# OIDC4AC: Extending OpenID Connect for Structured and Negotiable Authentication Contexts

**Brendon Vicente R. Silva**[1], **Frederico Schardong**[1], **Ricardo F. Custódio**[1]

[1]Laboratório de Segurança em Computação (LabSEC)
Universidade Federal de Santa Catarina (UFSC)

bredstone13@gmail.com, frederico.schardong@ufsc.br,
ricardo.custodio@ufsc.br

***Abstract.*** *OpenID Connect (OIDC) is a widely adopted authentication protocol on the web. Despite its popularity, it provides limited expressiveness for describing how users are authenticated, constraining its applicability in high-assurance, auditable scenarios. To understand the extent of this gap, a Multivocal Literature Review was conducted, revealing the absence of an integrated, OIDC-native model for representing factors and specifying authentication requirements. To address these gaps, this paper proposes OpenID Connect for Authentication Context (OIDC4AC), a new OIDC protocol extension that enables fine-grained specification and detailed representation of authentication factors, improving interoperability and auditability.*

## 1. Introduction

Digital identity management underpins how users are authenticated and how access is granted across platforms online. Technological advancements and shifting user expectations have changed how identities are managed over the decades. Simple credential exchanges have given way to sophisticated architectures that interconnect multiple services, devices, and agents into cohesive ecosystems [Ometov et al. 2018].

In this context, standards such as OAuth 2.0 [Hardt 2012] and OpenID Connect (OIDC) [Sakimura et al. 2014] have become foundational for secure authentication and authorization on the web. However, as the digital landscape evolves, these protocols face new challenges [Parecki et al. 2019]. Particularly, they provide limited control over how authentication methods[1] are represented and negotiated[2] between parties, constraining their applicability in scenarios that demand detailed authentication contexts, such as those requiring regulatory compliance, risk-informed decision making, or auditability.

OIDC conveys authentication and user information through *claims*, pieces of information embedded in tokens conveying attributes about the user or the authentication event, transmitted from the Identity Provider (IdP) to the application or service consuming the authentication response, known as the Relying Party (RP). Among the claims defined in OIDC Core[3], the *Authentication Method References* (`amr`) allows IdPs to indicate

[1]In this paper, the terms *authentication factor* and *authentication method* are used interchangeably.

[2]In this paper, *negotiation* refers to the protocol-level expression of authentication requirements and receipt of their acceptance or rejection, without iterative proposal rounds.

[3]This paper uses *OIDC* to refer to the OpenID family of specifications, whereas *OIDC Core* refers to the original OpenID Connect Core specification [Sakimura et al. 2014].

which authentication factors were employed. However, it consists of flat identifiers without a structured way to describe contextual information about the authentication event.

To illustrate, consider a user accessing an application. Upon reaching the login page, the user opts to authenticate via GitHub. Acting as an intermediary IdP, GitHub redirects the user to Google for identity verification, where they enter their password and confirm their identity via an SMS code. Once verified, the user is returned to GitHub, where they must complete an additional TOTP challenge before being finally granted access. Despite this multi-step journey involving two providers and three distinct factors, the token received by the platform contains only `amr: ["pwd", "sms", "otp"]`. These identifiers confirm that the methods were used but reveal nothing about which IdP validated each factor, under which trust framework, or in what order. The RP receives a minimal and ambiguous representation of the process, limiting its ability to enforce fine-grained access policies, demonstrate regulatory compliance, or conduct forensic audits.

OIDC also lacks a standardized mechanism for RPs to request or negotiate detailed authentication requirements with IdPs. At the protocol level, RPs cannot express preferences or constraints regarding authentication methods, nor define factor-specific requirements such as assurance thresholds or biometric liveness criteria [Sakimura et al. 2014]. This constrains the ability of RPs to enforce security policies, adapt to evolving threat landscapes, or meet regulatory mandates that require demonstrable assurance levels. As a result, organizations often resort to proprietary solutions, fragmenting the ecosystem and undermining the interoperability that OIDC aims to provide.

To investigate whether existing literature and specifications address these limitations, this paper conducts a *Multivocal Literature Review* (MLR) [Garousi et al. 2019], examining academic and technical sources related to authentication factor representation and negotiation in identity protocols. The MLR reveals that no existing solution provides a standardized, integrated model enabling both the representation and negotiation of authentication factors within OIDC or similar JSON-based ecosystems. This gap motivates the proposal of the *OpenID Connect for Authentication Context*[4] (OIDC4AC), a protocol extension that preserves full compatibility with existing OIDC deployments and supports regulatory alignment with frameworks such as the General Data Protection Regulation (GDPR) [European Commission 2016] and the electronic Identification, Authentication and Trust Services (eIDAS) Regulation [European Commission 2014].

Section 2 presents the background; Section 3 defines the problem space; Section 4 describes the MLR; Section 5 presents OIDC4AC; Section 6 discusses interoperability and compatibility; Section 7 presents its implementation and experimental evaluation; Section 8 examines security, privacy, and regulatory alignment; Section 9 compares OIDC4AC against protocols identified in the MLR; and Section 10 concludes the paper.

## 2. Background

### 2.1. OAuth 2.0

*OAuth 2.0* is an authorization framework that allows third-party applications to access users' resources without requiring the disclosure of credentials [Hardt 2012]. It defines standardized message exchanges and token issuance between four entities: (i) *resource*

[4]The full OIDC4AC specification can be found at this location.

