N. Aillery
C. Marais
Orange
September 19, 2016

OpenID Connect User Questioning API 1.0
draft-user-questioning-api-02

Abstract

This specification defines a specific endpoint used by a Client (i.e. Service Provider) in order to question a End-User and get his Statement (i.e. his answer).

Table of Contents

1. Introduction

This specification defines a specific endpoint used by a Client (i.e. Service Provider) in order to question a End-User and get his Statement (i.e. his answer).

This endpoint is specified as an OAuth 2.0-protected Resource Server accessible with an Access Token.

The way the Access Token has been obtained by the Client is out of scope of this specification.

Whether the End-User is currently using the Client or not is also out of scope of this specification.

The User Questioning API is an asynchronous API. There are 3 main ways to get the End-User's Statement: the first one requires some polling of the API, the second requires the Client to expose a callback endpoint and the third one requires a user interaction on the Client.

1.1. Requirements Notation and 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].

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value.

1.2. Terminology

This specification uses the terms "Access Token", "Resource Server", and "Client" defined by OAuth 2.0 [RFC6749]. This specification also defines the following terms:

Questioned User ** End-User receiving the question and requested to give his Statement.

1.3. Overview

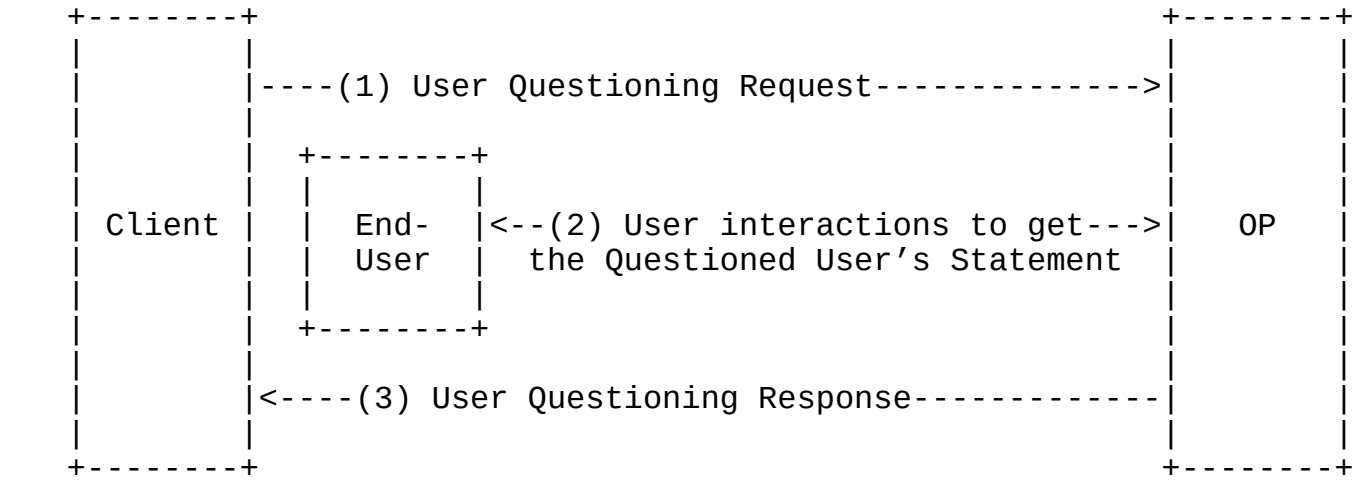
The User Questioning protocol, in abstract, follows the following steps.

1. The Client sends a User Questioning Request to the OpenID Provider (OP).

2. The OP interacts with the Questioned User and obtains his Statement.

3. The OP responds to the Client with a User Questioning Response.

These steps are illustrated in the following diagram:



1.4. Example of use cases involving the User Questioning API

The following use cases are non-normative examples to illustrate the usage of the User Questioning API by a Client.

1. The Client can be a bank and the User Questioning API is used to challenge the End-User when he wants to pay on Internet in order to secure the transaction. This is similar to 3D-Secure. The question could be: "Do you allow a payment of x euros to party y?".

2. The Client can be a bank and the User Questioning API is used to challenge the End-User when he wants to add a new payee for a bank transfer. The question could be: "Do you allow party y to be added to your payees?".

3. The Client can be a drive-in food market and the User Questioning API is used to ask the End-User if he accepts the exchange of one

missing product by another. The question could be: "Do you agree to get product x instead of product y?".

4. The Client can be a ticketing platform and the User Questioning API is used to prevent transactions by bots. The question could be: "Do you confirm that you are currently booking a ticket?".

5. The Client can be an airline company and the User Questioning API is used to be sure that the End-User is notified of a delay. The question could be: "Your flight is postponed. Can you confirm that you are aware?".

