechanisms, but not in the same transaction.

3. Asynchronous Poll mode

When the Client requires an Asynchronous Poll mode, the following flow can be synchronous or asynchronous. If the flow is synchronous, the Client gets the Access Token in the initial response, as illustrated in Section 1.3.1. If the flow is asynchronous, the Client has to regularly call the "Token Endpoint", as illustrated in Section 1.3.2.

3.1. Access Token Request

The initial Access Token Request to the token endpoint is defined in Section 2.1 of [RFC7523]. This specification proposes a specific "grant_type" for the Asynchronous Poll mode:

grant_type
: MANDATORY. In order to enable the Asynchronous Poll mode, the "grant_type" value MUST be set to: "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:poll_mode". Using the grant_type "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:poll_mode" means that, in case of asynchronous response, the Client will use the Asynchronous Poll mode, defined in Section 3.

Authentication of the Client is optional, as described in Section 3.2.1 of OAuth 2.0 [RFC6749]. If the Client authenticated for the Initial Access Token Request defined in Section 3.1, it MUST authenticate for the Polling Request.

The following is a non-normative example of an initial Access Token Request.

```
POST /token.oauth2 HTTP/1.1
Host: example.as.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type
   %3Ajwt-bearer%3Aversatile%3Apoll_mode
&assertion=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
   ZhwP[...omitted for brevity...].
   paC4[...omitted for brevity...]
&scope=scope_a%20scope_b
```

3.2. Access Token Responses

If the grant is approved, the token endpoint responds with a success response defined in Section 3.2.1; if the grant has not been approved yet, the token endpoint responds with a pending response defined in

Section 3.2.2; otherwise it responds with an error, as defined in Section 3.2.3.

3.2.1. Successful Response

If the grant has been approved, the token endpoint responds with a success response defined in Section 5.1 of [RFC6749].

The following is a non-normative example of a Successful Response.

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

{
    "access_token": "SlAV32hkKG",
    "token_type": "Bearer",
    "refresh_token": "8xLOxBtZp8",
    "expires_in": 3600
}
```

3.2.2. Pending Response

If the grant has not been aproved yet, the token endpoint responds with a HTTP 202 Accepted response. The response body contains JSON structure, with the following attributes.

transaction_id
: MANDATORY. The "transaction_id" value is used to identify the transaction in the Polling Request, defined in Section 3.3. The "transaction_id" SHOULD be a random value with a level of entropy high enough to prevent guessing and replay by a Client. The "transaction_id" value is a case sensitive string.

expires_in
: OPTIONAL. The "expires_in" value is lifetime in seconds of the "transaction_id". The "expires_in" value is an integer.

interval
: OPTIONAL. The "interval" value is the minimum amount of time in seconds that the client SHOULD wait between polling requests to the token endpoint. The "interval" value is an integer.

The following is a non-normative example of a Pending Response.

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "transaction_id": "b0f9f8022e344f9984dcc7d3d4d4",
    "expires_in": 1800,
    "interval": 5
}
```

3.2.3. Error Response

If the Access Token Request can not be processed, the token endpoint responds with an error, as defined in Section 5.2 of [RFC6749].

The following is a non-normative example of an Error Response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
   "error":"invalid_grant",
   "error_description":"Audience validation failed"
}
```

3.3. Polling Request

In order to detect that a pending grant has been approved, the Client can poll the token endpoint. If the grant has been approved, the token endpoint reponds in the response. Else, the token endpoint responds with HTTP 400 Bad Request.

A Client configured with a "client_notification_endpoint" MUST NOT send a Polling Request.

The Polling Request to the token endpoint is defined in Section 2.1 of [RFC7523]. This specification proposes a specific "grant_type", an additional parameter "transaction_id" and the limitation to those two attributes:

grant_type
: MANDATORY. In order to indicate a Polling Request, the "grant_type" value MUST be set to "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:polling".

transaction_id

MANDATORY. The "transaction_id" value contains the "transaction_id" received from the Pending Response (Section 3.2.2). The "transaction_id" value is a case sensitive string.

The parameters "grant_type" and "transaction_id" are the only attributes that the token endpoint SHOULD consider in a Polling Request.

Authentication of the Client is optional, as described in Section 3.2.1 of OAuth 2.0 [RFC6749].

The following is a non-normative example of an Polling Request.

