Web Authorization Protocol
Internet-Draft
Intended status: Informational
Expires: 1 January 2026

J. Lombardo
AWS
A. Babeanu
IndyKite
Y. Zehavi
Raiffeisen Bank International
G. Fletcher
Practical Identity
30 June 2025

OAuth 2.0 step-up authorization challenge proto
draft-lombardo-oauth-step-up-authz-challenge-proto-02

Abstract

It is not uncommon for resource servers to require additional information like details of delegation authorization, or assurance proof of the delegation of authorization mechanism used according to the characteristics of a request. This document introduces a mechanism that resource servers can use to signal to a client that the data and metadata associated with the access token of the current request does not meet its authorization requirements and, further, how to meet them. This document also codifies a taxonomy to guide the client into starting a new request towards the authorization server in order to get issued, if applicable, a new set of tokens matching the requirements.

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://identitymonk.github.io/draft-lombardo-oauth-step-up-authz-challenge/draft-lombardo-oauth-step-up-authz-challenge-proto.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-lombardo-oauth-step-up-authz-challenge-proto/.

Discussion of this document takes place on the Web Authorization Protocol 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/identitymonk/draft-lombardo-oauth-step-up-authz-challenge.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 1 January 2026.

Table of Contents

1. Introduction

In simple authorization scenarios, an authorization server will determine what claims to embed in the tokens to issue on the basis of aspects such as the scopes requested, the resource, the identity of the client, and other characteristics known a provisioning time. [RFC9470] helped improve the feedback a resource server can provide to a client in case the user authentication method, authentication class, or the freshness of the authentication event did not meet the requirements expected by the resource server. Although those approaches are viable in many situations, it falls short in several important circumstances, for instance, in [FAPI2.0-Security-Profiles] or [hl7.fhir.uv.smart-app-launch] regulated APIs when they require peculiar client authentication mechanisms to be enforced or transaction specific details to be present in the token. These requirements may depend upon resource access rules or policies implemented at a policy decision point it relies on, using a logic that is opaque to the authorization server.

This document extends the collection of error codes defined by [RFC6750] and by [RFC9470] with one new error code, insufficient_authorization, which can be used by resource servers to signal to clients that the access token presented with the request does not meet the authorization requirements of the resource server. This document also introduces authorization step-up challenges and associated payload definitions. The resource server can use these payloads to explicitly communicate to the client its authorization requirements.

The client can then use this information to reach back to the authorization server with a new authorization request that specifies the additional authorization details required for issuing tokens useable at the resource. This document does not describe any new methods to perform this additional authorization request but will rely on OAuth 2.0 Rich Authorization Request [RFC9396], OAuth 2.0 Pushed Authorization Request [RFC9126], or OAuth 2.0 JWT-Secured Authorization Request [RFC9101] for this purpose. These extensions will make it possible to implement interoperable step up authorization flows with minimal work from resource servers, clients, and authorization servers.

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.

This specification uses the terms "access token", "authorization server", "authorization endpoint", "authorization request", "client", "protected resource", and "resource server" defined by [RFC6749].

3. Protocol Overview

The following is an end-to-end sequence of a typical step up authorization scenario implemented according to this specification. The scenario assumes that the client obtained an access token for the protected resource before the sequence described below takes place.