2. User Statement Token

Here after is described the User Statement Token :

Attribute	Type	Description	Example
status	string	Status code of the Question. Possible values are described in Section 2.2.	"pending"
statement	string	Statement made by the Questioned User. Possible values are described in Section 2.3.	"accepted"
creation_date	timestamp	Date indicating when the User Questioning Request has been received by the OP. Its value is a	1311281970

		JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.	
last_modification_date	timestamp	Date indicating the last change of the status. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.	1311281970
statement_date	timestamp	Date indicating when the End-User gave his Statement on the Question. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC	1311282970

		until the date/time.	
user_id	string	Unique identifier allowing to identify the Questioned User (e.g. Mobile phone , sub, ...). Ignored if the Access Token is tied with a specific End-User, mandatory otherwise.	3444975f-1137-47dc-908f-942ac85ab98f
user_id_type	string	Indicate the type of the End-User’s identifier used for User Questioning. Ignored if the Access Token is tied with a specific End-User, mandatory otherwise. Possible values are described in Section 2.1.	MSISDN
question_displayed	string	Question that was displayed to the	An example message to display

		Questioned User.	
used_qcr	string	Questioning context class reference : Level of Assurance used by the OP (can be 2 3 4).	2
used_qmr	string	Questioning method used by the OP for the User Questioning.	SMS_OTP

User Statement Token

2.1. user_id_type

user_id_type member can take the following values :

Value	Description	Example
MSISDN	Represents the Mobile Phone number corresponding to the Questioned User. The way this MSISDN has been captured by the Client is out of scope of this specification.	33612345678
sub	Represents the sub corresponding to the Questioned User. The way this sub has been captured by the Client is out of scope of this specification.	8d858e0a-c91b-426a-92e8-462d3876df7d

user_id_type possible values

2.2. status

status member can take the following values :

Value	Description
pending	This status name is returned when the User Questioning is ongoing.
terminated	This status name is returned when the User Questioning is finished.

status possible values

2.3. statement

By default, statement member can take the following values :

Value	Description
accepted	This statement is returned when the Questioned User's Statement is an acceptation.
denied	This statement is returned when the Questioned User's Statement is a deny.

status possible values

The statement can take other values in order to address specific use cases. For example, the question could ask for a number or a choice amongst several possibilities. These specific statement values SHOULD be specified in extensions of this specification or in an ad-hoc agreement between the Clients and the OP.

3. User Questioning flows

This document specifies three User Questioning flows:

Pulled-By-Client flow: In this flow, after the User Questioning Request, the Client must call the OP in order to get the User Questioning Response. Refer to Section 3.1 for more details.

Pushed-To-Client flow: In this flow, after the User Questioning Request, the OP calls the Client to deliver the User Questioning Response. Refer to Section 3.2 for more details.

Terminated-By-Client flow: In this flow, after the User Questioning Request, the Client must call the OP with an additional verification_code in order to get the User Questioning Response. Refer to Section 3.3 for more details.

The flow to use is decided by the OP and depends on the User Questioning Request and the available User Questioning mechanisms:

Pulled-By-Client flow: If the Client does not include a client_notification_endpoint in the User Questioning Request and if the User Questioning mechanism selected by the OP does not require additional information from the Client (e.g. verification_code). Refer to Section 3.1 for more details.

Pushed-To-Client flow: If the Client includes a client_notification_endpoint in the User Questioning Request and if the User Questioning mechanism selected by the OP does not require additional information from the Client (e.g. verification_code). Refer to Section 3.2 for more details.

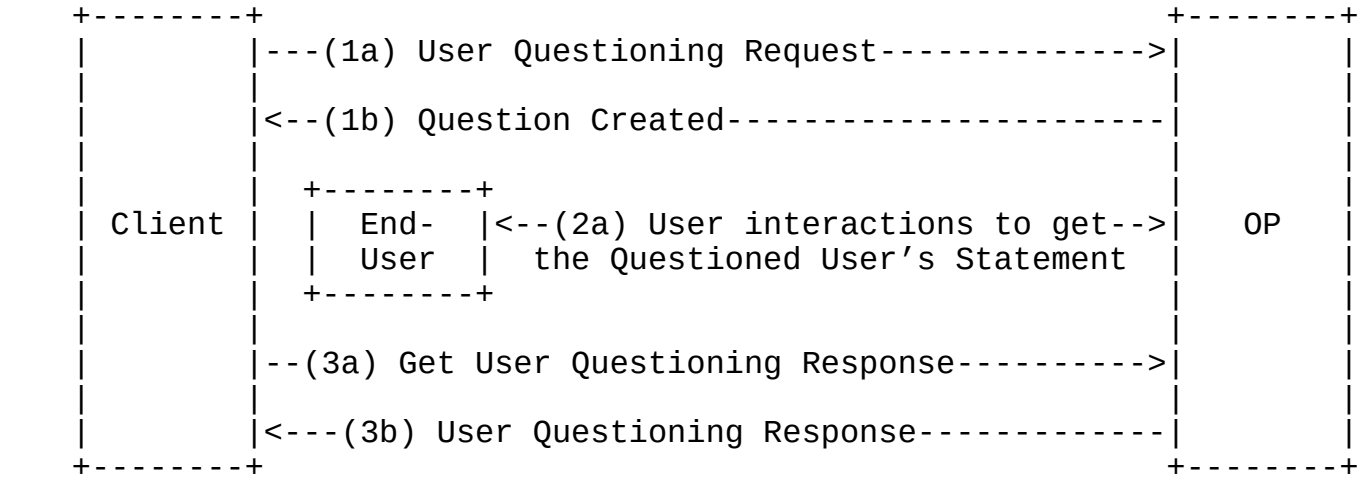
Terminated-By-Client flow: If the User Questioning mechanism selected by the OP requires additional information from the Client (e.g. verification_code). Refer to Section 3.3 for more details.

