**Digital Forensics As A Service**

**Introduction**

To enable contextual and secure communication between distributed and polyglot service providers and between front-end user experience and back-end systems, we leverage standardized Starbucks claims conveyed in JSON Web Token (JWT) format and encapsulated in a signed JSON Web Signature (JWS) envelope. Signature validation (asymmetric) of a trusted issuer enables assurance that claims have not been tampered with in-flight (i.e. signature over hashed claim set) and validation of the issuer identity; taken together, the receiver may trust the claims.

See JSON Web Token Introduction for general background on JWT format.

Important: The ultimate "security" of DFaaS JWT depends on secure and consistent cryptographic key management processes.

Important: DFaaS JWT conveys both the Subject (sub) in the current session and the service Client (s-cid). In some cases, the Subject and the Caller may be the same (e.g. a system making a call on its own behalf); in other cases, the Subject and Caller will be different (e.g. a system making a call on behalf of an identified End-User such as a Customer or Partner).

Identifiers used in the DFaaS JWT format fall into two categories:

1. Unique user identifiers (UIDs) from an Identity Provider (IdP) – For Subject (sub) when the subject is a Customer or Partner (or 3rd-party Merchant)

2. Unique DFaaS Entities (DNs/RDNs) – Always for Issuer (iss), Client (s-cid), and Audience (aud); for Subject when the subject is a DFaaS entity acting on its own behalf

Subject Tags (s-stag) used in the DFaaS JWT format represent domain-independent "fact" statements or tags (i.e. not application specific) that can be asserted about the current subject context. DFaaS Subject Tags are centrally managed (enumerated) codes that can be understood by all DFaaS participants. The intent of subject tags is to enable a receiver to reason and decision around the nature of the Subject, Channel, and Market in the current call.

Diagram of relationship of JWT format to roles and entities: DFaaS JWT - Key Identifier and Context Relationships See Customer Technology Platform - DFaaS IAM and Network-Level Controls (Target) for depiction of DFaaS Token passing between DFaaS frontend and backend (B2C) and between domain components (A2A) in a logical target architecture. Target represents deployment of next-generation customer IAM deployment addressing holistic concerns cross-cloud, crossstack, and cross-team; target emphasizes externalization of policy and enforcement from business logic (for agility and consistency) and seeks to embed or bridge to different cloud hosting providers and architectures in-use at Starbucks.

**JSON Web Signature (JWS)**

The JWT Claim Set will be conveyed in a digitally-signed JWS (RFC 7515) envelope using JWS JSON Compact Serialization. The cryptographic algorithm used to secure the JWS will be asymmetric (PKI). By this means, the issuer (owner of private key) will be authenticated downstream (with the corresponding public key) and the JWS protected header and JWT payload can be confirmed as untampered (i.e. as the issuer sent).

**Cryptographic Algorithm**

The cryptographic algorithm used to secure the JWS will be asymmetric (PKI) leveraging RSA PKCS#1 signature with SHA-256 hashing. The corresponding algorithm type enumeration defined in the JWS specification is "RS256".

Issuer keys shall be not less than 2048-bit in length.

JWS Protected Header

{

"alg": "RS256",

"typ": "JWT",

"x5t": "b9abba343728b7210ed5d52e0b401b73496327fc"

}

**Key Identification**

Downstream JWT validators will authenticate the issuing DFaaS Entity (iss) and trust the validity of conveyed claim set by successful JWS signature validation with the issuer's public key.

To support operational key lifecycles (e.g. key rotation) a given Issuer entity may have multiple valid keys for a time window necessary to complete rotation. The validator will select the issuer public key to be used in signature validation by:

1. Parsing the issuing DFaaS Entity (iss) from the unvalidated JWT

2. Parsing the public certificate thumbprint (x5t) from the JWS protected header; the thumbprint is the base64 URL-safe encoding of the SHA-1 hash of the public certificate as per RFC 7515 and RFC 5280 (and as implemented by standard tools such as openssl)

3. Obtaining the public certificate (that encloses the public key) by match of public certificate thumbprint AND issuing DFaaS Entity

Notes:

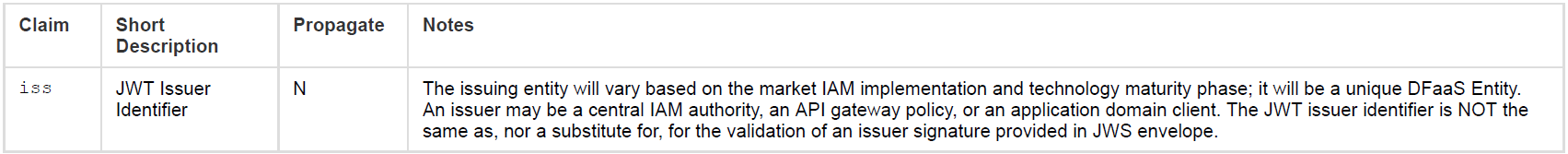
The X.509 certificates for issuing DFaaS Entities shall have the Certificate Subject encoded as the DFaaS Entity distinguished name (allowing match in #3)