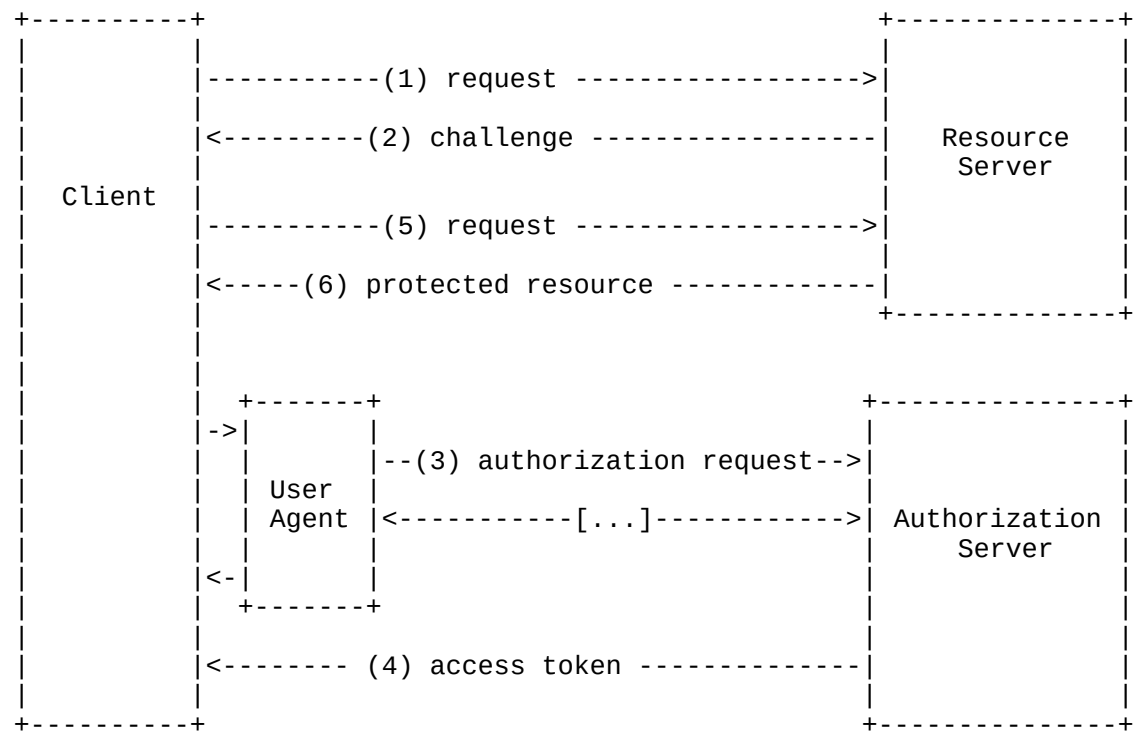

Figure 1: Abstract Protocol Flow

1. The client requests a protected resource, presenting an access token.

2. The resource server determines that the circumstances in which the presented access token was obtained offer insufficient authorization details, wrong grant flow, or inadequate client authentication mechanism; it therefore denies the request and returns a challenge describing (using a combination of error code and payload details) what authorization requirements must be met to allow the request. It is possible here for the resource server to rely on the responses from an external policy decision point.

3. The client redirects the user agent to the authorization server with an authorization request that includes the authorization details indicated by the resource server in the previous step.

4. A new and adequate authorization sequence takes place between the user agent, the client and the authorization server, resulting in the issuance of a new access token that encapsulates the

authorization level, or ceremonies requested by the resource server. The authorization server uses the payload for the resource server's response forwarded by the client to initiate the right grant flow type or to ensure that the authorization details are met. The authorization server may here also contact an external policy decision point to request evaluation of complex business access policies. The newly minted access token contains or references information about the authorization elements required, including but not limited to the claims defined in [I-D.lombardo-oauth-client-extension-claims].

5. The client repeats the request from step 1, presenting the newly obtained access token.

6. The resource server finds that the authorization details, grant flow or client authentication mechanism used during the acquisition of the new access token complies with its requirements and returns the representation of the requested protected resource.

Such protocol flow is coherent with the expectations of [FAPI2.0-Security-Profiles] and section 2.1.10.2.1 of [hl7.fhir.uv.smart-app-launch].

The validation operations mentioned in steps 2 and 6 imply that the resource server has a way of evaluating the authorization requirements that occurred during the ceremonies that led to the issuance of the access token. In the context of this document, the assessment by the resource server of the specific authorization mechanisms used to obtain a token for the requested resource is called an "authorization state".

This document does not describe how the resource server performs this assessment of the authorization state, whether the access token is a JSON Web Token (JWT) [RFC9068] or is validated via introspection [RFC7662] or whether the resource provider is performing this assessment natively or by offloading the assessment to a policy decision point as defined in [D-OpenID-AuthZEN], [XACML] or NIST's ABAC [SP.800-162]. This document rather describes how the resource provider tells the client what type of authorization it needs to get from the authorization server for a request.