3.1. Pulled-By-Client Flow

In this flow, the Client MUST NOT include a client_notification_endpoint in the User Questioning Request. The Client will poll the OP until the User Questioning is finished (accepted, denied, error...).

If a verification_code is needed for the mecanism chosen by the OP, the flow will be the Terminated-By-Client Flow (cf. Section 3.3).

3.1.1. Pulled-By-Client Flow steps



3.1.2. User Questioning Endpoint

3.1.2.1. (1a) User Questioning Request

The Client sends the User Questioning Request using HTTP GET or HTTP POST.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain the following attributes, either in the query string for the HTTP GET method or form serialized in the body for the HTTP POST method.

Attribute name	Presence	Type	Description	Example
user_id	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User	string	Unique identifier allowing to identify the Questioned User (e.g. Mobile phone , sub, ...). Ignored if the Access Token is tied with a specific End-User, mandatory otherwise.	3444975f-1137-47dc-908f-942ac85ab98f
user_id_type	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token	string	Indicate the type of the End-User’s identifier used for User Questioning.	MSISDN

	en is not tied with a End-User		Ignored if the Access Token is tied with a specific End-User, mandatory otherwise. Possible values are described in Section 2.1.		
question_to_display	MANDATORY	string	Question to be displayed to the Questioned User.	An example message to display	
wished_qcr	MANDATORY	string	Questioning context class reference : Level of Assurance wished by the Client (can be 2 3 4).	3	
wished_qmr	OPTIONAL	string	Questioning method wished	SMS_OTP	

```
   |             |         |        | by the  |                         |
   |             |         |        | Client  |                         |
   +-------------+---------+--------+---------+-------------------------+
```

In this flow, the Client MUST NOT include a client_notification_endpoint in the User Questioning Request.

3.1.2.1.1. Example with an access_token tied with a specific End-User

This example describes a context where the access_token is tied with a specific End-User. If the Client add user_id / user_id_type in the request, these parameters will be ignored.

The following is a non-normative example using HTTP GET.

```
GET /questions?
   question_to_display=An%20example%20message%20to%20display
   &wished_qcr=3
Host: server.example.com
Accept: application/json
Authorization: Bearer SlAV32hkKG
```

This example describes a context where the access_token is tied with a specific End-User. If the Client add user_id / user_id_type in the request, these parameters will be ignored.

The following is a non-normative example using HTTP POST.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Accept: application/json
Authorization: Bearer SlAV32hkKG

question_to_display=An%20example%20message%20to%20display&wished_qcr=3
```

3.1.2.1.2. Example with an access_token NOT tied with a specific End-User

This example describes a context where the access_token is not tied with a specific End-User. The Client MUST add user_id and user_id_type in the request.

The following is a non-normative example using HTTP GET.

```
   GET /questions?
      user_id=33612345678
      &user_id_type=MSISDN
      &question_to_display=An%20example%20message%20to%20display
      &wished_qcr=3
   Host: server.example.com
   Accept: application/json
   Authorization: Bearer SlAV32hkKG
```

The following is a non-normative example using HTTP POST.

