Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 23 July 2026

Y. Zehavi
Raiffeisen Bank International

19 January 2026

OAuth 2.0 RAR Metadata and Error Signaling
draft-zehavi-oauth-rar-metadata-01

Abstract

OAuth 2.0 Rich Authorization Requests (RAR), as defined in [RFC9396], enables fine-grained authorization requests, using structured JSON objects.

While RAR [RFC9396] standardizes the exchange and handling of authorization details, it does not define a mechanism for clients to discover how to construct valid authorization details types.

This document defines a machine-readable metadata format for authorization servers to provide authorization details type documentation and JSON Schema [JSON.Schema] definitions, as well as interoperable discovery via OAuth Resource Server Metadata [RFC9728].

This document also defines a new WWW-Authenticate normative OAuth error code, insufficient_authorization_details, enabling resource servers to indicate inadequate authorization details as the cause of failure.

It also defines an OPTIONAL response body which MAY be returned alongside the insufficient_authorization_details error, providing an informative yet actionable authorization details object, which can be used directly in a subsequent OAuth request.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://yaron-zehavi.github.io/oauth-rich-authorization-requests-metadata/draft-zehavi-oauth-rar-metadata.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-zehavi-oauth-rar-metadata/.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (mailto:oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/. Subscribe at https://www.ietf.org/mailman/listinfo/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/yaron-zehavi/oauth-rich-authorization-requests-metadata.

Status of This Memo

This Internet-Draft will expire on 23 July 2026.

Copyright Notice

Table of Contents

1. Introduction

OAuth 2.0 Rich Authorization Requests (RAR) [RFC9396] allows OAuth clients to request structured, fine-grained authorization, beyond the coarse-grained access offered by simple scopes, which has enabled advanced authorization models across many domains, such as open banking & API marketplaces.

However, RAR [RFC9396] does not specify how a client learns how to construct syntactically valid authorization details objects. As a result, clients must rely on out-of-band documentation or static ecosystem profiles, limiting interoperability and preventing dynamic client behavior.

This document addresses this gap by:

* Defining a new authorization server endpoint: authorization_details_types_metadata_endpoint, providing metadata for authorization details types, including human-readable documentation as well as embedded JSON Schema definitions [JSON.Schema].

* Adding accepted authorization details types to OAuth 2.0 Protected Resource Metadata [RFC9728] response, facilitating RAR metadata discovery.

* Defining a standardized error signaling mechanism using the WWW-Authenticate response header, allowing resource servers to specify insufficient_authorization_details as the cause of error.

* Defining an OPTIONAL response body, included with an insufficient_authorization_details error, providing an informative authorization details object, whose inclusion in a new OAuth request shall result, if approved, in an access token satisfying the endpoint’s requirements.

The OPTIONAL providing of actionable authorization details objects by resource servers enables:

* High interoperability and simplification by relieving clients from having to figure out how to construct valid authorization details objects, instead providing them with required authorization_details object, to be included in a subsequent OAuth request.

* Support for including ephemeral, interaction-specific details originating from the resource domain, in the authorization details object, such as for example a risk score, a risk profile or an internal interaction identifier. Resource servers MAY use this to guide authorization servers as to the required authentication strength and consent flow.

2. Conventions and Definitions

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.