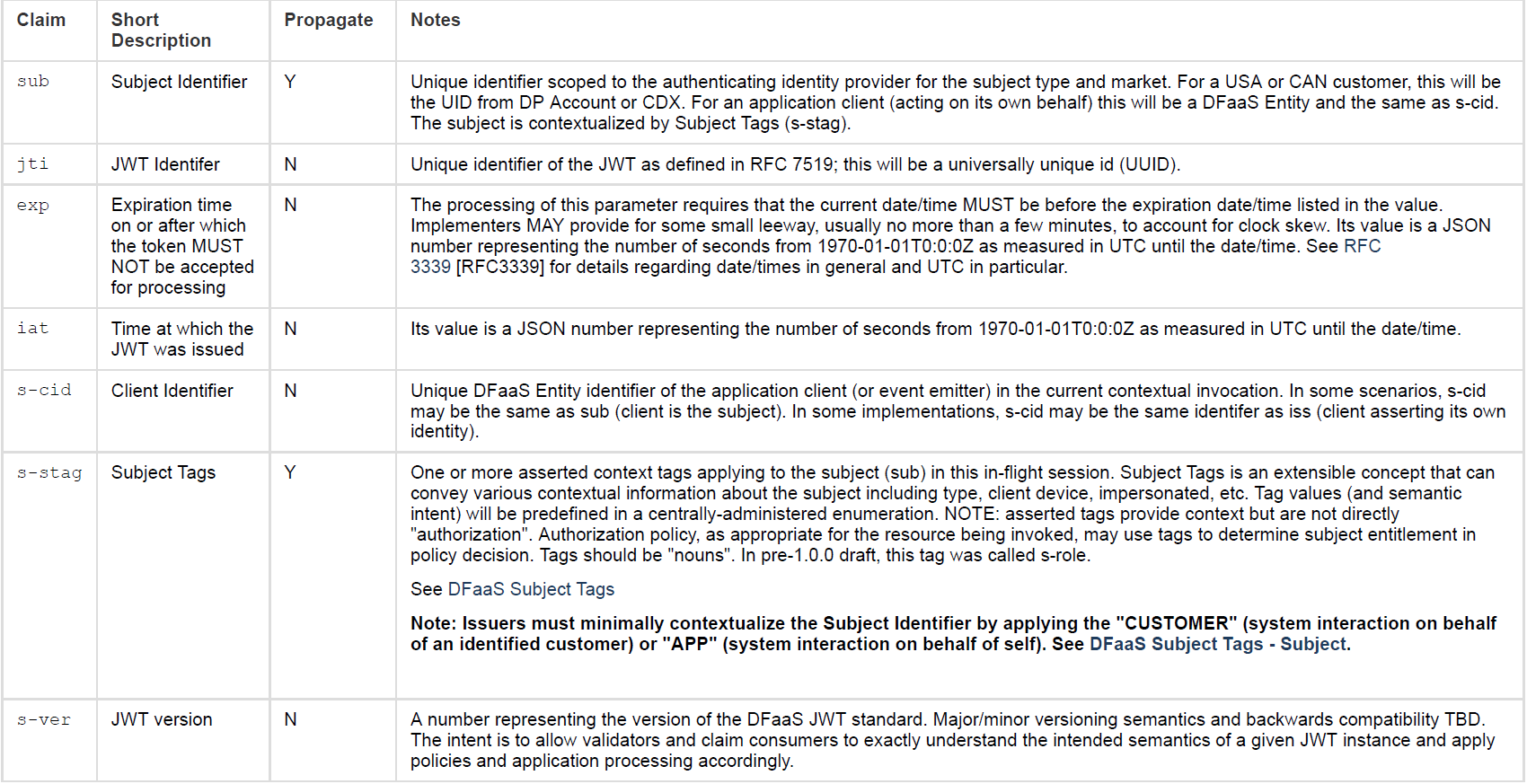
The public certificate thumbprint is the base64 URL-safe encoding of the SHA-1 hash of the public certificate as per RFC 7515 and RFC 5280 (and as implemented by standard tools such as openssl

The source (or provisioning lifecycle) for a validator to retrieve the exact public key necessary to validate the signature (Key identification ) is not documented in this specification; however the validator shall in any case select public key by combination of public certificate thumbprint and public certificate subject. The ultimate surety/strength of the DFaaS JWT solution is rooted in the authoring and secure distribution process of key material.