```
 POST /questions HTTP/1.1
 Host: server.example.com
 Content-Type: application/x-www-form-urlencoded
 Accept: application/json
 Authorization: Bearer SlAV32hkKG

 user_id=33612345678&user_id_type=MSISDN
 &question_to_display=An%20example%20message%20to%20display&wished_qcr=3
```

3.1.2.2. (1b) Question Created

3.1.2.2.1. Successful response

The Client receives a HTTP 201 Created response.

The response body contains a JSON object with the following attributes.

Parameter name	Presence	Type	Description	Example
id	MANDATORY	string	Unique identifier of the Question.	984dcc7d-3d4d-4b0f-9f80-22e344f9a956
status	MANDATORY	string	Status code of the Question (note that Error codes are handled by the HTTP protocol).	"pending"

Parameters

In this flow, there are two possible status in the received JSON object:

{"status":"pending"} This is the temporary status when the User Questioning Response is not ready.

{"status":"verification_code_required"} This is a temporary status when an additional verification_code is required.

If the status is verification_code_required, then the current flow is the Terminated-By-Client Flow (Section 3.3).

3.1.2.2.1.1. Example

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"pending",
}
```

3.1.2.2.2. Error response

The Client receives a HTTP 400 Bad Request as specified in Section 4.2.

3.1.2.3. (2a) User interactions to get the Questioned User’s Statement

The way the User Questioning Endpoint obtains the Questioned User’s Statement for the Question is out of the scope of this specification.

3.1.2.4. (3a) Get User Questioning Response

The Client get the User Questioning Response using HTTP GET.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

In order to detect that User Questioning Response is ready, the Client MUST request the User Questioning Response. If User Questioning Response is ready, it will be returned in the response. Else, the OP responds with HTTP 304 Not Modified.

3.1.2.4.1. Example

The following is a non-normative example.

```
GET /questions/84c1d9d6-62e5-4803-ac0e-36b858c HTTP/1.1
Host: server.example.com
Accept: application/json
Authorization: Bearer SlAV32hkKG
```

3.1.2.5. (3b) User Questioning Response

The Client receives the User Questioning Response in a HTTP 200 OK or HTTP 304 Not Modified response.

3.1.2.5.1. User Questioning Response is not ready

If the User Questioning Response is not ready, the Client will get a HTTP 304 Not Modified.

3.1.2.5.1.1. Example when User Questioning Response is not ready

The following is a non-normative example.

```
HTTP/1.1 304 Not Modified
```

3.1.2.5.2. User Questioning Response is ready

The Client receives the User Questioning Response in a HTTP 200 OK response.

The User Questioning Response contains a User Statement Token in the response body Section 2.

3.1.2.5.2.1. Example of User Questioning Response with Accepted Statement

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"accepted",
    "creation_date":"1311281970",
    "last_modification_date":"1311282970",
    "statement_date":"1311282970",
    "question_displayed":"An example message to display",
    "used_qcr":"2",
    "used_qmr":"SMS_OTP"
}
```

3.1.2.5.2.2. Example of User Questioning Response with Denied Statement

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"denied",
    "creation_date":"1311281970",
    "last_modification_date":"1311282970",
    "statement_date":"1311282970",
    "question_displayed":"An example message to display",
    "used_qcr":"2",
    "used_qmr":"SMS_OTP"
}
```

3.1.2.5.3. Error response

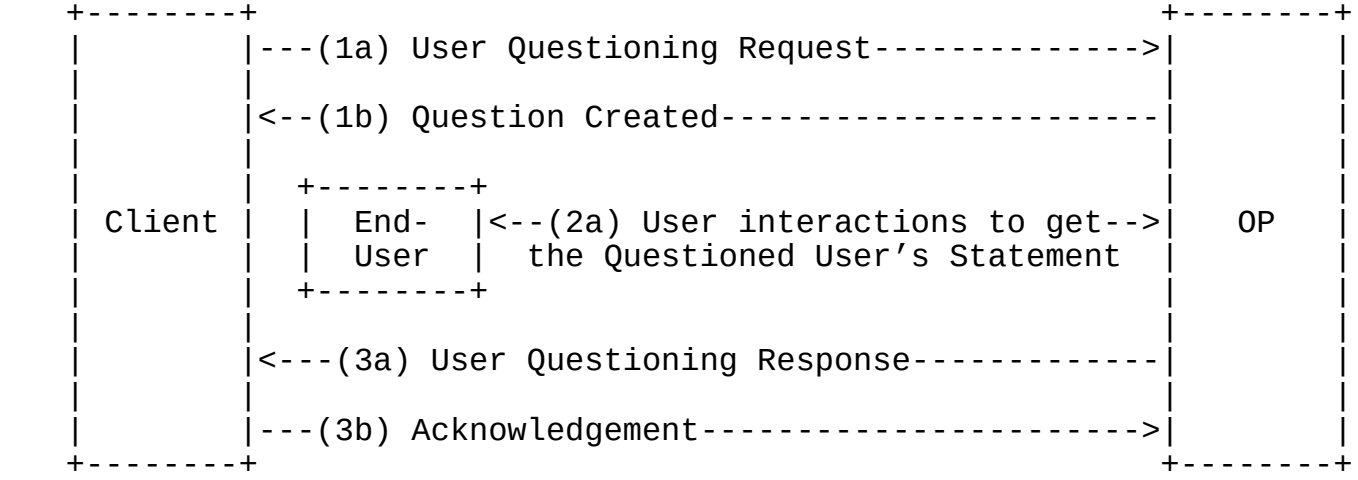
The Client receives a HTTP 400 Bad Request as specified in Section 4.2.