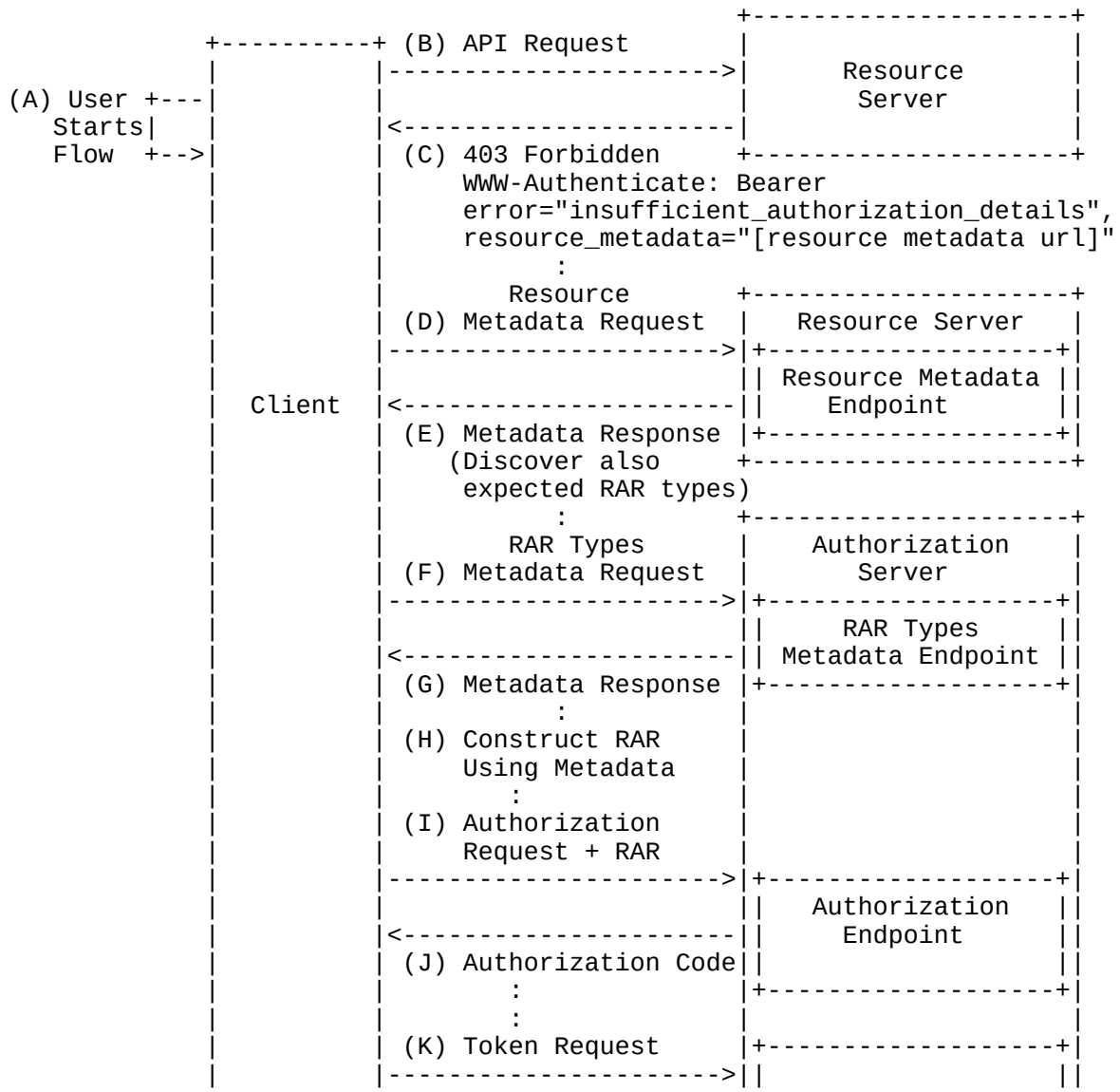
3. Protocol Overview

There are two main proposed flows:

   * Client *learns* to construct valid authorization details objects using authorization details types metadata.

   * Client *obtains an actionable authorization details object* from resource server's error response.

3.1. Client learns to construct valid authorization details objects using metadata



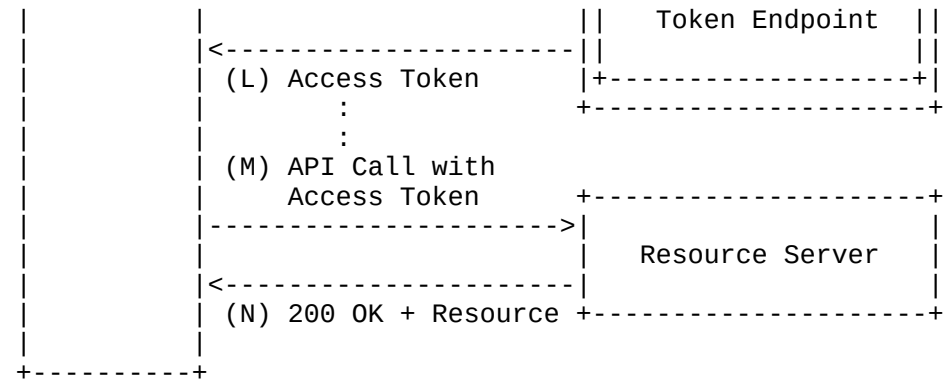


Figure: Client learns to construct valid authorization details objects from metadata

* (A) The user starts the flow.