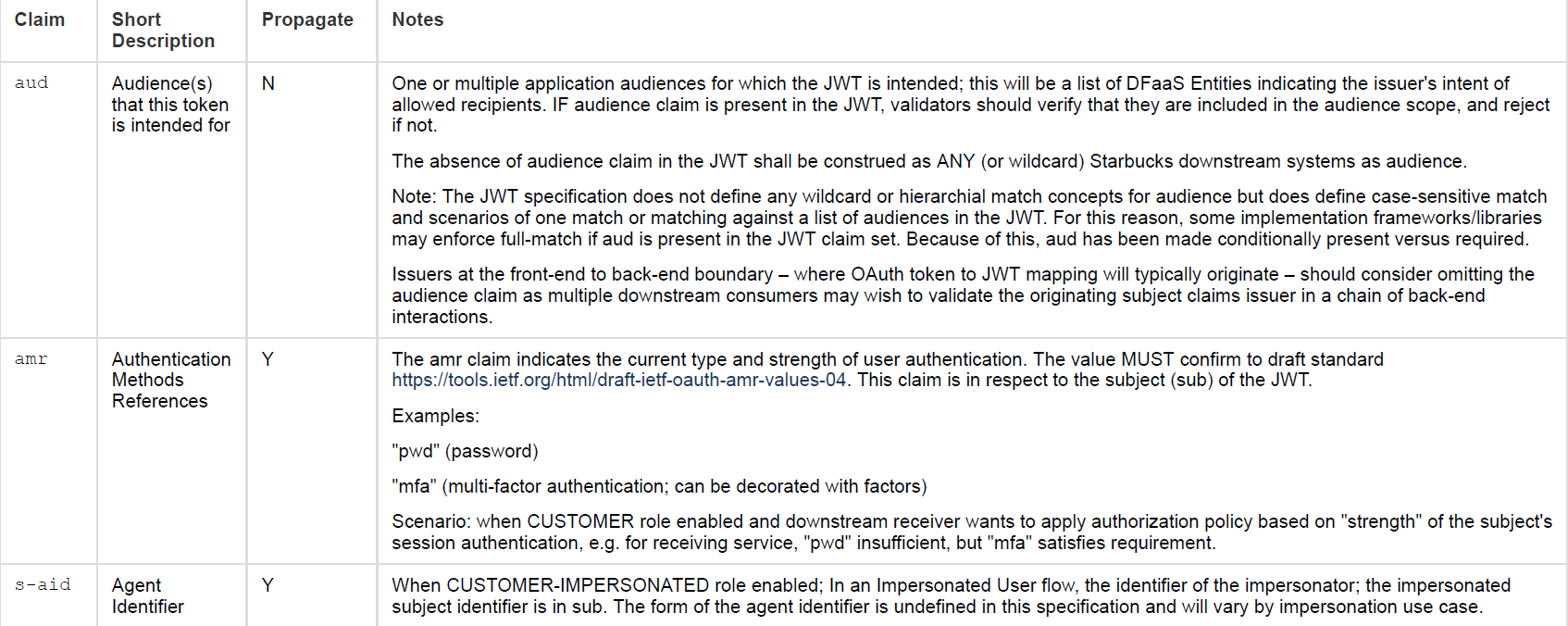
DFaaS JWT Specification

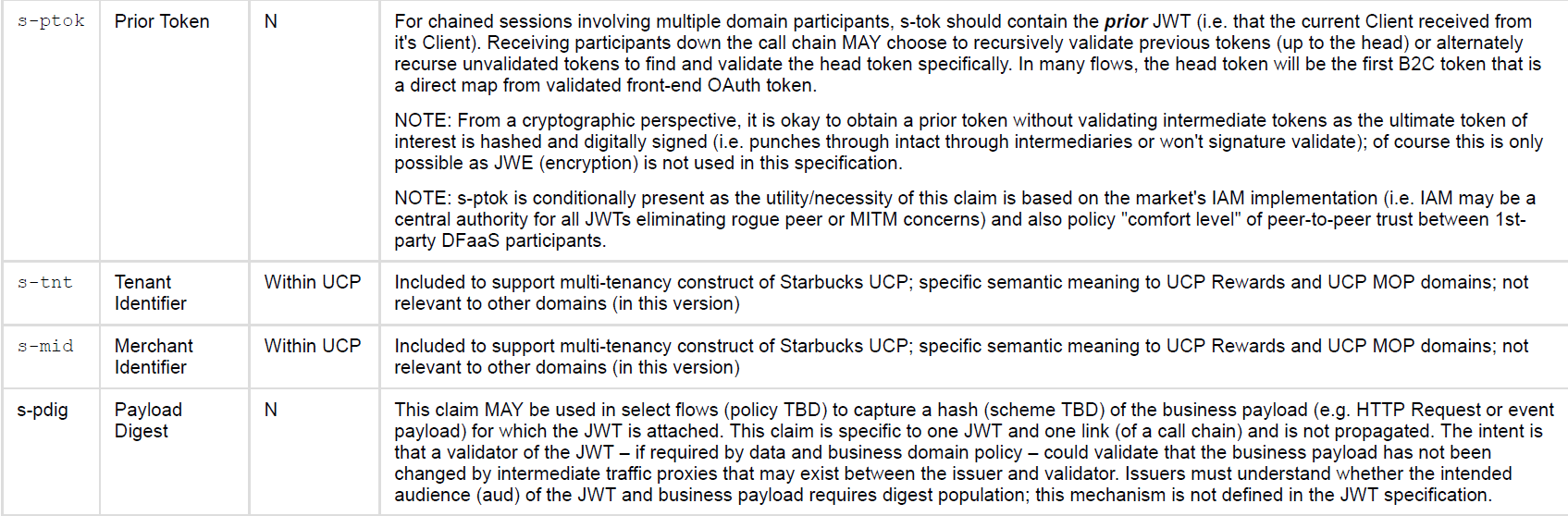
Claim Set





Conditionally Present





**Usage in HTTP Request**

The JWS (and contianed JWT) shall be passed as a request Authorization header (bearer):

Authorization:

Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImI5YWJiYTM0MzcyOGI3MjEwZWQ1ZDUyZTBiNDAxYjczNDk2MzI3ZmMifQ.eyJpc3MiOiJj

**Claim Set Propagation**

In a multi-participant call chain, e.g. Client A calls Service B which in turn calls Service C, each link in the chain has a new JWT that defines the Client (s-cid), Issuer (iss), and Audience (aud) of that specific link. When the intent is to propagate the Subject from start to finish in the chain, e.g. a Customer subject, the issuer in each link should copy unmodified (propagate) the claim types marked "Y" in the propogate column.

Additionally, each JWT issuer (accept the first link) should propagate the JWT received from its immediate client (in JWS JSON Compact Serialization notation) in the s-ptok claim of the JWT for the next link. For example Service B should include Client A's JWT in the next link to Service C.

If a participant downstream in the call chain (based on domain-specific operational trust policy) does not trust the previous chain participants to reliably propagate subject claims, it may choose to recurse the s-ptok claim (a JWS-signed JWT) to selectively validate upstream issuers and upstream claim sets.

Note: In target architecture, the desire is to offload (or externalize) issuance, validation, and propagation to a trusted security/IAM provider technology/service. This changes the trust model cardinality in multi-participant scenarios.

**JWT Identifier and Expiry**

Each JWT shall be identified by a 128-bit universally unique identifer (UUID). The default time-to-live (TTL) of a JWT shall be +60 seconds from creation

General Design assumptions