*owner*, the end-user who holds protected data and wishes to share it; (ii) *client*, an application requesting access to that data; (iii) *resource server*, which hosts and provides access to the protected data; and (iv) *authorization server*, the IdP, which authenticates the resource owner, obtains their consent, and issues tokens to the client.

OAuth 2.0 defines authorization grant types through which clients can obtain access tokens: opaque strings encapsulating attributes such as validity period and scope, thereby enabling fine-grained authorization control. Among them, the *Authorization Code Grant* is recommended for server-side clients capable of securely storing credentials, as it minimizes token exposure by keeping them out of the user agent [Lodderstedt et al. 2025]. In this flow, illustrated in Figure 1, the client first redirects the user to the authorization server for authentication and consent. Upon success, the server issues a short-lived authorization code transmitted back to the client through the user's browser. This code carries no attributes, user data, or usable privileges on its own. The client then exchanges it directly with the IdP over a direct Hypertext Transfer Protocol Secure (HTTPS) [Rescorla 2000] connection to obtain an access token. In this process, the client must authenticate itself to the IdP, ensuring that only legitimate and previously registered clients can obtain tokens.

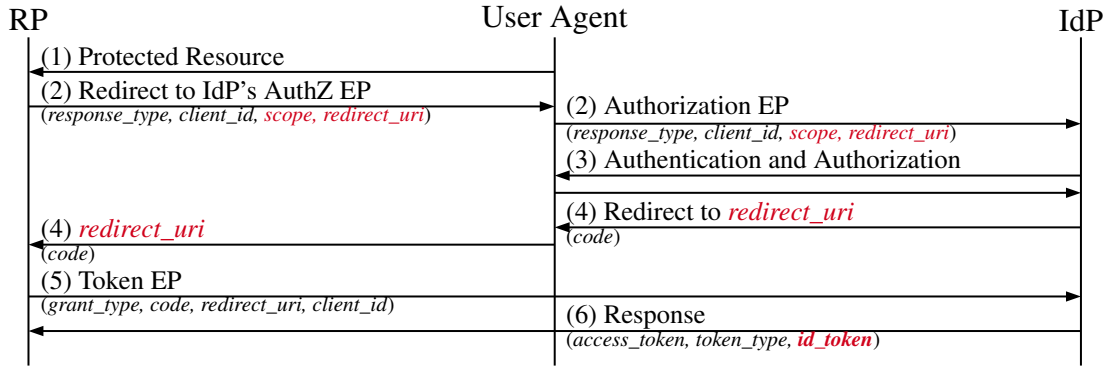


**Figure 1. Authorization code grant. Mandatory parameters are shown in black; OIDC additions are shown in red. The bold `id_token` is a new, mandatory OIDC value. Source: adapted from [Schardong et al. 2022].**

## 2.2. OpenID Connect

While OAuth 2.0 provides a robust foundation for authorization and access delegation, it does not address identity representation [Hardt 2012]. To fill this gap, *OpenID Connect 1.0* was introduced as an identity layer on top of OAuth 2.0, enabling RPs to verify end-user identity and obtain profile information in a standardized way [Sakimura et al. 2014].

As illustrated in Figure 1, OIDC preserves OAuth 2.0's entities and message flows while adding new mandatory parameters and artifacts. Among its contributions is the *ID Token*, a digitally signed JSON Web Token (JWT) [Jones et al. 2015] composed of *claims*: JavaScript Object Notation (JSON) [Bray 2017] based statements about the authentication event and, optionally, about the end-user. To provide context regarding the authentication event, OIDC defines claims that superficially describe how the user was authenticated. Two primary mechanisms for this purpose are the *Authentication Method Reference* (`amr`) and the *Authentication Context Class Reference* (`acr`).

The `amr` claim indicates which authentication factors were used during the authentication as a list of string identifiers (*e.g.*, `["pwd", "otp"]`), each representing an employed method. Although OIDC does not prescribe a vocabulary for these strings, it recommends adherence to the Authentication Method Reference Values registry defined in RFC 8176 [Jones et al. 2017]. This registry standardizes a set of short identifiers, such as `sms` for SMS-based verification, and `fpt` for fingerprint-based authentication. Since conformance to this specification is not mandatory, implementations may extend or interpret `amr` values in provider-specific ways, often relying on proprietary conventions.

The `acr` claim conveys the overall class of the authentication event. Unlike `amr`, which enumerates individual methods, `acr` provides a higher-level abstraction of the authentication's context. Typically, different `acr` values correspond to different authentication journeys, each with its own requirements. For instance, a `urn:example:low` value might indicate password-based authentication, while `urn:example:high` could signify multi-factor authentication involving biometrics and hardware tokens. Implementers often adopt standardized frameworks for `acr` values to align their implementation with regulatory requirements and industry best practices. OIDC does not enforce a specific vocabulary for `acr` values. RPs may request specific `acr` values during authentication, thereby expressing the minimum acceptable authentication context for a given request.

## 3. Problem Statement

Although OIDC provides a standardized foundation for identity exchange, it exhibits *semantic* gaps in representing and negotiating authentication factors. These gaps manifest across two complementary dimensions: (i) the inability of applications to express detailed authentication requirements; and (ii) the inability of IdPs to report structured, contextually rich information about how authentication was performed.

Beyond requesting high-level context classes through the `acr` claim, RPs have limited influence over the authentication process. They cannot specify which methods, policies, or contextual parameters the IdP should apply, and must rely on the IdP's internal policies and retrospective report rather than expressing what is required based on its own risk assessments. This limitation becomes more relevant when RPs need to enforce specific strategies, such as multi-factor or step-up authentication for sensitive transactions. There is no provision for negotiation on *which* factors should be used, *how* they should be combined, or *under what conditions* they should be applied.