The terms "authorization state" and "step up" are metaphors in this specification. These metaphors do not suggest that there is an absolute hierarchy of authorization states expressed in interoperable fashion. The notion of a state emerges from the fact that the resource server may only want to accept certain authorization mechanisms. When presented with a token derived from particular

authorization mechanisms (i.e., a given authorization state) that it does not want to accept (i.e., below the threshold it will accept), the resource server seeks to step up (i.e., renegotiate) from the current authorization state to one that it may accept. The "step up" metaphor is intended to convey a shift from the original authorization state to one that is acceptable to the resource server.

As for [RFC9470], although the case in which the new access token supersedes old tokens by virtue of a higher authorization state is common, in line with the connotation of the term "step up authorization", it is important to keep in mind that this might not necessarily hold true in the general case. For example, for a particular request, a resource server might require a higher authorization state and a shorter validity, resulting in a token suitable for one-off calls but leading to frequent prompts: hence, offering a suboptimal user experience if the token is reused for routine operations. In such a scenario, the client would be better served by keeping both the old tokens, which are associated with a lower authorization state, and the new one: selecting the appropriate token for each API call. This is not a new requirement for clients, as incremental consent and least-privilege principles will require similar algorithms for managing access tokens associated with different scopes and permission levels. This document does not recommend any specific token-caching strategy: that choice will be dependent on the characteristics of every particular scenario and remains application-dependent as in the core OAuth cases. Furthermore, OAuth 2.0 [RFC6749] assumes access tokens are treated as opaque by clients. The token format might thus be unreadable to the client or might change at any time to become unreadable. So, during the course of any token-caching strategy, a client must not attempt to inspect the content of the access token to determine the associated authentication information or other details (see Section 6 of [RFC9068] for a more detailed discussion).

4. Step-Up Authorization Challenge

4.1. HTTP Error Status Code

The resource server responds with a 403 HTTP status code using the Bearer authentication scheme’s error parameter (from [RFC6750]) when a request compliant with this specification does not meet its authorization state requirements.

4.2. WWW-Authenticate Header Error Code

Following other standards like OAuth 2.0 Demonstrating Proof of Possession (DPoP)[RFC9449] and in OAuth 2.0 Step Up Authentication Challenge Protocol[RFC9470], the error code of the WWW-Authenticate HTTP Header MUST be set to insufficient_authorization.

This will signal that the authorization mechanisms used for the issuance of the access token presented with the request do not meet the authorization state requirements of the protected resource. The client is provided with a response that describes which new authorization mechanisms and details SHOULD be used in order to gain access to the requested resource. The client SHOULD then initiate a new ceremony with the authorization server, that comply with the stated resource requirements.

Note: the logic through which the resource server determines that the current request does not meet the authorization state requirements of the protected resource, and associated functionality are out of scope for this document.

4.3. WWW-Authenticate Header Description

The description field of the WWW-Authenticate HTTP Header MUST be set to The authorization level requires more details.

4.4. WWW-Authenticate Header Challenge

This document specifies two types of challenge that MAY be used individually or conjointly.

4.4.1. resource_metadata_uri Challenge

The resource server MAY return a challenge with the key resource_metadata_uri. When this challenge is present, its value MUST be set to the OAuth2 Protected Resource Metadata Endpoint URI for the resource server as defined in [RFC9728].

The metadata endpoint MAY provide information on authorization details expected by the resource server.

Note: the logic through which the client determines how to use any information on authorization details expected by the resource server from the metadata endpoint and how to map it to a viable authorization request for the authorization server is out of the scope of this document.

4.4.2. body_instructions Challenge

The resource server MAY return a challenge with the key body_instructions When this challenge is present, its value MUST be set to false if no body is associated to the response from the resource server, or true if a body is associated to the response from the resource server.

