Network Working Group
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

A. Agarwal
Skyfire
M. Jones
Self-Issued Consulting

2 March 2026

KYAPay Profile
draft-skyfire-kyapayprofile-00

Abstract

This document defines a profile for agent identity and payment tokens in JSON web token (JWT) format. Authorization servers and resource servers from different vendors can leverage this profile to consume identity and payment tokens in an interoperable manner.

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://skyfire-xyz.github.io/kyapay-ietf-draft/draft-skyfire-kyapayprofile.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-skyfire-kyapayprofile/.

Source for this draft and an issue tracker can be found at https://github.com/skyfire-xyz/kyapay-ietf-draft.

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 3 September 2026.

Table of Contents

1. Introduction

As software agents evolve from pre-orchestrated workflow automations to truly autonomous or semi-autonomous assistants, they require the ability to identify themselves -- and more importantly, identify their human principals -- to external systems. Agents acting on behalf of users to discover services, create accounts, or execute actions currently face significant operational barriers.

The KYAPay token addresses these challenges by providing a standard envelope to carry verified identity and payment information. By utilizing "kya" (Agent Identity) and "pay" (Payment) tokens, agents can identify their human principals to services, sites, bot managers, customer identity and access management (CIAM) systems, and fraud detectors. This enables agents to bypass common blocking mechanisms and access services that were previously restricted to manual human interaction.

KYAPay does not aim to define agentic identity in its entirety. Rather, it specifies a standard and extensible JWT profile for a token that can be used to securely share human principal and agent identity information with websites and APIs. KYAPay tokens provide a strong signal of human presence behind agentic requests that are otherwise indistinguishable from programmatic and potentially malicious bot requests.

Note that, in the future, the payment token functionality could be split into a separate specification, if desired by a working group adopting the specification. It is retained here at present for ease of reviewing.

1.1. Use Cases for the KYAPay Token

Enabling agents to access websites and APIs on behalf of the human principals they represent is a design goal of KYAPay tokens. Today's internet is designed primarily for humans, meaning that automated systems are often classified as malicious and blocked by web security infrastructure. However, the rise of AI agents has introduced a new paradigm where programmatic clients legitimately access websites and APIs on behalf of human principals. Because these agents can be hard to distinguish from traditional bots, they are often inadvertently blocked, creating a need for the web security ecosystem to distinguish between legitimate agentic traffic and truly malicious activity. KYAPay tokens are designed to address this challenge by enabling agents to convey verified identity and payment credentials. These tokens can provide web security systems and merchants with a strong signal that the requests are authorized by a human, allowing them to safely permit legitimate programmatic transactions while aggressively blocking undesired traffic.

Enabling agents to create accounts and/or log in to accounts on behalf of their human principals is a related design goal. To achieve this, systems can utilize a token exchange workflow [RFC8693]. In this process, a Security Token Service (STS), Identity Provider (IdP), or OAuth Authorization Server verifies incoming KYA tokens and extracts claims associated with the human principal, such as email addresses. The authorization server then performs a token exchange, swapping the KYA token for a standard OAuth Access Token, which the agent subsequently uses to interact with the target service. Crucially, this architecture allows the service to know that the agent is acting on behalf of the user, making it possible to differentiate between direct, human-present sessions and human-initiated, agentic sessions for authorization, auditing, and security purposes.

Enabling agents to have ubiquity of access across the Internet just like their human principals is a related design goal. Automation typically scales as it achieves higher reliability and lower cost-to-entry. Unlike the structured logic required by cron jobs or low-code / no-code platforms, agentic automation leverages LLMs to execute tasks via natural language, effectively removing the software-skill barrier. As model reasoning improves and infrastructure scales, these agents become increasingly dependable and affordable for the human principal. To maximize utility, agents require ubiquitous

Internet access, a feat made possible by KYAPay Token Issuers. By providing a client-side verification framework analogous to the server-side role of Certificate Authorities (CAs), KYAPay builds a standardized network of acceptance across the web security ecosystem. This allows for the seamless attestation of both the agent's and the human principal's identity, ensuring secure, cross-domain task execution without the friction of fragmented authentication silos.