```
POST /token.oauth2 HTTP/1.1
Host: example.as.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
   %3Aversatile%3Apolling
&transaction_id=b0f9f8022e344f9984dcc7d3d4d4
```

3.4. Polling Responses

If the grant has been approved, the token endpoint responds with a Successful Polling Response, defined in Section 3.4.1. Else, the token endpoint responds with an Error Polling Response defined in Section 3.4.2.

3.4.1. Successful Polling Response

If the grant has been approved, the token endpoint responds with a success response defined in Section 5.1 of [RFC6749].

The following is a non-normative example of a Successful Polling Response.

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

{
    "access_token": "SlAV32hkKG",
    "token_type": "Bearer",
    "refresh_token": "8xLOxBtZp8",
    "expires_in": 3600
}
```

3.4.2. Error Polling Response

If the grant has not been approved yet or if the Polling Request can not be processed, the token endpoint responds with an error, as defined in Section 5.2 of [RFC6749], with the following additional error codes, specific for the versatile JWT profile:

authorization_pending
: The Access Token Request is still pending as the authorization has not been gained yet. The client should repeat the Access Token Request to the token endpoint.

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

expired_transaction
: The "transaction_id" has expired. The client will need to make a new Access Token Request (Section 3.1).

invalid_transaction_id
: The Client sent a polling request for a "transaction_id" that does not exist or is not valid for the requesting Client.

forbidden_polling
: The Client sent a Polling Request whereas it set the "grant_type" to "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:push_mode" in the Access Token Request.

The error codes "authorization_pending" and "slow_down" are considered soft errors. The client should continue to poll the token endpoint by repeating the Polling Request (Section 3.3) when receiving soft errors, increasing the time between polls if a "slow_down" error is received. Other error codes are considered hard errors; the client should stop polling and react accordingly. The interval at which the client polls MUST NOT be more frequent than the

"interval" parameter returned in the Pending Response (Section 3.2.2).

The following is a non-normative example of an Error Polling Response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
{
   "error":"invalid_transaction_id",
   "error_description":"The transaction_id is not valid."
}
```

4. Asynchronous Push mode

When the Client requires an Asynchronous Push mode, the following flow can be synchronous or asynchronous. If the flow is synchronous, the Client gets the Access Token in the initial response, as illustrated in Section 1.3.1. If the flow is asynchronous, The Client will be notified on the "Client Notification Endpoint"., as illustrated in Section 1.3.3.

4.1. Access Token Request

The initial Access Token Request to the token endpoint is defined in Section 2.1 of [RFC7523]. This specification proposes a specific "grant_type" and an additional parameter "client_notification_token" for the Asynchronous Push mode:

grant_type
: MANDATORY. In order to enable the Asynchronous Push mode, the "grant_type" value MUST be set to: "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:push_mode". Using the grant_type "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:push_mode" means that, in case of asynchronous response, the Client will use the Asynchronous Push mode, defined in Section 4.

client_notification_token
: MANDATORY if the Client uses the Asynchronous Push mode described in Section 4. MUST NOT be used otherwise. The "client_notification_token" is an opaque token issued by the Client. The Client SHOULD NOT reuse a "client_notification_token". The Client SHOULD use a random value with a level of entropy high enough to cover its security requirements. The "client_notification_token" is a Bearer Token used by the Client to authorize the OP when the OP sends the Token Response to the Client Notification Endpoint. The "client_notification_token" value is a case sensitive string.

Authentication of the Client is optional, as described in Section 3.2.1 of OAuth 2.0 [RFC6749].

The following is a non-normative example of an initial Access Token Request.