* (B) The client calls an API with an access token.

* (C) Resource server returns HTTP 403 forbidden including a WWW-Authenticate header with error code insufficient_authorization_details and the resource metadata url (OAuth 2.0 Protected Resource Metadata [RFC9728]).

* (D-E) The client discovers expected authorization details types from resource metadata endpoint’s response.

* (F-G) The client consumes authorization details type metadata from authorization server’s authorization_details_types_metadata_endpoint.

* (H-I) The client constructs a valid authorization details object and makes an OAuth + RAR [RFC9396] request.

* (J) Authorization server returns authorization code.

* (K-L) The client exchanges authorization code for access token.

* (M) The client makes an API request with the (RAR) access token.

* (N) Resource server validates access token and returns successful response.

3.2. Client obtains authorization details object from resource server’s error response

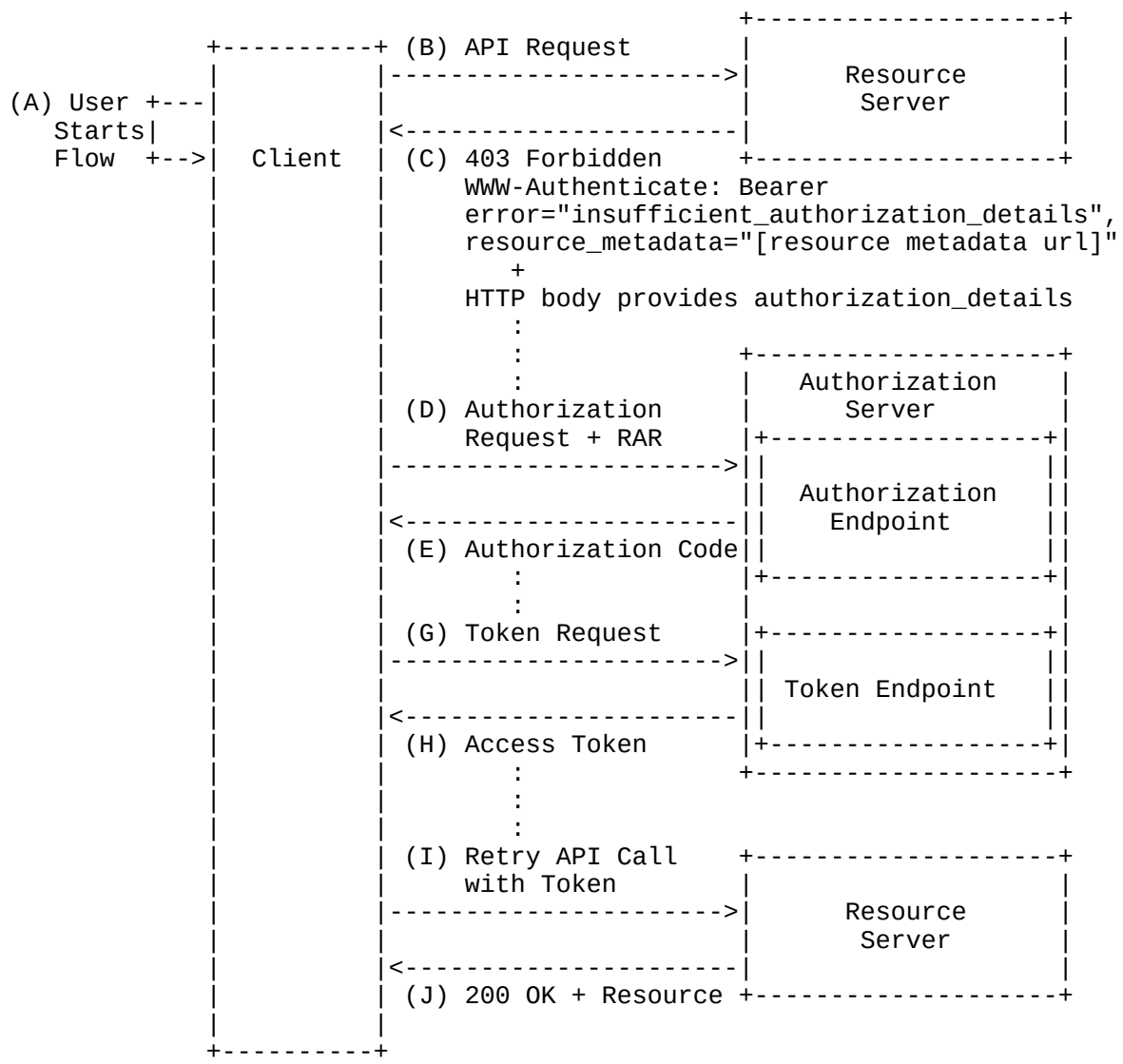


Figure: Client obtains authorization details object from resource server’s error response

* (A) The user starts the flow.

* (B) The client calls an API with an access token.

* (C) Resource server returns HTTP 403 forbidden including a WWW-Authenticate header with error code insufficient_authorization_details and in the response body *includes the authorization details object requiring approval*.

* (D) The client uses the obtained authorization details object in a new OAuth + RAR [RFC9396] request.

* (E) Authorization server returns authorization code.

* (G-H) The client exchanges authorization code for access token.

* (I) The client makes an API request with the (RAR) access token.

* (J) Resource server validates access token and returns successful response.

4. OAuth 2.0 Protected Resource Metadata [RFC9728]

This document specifies that the metadata attribute: authorization_details_types_supported, defined by RAR [RFC9396], shall be included as an OPTIONAL response attributes in Protected Resource Metadata [RFC9728].

Note: When resource servers accept access tokens _from several authorization servers_, interoperability is maintained as client can discover each authorization server’ supported authorization details types.

The following is a non-normative example response with the added authorization_details_types_supported attribute:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "resource":
  "https://resource.example.com",
  "authorization_servers":
    ["https://as1.example.com",
     "https://as2.example.net"],
  "bearer_methods_supported":
    ["header", "body"],
  "scopes_supported":
    ["profile", "email", "phone"],
  "resource_documentation":
    "https://resource.example.com/resource_documentation.html",
  "authorization_details_types_supported":
    ["payment_initiation"]
}
```

5. Authorization Details Types Metadata Endpoint

The following authorization server metadata [RFC8414] parameter is introduced to signal the server’s support for Authorization Details Types Metadata:

"authorization_details_types_metadata_endpoint": OPTIONAL. The URL of the Authorization Details Types Metadata endpoint.

5.1. Authorization Details Types Metadata Endpoint Response

The Authorization Details Types Metadata endpoint’s response is a JSON document with the key authorization_details_types_metadata whose attributes are authorization details type identifiers.

Each identifier is an object describing a single authorization details type.

```
{
  "authorization_details_types_metadata": {
    "type": {
    "version": "...",
    "description": "...",
    "documentation_uri": "...",
    "schema": { },
    "schema_uri": "...",
    "examples": [ ]
    }
  }
}
```

Attributes definition:

"version": OPTIONAL. String identifying the version of the authorization details type definition. The value is informational and does not imply semantic version negotiation.

"description": OPTIONAL. String containing a human-readable description of the authorization details type. Clients MUST NOT rely on this value for authorization or validation decisions.

"documentation_uri": OPTIONAL. URI referencing external human-readable documentation describing the authorization details type.

"schema": The schema attribute is a JSON Schema document [JSON.Schema] describing a single authorization detail object. The schema MUST validate a single authorization detail object and MUST constrain the type attribute to the authorization detail type identifier. This attribute is REQUIRED unless schema_uri is specified. If this attribute is present, schema_uri MUST NOT be present.

"schema_uri": The schema_uri attribute is an absolute URI, as defined by RFC 3986 [RFC3986], referencing a JSON Schema document describing a single authorization details object. The referenced schema MUST satisfy the same requirements as the schema attribute. This attribute is REQUIRED unless schema is specified. If this attribute is present, schema MUST NOT be present.

"examples": OPTIONAL. An array of example authorization details objects. Examples are non-normative.

See Examples Appendix A.1 for non-normative response example.

6. Resource Server Error Signaling of Inadequate authorization_details

This document defines a new normative OAuth error code, insufficient_authorization_details, which resource servers SHALL return using the WWW-Authenticate header, to signal access is denied due to missing or insufficient authorization details.

Example HTTP response:

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization_details",
    resource_metadata="https://resource.example.com/
    .well-known/oauth-protected-resource/payments"
```

6.1. OPTIONAL authorization_details in response body

Resource server MAY provide alongside the insufficient_authorization_details error, an HTTP response body of content type application/json, containing the required authorization details to satisfy the currently failing request.