Enabling the ecosystem of web security vendors to engage in finer-grained and deliberate bad-actor mitigation is a related design goal. KYA tokens provide a layered, verified, and extensible identity stack specifically engineered for autonomous agents. This framework allows the web security ecosystem to distinguish among individual agent instances, the platforms they run on, and the human principals behind them. By establishing this level of granular visibility, security systems can transition from broad defensive measures to specific mitigation; rather than being forced to block an entire platform, administrators can now isolate and neutralize a single malicious human user or a malfunctioning software instance without disrupting legitimate traffic.

Note that the protocols using these tokens to achieve these goals are not defined by this specification. The interoperable use of them for these purposes will require further specification.

Early production deployments of KYAPay tokens are described at https://kyapay.org.

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.

The claims iss, iat, exp, aud, and jti are defined by [RFC7519]. The header parameters alg, kid, and typ are defined by [RFC7515].

The alg value ES256 is a digital signature algorithm defined in Section 3.4 of [RFC7518].

2.1. Roles

Agent: An application, service, or specific software process, executing on behalf of a Principal.

Agent Identity: A unique identifier and a set of claims describing

an agent. Grouped into the aid claim for convenience. Because an agent can be public or confidential (as described in Section 2.1 of [RFC6749]), the level of assurance for these claims varies dramatically. Agents also vary in terms of longevity -- they can have stable long-running identities (such as those of a server-side confidential client), or they can be transient and ephemeral, and correspond to individual API calls or compute workloads.

Agent Platform: The service provider and runtime environment hosting the Agent, such as a cloud compute provider or AI operator service. Assertions about the agent platform are grouped into the apd claim, and are primarily used to identify the Principal entity operating the platform, allowing consumers of the token to apply reputation-based logic or offer platform-specific services.

Principal: A legal entity (human or organization) on whose behalf / in whose authority an agent or service is operating.

2.1.1. Buy-Side Roles

Buyer Agent: An Agent performing tasks on behalf of a Buyer Principal, that has its own Agent Identity, grouped into the aid claim.

Buyer Agent Platform: The Agent Platform hosting the Buyer Agent. Some use cases require the Platform to have its own verified identity assertions, grouped into the apd claim.

Buyer Principal: A legal entity (human or organization) behind the purchase / consumption of a product or service. The Principal typically interacts with the seller via a Buyer Agent. Many sellers are required to be able to determine the Buyer Identity in order to comply with KYC/AML regulations, accounting standards, and to maintain a direct customer relationships. The buyer principal’s identity is grouped into the hid claim.

Buyer Identity: The aggregate verified identity assertions of the buy-side entities, typically encompassing the Buyer Principal, the Buyer Agent Platform, and the Buyer Agent itself. This composite identity is conveyed via the KYA token, allowing the seller to verify the entire chain of responsibility behind a request. The buyer identity utilizes the hid, apd, and aid claims.

2.1.2. Sell-Side Roles

Seller Agent: An Agent performing tasks on behalf of a Seller

Principal, directly interacting with Buyer Agents to facilitate discovery and purchase. Typically runs on Internet-connected infrastructure, and discoverable via service directories. Seller agent identity claims are also grouped into the aid claim if KYA tokens are generated for the sellers.

Seller Agent Platform: The Agent Platform that hosts Seller Agents. Some use cases require the Platform to have its own verified identity assertions, grouped into the apd claim.

Seller Principal: A human principal (individual or organization) that that owns the product, service, API, website, or content being consumed or sold, and serves as the ultimate beneficiary of a transaction. The seller principal’s identity is grouped into the hid claim.

Seller Identity: The aggregate verified identity assertions of the sell-side entities, typically encompassing the Seller Principal, the Seller Agent Platform, as well as the Seller Agent Identity. These various aspects of Seller Identity allow Buyers and Buyer Agents to perform reputation-based logic, to verify that they are interacting with the authorized (and expected) counter-party, and to fulfill KYC/AML regulation requirements. The seller identity utilizes the hid, apd, and aid claims.

2.1.3. Ecosystem Infrastructure Roles

Identity Token Issuer: A trusted neutral entity that conducts Know Your Customer (KYC) and Know Your Business (KYB) (for organizations) verifications. It is responsible for issuing cryptographically signed kya tokens that attest to the identity of the Principal, Agent, and Agent Platform, for both Buyers and Sellers.