The mechanism for reporting authentication events is similarly limited: the `amr` claim provides a minimal way for indicating which authentication factors were employed. However, it lacks the capacity to convey *how* each factor was executed, *by whom*, or *under which circumstances*. For example, in scenarios where authentication is delegated across multiple IdPs, it cannot indicate which one performed the verification, how assurance was evaluated, or whether the process followed a trust framework. As a result, RPs receive a generic representation that omits contextual data, undermining confidence and reliability.

To illustrate these limitations, consider a digital signature platform issuing qualified certificates under eIDAS [European Commission 2014]. The application must ensure authentication using facial biometrics processed by a certified algorithm and an OTP delivered via SMS [M'Raihi et al. 2011]. Under OIDC, the RP cannot express these requirements in its request, nor verify from the returned token whether the IdP adhered to those

methods or policies. The RP receives only `amr`: `["face", "otp"]`, which confirms that these methods were used but lacks information about which algorithm was applied, which OTP delivery mechanism was used, and which policy governed the authentication.

These limitations have significant implications: (i) different IdPs may interpret `amr` values inconsistently, use proprietary identifiers, or omit data, preventing cross-domain comparability; (ii) the absence of structured mechanisms for expressing transparent and traceable authentication may impede alignment with regulatory frameworks such as eIDAS [European Commission 2014] and NIST SP 800-63-3 [Temoshok et al. 2025], which emphasize these principles; (iii) without provenance, timestamps, and trust frameworks, RPs cannot formally request combinations of factors or express detailed constraints on how they should be executed, which may limit specific use cases; and (iv) to compensate for these gaps, developers often create *ad hoc* solutions, such as custom claims or proprietary endpoints, leading to fragmentation and reduced interoperability.

## 4. Related Work

To validate the problem stated and assess the extent to which existing literature addresses it, we conducted a *Multivocal Literature Review* [Garousi et al. 2019]. Unlike *Systematic Literature Reviews* (SLRs) [Petersen et al. 2015], which restrict their scope to peer-reviewed academic publications, an MLR explicitly incorporates *Gray Literature* (GL), including technical specifications, community drafts, and industry white papers. This broader scope is particularly suitable for identity protocols, where influential advances often originate from standardization bodies rather than from academic venues.

### 4.1. Planning

The planning comprised five steps: (i) defining the objectives and Research Questions (RQs); (ii) formulating a search strategy; (iii) selecting sources consistent with the study's scope; (iv) applying Inclusion and Exclusion Criteria (ICs and ECs) to filter retrieved material; and (v) conducting backward and forward snowballing through citation analysis.

The *objective* of the MLR was fourfold: (i) to analyze how authentication factors are represented within existing web-based Single Sign-On (SSO) and federated identity protocols; (ii) to identify semantic and structural limitations in these representations; (iii) to explore mechanisms through which RPs can request or negotiate authentication factors; and (iv) to examine identity-related standards and extensions that can inform the design of a richer, interoperable model for representing and requesting authentication context.

Following the Goal-Question-Metric (GQM) paradigm [Basili et al. 1994] and based on these objectives, the *Research Questions* were formulated: (RQ 1) How do web-based SSO or federated identity protocols represent authentication factors? (RQ 2) How can authentication factors be requested or negotiated in protocols such as OIDC? (RQ 3) What semantic limitations exist in current mechanisms for representing authentication factors (*e.g.*, opacity, lack of granularity, ambiguity)? and (RQ 4) What standards and data structures are used in adjacent OIDC extensions to represent authentication factors?

The search strategy was differentiated between academic and technical sources to accommodate the distinct nature of contributions in each domain. For academic work, a structured string was developed based on the Population, Intervention, Comparison, Outcome, Context (PICOC) framework [Kitchenham and Charters 2007] and iteratively

refined to balance precision and recall, ensuring that only relevant studies were included. The following *search string* was applied to *title*, *abstract*, and *keywords* of publications indexed in Scopus Preview, IEEE Xplore, Web of Science, ACM Digital Library, Wiley Online Library, and Taylor & Francis Online.

```
("OAuth" OR "OpenID Connect" OR "OIDC" OR "SAML") AND ("authentication
   context" OR "authentication method" OR "authentication factor" OR
      "authentication detail" OR "ACR" OR "AMR") AND ("claim" OR
 "representation" OR "semantic" OR "data" OR "context" OR "assurance")
AND ("profile" OR "trust framework" OR "extension" OR "specification")
```

Given the role of technical specifications in the identity domain, the GL search string was designed to maximize recall and avoid excluding relevant works. The query was executed in the IETF Datatracker and RFC Editor over drafts and documents related to authentication protocols:

```
("authentication method" OR "authentication factor")
```

Additionally, active drafts and final specifications for OIDC and Security Assertion Markup Language (SAML) were retrieved from their repositories. In both cases, no predefined query string was applied to exhaustively examine each specification for mechanisms pertinent to authentication-factor representation and directly relevant to RQ 4.

