Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 11 July 2026

Y. Zehavi
Raiffeisen Bank International

7 January 2026

OAuth 2.0 RAR Metadata and Error Signaling
draft-zehavi-oauth-rar-metadata-00

Abstract

OAuth 2.0 Rich Authorization Requests (RAR), as defined in [RFC9396], enables clients to request fine-grained authorization 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 structure for advertising authorization details type documentation and JSON Schema [JSON.Schema] definitions via OAuth Authorization Server Metadata [RFC8414] and OAuth Resource Server Metadata [RFC9728].

In addition, this document defines a new OAuth error code, insufficient_authorization_details, enabling resource servers to return actionable authorization details objects to clients.

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

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. This has enabled advanced authorization models across domains, such as open banking & API marketplaces, and is well positioned for authorizing AI agents to perform state-changing actions.

However, RAR [RFC9396] does not specify how a client discovers the structure of supported authorization_detail types and how to construct syntactically valid authorization details.

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 metadata structure for authorization details types, containing both human-readable documentation as well as embedded JSON Schema [JSON.Schema] definitions.

* Discovery through Authorization Server Metadata [RFC8414], as well as via OAuth 2.0 Protected Resource Metadata [RFC9728].

* A standardized error signaling mechanism, allowing resource servers to return an authorization details object, to be included in a new Auth request, in order to accomplish a specific request.

It is up to implementers to decide if their clients MUST learn to construct valid authorization details objects, and if so whether authorization details types metadata should be provided by authorization servers, resource servers or both.

This document also outlines a solution pattern relieving clients from having to learn how to construct valid authorization details objects, instead providing clients the required authorization_details objects in resource servers' error responses. Clients then include the provided authorization details objects in subsequent OAuth requests.

This latter option is especially useful, as it enables resource servers to include ephemeral, interaction-specific details in the authorization details object, which are part of the resource domain, such as a risk score, risk profile or an internal interaction identifier.

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, either of them or both may be offered in parallel:

* Client learns to construct valid authorization details objects from authorization details types metadata provided by authorization servers, resource servers or both.

* Client obtains authorization details object from resource server's error response, providing an actionable authorization details object, whose inclusion in subsequent OAuth requests is required to accomplish a specific request.