Payment Token Issuer: A trusted entity responsible for facilitating the exchange of payments and credentials between the Buyer and Seller. It issues signed pay tokens that enable settlement via various schemes (Cards, Banks, Cryptocurrency), without exposing raw credentials or secrets.

3. KYAPay Token Schemas

3.1. Common Token Claims

The following are claims in common, used within the KYA (Know Your Agent), PAY (Payment), and KYA-PAY (combined Know Your Agent and Payment) Tokens.

iss: REQUIRED - URL of the token’s issuer. Used for discovering JWK Sets for token signature verification, via the /.well-known/ jwks.json suffix mechanism.

sub: REQUIRED - Subject Identifier. Must be pairwise unique within a given issuer.

aud: REQUIRED - Audience (used for audience binding and replay attack mitigation), uniquely identifying the seller agent. A single string value.

iat: REQUIRED - as defined in Section 4.1.6 of [RFC7519]. Identifies the time at which the JWT was issued. This claim must have a value in the past and can be used to determine the age of the JWT.

jti: REQUIRED - Unique ID of this JWT as defined in Section 4.1.7 of [RFC7519].

exp: REQUIRED - as defined in Section 4.1.4 of [RFC7519]. Identifies the expiration time on or after which the JWT MUST NOT be accepted for processing.

sdm: OPTIONAL - Seller domain, associated with the audience claim, the token is intended for.

srl: OPTIONAL - Seller resource locator - URL the agent is intended to access.

ori: OPTIONAL - URL of the token’s originator.

env: OPTIONAL - Issuer environment (such as "production" or "sandbox"). Additional values may be defined and used.

ssi: OPTIONAL - Seller Service ID that this token was created for.

btg: OPTIONAL - Buyer tag - an opaque reference ID internal to the buyer.

Additional claims MAY be defined and used in these tokens. The recipient MUST ignore any unrecognized claims.

3.2. KYA Token

The following identity related claims are used within KYA and KYA-PAY tokens:

hid: REQUIRED (Required for human identity use cases) - A map of

human identity claims (individual or organization).

apd: OPTIONAL - Agent Platform identity claims.

aid: REQUIRED - Agent identity claims.

scope OPTIONAL - String with space-separated scope values, per [RFC8693]

The following informative example displays a decoded KYA type token.

```
{
  "kid": "YjFdJgFNWj9AkUmtoXILwoeb37PsBuGWVK6_QvFLwJw", // JWK Key ID
  "alg": "ES256",
  "typ": "kya+jwt"
}.{
  "iss": "https://example.com/issuer", // Issuer URL
  "iat": 1742245254,
  "exp": 1773867654,
  "jti": "b9821893-7699-4d24-af06-803a6a16476b",
  "sub": "bb713104-c14e-460f-9b7c-f8140fa9bea4", // Buyer Agent Account ID
  "aud": "7434230d-0861-46f2-9c2c-a6ee33d07f17", // Seller Agent Account ID

  "env": "production",
  "ssi": "bc3ff89f-069b-4383-82a9-8cfe53c55fc3", // Seller Service ID
  "btg": "4f6cbd39-215c-4516-bf33-cab22862ee60", // Buyer Tag (Internal Reference ID)

  "hid": {
    "email": "buyer@buyer.com"
  },
  "apd": {
    "id": "d3306fc0-602b-47e6-9fe2-3d55d028fbd2"
    "name": "Acme Shopping Agents", // Agent platform name
    "email": "platform@acme.com", // Email address for the agent platform
    "phone_number": "+12345677890", // Phone number for the agent platform
    "organization_name": "Acme Shopping Inc.", // Legal name of the agent platform
    "verifier": "https://www.verifier.com/", // URL of the Identity verifier
    "verified": true, // Outcome of the verifier's KYA verification
    "verification_id": "a23c1fe4-a4b7-442d-8bca-3c8fad5ec3a6" // Verifier's verification
ID
  },
  "aid": {
    "name": "Acme Agent Extraordinaire",
    "creation_ip": "54.86.50.139", // IP Address where token was created
    "source_ips": ["54.86.50.139-54.86.50.141", "1.1.1.0/24",
      "2001:db8:abcd:0012::/64", "acme.com"]
      // IP addresses from which the buyer agent will make requests to the seller
  }
}
```