All retrieved records were consolidated, deduplicated, and screened. Materials were included if they satisfied at least one IC and excluded if they matched any EC. The inclusion criteria were: (IC 1) the study explicitly analyzes mechanisms for representing authentication factors in web-based SSO or federated identity protocols; (IC 2) it describes mechanisms for requesting or negotiating authentication factors, including logical relationships or parameter-level details; (IC 3) it identifies semantic or structural limitations in how authentication methods are represented or requested; and (IC 4) it presents or extends standards addressing factor representation or negotiation. The exclusion criteria were: (EC 1) the study does not discuss authentication factor representation or negotiation; (EC 2) it does not detail how specific factors or parameters are represented or requested; (EC 3) it is limited to performance, usability, or implementation aspects without conceptual contributions; (EC 4) it addresses network-level or device-level authentication unrelated to web-based SSO or federated identity; and (EC 5) its contribution is already covered by another included work in greater depth.

### 4.2. Execution

The *search phase* was conducted on September 20, 2025, across all WL databases and GL repositories, yielding 15 papers and 105 technical specifications. The *screening* and *data extraction* stages applied the ICs and ECs to these records. Each record was first evaluated by title and abstract, and those passing this initial screening were analyzed in full. Backward and forward snowballing were conducted using Google Scholar, and newly retrieved works were subjected to the same eligibility assessment.

Data extraction was carried out using a *Data Extraction Form*[5] (DEF), ensuring consistency and reproducibility across all analyzed materials. The overall workflow of the review is summarized in Figure 2, which illustrates the number of records at each stage.

[5]The complete DEF and the resulting table listing all analyzed studies are available at this webpage.

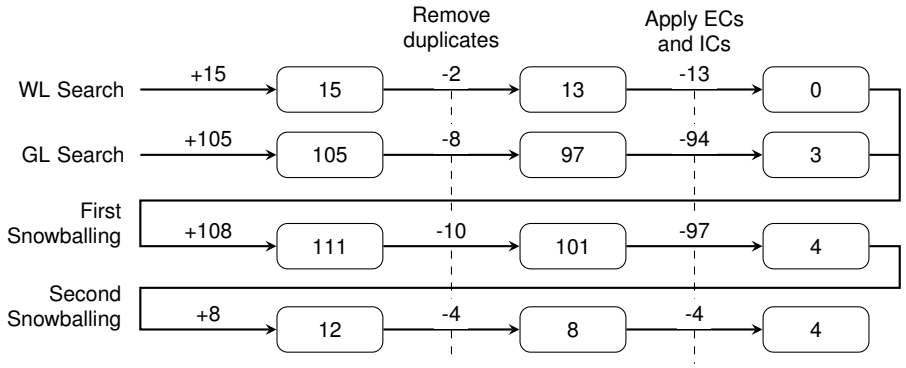


**Figure 2. Number of records in each stage of the MLR.**

### Threats to Validity

This study is subject to threats to validity that may affect the completeness and accuracy of its findings [Petersen et al. 2015]. First, some studies may have been missed due to terminology inconsistencies or incomplete indexing; this risk was mitigated through backward and forward snowballing, allowing the identification of additional sources beyond the initial results. Second, document classification involved researchers' judgment, especially when assessing whether a specification addressed representation or negotiation of authentication factors; to minimize potential inaccuracies, a DEF was designed to ensure consistency across analyzed materials. Finally, the concepts of representation and negotiation of authentication factors were operationalized according to Section 3, and interpretation differences between specifications may influence comparability.

## 4.3. MLR Results

After applying the procedure described in Section 4.2, four specifications were retained for detailed analysis. *RFC 8176* [Jones et al. 2017] defines the Internet Assigned Numbers Authority (IANA) registry for `amr` values, providing a vocabulary of short identifiers for authentication methods used in OIDC tokens. *OpenID Connect User Questioning API 1.0* [Aillery et al. 2020] extends OIDC with an interaction model for RPs to send questions to authenticated users and verify their answers. It introduces two claims, `used_amr` and `wished_amr`, that denote the authentication methods actually used and those preferred or required by the RP. *Authentication Context for SAML 2.0* [Kemp et al. 2005] defines eXtensible Markup Language (XML) [Bray et al. 2008] schemas for representing authentication context classes and declarations. Classes identify predefined authentication contexts, while declarations can encode structured details about authentication methods, protection mechanisms, transport protocols, and governing policies. SAML's `RequestedAuthnContext` mechanism [Cantor et al. 2005] allows RPs to refer to such predefined contexts. The RAC extension [Madsen et al. 2007] introduces the `RequestedACCombination` element and the `RACComparison` attribute to express logical relationships among requested contexts.

### RQ 1 – Representation of Authentication Factors

In OIDC, authentication factors are represented using flat, enumerative structures describing only *which* methods were used. Although [Jones et al. 2017] promotes syntactic in-

teroperability by standardizing identifiers used in `amr`, it provides no structure for contextual or parameter-level details. [Aillery et al. 2020] introduces the claims `used_amr` and `wished_amr`, which extend OIDC's capability to represent authentication methods, but still rely on the same model defined in RFC 8176, inheriting its limitations.

In contrast, [Kemp et al. 2005] describes a schema-driven approach that enables richer semantic expression. SAML assertions can include data such as credential usage and governing policy, allowing IdPs to describe the factors used and the context in which they were executed. Nonetheless, this approach remains tied to XML schemas.

**RQ 2 – Request and Negotiation of Authentication Factors**

In OIDC, [Aillery et al. 2020] partially addresses factor negotiation by introducing the `wished_amr` claim, enabling RPs to express a list of desired authentication methods. The IdP may then select one or more of these methods to authenticate the user before allowing them to answer RP-submitted questions. Although this claim introduces a form of negotiation, its semantics are limited to ordered preference. It does not define logical operators (*e.g.*, `AND`/`OR`) or composite structures, nor does it support parameter-level requests, such as specifying an OTP delivery channel, biometric algorithm, or device constraints.