```
POST /token.oauth2 HTTP/1.1
Host: example.as.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type
   %3Ajwt-bearer%3Aversatile%3Apush_mode
&assertion=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
   ZhwP[...omitted for brevity...].
   paC4[...omitted for brevity...]
&scope=scope_a%20scope_b
&client_notification_token=vO4n2DAeRrcWE0VAlo4I
```

4.2. Access Token Responses

If the grant is approved, the token endpoint responds with a success response defined in Section 4.2.1; if the grant has not been approved yet, the token endpoint responds with a pending response defined in Section 4.2.2; otherwise it responds with an error, as defined in Section 4.2.3.

4.2.1. Successful Response

If the grant has been approved, the token endpoint responds with a success response defined in Section 5.1 of [RFC6749].

The following is a non-normative example of a Successful Response.

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

{
    "access_token": "SlAV32hkKG",
    "token_type": "Bearer",
    "refresh_token": "8xLOxBtZp8",
    "expires_in": 3600
}
```

4.2.2. Pending Response

If the grant has not been aproved yet, the token endpoint responds with a HTTP 202 Accepted response. The response body contains JSON structure, with the following attributes.

transaction_id
: MANDATORY. The "transaction_id" value is used to identify the transaction in the Notifications, defined in Section 4.3. The "transaction_id" SHOULD be a random value with a level of entropy high enough to prevent guessing. The "transaction_id" value is a case sensitive string.

expires_in
: OPTIONAL. The "expires_in" value is lifetime in seconds of the "transaction_id". The "expires_in" value is an integer.

The following is a non-normative example of a Pending Response.

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "transaction_id": "b0f9f8022e344f9984dcc7d3d4d4",
    "expires_in": 1800
}
```

4.2.3. Error Response

If the Access Token Request can not be processed, the token endpoint responds with an error, as defined in Section 5.2 of [RFC6749], with the following additional error code, specific for the versatile JWT profile:

forbidden_mode
: The Client set the "grant_type" to "urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:push_mode" in the Access Token Request, but did not register a "client_notification_endpoint".

The following is a non-normative example of an Error Response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
   "error":"invalid_grant",
   "error_description":"Audience validation failed"
}
```

4.3. Notifications

If the Client registered a "client_notification_endpoint", the OP MUST notify the Client when a pending grant has been approved or an error occured. The Notification is sent to the Client using a HTTP POST on the Client Notification Endpoint.

Communication with the Client Notification Endpoint MUST utilize TLS.

A Client configured with a "client_notification_endpoint" MUST wait for a Notification on its "client_notification_endpoint".

4.3.1. Notification of Success

The OP sends the Notification of Success to the Client using HTTP POST.

The HTTP POST request MUST contain the following header:

Authorization
: MANDATORY. The "Authorization" value contains the "client_notification_token" specified in the Access Token Request and used as a [RFC6750] Bearer Token. The "Authorization" value is a case sensitive string.

A Notification of Success contains a JSON object, as defined in Section 5.1 of [RFC6749], with the additional attribute "transaction_id".

A Notification is a Notification of Error if the JSON object contains an "error" attribute. Otherwise, it’s a Notification of Success.

transaction_id
: MANDATORY. The "transaction_id" links the Notification with a Pending Response (Section 4.2.2).

The following is a non-normative example of a Notification of Success.

```
POST /notification  HTTP/1.1
Host: example.client.com
Authorization: Bearer vO4n2DAeRrcWE0VAlo4I
Content-Type: application/json

{
    "transaction_id": "b0f9f8022e344f9984dcc7d3d4d4",
    "access_token": "SlAV32hkKG",
    "token_type": "Bearer",
    "refresh_token":  "8xLOxBtZp8",
    "expires_in": 3600
}
```

4.3.2. Notification of Error

The OP sends the Notification of Error to the Client using HTTP POST.

The HTTP POST request MUST contain the following header:

Authorization
: MANDATORY. The "Authorization" value contains the "client_notification_token" specified in the Access Token Request and used as a [RFC6750] Bearer Token. The "Authorization" value is a case sensitive string.

A Notification of Error contains a JSON object, as defined in Section 5.2 of [RFC6749], with the additional attribute "transaction_id".

