CAI Claims

Leonard Rosenthol

April 16, 2020

Table of Contents

# Claims

## Introduction

As described in the [Glossary](Glossary.md), a claim is “A JSON-formatted data structure representing the assertions of fact by an *actor* concerning an *asset* at a specific time and for a specific reason”. *Claims* in the CAI ecosystem are equivalent to (and compatible with) a [W3C Verifiable Credential](https://www.w3.org/TR/vc-data-model/), however since our claims aren’t about people we don’t use the term credential.

## Technologies

* [JSON](https://tools.ietf.org/html/rfc8259)
* [JSON-LD](https://www.w3.org/TR/json-ld11/)
* [JSON Web Signatures](https://tools.ietf.org/html/rfc7515) (JWS)
* [JSON Web Signature Unencoded Payload Option](https://tools.ietf.org/html/rfc7797) (JWS-UPO)
* [JSON Web Token](https://tools.ietf.org/html/rfc7519) (JWT)
* [Verifiable Credentials](https://www.w3.org/TR/vc-data-model/) (VC)
* [eXtensible Metadata Platform](https://www.adobe.com/products/xmp.html) (XMP)

## XMP

Every asset, for which a claim is being made, shall contain embedded XMP. If the asset does not contain XMP at the time a claim is made, the *claims recorder* shall create it prior to signing the claim. The [Adobe XMP Toolkit SDK](https://github.com/adobe/XMP-Toolkit-SDK/) can be used to create and modify XMP in [various asset types](https://github.com/adobe/XMP-Toolkit-SDK/tree/master/XMPFiles/source/FileHandlers).

As defined in the [ISO 16684-1 standard](https://www.iso.org/standard/75163.html), the XML+RDF serialization of the metadata shall be uncompressed and can be located starting with the bytes <?xpacket begin= and ending with the bytes <?xpacket end="w"?>.

## Claim Internals

### JWT Claim Set

A claim is defined as a standard JWT claim set (https://tools.ietf.org/html/rfc7519#section-4) that also follows the requirements for a *VC* (6.3.1 of the VC spec) with CAI as the \_@context\_ and *credentialSubject*. Claims can either be signed or unsigned. An unsigned claim may contain any values (*for now*), though it is RECOMMENDED to include the *actions* that preceded this claim.

**Example Claim**

{  
 "jti": "3e061079a991071a5d2dcfd2ee1c6794",  
 "iss": "http://cai.adobe.com",  
 "iat": 1516239022,  
 "vc" : {  
 "@context": [  
 "https://www.w3.org/2018/credentials/v1",  
 "https://ns.adobe.com/cai"  
 ],  
 "type": ["VerifiableCredential", "AuthenticContent"],  
 "credentialSubject": {  
 "actions": [ { "stEvt:action": "filter\_applied", "stEvt:when": "2020-02-11T09:00:00" } ],  
 "signature" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..t3VhQ7QsILDuV\_HNFSMI-Fb2FoT7fuzalpS5AH8A9c0",  
 "url": "https://cai\_resolver.adobe.com/A1B2C3D4E5",  
 "parent\_url": "https://cai\_resolver.adobe.com/123456789123456789000"  
 }  
 }  
}

The JWT claim set shall include: - *jti* - a unique identifier for the JWT. (NOTE: as per VC, this is equivalent to the VC id property) - *iss* - identifies the actor that issued the JWT as a case-sensitive string containing a StringOrURI. (NOTE: as per VC, this is equivalent to the VC issuer property) - *iat* - identifies the time at which the JWT was issued. Its value MUST be a number expressing a NumericDate - *vc* - this is the verifiable credential itself, which is a valid JSON-LD object including *type* and *credentialSubject*.

## Adobe’s view of DID

### Overview

The W3C specification for DID defines them as:

Decentralized identifiers (DIDs) are a new type of identifier to provide verifiable, decentralized digital identity. These new identifiers are designed to enable the controller of a DID to prove control over it and to be implemented independently of any centralized registry, identity provider, or certificate authority.

## Command Line Parameters

### Standard Parameters

|  |  |
| --- | --- |
| Param | Description |
| -h | print help information |
| --help | print help information |
| -o [FILE|DIR] | The explicit filename to save to OR a directory where logically named output will be placed |
| --log [FILE] | Instead of logging to stdout, write to a specified file instead |
| --pages | range/list of pages to be processed (eg. 1-5, 2,4,7). NOTE: page range is normally one-based unless –zero is also used, then it’s zero-based |