SAML's `RequestedAuthnContext` mechanism [Cantor et al. 2005] allows RPs to request a predefined context class or declaration, similar to requesting an `acr` value in OIDC. It does not provide a generic, inline syntax for selecting or constraining individual attributes within that context. RAC [Madsen et al. 2007] adds logical combinations and comparisons of requested context-class references, but does not change this model.

**RQ 3 – Semantic Limitations**

Among OIDC specifications, both share a fundamental limitation: lack of contextual depth. While [Aillery et al. 2020] avoids explicit discussion of representational or interoperability limitations, [Jones et al. 2017] explicitly states that its registry is "intentionally small and non-exhaustive", and acknowledges that identical `amr` values may correspond to different implementations, such as the `otp` identifier, which could refer to either TOTP or HOTP authentication [M'Raihi et al. 2011, M'Raihi et al. 2005]. Although the RFC does not explicitly discuss interoperability risks, its coarse-grained vocabulary leads to divergent interpretations, producing only superficial equivalence across IdPs.

In SAML, [Kemp et al. 2005] warns that interpretation of context classes and declarations depends on a common understanding between parties, implicitly recognizing that, without shared profiles or registries, semantic misalignment can occur. [Madsen et al. 2007] further acknowledges limitations in SAML, pointing out that it lacks the ability to express combinations of authentication contexts without creating new URIs for every possible permutation. Based on this, it introduces mechanisms for logical combination of contexts, but does not define generic parameter-level request semantics.

**RQ 4 – Standards and Data Structures**

[Jones et al. 2017] and [Aillery et al. 2020] both operate within the OIDC framework, which is inherently JSON-based. They use JSON arrays of strings to represent authentication methods, prioritizing lightweight interoperability and ease of implementation. [Kemp et al. 2005] and [Madsen et al. 2007], on the other hand, are grounded in SAML,

relying on XML for data representation. They use XML schemas to define structured context classes and declarations, enabling richer semantic expression but at the cost of increased complexity and reduced compatibility with JSON-centric ecosystems.

### 4.4. MLR Synthesis

The MLR reveals a persistent tension across existing specifications. OIDC-native mechanisms are lightweight and JSON-compatible, but represent authentication methods as flat string identifiers, offering limited semantic expressiveness and no standardized support for fine-grained method-level requests. SAML-based mechanisms provide richer context representations and logical composition, yet their request model relies on references to predefined classes or declarations, with semantics defined by XML schemas and shared profiles. Neither approach combines structured, factor-level requests with protocol-native JSON compatibility in the way required by high-assurance and auditable deployments.

## 5. Proposal: OpenID Connect for Authentication Context

To address the identified gaps, we propose the *OpenID Connect for Authentication Context*, a protocol extension that introduces a unified model for structured representation and negotiation of authentication methods. It enables RPs to request specific factors, logical combinations thereof, and method-level properties in a machine-readable form, while allowing IdPs to report structured, contextualized data about each method employed.

Three principles guide the design of OIDC4AC: (i) *compatibility* ensures seamless integration with existing OIDC deployments by preserving core protocol mechanisms; (ii) *semantic clarity* addresses the ambiguity of flat identifiers through a well-defined vocabulary for method metadata and properties; and (iii) *progressive extensibility* allows IdPs, trust frameworks, and future specifications to define additional fields as needed, ensuring adaptability to diverse assurance models and evolving authentication technologies.

### 5.1. Structured Representation of Authentication Factors

In OIDC4AC, authentication factors are represented using *AMR Details Objects*, each corresponding to one authentication method employed. A complete authentication event is represented as a collection of such descriptors conveyed through a claim named `amr_details`. This representation allows RPs to perform deterministic evaluation of authentication events, apply method-level requirements, and interpret contextual attributes relevant to assurance, compliance, and policy evaluation. The protocol explicitly separates *which* methods were used, conveyed by the existing `amr` claim, from *how* they were executed, conveyed by `amr_details`. As the `amr` claim remains unchanged, existing RPs and IdPs can continue to operate without modification, while more advanced implementations can leverage the extended semantics provided by the specification.

Each AMR Details Object is composed of three components: (i) a *Method Identifier*; (ii) *Authentication Method Metadata*, which provides method-independent contextual information about the authentication event (*e.g.*, timestamp, location, and device information); and optionally, (iii) *Authentication Method Properties*, which contains method-specific attributes, such as hashing algorithms for passwords or quality metrics for biometrics, that detail the execution of the authentication method.

### AMR Details Object Schema

The `amr_details` claim is a JSON array of AMR Details Objects, each describing one successful execution of an authentication method. The Method Identifier is represented by the `amr_identifier` member, which must match one of the values in the `amr` claim. This ensures each factor can be unambiguously detailed while preserving `amr` for backward compatibility. Each AMR Details Object includes Authentication Method Metadata through the `amr_metadata` member, a JSON object that contains at least the timestamp of the successful execution. Additional method-independent contextual information supports auditability and provenance, allowing RPs to assess which method was used and under what conditions; its disclosure depends on IdP policies and the applied trust framework. Authentication Method Properties are conveyed through `amr_properties`, a JSON object whose structure is specific to the method identified by `amr_identifier`. Properties for OTP methods might describe the delivery channel or code characteristics, while biometric properties might describe matching confidence or liveness detection.

Implementations may define additional `amr_metadata` and `amr_properties` fields if their semantics are documented and non-registered names follow a collision-resistant naming convention. The complete specification[4] complements the presented structure with normative JSON Schemas, metadata registries, and method-specific vocabularies. Figure 3 illustrates a password authentication, showing how `amr_details` conveys contextual information such as the trust framework and governing policy.