1. Try to use claim names already defined in the JWT community where possible (RFC 7519, OIDC 1.0, etc.) – for SBUX custom private claims use a shorthand prefix (e.g. "s-") to provide best-effort collision avoidance while avoiding lengthy public name prefixing (e.g. http://www.starbucks.com) and/or formal registration.

2. Assume tracking items, e.g. DFaaS Correlation ID are not included in the JWT but rather their own protocol-specific headers

3. The DFaaS JWT and B2C Access Token are separate sets of claims rationalized to respective domain usage and "front end" versus "back end" concerns. SBUX will encode a mapping between front end and backend claims using policy rules administered centrally.

4. Typical subject attributes such as first name, given name, etc. will not be conveyed in the JWT but rather in the business payload or via customer system of record lookup.

5. DFaaS IAM implementation may vary from "target" architecture by phase and by market; thus, no assumption has been made about central issuing authority or end-to-end view of session context (as might be provided by a security "fabric" capability).

6. Attempt to keep number of claims (especially custom claims) to a minimum and to avoid mingling of numerous claim types with complex business logic and process (which may and will evolve over time).

7. Leverage multi-value s-stag as an extensibility mechanism that is flexible to describe the current subject's context.

8. Provide a cloud, stack, and language agnostic mechanism rooted in sound crypto principles

**Background and Reference Material**

Resources

* JSON Web Token Standard: RFC 7519
* JSON Web Signature Standard: RFC 7515
* JSON Web Key Standard: RFC 7517
* JSON Web Tokens Claims Registry
* Registered Assignments
* Authentication Reference Methods Draft IETF standard
* Initial Registry Contents
* A String Representation of Distinguished Names: RFC 1779
* Internet X.509 Public Key Infrastructure Certificate: RFC 5280
* Example from Google IoT
* Standard Claim Reference