Note:

* Authorization details objects provided by a resource server in an error response are intended for its trusted authorization servers, as advertised by the Resource Server’ s metadata endpoint.

* Resource servers SHALL provide authorization_details objects only if *all* trusted authorization servers accept the *authorization details type* used.

HTTP response body definition:

"authorization_details": OPTIONAL. Array of authorization details objects, matching the format specified in RAR [RFC9396] for the authorization_details request parameter.

Clients MAY use the provided authorization_details in a subsequent OAuth request to obtain an access token satisfying the resource’s requirements.

Example resource server response with OPTIONAL authorization_details:

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization_details",
    resource_metadata="https://resource.example.com/
    .well-known/oauth-protected-resource/payments"
Content-Type: application/json
Cache-Control: no-store

{
  "authorization_details": [{
    "type": "payment_initiation",
    "instructedAmount": {
      "currency": "EUR",
      "amount": "100.00"
    },
    "creditorAccount": {
      "iban": "DE02120300000000202051"
    }
  }]
}
```

7. Processing Rules

7.1. Client Processing Rules

* If encountering error insufficient_authorization_details, check if body.authorization_details exists and if provided MAY include in subsequent OAuth request.

* Otherwise consult metadata:

  - Fetch resource metadata to discover accepted authorization servers and supported *authorization_details types*.

  - Fetch authorization server metadata to discover authorization_details_types_supported.

  - Fetch authorization server’s authorization_details_types_metadata_endpoint to obtain metadata and schema

  - Locate schema or retrieve schema_uri.

* Construct authorization details conforming to the schema and include in subsequent OAuth request.

7.2. Resource Server Processing Rules

* Advertise in resource metadata authorization_details_types_supported, where relevant.

* Verify access tokens against required authorization details.

* If insufficient, return HTTP 403 with WWW-Authenticate: Bearer error="insufficient_authorization_details".

* OPTIONALLY provide also an HTTP body with an informative actionable authorization_details object.

8. Security Considerations

8.1. Cacheability and Intermediaries

HTTP 403 responses with response bodies may be cached or replayed in unexpected contexts. Recommended mitigation is resource servers SHALL use Cache-Control: no-store response header.

9. IANA Considerations

9.1. OAuth 2.0 Bearer Token Error Registry

Error Code	Description
insufficient_authorization_details	The request is missing required authorization details or the provided authorization details are insufficient.

Table 1

9.2. OAuth Metadata Attribute Registration

The metadata attribute authorization_details_types_metadata_endpoint is defined for OAuth authorization server metadata as a URL.

10. Normative References

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

[JSON.Schema]
Wright, Ed, A., Andrews, Ed, H., Hutton, Ed Postman, B., and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents", June 2022, <https://json-schema.org/draft/2020-12/json-schema-core>.

[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/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/rfc/rfc8414>.

[RFC9396] Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <https://www.rfc-editor.org/rfc/rfc9396>.

[RFC9728] Jones, M.B., Hunt, P., and A. Parecki, "OAuth 2.0 Protected Resource Metadata", RFC 9728, DOI 10.17487/RFC9728, April 2025, <https://www.rfc-editor.org/rfc/rfc9728>.

Appendix A. Examples

This section provides non-normative examples of how this specification may be used to support specific use cases.

A.1. Metadata Examples

A.1.1. Example authorization_details_types_metadata_endpoint response with Payment Initiation

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "authorization_details_types_metadata": {
        "payment_initiation": {
            "version": "1.0",
            "description": "Authorization to initiate a single payment from a payer accou
nt to a creditor account.",
            "documentation_uri": "https://example.com/docs/payment-initiation",
            "schema": {
                "$schema": "https://json-schema.org/draft/2020-12/schema",
                "title": "Payment Initiation Authorization Detail",
                "type": "object",
                "required": [
                    "type",
                    "instructed_amount",
                    "creditor_account"
                ],
                "properties": {
                    "type": {
                        "const": "payment_initiation",
                        "description": "Authorization detail type identifier."
                    },
                    "actions": {
                        "type": "array",
                        "description": "Permitted actions for this authorization.",
                        "items": {
                            "type": "string",
                            "enum": ["initiate"]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                    },
                    "instructed_amount": {
                        "type": "object",
                        "description": "Amount and currency of the payment to be initiate
d.",
                        "required": ["currency", "amount"],
                        "properties": {
                            "currency": {
                                "type": "string",
                                "description": "ISO 4217 currency code.",
                                "pattern": "^[A-Z]{3}$"
                            },
```

```
                            "amount": {
                                "type": "string",
                                "description": "Decimal monetary amount represented as a
string.",
                                "pattern": "^[0-9]+(\\.[0-9]{1,2})?$"
                            }
                        },
                        "additionalProperties": false
                    },
                    "creditor_account": {
                        "type": "object",
                        "description": "Account to which the payment will be credited.",
                        "required": ["iban"],
                        "properties": {
                            "iban": {
                                "type": "string",
                                "description": "International Bank Account Number (IBAN).
",
                                "pattern": "^[A-Z0-9]{15,34}$"
                            }
                        },
                        "additionalProperties": false
                    },
                    "remittance_information": {
                        "type": "string",
                        "description": "Unstructured remittance information for the payme
nt.",
                        "maxLength": 140
                    }
                },
                "additionalProperties": false
            }
        }
    }
}
```