**Figure 3. Password authentication through the `amr_details` claim.**

```
{
  "amr": ["pwd"],
  "amr_details": [{
    "amr_identifier": "pwd",
    "amr_metadata": { "iss": "https://idp.com", "time": "2025-10-14T12:03:25Z" },
    "amr_properties": { "pwd_created_at": "2025-09-12T08:41:17Z",
      "pwd_policy_id": "pwd-policy" }
  }]
}
```

## 5.2. Requesting Authentication Factors

OIDC4AC expresses standardized and semantically rich authentication requirements through the AMR Details Object structure, included in authentication requests using OIDC's existing `claims` parameter. This allows RPs to declaratively specify not only which authentication factors are desired, but also the conditions under which they should be accepted. By building on OIDC's native extensibility mechanisms, this approach remains compatible with existing implementations, since RPs or IdPs that do not support OIDC4AC can simply ignore the additional request semantics.

In requests, an AMR Details Object does not represent the outcome of an authentication event, but rather a *requirement expression* describing the authentication factors that the RP wishes to be employed. While the representation of authentication methods in responses is defined as a JSON array, OIDC4AC intentionally defines the `amr_details` claim in requests as a JSON object. This object acts as a structured filter or logical template that the IdP evaluates and attempts to satisfy when processing the request. This de-

sign choice is necessary to support the expression of logical combinations and constraints over authentication factors, which cannot be naturally represented using a flat list.

Within this request object, the RP can refer to authentication factors, metadata, and properties using the same identifiers and vocabulary defined for reporting authentication events, ensuring semantic continuity between requests and responses. OIDC4AC also inherits OIDC's established semantics for requesting claims, including the use of the `value` and `essential` parameters to indicate required values and mandatory requirements, respectively. In addition, it introduces a set of operators and constraints that allow RPs to express more precise authentication policies, including:

**`one_of` and `all_of`** logical operators that allow the RP to define relationships among requirements. The `one_of` operator indicates that at least one of the specified claims must be satisfied, while `all_of` requires all to be fulfilled, enabling the expression of policies based on logical *OR* and *AND* combinations.

**`min` and `max`** quantitative constraints applicable to numeric attributes, such as OTP length, iteration counts, or assurance-related scores.

**`max_age`** a temporal constraint indicating how recent a time-based attribute must be, for example requiring a factor to have been used within a given time window.

By incorporating these constructs into the `amr_details`, RPs can articulate complex authentication requirements such as those illustrated in Figure 4. In this example, the RP requests facial biometric authentication with a minimum quality score, combined with either an OTP or a password. If OTP is selected, the RP also requests additional metadata about the algorithm used to generate the code and the OTP length. As in the OIDC specification, `null` values indicate that no specific constraint is imposed.

**Figure 4. Requesting authentication factors using the `amr_details` claim.**

```
{
  "claims": {
    "id_token": {
      "amr_details": {
        "all_of": [
          { "amr_identifier": { "value": "face" },
            "amr_properties": {
              "face_match_score": { "min": 0.8 },
              "face_image_quality": { "min": 0.7 } } },
          { "one_of": [
            { "amr_identifier": { "value": "pwd" } },
            { "amr_identifier": { "value": "otp" },
              "amr_properties": { "otp_length": null,
                "otp_algorithm": null } }
          ] }
        ]
      }
    }
  }
}
```

When processing such requests, an IdP must attempt to satisfy the requested conditions to the greatest possible extent. The IdP may provide more information than requested if such data is mandatory under its own policy, or less information if some parameters are optional or unsupported. If a requested combination of factors cannot be

fulfilled and the request is not marked as `essential`, the IdP may proceed with an alternative authentication method and return the corresponding `amr_details` claim. If the request is marked as `essential` and cannot be satisfied, an error indicating that the requirements could not be met must be returned.

## 6. Interoperability and Compatibility

OIDC4AC preserves interoperability with existing OIDC deployments while introducing a structured mechanism for representing and requesting authentication factors. The `amr_details` claim is a non-disruptive, optional addition that does not alter OIDC Core's normative behavior, supporting backward compatibility and incremental adoption.

To allow RPs to dynamically determine whether an IdP supports OIDC4AC and which authentication-context capabilities it offers, the extension leverages OIDC Discovery by defining additional provider metadata parameters [Sakimura et al. 2023]. This mechanism exposes a standardized `/.well-known/openid-configuration` document through which an IdP publishes its endpoints and supported features. By consulting this metadata before initiating authentication, an RP can determine whether the provider can return `amr_details`, process authentication-method requirements, and support the methods and properties relevant to its request. This avoids prior bilateral configuration and prevents the RP from relying on unsupported `essential` requirements.

These capabilities are represented through Discovery metadata. The presence of `amr_details` in `claims_supported` indicates that the provider may return the claim in an ID Token or UserInfo response, but not that it processes authentication-method requirements. That capability is indicated by `amr_details_request_supported`. Additional parameters describe supported methods, properties and values, trust frameworks, assurance levels, and location attributes, allowing RPs to tailor their requests.

## 7. Implementation and Experimental Evaluation

To assess the practical feasibility of OIDC4AC, the extension was implemented as an experimental feature in Keycloak[6], an open-source Identity and Access Management platform that supports OIDC. The implementation integrates OIDC4AC with the authorization flow and supports the delivery of `amr_details` in ID Token and UserInfo responses. A separate interactive laboratory acts as an RP and enables the main protocol flows to be exercised while requests and returned tokens can be inspected[7].