Figure 1: A KYA type token

3.2.1. hid - Human Identity Sub-Claims

The Human Identity (hid) claim contains sub-claims identifying the human principal (individual or organization) as follows.

email: REQUIRED - Email address associated with the human individual or organization

given_name: OPTIONAL - Given name(s) or first name(s) of the human principal if they are an individual.

middle_name: OPTIONAL - Middle name(s) of the human principal if they are an individual.

family_name: OPTIONAL - Surname(s) or last name(s) of the human principal if they are an individual.

phone_number: OPTIONAL - Phone number associated with the human individual or organization.

organization_name: OPTIONAL - Name of the organization.

verifier: OPTIONAL - URL of the Identity Verifier

verified: OPTIONAL - Boolean Verification status. True if verified, otherwise false.

verification_id: OPTIONAL - Verification identifier. Identifier for the verification performed, such as a GUID.

Additional sub-claims MAY be defined and used. The recipient MUST ignore any unrecognized sub-claims.

3.2.2. Agent Platform Identity apd Sub-Claims

The apd claim is optional. If present, it contains the following sub-claims.

id: REQUIRED - Agent Platform identifier.

name: REQUIRED - Agent Platform name.

email: OPTIONAL - Email associated with agent platform.

phone_number: OPTIONAL - Phone number associated with agent platform.

organization_name: OPTIONAL - Legal name associated with agent platform.

verifier: OPTIONAL - URL of the Identity Verifier

verified: OPTIONAL - Boolean Verification status. True if verified, otherwise false.

verification_id: OPTIONAL - Verification identifier. Identifier for the verification performed, such as a GUID.

Additional sub-claims MAY be defined and used. The recipient MUST ignore any unrecognized sub-claims.

3.2.3. Agent Identity aid Sub-Claims

The aid claim is optional. If present, it contains the following sub-claims.

name: REQUIRED - Agent name. The name should reflect the business purpose of the agent.

creation_ip: REQUIRED - The public IP address of the system / agent that requested the token. Its value is a string containing the public IPv4 or IPv6 address from where the token request originated. It MUST be captured directly from the token request.

source_ips: OPTIONAL - Valid public IP address, or range of public IP addresses, from where the system / agent’s requests to merchants / services will originate. Array of comma-separated IPv4 addresses or ranges, IPv6 addresses or ranges, or domain names resolvable to an IP address via DNS. IPv4 and IPv6 addresses can be a single IPv4 or IPv6 address or a range of IPv4 or IPv6 addresses in CIDR notation or start-and-end IP pairs.

Additional sub-claims MAY be defined and used. The recipient MUST ignore any unrecognized sub-claims.

3.3. PAY Token

The following payment related claims are used within PAY and KYA-PAY type tokens:

spr: OPTIONAL - JSON string representing seller service price in currency units.

sps: OPTIONAL - Seller pricing scheme, which represents a way for

the seller list how it charges for its service or content. One of pay_per_use, subscription, pay_per_mb, or custom. Additional values may be defined and used.

amt: REQUIRED - JSON string representing token amount in currency units.

cur: REQUIRED - Currency unit, represented as an ISO 4217 three letter code, such as "EUR".

val: REQUIRED - JSON string representing token amount in settlement network’s units.

mnr: OPTIONAL - JSON number representing maximum number of requests when sps is pay_per_use.

stp: REQUIRED - Settlement type (one of coin or card). Additional values may be defined and used.

sti: REQUIRED - Meta information for payment settlement, depending on settlement. type.

3.3.1. Agent Identity sti Sub-Claims

The sti claim is optional. If present, it MAY contain the following sub-claims, all of which are OPTIONAL.

type: REQUIRED - "type" is dependent on the "stp" value; for "coin" - "usdc"; for "card" - "visa_vic" or "mastercard_scof". Additional values may be defined and used.

paymentToken: OPTIONAL - String containing Virtual Payment Card Number in ISO/IEC 7812 format. 12-19 characters.

tokenExpirationMonth: OPTIONAL - String containing two-digit Expiration Month Number.

tokenExpirationYear: OPTIONAL - String containing four-digit Expiration Year.

tokenSecurityCode: OPTIONAL - String containing 3 or 4 digit CVV code.

Additional sub-claims MAY be defined and used. The recipient MUST ignore any unrecognized sub-claims.