4.4.2.1. Challenge Associated Body Content

If a body is attached to the to the response from the resource server, it MUST be formatted as an AuthZEN [D-OpenID-AuthZEN] response which MUST contain the following keys:

decision: _REQUIRED_ - MUST be set to false.

context: _REQUIRED_ - JSON structure as described below

The context MAY contain the following information

error_msg: _OPTIONAL_ - a human readable String that summarize the details required

details: _OPTIONAL_ - a JSON structure representing the expected parameters to be passed by the client to the authorization server if a new authorization request is attempted. This JSON structure MUST be formatted using the syntax defined by [eKYC.IDA]

4.5. Non Normative Examples Of Step-Up Authorization Challenge

The following are non normative examples of some step-up authorization challenges:

4.5.1. resource_metadata_uri challenge and body_instructions challenge expressing a missing expected access token scope:

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization", error_description="The autho
rization level is not met", resource_metadata_uri="https://www.example.com/.well-known/oa
uth-protected-resource", body_instructions=true

{
  "decision": false,
  "context": {
        "error_msg": "Missing expected access token scope",
        "details": [{
    "loc": "/scope",
    "method": "simple",
    "values": ["resource:read", "resource:write"]
    }]
  }
}
```

4.5.2. body_instructions challenge expressing missing authorization_details:

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization", error_description="The autho
rization level is not met",  body_instructions=true

{
    "decision": false,
    "context": {
        "error_msg": "Missing authorization_details",
        "details": [{
      "loc": "/authorization_details",
      "method": "simple",
      "value": [{
        "type": "payment_initiation",
          "actions": [
            "initiate",
            "status",
            "cancel"
          ],
          "locations": ["https://example.com/payments"],
          "instructedAmount": {
            "currency": "EUR",
            "amount": "123.50"
          },
          "creditorName": "Merchant A",
          "creditorAccount": {
            "iban": "DE02100100109307118603"
          },
          "remittanceInformationUnstructured": "Ref Number Merchant"
      }]
    }]
  }
}
```

4.5.3. body_instructions challenge expressing missing email as an expected access token claim

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization", error_description="The autho
rization level is not met",  body_instructions=true

{
    "decision": false,
    "context": {
        "error_msg": "Missing expected access token scope",
        "details": [{
      "loc": "/email",
      "method": "exists"
    }]
  }
}
```

4.5.4. body_instructions challenge expressing missing gty, ccr, and cmr as an expected access token claim

```
HTTP/1.1 403 Forbidden
WWW-Authenticate: Bearer error="insufficient_authorization", error_description="The autho
rization level is not met",  body_instructions=true