Unit, integration and end-to-end tests cover the implementation. The combined suites follow the main protocol path, from Discovery and authentication-request processing to claim delivery through ID Tokens and UserInfo, under different combinations of authentication requirements, properties, and freshness conditions. They also assess session-related behavior, error handling, and operation with the extension disabled. Source code, test cases, and documentation are publicly available[8].

## 8. Security, Privacy and Regulatory Alignment

OIDC4AC provides a mechanism for conveying structured authentication context information, but it does not define an assurance, authorization, or regulatory framework. Legal

[6]`https://www.keycloak.org/`
[7]The interactive laboratory can be found at this location.
[8]Full Keycloak implementation can be found at this location.

requirements, liability models, trust frameworks, operational policies, consent mechanisms, and user interface behavior remain outside its scope and must be addressed by deployments, federation agreements, or applicable regulation. The protocol's role is limited to making authentication information available in a structured and interoperable form.

This information can support alignment with frameworks that emphasize accountability, traceability, and proportional security. Under GDPR, for instance, organizations must demonstrate that appropriate measures are in place [European Commission 2016]; similarly, eIDAS and NIST SP 800-63 emphasize assurance and interoperability as foundational requirements [European Commission 2014, Temoshok et al. 2025]. By conveying authentication metadata in a structured form, `amr_details` provides input for RPs and auditors assessing how and under which conditions authentication was performed.

As an OIDC claim, `amr_details` inherits the integrity and issuer-authentication guarantees of a properly validated ID Token. The token signature allows the RP to verify that the claim was issued by the IdP and was not altered after issuance, but it does not independently verify that the reported authentication event occurred as described. The evidentiary value of `amr_details` therefore depends on the RP's trust in the IdP, including its authentication processes under the applicable trust framework.

`amr_details` describes a past authentication event and should not be treated as reusable proof of a new session. RPs should evaluate the freshness of the reported event using its timestamp and any applicable `max_age` requirement, in addition to the usual validation of the ID Token. Its richer metadata also concentrates sensitive information in the ID Token and may enable fingerprinting or correlation across RPs when fine-grained values are disclosed repeatedly. Biometric properties may reveal information about biometric processing, even when raw biometric data is not transmitted. Their disclosure should be justified by a specific purpose and constrained by the applicable legal and trust framework. Deployments should follow data-minimization principles: IdPs should disclose only fields required for the RP's stated purpose, and RPs should request only what is necessary to evaluate their policy. Trust frameworks and policies may further restrict which fields can be requested, who may receive them, and how long they may be retained.

## 9. Comparative Analysis with Existing Standards

To evaluate the contribution of OIDC4AC, Table 1 compares it with the specifications selected in the MLR. The comparison is qualitative and based on the analytical dimensions used throughout the review: token format adaptability, authentication request expressiveness, contextual metadata coverage, extensibility, regulatory fit, and OIDC compatibility. Each dimension was classified according to the degree to which the analyzed work supports the corresponding capability.[9] *Very Low* indicates that the capability is absent or only indirectly supported; *Low* indicates limited support through flat identifiers, predefined classes, or implementation-specific agreements; *Medium* indicates partial support with structured representation or limited composition mechanisms; and *High* indicates explicit, protocol-level support with structured semantics and extensibility.

The comparison shows that the selected specifications address complementary parts of the problem, but do not combine them in a single OIDC-native model. Existing

[9]The ratings are derived from the evidence synthesized in RQ 1–RQ 4 (Section 4.3).

**Table 1. Comparison of OIDC4AC and related standards.**
**Table legend: VL = Very Low, L = Low, M = Medium, H = High.**

| Work | Token Format | Request Semantics | Context Detail | Extension Support | Regulatory Alignment | OIDC Native |
|---|---|---|---|---|---|---|
| [Jones et al. 2017] | JSON | VL | VL | VL | ✗ | ✓ |
| [Aillery et al. 2020] | JSON | L | L | VL | ✗ | ✓ |
| [Kemp et al. 2005] | XML | L | M | H | M | ✗ |
| [Madsen et al. 2007] | XML | M | M | H | M | ✗ |
| **OIDC4AC** | JSON | H | H | H | ✓ | ✓ |

OIDC mechanisms preserve JSON compatibility and simplicity, but provide limited structure for method-level context and negotiation. SAML-based mechanisms provide richer semantics and composition mechanisms, but remain less aligned with JSON-based OIDC deployments. OIDC4AC is positioned between these approaches by combining structured representation, method-level negotiation, and compatibility with existing OIDC flows.

## 10. Final Remarks

This paper identified a semantic gap in current identity protocols regarding the structured representation and negotiation of authentication methods. The MLR showed that OIDC-based mechanisms preserve lightweight JSON compatibility but provide limited expressiveness for method-level context, while SAML-based approaches offer richer structures at the cost of XML-bound schemas and limited alignment with modern OIDC deployments. To address this gap, we proposed OpenID Connect for Authentication Context, an extension that introduces `amr_details` as a mechanism for representing and requesting authentication context in OIDC. In responses, `amr_details` allows IdPs to report method-level metadata and properties about the authentication methods used. In requests, it enables RPs to express requirements through OIDC's existing `claims` parameter, including logical combinations and property-level constraints.