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

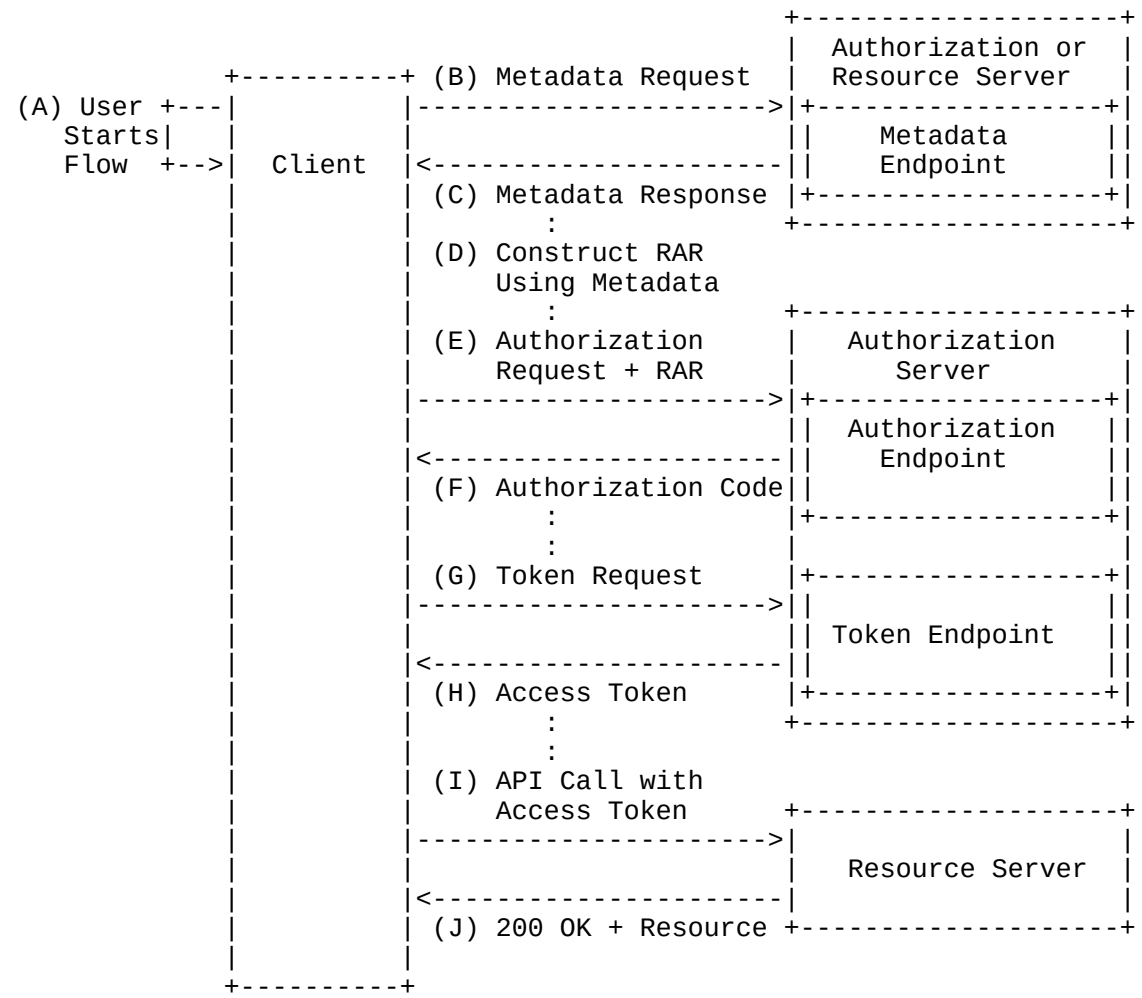

Figure: Client learns to construct valid authorization details objects from metadata

* (A) The user starts the flow.

* (B-C) The client discovers authorization details type metadata from Authorization Server Metadata [RFC8414], or from resource server Protected Resource Metadata [RFC9728]

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

* (F) Authorization server returns authorization code.