{
    "decision": false,
    "context": {
        "error_msg": "Missing token claims - gty, ccr, cmr",
        "details": [{
    "loc": "/gty",
    "method": "exists"
    }, {
    "loc": "/ccr",
    "method": "exists"
    }, {
    "loc": "/cmr",
    "method": "exists"
    }]
  }
}
```

Note: gty, ccr, and cmr are claims defined by [I-D.lombardo-oauth-client-extension-claims]

5. Client Action Following A Step-Up Authorization Challenge

This document does not define how the client should respond to a step-up authorization challenge. It is up to the logic of the client to decide what is the most appropriate grant type flow to start in order to try to obtain a new set of tokens from the authorization server.

6. Deployment Considerations

This specification facilitates the communication of requirements from a resource server to a client, which, in turn, can enable a more granular and appropriate Authorization Request at the Authorization Server using either an OAuth 2.0 [RFC6749] defined grant flows, a Rich Authorization Request [RFC9396], a Push Authorization Request [RFC9126], or a JWT-Secured Authorization Request [RFC9101] as it sees fit. However, it is important to realize that the user experience achievable in every specific deployment is a function of the policies each resource server and authorization server pair establishes. Imposing constraints on those policies is out of scope for this document. It is therefore perfectly possible for resource servers and authorization servers to impose requirements that are

impossible for subjects or clients to comply with or that lead to an undesirable user-experience outcome.

7. Resource Server Metadata

Resource servers can advertise their support of this specification by including in their OAuth protected resource metadata document, as defined in [RFC9728], the value step_up_authorization_supported. The presence of step_up_authorization_supported in the resource server metadata document signals that the resource server MAY honor the issuance of step-up authorization challenge if its sees fit.

8. Security Considerations

This specification adds to previously defined OAuth mechanisms. Their respective security considerations apply:

* OAuth 2.0 [RFC6749],
* JWT access tokens [RFC9068],
* Bearer WWW-Authenticate [RFC6750],
* Token introspection [RFC7662],
* OAuth2 Protected Resource Metadata [RFC9728],
* Rich Authorization Request [RFC9396],
* Push Authorization Request [RFC9126],
* JWT-Secured Authorization Request [RFC9101] and
* AuthZEN [D-OpenID-AuthZEN]

8.1. Scope

This specification does not attempt to define the mechanics by which access control is made by the resource provider and how the result of such access control evaluation should be translated into one of the challenges defined in Section 4.

This specification does not attempt to define the mechanics by which extended authorization requests are processed and validated by the authorization server.

8.2. Validation Of Token

For this specification, the resource provider MUST examine the incoming access token and enforce the conventional token-validation logic - be it based on JWT validation, introspection, or any other method - before determining whether or not a challenge should be returned.

8.3. Step-Up Authorization Challenge Payload

Following this document, response from the resource server to the client might unintentionally disclose information about the subject, the resource, the action to be performed, as long as context-specific data such as but not limited to authorization details that an attacker might use to gain knowledge about their target.