A Notification is a Notification of Error if the JSON object contains an "error" attribute. Otherwise, it’s a Notification of Success.

transaction_id MANDATORY. The "transaction_id" links the Notification with a Pending Response (Section 4.2.2).

The following is a non-normative example of a Notification of Error.

```
POST /notification  HTTP/1.1
Host: example.client.com
Authorization: Bearer vO4n2DAeRrcWE0VAlo4I
Content-Type: application/json

{
   "transaction_id": "b0f9f8022e344f9984dcc7d3d4d4",
   "error":"invalid_grant",
   "error_description":"Audience validation failed"
}
```

4.3.3. Processing the Authorization

The "Authorization" value is a "client_notification_token". This "client_notification_token" enables to retrieve the "transaction_id" received in the Pending Response (Section 4.2.2). The "transaction_id" in the Notification MUST match with the "transaction_id" received in the Pending Response (Section 4.2.2).

4.4. Acknowledgement

The acknowledgement MUST be a HTTP 200 OK. This HTTP response SHOULD be ignored by the OP.

The following is a non-normative example of an Acknowledgement.

```
HTTP/1.1 200 OK
```

5. Interoperability Considerations

For Interoperability Considerations, please refer to:

- Section 7 of [RFC7521]
- Section 5 of [RFC7523]

6. Security Considerations

For Security Considerations, please refer to:

- Section 10 of [RFC6749]
- Section 8 of [RFC7521]
- Section 6 of [RFC7523]
- Section 11 of [JWT]

7. Privacy Considerations

For Privacy Considerations, please refer to:

- Section 7 of [RFC7523]
- Section 12 of [JWT]

8. IANA Considerations

8.1. OAuth Parameter Registration

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

8.1.1. Registry Contents

- Parameter name: client_notification_token
- Parameter usage location: Access Token Request
- Change Controller: IESG
- Specification Document: Section 4.1 of this specification

- Parameter name: transaction_id
- Parameter usage location: Polling Request
- Change Controller: IESG
- Specification Document: Section 3.3 of this specification

8.2. OAuth URI Registration

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

8.2.1. Registry Contents

- URN: urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:push_mode
- Common Name: Push mode of the Versatile JWT Bearer Token Grant Type Profile for OAuth 2.0
- Change Controller: IESG
- Specification Document: Section 4.1 of this specification

- URN: urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:poll_mode
- Common Name: Poll mode of the Versatile JWT Bearer Token Grant Type Profile for OAuth 2.0
- Change Controller: IESG
- Specification Document: Section 3.1 of this specification

- URN: urn:ietf:params:oauth:grant-type:jwt-bearer:versatile:polling
- Common Name: Polling for Versatile JWT Bearer Token Profile for OAuth 2.0
- Change Controller: IESG
- Specification Document: Section 3.3 of this specification

8.3. OAuth Extensions Error Registration

This specification registers the following values in the IANA "OAuth Extensions Error Registry" registry [IANA.OAuth.Parameters] established by [RFC6755].

8.3.1. Registry Contents

- Error name: invalid_transaction_id
- Error usage location: Error Polling Response
- Related protocol extension: this specification
- Change controller: IETF
- Specification Document: Section 3.4.2 of this specification

- Error name: forbidden_polling
- Error usage location: Error Polling Response
- Related protocol extension: this specification
- Change controller: IETF
- Specification Document: Section 3.4.2 of this specification

- Error name: forbidden_mode
- Error usage location: Error Response
- Related protocol extension: this specification
- Change controller: IETF
- Specification Document: Section 4.2.3 of this specification

9. References

9.1. Normative References

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

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, May 2015,
<http://www.rfc-editor.org/info/rfc7519>.

[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>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <http://www.rfc-editor.org/info/rfc6750>.

[RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, May 2015, <http://www.rfc-editor.org/info/rfc7521>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <http://www.rfc-editor.org/info/rfc7523>.

9.2. Informative References

[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>.

Acknowledgements

Authors' Addresses

Nicolas Aillery
Orange

EMail: nicolas.aillery@orange.com

Charles Marais
Orange

EMail: charles.marais@orange.com