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

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

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

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

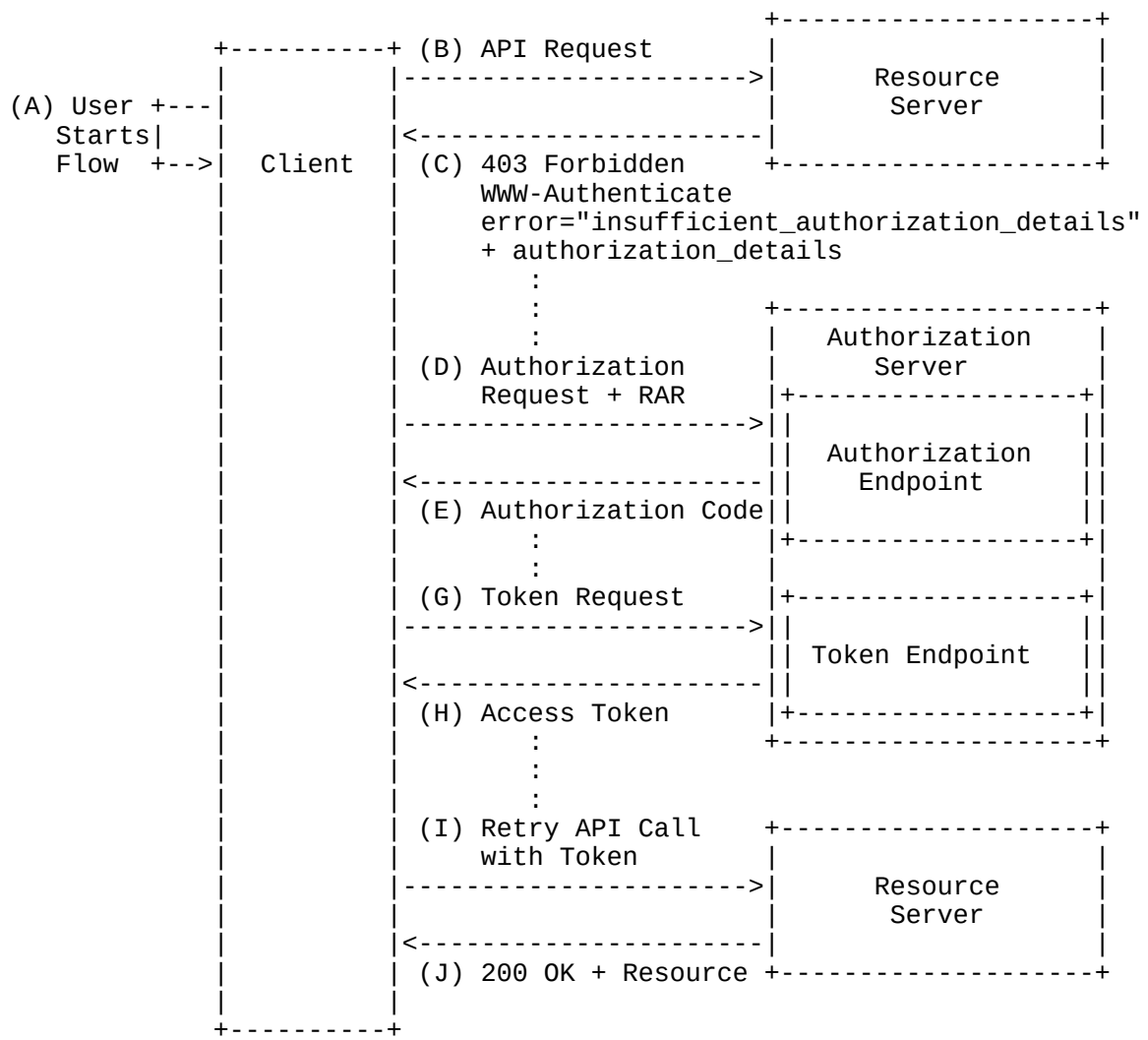

Figure: Client obtains authorization details object from resource server’s error response

* (A) The user starts the flow.

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

* (C) Resource server returns HTTP 403 forbidden because the access token does not contain required authorization details. Resource server’s response includes a WWW-Authenticate header with 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 API request with access token.

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

4. Authorization Details Type Metadata

4.1. Overview

This document defines a new metadata attribute, authorization_details_types_metadata, which provides documentation and validation information for authorization details types used with Rich Authorization Requests [RFC9396].

The new metadata property MAY be published by OAuth authorization servers using Authorization Server Metadata [RFC8414] as well as by Resource Servers using Protected Resource Metadata [RFC9728].

Clients MAY use this metadata to dynamically construct valid authorization_details objects.

This document also proposes the existing authorization_details_types_supported metadata attributed defined by RAR [RFC9396], be included in Protected Resource Metadata [RFC9728].

4.2. Metadata Location

The authorization_details_types_metadata attribute may be included in:

* Authorization Server Metadata [RFC8414], to describe authorization details types supported by the authorization server. If present, its keys MUST be a subset of the values listed in authorization_details_types_supported as defined in [RFC9396].

* OAuth 2.0 Protected Resource Metadata [RFC9728], to describe authorization details types accepted by the protecred resource.

4.3. Metadata Structure

The authorization_details_types_metadata attribute is a JSON object whose keys are authorization details type identifiers. Each value 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.