Implementers should use care in determining what to disclose in the challenge and in what circumstances.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[D-OpenID-AuthZEN]
GAzitt, O., Ed., Brossard, D., Ed., and A. Tulshibagwale, Ed., "Authorization API", 2025, <https://openid.github.io/authzen/>.

[eKYC.IDA] Fett, D. D., Ed., "OpenID Connect Advanced Syntax for Claims (ASC) 1.0", n.d., <https://openid.bitbucket.io/ekyc/openid-connect-advanced-syntax-for-claims.html>.

[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>.

[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>.

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

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC9068] Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021, <https://www.rfc-editor.org/rfc/rfc9068>.

[RFC9470] Bertocci, V. and B. Campbell, "OAuth 2.0 Step Up Authentication Challenge Protocol", RFC 9470, DOI 10.17487/RFC9470, September 2023, <https://www.rfc-editor.org/rfc/rfc9470>.

[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>.

10.2. Informative References

[FAPI2.0-Security-Profiles] Fett, D. D., Ed., Tonge, D., Ed., and J. Heenan, Ed., "FAPI 2.0 Security Profile", n.d., <https://openid.net/specs/fapi-2_0-security-02.html>.

[hl7.fhir.uv.smart-app-launch] "HL7 FHIR SMART App Launch", n.d., <https://www.hl7.org/fhir/smart-app-launch/app-launch.html#obtain-authorization-code>.

[I-D.ietf-oauth-v2-1]
Hardt, D., Parecki, A., and T. Lodderstedt, "The OAuth 2.1 Authorization Framework", Work in Progress, Internet-Draft, draft-ietf-oauth-v2-1-13, 28 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-13>.

[I-D.lombardo-oauth-client-extension-claims]
Lombardo, J. and A. Babeanu, "OAuth 2.0 client extension claims", Work in Progress, Internet-Draft, draft-lombardo-oauth-client-extension-claims-01, 28 June 2025, <https://datatracker.ietf.org/doc/html/draft-lombardo-oauth-client-extension-claims-01>.

[MCP] "Model Context Protocol", n.d., <https://modelcontextprotocol.io/specification/2025-06-18/basic>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/rfc/rfc7662>.

[RFC9101] Sakimura, N., Bradley, J., and M. Jones, "The OAuth 2.0 Authorization Framework: JWT-Secured Authorization Request (JAR)", RFC 9101, DOI 10.17487/RFC9101, August 2021, <https://www.rfc-editor.org/rfc/rfc9101>.

[RFC9126] Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <https://www.rfc-editor.org/rfc/rfc9126>.

[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>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

[SP.800-162] Hu, V., Ed., Ferraiolo, D., Ed., Kuhn, R., Ed., Schnitzer, A., Ed., Sandlin, K., Ed., Miller, R., Ed., and K. Scarfone, Ed., "Guide to Attribute Based Access Control (ABAC) Definition and Considerations", 2014, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf>.

[XACML] Godik, S., Ed. and T. M. (Ed.), Ed., "eXtensible Access Control Markup Language (XACML) Version 1.1", 2006, <https://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf>.

Appendix A. Use Cases

A.1. LLM Agent accessing a service via an LLM Tool on behalf of a user

LLM agents, including those based on large language models (LLMs), are designed to manage user context, memory, and interaction state across multi-turn conversations. To perform complex tasks, these agents often integrate with external systems such as SaaS applications, internal services, or enterprise data sources. When accessing these systems, the agent operates on behalf of the end user, and its actions are constrained by the user's identity, role, and permissions as defined by the enterprise. This ensures that all data access and operations are properly scoped and compliant with organizational access controls.