3.2. Pushed-To-Client Flow

In this flow, the Client MUST include a client_notification_endpoint in the User Questioning Request. The Client will be informed at this endpoint by the OP when the User Questioning is finished (accepted, denied, error...).

If a verification_code is needed for the mecanism chosen by the OP, the flow will be the Terminated-By-Client Flow (cf. Section 3.3), despite the client_notification_endpoint in the request.

3.2.1. Pushed-To-Client Flow Steps



3.2.2. User Questioning Endpoint

3.2.2.1. (1a) User Questioning Request

The Client sends the User Questioning Request using HTTP POST.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain a JSON object in the request body, with the following attributes.

Attribute name	Presence	Type	Description	Example
user_id	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User	string	Unique identifier allowing to identify the Questioned User (e.g. Mobile phone , sub, ...). Ignored if the Access Token is tied with a specific End-User, mandatory otherwise.	3444975f-1137-47dc-908f-942ac85ab98f
user_id_type	FORBIDDEN if the access_token	string	Indicate the type of the End-User’s i	MSISDN

	is tied with a End-User, MANDATORY if the access_token is not tied with a End-User		dentifier used for User Questioning. Ignored if the Access Token is tied with a specific End-User, mandatory otherwise. Possible values are described in Section 2.1.	
question_to_display	MANDATORY	string	Question to be displayed to the Questioned User.	An example message to display
wished_qcr	MANDATORY	string	Questioning context class reference : Level of Assurance wished by the Client (can be 2 3 4).	3
wished_qmr	OPTIONAL	string	Questioning method	SMS_OTP

			wished by the Client	
client_notification_endpoint	MANDATORY	uri	URL exposed by the Client's backend to receive a notification from OP when a User Questioning is finished (accepted, denied, error...). Needed if the Client wants to use Pushed-To-Client Flow (See Section 3.2).	https://client.example.com/questions

3.2.2.1.1. Example with an access_token tied with a specific End-User

This example describes a context where the access_token is tied with a specific End-User. If the Client add user_id / user_id_type in the request, these parameters will be ignored.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SlAV32hkKG

{
    "question_to_display":"An example message to display",
    "wished_qcr":"3",
    "client_notification_endpoint":"https://server.client.com/questions"
}
```

3.2.2.1.2. Example with an access_token NOT tied with a specific End-User

This example describes a context where the access_token is not tied with a specific End-User. The Client MUST add user_id and user_id_type in the request.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SlAV32hkKG

{
    "user_id":"33612345678",
    "user_id_type":"MSISDN",
    "question_to_display":"An example message to display",
    "wished_qcr":"3",
    "client_notification_endpoint":"https://server.client.com/questions"
}
```

3.2.2.2. (1b) Question Created

3.2.2.2.1. Successful response

The Client receives a HTTP 201 Created response.

The response body contains a JSON object with the following attributes.

Parameter name	Presence	Type	Description	Example
id	MANDATORY	string	Unique identifier of the Question.	984dcc7d-3d4d-4b0f-9f80-22e344f9a956
status	MANDATORY	string	Status code of the Question (note that Error codes are handled by the HTTP protocol).	"pending"

Parameters

In this flow, there are two possible status in the received JSON object:

{"status":"pending"} This is the temporary status when the User Questioning Response is not ready.

{"status":"verification_code_required"} This is a temporary status when an additional verification_code is required.

If the status is verification_code_required, then the current flow is the Terminated-By-Client Flow (Section 3.3).

3.2.2.2.1.1. Example with an access_token tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"pending",
}
```

3.2.2.2.1.2. Example with an access_token not tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"pending",
}
```

3.2.2.2.2. Error response

The Client receives a HTTP 400 Bad Request as specified in Section 4.2.

3.2.2.3. (2a) User interactions to get the Questioned User’s Statement

The way the User Questioning Endpoint obtains the Questioned User’s Statement for the Question is out of the scope of this specification.

3.2.3. Client Notification Endpoint

3.2.3.1. (3a) User Questioning Response

3.2.3.1.1. Successful response

The OP sends the User Questioning Response to the Client using HTTP POST.

The User Questioning Response MUST contain a JSON object in the request body, with the following attributes.