3.3.2. PAY Token Example

The following informative example displays a decoded PAY type token.

```
{
  "kid": "FgT4q8c5IqbBCCjcho5JdeGQvuK1keMDFc9IwCm8J7Y", // JWK Key ID
  "alg": "ES256",
  "typ": "pay+jwt"
}.{
  "iss": "https://example.net/pay_token_issuer", // Issuer URL
  "iat": 1742245254,
  "exp": 1773867654,
  "jti": "b9821893-7699-4d24-af06-803a6a16476b",
  "sub": "8b810549-7443-494f-b4ad-5bc65871e32b", // Buyer Agent Account ID
  "aud": "37888095-2721-48d9-a2df-bfe4075f223a", // Seller Agent Account ID

  "env": "sandbox",
  "ssi": "274efc47-024e-466f-b278-152d2ee73955", // Seller Service ID
  "btg": "16c135ce-a99a-453d-a7b5-4958fd91de5f", // Buyer Tag (Internal Reference ID)

  "spr": "0.01",
  "sps": "pay_per_use",
  "amt": "15",
  "cur": "USD",
  "val": "15000000",
  "mnr": 1600,
  "stp": "card",
  "sti": {
    "type": "visa_vic",
    "paymentToken": "1234567890123456",
    "tokenExpirationMonth": "03",
    "tokenExpirationYear": "2030",
    "tokenSecurityCode": "123",
    "verifier": "https://verifier.example.info", // URL of payment method verifier
    "verified": true, // Outcome of the verifier’s payment method verification
    "verification_id": "3a6e1b76-8f78-4c24-b1bd-dc78a8cc3711" // Identifier for the verif
ication performed, such as a GUID.
  }
}
```

Figure 2: A PAY type token

3.4. KYA-PAY Token

The following informative example displays a decoded KYA-PAY type token.

```
{
  "kid": "YjFdJgFNWj9AkUmtoXILwoeb37PsBuGWVK6_QvFLwJw", // JWK Key ID
  "alg": "ES256",
  "typ": "kya-pay+jwt"
}.{
  "iss": "kya-pay.example.org", // Issuer URL
  "iat": 1742245254,
  "exp": 1773867654,
  "jti": "b9821893-7699-4d24-af06-803a6a16476b",
  "sub": "f24a431d-108c-46e6-9357-b428c528210e", // Buyer Agent Account ID
  "aud": "5e00177d-ff7f-424b-8c83-2756e15efbed", // Seller Agent Account ID

  "env": "production",
  "ssi": "3e6d33a1-438e-482e-bba5-6aa69544727d", // Seller Service ID
  "btg": "c52e0ef2-e27d-4e95-862e-475a904ae7b2", // Buyer Tag (Internal Reference ID)

  "hid": {
    "email": "maryjane@buyer.example.com",
    "given_name": "Mary",
    "middle_name": "Jane",
    "family_name": "Doe",
    "phone_number": "+1-425-555-1212",
    "verified": false
  },
  "apd": {
    "id": "4b087db2-b6e5-48b8-8737-1aa8ddf4c4fe", // Agent platform ID
    "name": "Acme Shopping Agents", // Agent platform name
    "email": "platform@acme.com", // Email address for the agent platform
    "phone_number": "+12345677890", // Phone number for the agent platform
    "organization_name": "Acme Shopping Inc.", // Legal name of the agent platform
    "verifier": "https://www.verifier.com/", // URL of the Identity verifier
    "verified": true, // Outcome of the verifier's KYA verification
    "verification_id": "a23c1fe4-a4b7-442d-8bca-3c8fad5ec3a6" // Verifier's verification
ID
  },
  "aid": {
    "name": "Agentic Excellence Я Us",
    "creation_ip": "128.2.42.95", // IP Address where token was created
    "source_ips": ["54.86.50.139-54.86.50.141", "1.1.1.0/24",
      "2001:db8:abcd:0012::/64", "agentic-excellence.example.com"]
      // IP addresses from which the buyer agent will make requests to the seller
  },

  "spr": "0.01",
  "sps": "pay_per_use",
  "amt": "15",
  "cur": "USD",
  "val": "15000000",
  "mnr": 1600,
```

```
  "stp": "card",
  "sti": {
    "type": "visa_vic",
    "paymentToken": "1234567890123456",
    "tokenExpirationMonth": "03",
    "tokenExpirationYear": "2030",
    "tokenSecurityCode": "123"
  }
}
```

Figure 3: A KYA-PAY type token

4. Token Validation

4.1. Validating KYA and PAY Tokens

4.1.1. JWT Header Validation

1. alg - JWTs MUST be signed using allowed JWA algorithms (currently, ES256).