When using an LLM Tool, the LLM Agent is consuming an external API which has its own access control logic and policies on what conditions should be met for the access to the resources and the generation of the enriched information that the AI Agent can send return to the user who prompted it, with potential support of the LLM. Some access control conditions might require some authorization details as defined into, without being limited to, the examples of Rich Authorization Request [RFC9396].

A non normative example of such interaction would be at the functional level:

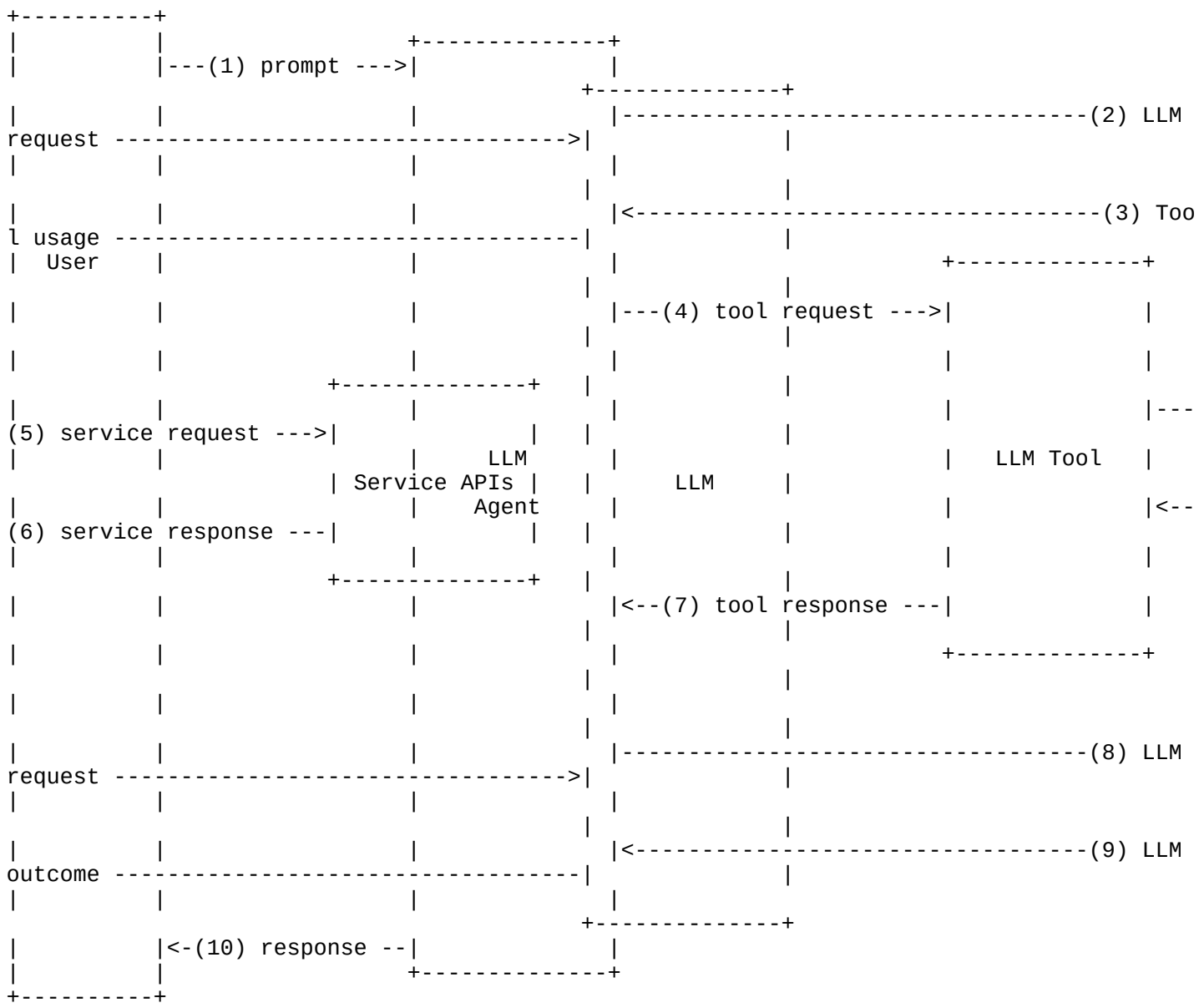

Figure 2: Abstract AI Agent Use Case Flow

A.1.1. Preconditions

* The LLM Agent has a registered OAuth 2.0 Client (com.example.llm-agent) with the Enterprise IdP (idp.example.com)
* The LLM Tool has a registered OAuth 2.0 Client (4960880b83dc9) with the Enterprise IdP (idp.example.com)
* The LLM Tool has a registered OAuth 2.0 Client (eb1e27d2df8b7) with External Service IdP (authorization-server.saas.net)
* The External Service APIs is protected by the Trust Domain controlled by the External Service IdP (authorization-server.saas.net)
* User already authenticated at the Enterprise IdP (idp.example.com) and delegated its authorization to the LLM Agent

* The LLM Agent is in possession of an Identity Token, an Access Token, and a Refresh Token issued by the Enterprise IdP (idp.example.com)

* We assume that the Access Token is valid for the duration of this example and possess the appropriate scopes and claims to be authorized to call the LLM Tool

A.1.2. LLM Agent receives a response from the LLM

LLM Agent receives a directive to use the LLM Tool with a specific payload and it calls the external LLM Tool provided by an Enterprise internal IT with a valid access token.

POST /Pay Host: tool.example.com Authorization: Bearer eyJhbGciOiJIUz
I1NiIsInR5cCI6ImF0K2p3dCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG
4gRG9lIiwiYWRtaW4iOnRydWUsImlhdCI6MTUxNjIzOTAyMn0.S9QVKeqooCU7nRahQsP
hnV7xOrkqsKB7YKQEReyaWT8

to=DE02100100109307118603& amount=123.50

A.1.3. LLM Tool receives the request

LLM tool tries to call the service with the External Service APIs with the Access Token received using the JWT Bearer Authentication scheme (5) and is issued an authentication challenge.

HTTP/1.1 403 Forbidden WWW-Authenticate: Bearer
error="new_authorization_needed", error_description="A new
authorization request is needed"

{ "decision": false, "context": { "error_msg": "The user must be
authorized to initiate payment for Merchant A ", "details": {
"method": "urn:ietf:params:oauth:grant-ext:rar",
"authorization_details": [{ "type": "payment_initiation", "actions":
["initiate", "status", "cancel"], "locations": [
"https://example.com/payments"], "instructedAmount": { "currency":
"EUR", "amount": "123.50" }, "creditorName": "Merchant A",
"creditorAccount": { "iban": "DE02100100109307118603" },
"remittanceInformationUnstructured": "Ref Number Merchant" }] } } }

Note: How agents discover available tools is out of scope of this specification and this example

LLM Agent fetches the external tool resource’s OAuth 2.0 Protected Resource Metadata per [RFC9728] to dynamically discover an authorization server that can issue an access token for the resource.