Attribute name	Presence
id	MANDATORY
status	MANDATORY
creation_date	MANDATORY
last_modification_date	MANDATORY
statement_date	MANDATORY
user_id	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
user_id_type	OPTIONAL if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User
question_displayed	MANDATORY
used_qcr	MANDATORY
used_qmr	OPTIONAL
...any other...	IGNORED

In this flow, there are three possible status (cf. Section 2.2) in the User Questioning Response:

{"status":"accepted"} This is the final status when the End-User accepted the User Questioning Request.

{"status":"denied"} This is the final status when the End-User denied the User Questioning Request.

{"status":"error"} This is the final status when there is an error. Refer to Section 4 for more details.

3.2.3.1.1.1. Example of Accepted Statement

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.client.com
Accept: application/json

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"accepted",
    "creation_date":"1311281970",
    "last_modification_date":"1311282970",
    "statement_date":"1311282970",
    "question_displayed":"An example message to display",
    "used_qcr":"3",
    "used_qmr":"SIM_APPLET",
}
```

3.2.3.1.1.2. Example of Denied Statement

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.client.com
Accept: application/json

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"denied",
    "creation_date":"1311281970",
    "last_modification_date":"1311282970",
    "statement_date":"1311282970",
    "question_displayed":"An example message to display",
    "used_qcr":"3",
    "used_qmr":"SIM_APPLET",
}
```

3.2.3.1.2. Error response

The Client receives a HTTP POST as specified in Section 4.3.

3.2.3.2. (3b) Acknowledgement

The acknowledgement MUST be a HTTP 200 OK. The HTTP code is the only parameter to consider. The rest of the HTTP response should be ignored.

3.2.3.2.1. Example of the simplest acknowledgement

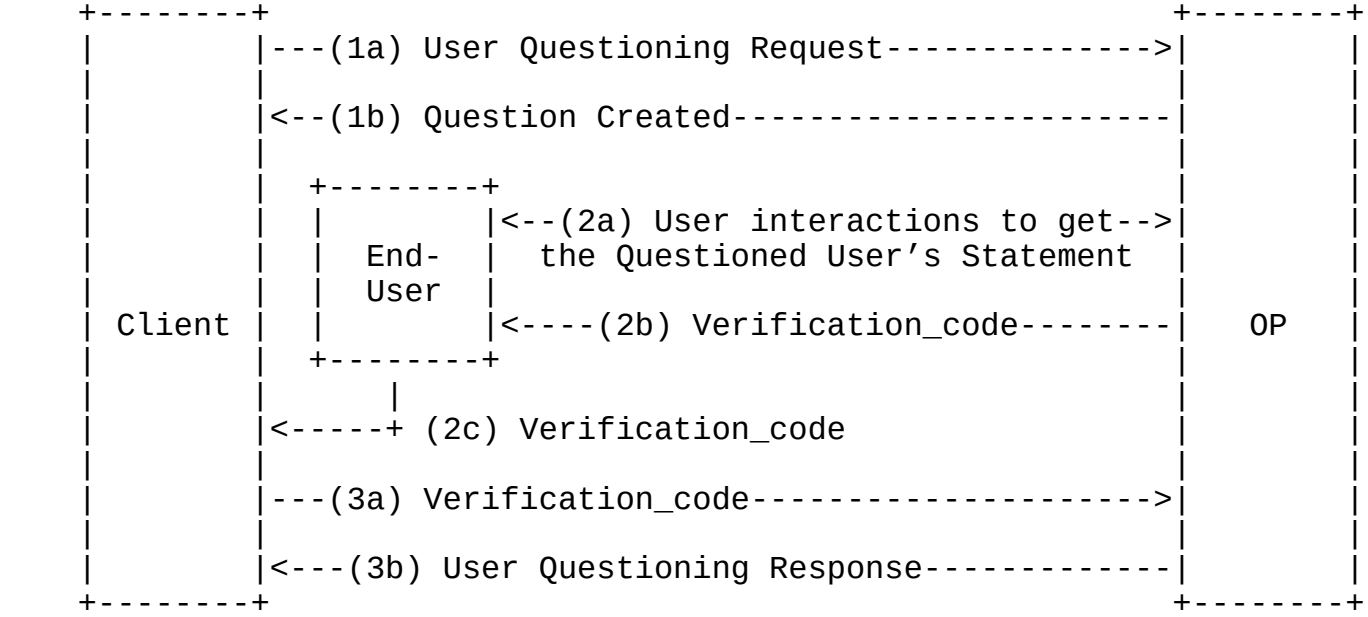
The following is a non-normative example.

HTTP/1.1 200 OK

3.3. Terminated-By-Client Flow

In this flow, after the first User Questioning Request, the Client must supply the OP with a verification_code in order to finish the User Questioning (accepted, denied, error...).