2. kid - The kid claim MUST be present, and set to a valid Key ID discoverable via the issuer’s (payload iss claim) JWK Set.

3. typ - The typ header parameter value MUST be one of: kya+jwt, pay+jwt, or kya-pay+jwt.

4.1.2. JWT Payload Validation

1. *Verify JWT Signature* - Valid JWTs MUST be signed with a valid key belonging To the token’s issuer (iss claim)

2. *Validate iss Claim* - Ensure that the token is signed by the expected valid issuer.

3. *Validate the exp Claim* - The verifier MUST validate that the token has not expired, within the verifier’s clock drift tolerance.

4. *Validate the iat Claim* - The verifier MUST validate that the token was issued in the past, within the verifier’s clock drift tolerance.

5. *Validate the jti Claim* - Ensure that the jti claim is present, and is a UUID.

6. *Validate the aud Claim* - Ensure that the aud identifies the recipient as the intended audience.

7. *Validate the env Claim* - Ensure that the Environment claim is set to an expected and use case appropriate value (such as production or sandbox)

4.2. Validating PAY Tokens

For tokens of type pay+jwt or kya-pay+jwt, perform the steps described in the Validating KYA and PAY Tokens section.

In addition, perform the following steps.

1. The val claim is greater than 0.

2. The amt claim is greater than 0.

3. The cur claim is set to a currency the seller supports (such as USD)

4. The sps claim, if present, matches the pricing scheme that you configured in the seller’s service

5. The spr claim, if present, matches the price that you configured in the seller’s service

5. Security Considerations

When validating the JWTs described in this specification, implementers SHOULD follow the best practices and guidelines described in [RFC8725].

6. Privacy Considerations

KYAPay tokens are designed to convey the information that an agent is acting on behalf of a principal - a person or organization. To do this, they will necessarily contain information about that principal that can be verified and utilized by participants in the system. Participants should therefore only share these tokens with other legitimate participants and not make their contents public or disclose them to unknown or untrustworthy parties.

Consent of the principal represented to participate in the interactions is vital. If I authorize an agent to shop for a widget at given price, it’s legitimate for the agent to carry enough information about me to the merchant to be able to do this for me. Whereas, if an agent claims to be shopping for me but does not have my authorization to do so, my privacy and possibly also my financial integrity are being violated.

The principle of minimal disclosure should be employed. Only the infomation needed to facilitate the intended interactions should be placed in the tokens and conveyed to participants.

7. IANA Considerations

7.1. JSON Web Token Claims Registration

This specification registers the following Claims in the IANA "JSON Web Token Claims" registry [IANA.JWT.Claims] established by [RFC7519].

7.1.1. "sdm" Claim

* Claim Name: sdm
* Claim Description: Seller domain the token is intended for
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.2. "srl" Claim

* Claim Name: srl
* Claim Description: Seller resource locator - URL the agent is intended to access
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.3. "ori" Claim

* Claim Name: ori
* Claim Description: URL of the token’s originator
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.4. "env" Claim

* Claim Name: env

* Claim Description: Issuer environment (such as "production" or "sandbox")
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.5. "btg" Claim

* Claim Name: btg
* Claim Description: Buyer tag, an opaque reference ID internal to the buyer
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.6. "hid" Claim

* Claim Name: hid
* Claim Description: JSON structure containing human identity claims
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.7. "apd" Claim

* Claim Name: apd
* Claim Description: JSON structure containing agent platform identity claims
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.8. "aid" Claim

* Claim Name: aid
* Claim Description: JSON structure containing agent identity claims
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#common-claims) of this specification

7.1.9. "spr" Claim

* Claim Name: spr
* Claim Description: JSON string representing seller service price in currency units
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#pay-token) of this specification

7.1.10. "sps" Claim

* Claim Name: sps
* Claim Description: Seller pricing scheme, which represents a way for the seller list how it charges for its service or content
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#pay-token) of this specification