```
GET /.well-known/oauth-protected-resource
Host: api.saas.net
Accept: application/json

HTTP/1.1 200 Ok
Content-Type: application/json

{
  "resource": "https://api.saas.net/",
  "authorization_servers": [ "https://authorization-server.saas.net" ],
  "bearer_methods_supported": [
    "header",
    "body"
  ],
  "scopes_supported": [
    "agent.tools.read",
    "agent.tools.write"
  ],
  "resource_documentation": "https://idp.saas.net/tools/resource_documentation.html"
}
```

LLM Agent discovers the Authorization Server configuration per [RFC9728].

```
GET /.well-known/oauth-authorization-server
Host: authorization-server.saas.net
Accept: application/json

HTTP/1.1 200 Ok
Content-Type: application/json

{
  "issuer": "https://authorization-server.saas.net",
  "authorization_endpoint": "https://authorization-server.saas.net/oauth2/authorize",
  "token_endpoint": "https://authorization-server.saas.net/oauth2/token",
  "jwks_uri": "https://authorization-server.saas.net/oauth2/keys",
  "registration_endpoint": "authorization-server.saas.net/oauth2/register",
  "scopes_supported": [
    "agent.read", "agent.write"
  ],
  "response_types_supported": [
    "code"
  ],
  "grant_types_supported": [
    "authorization_code", "refresh_token"
  ]
}
```

LLM Agent has learned all necessary endpoints and supported capabilites to obtain an access token for the external tool.

A.1.4. LLM Tool obtains a set of token from Authorization Server protecting the API

The LLM tool redirects the LLM Agent for an authorization request:

```
GET /oauth2/authorize?response_type=code
  &client_id=eb1e27d2df8b7
  &state=af0ifjsldkj
  &redirect_uri=https%3A%2F%2Ftool.example.com%2Fcb
  &code_challenge_method=S256
  &code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bwc-uCHaoeK1t8U
  &authorization_details=%5B%7B%22type%22:%20%22payment_initiation
  %22,%22actions%22:%20%5B%22initiate%22,%22status%22,%22cancel%22
  %5D,%22locations%22:%20%5B%22https://example.com/payments%22%5D,
  %22instructedAmount%22:%20%7B%22currency%22:%20%22EUR%22,%22amount
  %22:%20%22123.50%22%7D,%22creditorName%22:%20%22Merchant%20A%22,
  %22creditorAccount%22:%20%7B%22iban%22:%20%22DE02100100109307118603
  %22%7D,%22remittanceInformationUnstructured%22:%20%22Ref%20Number
  %20Merchant%22%7D%5D
Host: authorization-server.saas.net
```

> We don’t describe the way the user is authenticated as it follows Rich Authorization Request [RFC9396]

The LLM Tool will receive an Authorization Code that it will be able to exchange for a set of JWTs issued by the Authorization Server protecting the API.

The LLM Tool can then make a new request to the External Service APIs (5). If it can meet the APIs Access Control requirement, the flow will follow with a response (6).

Acknowledgments

The authors wants to acknowledge the support and work of the following individuals: Grese Hyseni (Raiffeisen Bank International, grese.hyseni@rbinternational.com), Henrik Kroll (Raiffeisen Bank International, henrik.kroll@rbinternational.com).

The authors wants also to recognize the trail blazers and thought leaders that created the ecosystem without which this draft proposal would not be able to solve customer pain points and secure usage of digital services, especially without being limited to: Vittorio Bertocci†, Brian Campbell (Ping Identity), Justin Richer (MongoDB), Aaron Parecki (Okta), Pieter Kasselman (SPRL), Dr Mike Jones (Self-Issued Consulting, LLC), Dr Daniel Fett (Authlete).

Authors' Addresses

Jean-Franois Lombardo
AWS
Canada
Email: jeffsec@amazon.com

Alexandre Babeanu
IndyKite
Canada
Email: alex.babeanu@indykite.com

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

George Fletcher
Practical Identity
United States of America
Email: george@practicalidentity.com