3.3.1. Terminated-By-Client Flow steps



3.3.2. User Questioning Endpoint

3.3.2.1. (1a) User Questioning Request

The Client sends the User Questioning Request using HTTP POST.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The User Questioning Request MUST contain a JSON object in the request body, with the following attributes.

Attribute name	Presence	Type	Description	Example

user_id	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User	string	Unique identifier allowing to identify the Questioned User (e.g. Mobile phone , sub, ...). Ignored if the Access Token is tied with a specific End-User, mandatory otherwise.	3444975f-1137-47dc-908f-942ac85ab98f
user_id_type	FORBIDDEN if the access_token is tied with a End-User, MANDATORY if the access_token is not tied with a End-User	string	Indicate the type of the End-User’s identifier used for User Questioning. Ignored if the Access Token is tied with a specific End-User, mandatory	MSISDN

			otherwise. Possible values are described in Section 2.1.		
question_to_display	MANDATORY	string	Question to be displayed to the Questioned User.	An example message to display	
wished_qcr	MANDATORY	string	Questioning context class reference : Level of Assurance wished by the Client (can be 2 3 4).	3	
wished_qmr	OPTIONAL	string	Questioning method wished by the Client	SMS_OTP	
client_notification_endpoint	IGNORED	uri	URL exposed by the Client's backend to receive a notification from OP when a	https://client.example.com/questions	

			User Questioning is finished (accepted, denied, error...). Needed if the Client wants to use Pushed-To-Client Flow (See Section 3.2).		

3.3.2.1.1. Example with an access_token tied with a specific End-User

This example describes a context where the access_token is tied with a specific End-User. If the Client add user_id / user_id_type in the request, these parameters will be ignored.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SlAV32hkKG

{
    "question_to_display":"An example message to display",
    "wished_qcr":"3"
}
```

3.3.2.1.2. Example with an access_token NOT tied with a specific End-User

This example describes a context where the access_token is not tied with a specific End-User. The Client MUST add user_id and user_id_type in the request.

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.example.com
Content-Type: application/json
Accept: application/json
Authorization: Bearer SlAV32hkKG

{
    "user_id":"33612345678",
    "user_id_type":"MSISDN",
    "question_to_display":"An example message to display",
    "wished_qcr":"3"
}
```

3.3.2.2. (1b) Question Created

3.3.2.2.1. Successful response

The Client receives a HTTP 201 Created response.

The response body contains a JSON object with the following attributes.

Parameter name	Presence	Type	Description	Example
id	MANDATORY	string	Unique identifier of the Question.	984dcc7d-3d4d-4b0f-9f80-22e344f9a956
status	MANDATORY	string	Status code of the Question (note that Error codes are handled by the HTTP protocol).	"pending"

Parameters

In this flow, there are two possible status in the JSON object received:

{"status":"pending"} This is the temporary status when the User Questioning Response is not ready.

If the status is "pending", then the current flow is either the Pulled-By-Client Flow (Section 3.1) or the Pushed-To-Client Flow (Section 3.2).

{"status":"verification_code_required"} This is a temporary status when an additional verification_code is required. This is the normal status for this flow.

3.3.2.2.1.1. Example with an access_token tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"verification_code_required",
}
```

3.3.2.2.1.2. Example with an access_token not tied with a specific End-User

The following is a non-normative example.

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"verification_code_required",
}
```

3.3.2.2.2. Error response

The Client receives a HTTP 400 Bad Request as specified in Section 4.2.

3.3.2.3. (2a) User interactions to get the Questioned User’s Statement

The way the User Questioning Endpoint obtains the Questioned User’s Statement for the Question is out of the scope of this specification.

3.3.2.4. (2b) User Questioning Endpoint Sends Verification_code to Questioned User

The way the User Questioning Endpoint sends the verification_code to the Questioned User is out of the scope of this specification.

3.3.2.5. (2c) Questioned User Sends Verification_code to Client

The way the End-User sends the verification_code to the Client is out of the scope of this specification.

3.3.2.6. (3a) Client Sends Verification_code to User Questioning Endpoint

The Client sends the verification_code to the OP using HTTP PUT.

The Access Token obtained from an OAuth Authorization request MUST be sent as a Bearer Token.

The HTTP request MUST contain the following attribute, form serialized in the body for the HTTP PUT method.

Attribute name	Presence
verification_code	MANDATORY
...any other...	IGNORED

3.3.2.6.1. Example

The following is a non-normative example.

```
PUT /questions/84c1d9d6-62e5-4803-ac0e-36b858c HTTP/1.1
Host: server.example.com
Accept: application/x-www-form-urlencoded
Content-Type: application/json
Authorization: Bearer SlAV32hkKG

verification_code=12345
```

3.3.2.7. (3b) User Questioning Response

3.3.2.7.1. Successful response

The Client receives the User Questioning Response in a HTTP 200 OK response.