4.4. Metadata Examples

4.4.1. Example RAR Metadata: Payment Initiation

```
{
    "authorization_details_types_supported": ["payment_initiation"],
    "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
            }
        }
    }
}
```

5. Resource Server Error Signaling with insufficient_authorization_details

5.1. Overview

This document defines a new OAuth error code, insufficient_authorization_details, for use when access is denied due to missing or insufficient authorization details.

5.2. Error Definition

The error MUST be conveyed using the WWW-Authenticate header and MUST include an authorization_details parameter.

5.3. authorization_details Error Parameter

The parameter MUST contain a JSON object or array, representing the required authorization details, whose inclusion in a subsequent OAuth request is required to satisfy the resource server's requirements for this specific request. The value MUST be base64url-encoded.

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization_details",
authorization_details="{base64url-encoded JSON of required RAR}"
```

6. Processing Rules

6.1. Client Processing Rules

* Fetch authorization_details_types_metadata from the authorization or resource server's metadata endpoints.
* Locate schema or retrieve schema_uri.
* Construct authorization details conforming to the schema.
* If resource server returns error insufficient_authorization_details, use provided authorization_details in subsequent OAuth request, then provide the obtained token to resource server.

6.2. Authorization Server Processing Rules

* Advertise authorization_details_types_metadata in metadata.
* Validate each authorization detail against schema.
* Enforce additional semantic checks.
* Reject missing or invalid details with standard OAuth error semantics.

6.3. Resource Server Processing Rules

* Advertise authorization_details_types_metadata.
* Verify tokens against required authorization details.
* If insufficient, return HTTP 403 with WWW-Authenticate: Bearer error="insufficient_authorization_details".
* Do not reveal additional sensitive information.

7. Security Considerations

8. IANA Considerations

8.1. 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. The resource server SHOULD include the required authorization_details

Table 1

8.2. OAuth Metadata Attribute Registration

The metadata attribute authorization_details_types_metadata is defined for OAuth authorization and resource server metadata, as a JSON object mapping authorization details types to documentation, schema, and examples.

9. 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. Payment initiation with RAR error signaling

A.1.1. Client initiates API request

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

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

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

A.1.2. Resource server signals insufficient_authorization_details

Resource server requires payment approval and responds with:

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization_details",
    resource_metadata="https://server.example.com/.well-known/oauth-protected-resource/pa
yments",
    authorization_details=W3sKICAgICJ0eXBlIjogInBheW1lbnRfaW5pdGlhdGlvbiIsCiAgICAibG9jYXR
pb25zIjogWwogICAgICAgICJodHRwczovL2V4YW1wbGUuY29tL3BheW1lbnRzIgogICAgXSwKICAgICJpbnN0cnVj
dGVkQW1vdW50IjogewogICAgICAgICJjdXJyZW5jeSI6ICJFVVIiLAogICAgICAgICJhbW91bnQiOiAiMTIzLjUwI
gogICAgfSwKICAgICJjcmVkaXRvck5hbWUiOiAiTWVyY2hhbnQgQSIsCiAgICAiY3JlZGl0b3JBY2NvdW50Ijogew
ogICAgICAgICJiaWMiOiAiQUJDSURFRkZYWFgiLAogICAgICAgICJpYmFuIjogIkRFMDIxMDAxMDAxMDkzMDcxMTg
2MDMiCiAgICB9LAogICAgImludGVyYWN0aW9uSWQiOiAiZjgxZDRmYWUtN2RlYy0xMWQwLWE3NjUtMDBhMGM5MWU2
YmY2IiwKCSJyaXNrUHJvZmlsZSI6ICJCLTcxIgp9XQ==
```

The base64 encoded authorization_details decodes to:

```
[{
    "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, riskProfile.

A.1.3. 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.1.4. Client re-attempts API request

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

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

A.1.5. 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

-00

* Document creation

Acknowledgments

Author’s Address

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