An experimental implementation in Keycloak and a separate interactive laboratory provide an initial empirical evaluation of OIDC4AC. The implementation and test suites provide evidence of the proposal's feasibility and functional behavior, but do not constitute formal verification, a security proof, a performance benchmark, or an interoperability assessment with independent implementations. Future work should address these limitations and further investigate brokered authentication flows and policy enforcement in high-assurance scenarios. OIDC4AC has also been presented to the OIDC Working Group as a candidate for further standardization, and community discussion and potential adoption by the foundation may inform future implementation decisions.

## Generative AI Usage Disclosure

ChatGPT[10] and Grammarly[11] were used for language and style editing. All content remains the authors' responsibility; no AI tool produced technical or conceptual content.

[10] `https://chatgpt.com/`
[11] `https://www.grammarly.com`

## References

Aillery, N., Marais, C., and Manger, J. (2020). OpenID Connect User Questioning API 1.0. Standard, OpenID Foundation. https://openid.net/specs/openid-connect-user-questioning-api-1_0.html. Accessed: 11 May 2026.

Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). The Goal Question Metric Approach. In Marciniak, J. J., editor, *Encyclopedia of Software Engineering*, volume 1, pages 528–532. John Wiley & Sons. ISBN: 1-54004-8.

Bray, T. (2017). The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259. DOI: 10.17487/RFC8259.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation. https://www.w3.org/TR/xml/. Accessed: 11 May 2026.

Cantor, S., Kemp, J., Philpott, R., and Maler, E. (2005). Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, OASIS.

European Commission (2014). Regulation (EU) No 910/2014 on electronic identification and trust services for electronic transactions in the internal market (eidas regulation). Official Journal of the European Union (OJ L 257, 28.8.2014, pp. 73–114). https://eur-lex.europa.eu/eli/reg/2014/910/oj. Accessed: 11 May 2026.

European Commission (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj. Accessed: 11 May 2026.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, 106:101–121. DOI: 10.1016/j.infsof.2018.09.006.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749. DOI: 10.17487/RFC6749.

Jones, M. B., Bradley, J., and Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519. DOI: 10.17487/RFC7519.

Jones, M. B., Hunt, P., and Nadalin, A. (2017). Authentication Method Reference Values. RFC 8176. DOI: 10.17487/RFC8176.

Kemp, J., Cantor, S., Mishra, P., Philpott, R., Maler, E., Cahill, C., Hughes, J., Lockhart, H., Beach, M., Metz, R., et al. (2005). Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, OASIS Security Services (SAML) Technical Committee. https://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf. Accessed: 11 May 2026.

Kitchenham, B. A. and Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report EBSE-2007-01, Keele University and Durham University. Version 2.3.

Lodderstedt, T., Bradley, J., Labunets, A., and Fett, D. (2025). Best Current Practice for OAuth 2.0 Security. RFC 9700. DOI: 10.17487/RFC9700.

Madsen, P., Patel, A., Lockhart, H., and Mishra, P. (2007). SAML 2.0 Protocol Extension for Requested Authentication Context. Technical report, OASIS Security Services (SAML) Technical Committee. https://docs.oasis-open.org/security/saml/SpecDrafts-Post2.0/sstc-saml-protocol-ext-rac-cs-01.html. Accessed: 11 May 2026.

M'Raihi, D., Hoornaert, F., Naccache, D., Bellare, M., and Ranen, O. (2005). HOTP: An HMAC-Based One-Time Password Algorithm. RFC 4226. DOI: 10.17487/RFC4226.

M'Raihi, D., Rydell, J., Pei, M., and Machani, S. (2011). TOTP: Time-Based One-Time Password Algorithm. RFC 6238. DOI: 10.17487/RFC6238.

Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., and Koucheryavy, Y. (2018). Multi-factor authentication: A survey. *Cryptography*, 2(1). DOI: 10.3390/cryptography2010001.

Parecki, A., Hardt, D., and Lodderstedt, T. (2019). OAuth 2.1. Slides, IETF 106, Singapore. https://www.ietf.org/proceedings/106/slides/slides-106-oauth-sessa-oauth-21-00.pdf. Accessed: 11 May 2026.

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18. DOI: 10.1016/j.infsof.2015.03.007.

Rescorla, E. (2000). HTTP Over TLS. RFC 2818. DOI: 10.17487/RFC2818.

Sakimura, N., Bradley, J., Jones, M., De Medeiros, B., and Mortimore, C. (2014). OpenID Connect Core 1.0 incorporating errata set 1. *The OpenID Foundation, specification*, 335. https://openid.net/specs/openid-connect-core-1_0.html. Accessed: 11 May 2026.

Sakimura, N., Bradley, J., Jones, M., and Jay, E. (2023). OpenID Connect Discovery 1.0 incorporating errata set 2. https://openid.net/specs/openid-connect-discovery-1_0.html. Accessed: 16 October 2025.

Schardong, F., Giron, A. A., Müller, F. L., and Custódio, R. (2022). Post-quantum electronic identity: Adapting openid connect and oauth 2.0 to the post-quantum era. In Beresford, A. R., Patra, A., and Bellini, E., editors, *Cryptology and Network Security*, pages 371–390, Cham. Springer International Publishing. DOI: 10.1007/978-3-031-20974-1_20.

Temoshok, D., Proud-Madruga, D., Choong, Y.-Y., Galluzzo, R., Gupta, S., LaSalle, C., Lefkovitz, N., and Regenscheid, A. (2025). Digital Identity Guidelines. Technical Report NIST Special Publication (SP) 800-63-4, National Institute of Standards and Technology, Gaithersburg, MD. DOI: 10.6028/NIST.SP.800-63-4.