The successful User Questioning Response MUST contain a User Statement Token in the response body Section 2.

3.3.2.7.1.1. Example of Accepted Statement

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "status":"accepted",
    "creation_date":"1311281970",
    "last_modification_date":"1311282970",
    "statement_date":"1311282970",
    "question_displayed":"An example message to display",
    "used_qcr":"2",
    "used_qmr":"SMS_OTP"
}
```

3.3.2.7.1.2. Example of Denied Statement

The following is a non-normative example.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "status":"denied",
    "creation_date":"1311281970",
    "last_modification_date":"1311282970",
    "statement_date":"1311282970",
    "question_displayed":"An example message to display",
    "used_qcr":"2",
    "used_qmr":"SMS_OTP"
}
```

3.3.2.7.2. Error response

The Client receives a HTTP 400 Bad Request as specified in Section 4.2.

4. Errors

4.1. error_info Object

error_info is a JSON object which contains the following properties :

Value	Description	Example
error_code	REQUIRED. Code representing the error. See section Section 4.1.1 for possible values.	unknown_user
error_description	OPTIONAL. Human-readable ASCII encoded text description of the error.	The user is unknown
error_uri	OPTIONAL. URI of a web page that includes additional information about the error.	https://server.example.com/errors/unknown_user

error_info properties

4.1.1. error_code

error_code is a property of the error_info Object. It can takes the following values :

Value	Description
unknown_user	The Questioned User is unknown.
timeout	The User Questioning has expired (timeout - no action from Questioned User).
verification_code_failed	The verification_code provided was not correct.
verification_code_too_many_tries	The verification_code has already been check without success several times (the number of time is up to the OP).

error_code possible values

4.2. Error received by the Client in a HTTP 400 Bad Request

The Client receives the error in a HTTP 400 Bad Request.

The response body contains a JSON object with the following attributes.

Parameter name	Presence	Type	Description	Example
error_info	MANDATORY	JSON Object	See Section 4.1.	{"error_code":"unknown_user"}

Parameters

4.2.1. Example of error in a HTTP 400 Bad Request

This example can occur in the Pulled-By-Client (cf. Section 3.1), Pushed-To-Client (cf. Section 3.2) and Terminated-By-Client (cf. Section 3.3) flows.

The following is a non-normative example.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Location: https://server.example.com/questions/84c1d9d6-62e5-4803-ac
0e-36b858c
Content-Length: xxx

{
    "error_info":
    {
        "error_code":"unknown_user",
        "error_description":"The user is unknown",
        "error_uri":"https://server.example.com/errors/unknown_user"
    }
}
```

4.3. Error received by the Client in a HTTP POST

The Client receives the error in a HTTP POST.

The request body contains a JSON object with the following attributes.

Parameter name	Presence	Type	Description	Example
id	MANDATORY	string	Unique identifier of the Question.	984dcc7d-3d4d-4b0f-9f80-22e344f9a956
status	MANDATORY	string	Status code of the Question.	"error"
error_info	MANDATORY	JSON Object	See Section 4.1.	{"error_code":"unknown_user"}

Parameters

4.3.1. Example of error in a HTTP POST

This example can occur in the Pushed-To-Client flow (cf. Section 3.2).

The following is a non-normative example.

```
POST /questions HTTP/1.1
Host: server.client.com
Accept: application/json

{
    "id":"84c1d9d6-62e5-4803-ac0e-36b858c",
    "status":"error",
    "error_info":
    {
        "error_code":"unknown_user",
        "error_description":"The user is unknown",
        "error_uri":"https://server.example.com/errors/unknown_user"
    }
}
```

5. Signatures and Encryption

cf. OpenID Connect - chapter 10

6. Security Considerations

TBD

7. Privacy Considerations

TBD

8. IANA Considerations

This document makes no requests of IANA.

9. Normative References

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token (work in progress), May 2013.

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, "OpenID Connect Standard 1.0", December 2013.

[OpenID.Discovery] Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0", September 2014.

[OpenID.Registration]
Sakimura, N., Bradley, J., and M. Jones, "OpenID Connect Dynamic Client Registration 1.0", December 2013.

[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>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <http://www.rfc-editor.org/info/rfc2246>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[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>.

Appendix A. Acknowledgements

The following have contributed to the development of this specification.

Appendix B. Notices

Appendix C. Document History

[[To be removed from the final specification]]

-01

- Initial draft
- Added OIDF Standard Notice

Authors' Addresses

Nicolas Aillery
Orange

Email: nicolas.aillery@orange.com

Charles Marais
Orange

Email: charles.marais@orange.com