7.1.11. "amt" Claim

* Claim Name: amt
* Claim Description: JSON string representing token amount in currency units
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#pay-token) of this specification

7.1.12. "cur" Claim

* Claim Name: cur
* Claim Description: Currency unit, represented as an ISO 4217 three letter code, such as "EUR"
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com
* Reference: (#pay-token) of this specification

7.1.13. "val" Claim

* Claim Name: val

* Claim Description: JSON string representing token amount in settlement network’s units

* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

* Reference: (#pay-token) of this specification

7.1.14. "mnr" Claim

* Claim Name: mnr

* Claim Description: JSON number representing maximum number of requests

* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

* Reference: (#pay-token) of this specification

7.1.15. "stp" Claim

* Claim Name: stp

* Claim Description: Settlement type

* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

* Reference: (#pay-token) of this specification

7.1.16. "sti" Claim

* Claim Name: sti

* Claim Description: Meta information for payment settlement, depending on settlement

* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

* Reference: (#pay-token) of this specification

7.2. Media Types Registration

This section registers the following media types [RFC2046] in the IANA "Media Types" registry [IANA.MediaTypes] in the manner described in [RFC6838].

7.2.1. application/kya+jwt

* Type name: application
* Subtype name: kya+jwt
* Required parameters: n/a
* Optional parameters: n/a
* Encoding considerations: Uses JWS Compact Serialization as defined in [RFC7515]
* Security considerations: See Security Considerations in in [RFC7519]
* Interoperability considerations: n/a
* Published specification: (#kya-token) of this specification
* Applications that use this media type: Applications using Know Your Agent tokens
* Additional information:
  - Magic number(s): n/a
  - File extension(s): n/a
  - Macintosh file type code(s): n/a
* Person & email address to contact for further information: TBD
* Intended usage: COMMON
* Restrictions on usage: none
* Author: Michael B. Jones - michael_b_jones@hotmail.com
* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

7.2.2. application/pay+jwt

* Type name: application
* Subtype name: pay+jwt
* Required parameters: n/a

* Optional parameters: n/a

* Encoding considerations: Uses JWS Compact Serialization as defined in [RFC7515]

* Security considerations: See Security Considerations in in [RFC7519]

* Interoperability considerations: n/a

* Published specification: (#pay-token) of this specification

* Applications that use this media type: Applications using Pay tokens

* Additional information:

  - Magic number(s): n/a

  - File extension(s): n/a

  - Macintosh file type code(s): n/a

* Person & email address to contact for further information: TBD

* Intended usage: COMMON

* Restrictions on usage: none

* Author: Michael B. Jones - michael_b_jones@hotmail.com

* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

7.2.3. application/kya-pay+jwt

* Type name: application

* Subtype name: kya-pay+jwt

* Required parameters: n/a

* Optional parameters: n/a

* Encoding considerations: Uses JWS Compact Serialization as defined in [RFC7515]

* Security considerations: See Security Considerations in in [RFC7519]

* Interoperability considerations: n/a

* Published specification: (#kya-pay-token) of this specification

* Applications that use this media type: Applications using KYA-Pay tokens

* Additional information:

  - Magic number(s): n/a

  - File extension(s): n/a

  - Macintosh file type code(s): n/a

* Person & email address to contact for further information: TBD

* Intended usage: COMMON

* Restrictions on usage: none

* Author: Michael B. Jones - michael_b_jones@hotmail.com

* Change Controller: Michael B. Jones - michael_b_jones@hotmail.com

8. References

8.1. Normative References

[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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/rfc/rfc7518>.

[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>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/rfc/rfc8693>.

8.2. Informative References

[IANA.JWT.Claims] IANA, "JSON Web Token Claims", n.d., <https://www.iana.org/assignments/jwt>.

[IANA.MediaTypes] IANA, "Media Types", n.d., <https://www.iana.org/assignments/media-types>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/rfc/rfc2046>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

Document History

[[to be removed by the RFC Editor before publication as an RFC]]

-00

* Initial Internet Draft.

Contributors

Dmitri Zagidulin

Authors' Addresses

Ankit Agarwal
Skyfire
Email: ankit@skyfire.xyz

Michael B. Jones
Self-Issued Consulting
Email: michael_b_jones@hotmail.com
URI: https://self-issued.info/