A.2. Payment initiation with RAR error signaling

A.2.1. Client initiates API request

Client uses access token obtained at login to call payment initiation API

```
POST /payments HTTP/1.1
Host: resource.example.com
Content-Type: application/json
Authorization: Bearer eyj... (access token from login)

{
    "type": "payment_initiation",
    "locations": [
        "https://resource.example.com/payments"
    ],
    "instructedAmount": {
        "currency": "EUR",
        "amount": "123.50"
    },
    "creditorName": "Merchant A",
    "creditorAccount": {
        "bic": "ABCIDEFFXXX",
        "iban": "DE02100100109307118603"
    }
}
```

A.2.2. Resource server signals insufficient_authorization_details with actionable RAR object

Resource server requires payment approval and responds with:

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization_details",
    resource_metadata="https://resource.example.com
    /.well-known/oauth-protected-resource/payments"
Content-Type: application/json
Cache-Control: no-store

{
    "authorization_details": [{
      "type": "payment_initiation",
      "locations": [
          "https://example.com/payments"
      ],
      "instructedAmount": {
          "currency": "EUR",
          "amount": "123.50"
      },
      "creditorName": "Merchant A",
      "creditorAccount": {
          "bic": "ABCIDEFFXXX",
          "iban": "DE02100100109307118603"
      },
      "interactionId": "f81d4fae-7dec-11d0-a765-00a0c91e6bf6",
      "riskProfile": "B-71"
}]
}
```

Note: the resource server has added the ephemeral attributes interactionId and riskProfile.

A.2.3. Client initiates OAuth flow using the provided authorization_details object

After user approves the request, client obtains single-use access token representing the approved payment

A.2.4. Client re-attempts API request

```
POST /payments HTTP/1.1
Host: resource.example.com
Content-Type: application/json
Authorization: Bearer eyj... (payment approval access token)

{
    "type": "payment_initiation",
    "locations": [
        "https://resource.example.com/payments"
    ],
    "instructedAmount": {
        "currency": "EUR",
        "amount": "123.50"
    },
    "creditorName": "Merchant A",
    "creditorAccount": {
        "bic": "ABCIDEFFXXX",
        "iban": "DE02100100109307118603"
    }
}
```

A.2.5. Resource server authorizes the request

```
HTTP/1.1 201 Accepted
Content-Type: application/json
Cache-Control: no-store

{
    "paymentId": "a81bc81b-dead-4e5d-abff-90865d1e13b1",
    "status": "accepted"
}
```

Appendix B. Document History

-01

* Authorization details moved to HTTP body and made OPTIONAL

* Metadata pointer from resource metadata url, full authorization details types metadata on authorization server new endpoint

-00

* Document creation

Acknowledgments

Author’s Address

Yaron Zehavi
Raiffeisen Bank International
Email: yaron.zehavi@rbinternational.com