International Standard

ISO/IEC 21617-1

First edition
2025-01

Information technology — JPEG Trust —

Part 1: Core foundation

Reference number
ISO/IEC 21617-1:2025(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO 21617 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Current technologies permit the modification or synthetic creation of media assets. Some, like deep learning methods, can create media assets that are hard for people to distinguish from natural media assets. These technologies open new, creative opportunities that are useful for business and research usage. However, these technologies can also lead to issues relating to the use of manipulated media to spread misinformation or disinformation. Misuse of manipulated media can cause social unrest, spread rumours for political gain or encourage hate crimes.

Media modifications are not always negative as they are increasingly a normal and legal component of many production pipelines. However, in many application domains, creators need or want to declare the type of modifications that were performed on the media asset. A lack of such declarations in these situations may reveal the lack of trustworthiness of media assets or the intention to hide the existence of manipulations. To address such problems and attempt to avoid negative impacts, some companies, including social media platforms and news outlets, are developing mechanisms to clearly detect and annotate manipulated media when they are shared.

There is a need to have a standardized way to annotate media assets (regardless of the intent) and securely link the assets and annotations together. This document (JPEG Trust) ensures interoperability between a wide range of applications dealing with media asset creation and modification, providing a set of standard mechanisms to describe and embed information about the creation and modification of media assets.

Furthermore, a key aspect of understanding the trustworthiness of a media asset is the nature of trust itself. No single trust model can accommodate all the expressions of media asset trust in society. This means that the standard requires a flexible architecture for accommodating diverse trust models. Through the mechanism of user-defined trust profiles, this document empowers various communities to define trust models that meet the specific demands of their trust requirements.

This document (JPEG Trust) provides a comprehensive framework for individuals, organizations, and governing institutions interested in establishing an environment of trust for the media that they use, and to support trust in the media they share online. This framework addresses aspects of providing provenance information, extracting and evaluating trust indicators, and handling privacy and security concerns.

Information technology — JPEG Trust —

Part 1:
Core foundation

1 Scope

This document specifies a framework for establishing trust in media. This framework includes aspects of authenticity, provenance and integrity through secure and reliable annotation of the media assets throughout their life cycle.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10918-1, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 15444-1:2024, *Information technology — JPEG 2000 image coding system — Part 1: Core coding system*

ISO/IEC 18181-2:2024, *Information technology — JPEG XL image coding system — Part 2: File format*

ISO/IEC 18477-1, *Information technology — Scalable compression and coding of continuous-tone still images — Part 1: Core coding system specification*

ISO/IEC 18477-3, *Information technology — Scalable compression and coding of continuous-tone still images — Part 3: Box file format*

ISO/IEC 19566-4, *Information technologies — JPEG systems — Part 4: Privacy and security*

ISO/IEC 19566-5:2023, *Information technologies — JPEG systems — Part 5: JPEG universal metadata box format (JUMBF)*

ISO/IEC 19566-6, *Information technologies — JPEG systems — Part 6: JPEG 360*

ISO/IEC 19566-7, *Information technologies — JPEG systems — Part 7: JPEG linked media format (JLINK)*

ISO/IEC 19566-8, *Information technologies — JPEG systems — Part 8: JPEG Snack*

ISO/IEC 21122-1, *Information technology — JPEG XS low-latency lightweight image coding system — Part 1: Core coding system*

IETF RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*, available at: https://www.rfc-editor.org/info/rfc4122

W3C Recommendation JSON-LD 1.1, *A JSON-based Serialization for Linked Data*, available at: https://www.w3.org/TR/json-ld11/

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
actor
actors
human or non-human (hardware or software) that is participating in the media ecosystem.

EXAMPLE Camera (capture device), generation or editing software, cloud service or the person using such tools.

3.2
media asset
digital assets including images, videos, audio or text

3.3
media asset content
portion of a *media asset* (3.2) that represents the actual content, such as the pixel data of an image, along with any additional technical metadata required to understand or render the content (e.g. a colour profile or encoding parameters)

3.4
media asset metadata
portion of a *media asset* (3.2) that represents non-technical information about the media asset or its content, such as location, creator, annotations or IPR information

3.5
natural media asset
sensor acquired media asset

3.6
region of interest
ROI
subset within the *media asset content* (3.3) identified for a particular purpose

EXAMPLE the face portion of a portrait image, an extracted foreground object(s) or scene cuts of a video

3.7
synthetic media asset
synthetically generated media asset
media asset (3.2) generated at least partially by a computer program

3.8
generative AI media asset
media asset (3.2) created by means of artificial intelligence (AI) and machine learning (ML)

3.9
JPEG 1
common image compression data format and means of reference to ISO/IEC 10918-1

3.10
digital master
master media asset as intended by its creator

3.11
manipulated media asset
manipulated media
media asset (3.2) that has been changed with the intention to induce misinterpretation

3.12
media asset original
media asset (3.2) produced by a device or method without any modifications

3.13
media asset provenance
set of information about a *media asset* (3.2) including the trail of modifications starting from an *actor* (3.1)

EXAMPLE Media asset origin.

Note 1 to entry: Modifications that are missing from the asset's provenance are treated the same as an invalid or unverifiable provenance chain.

3.14
media asset source
non-human *actor* (3.1) that created the *media asset original* (3.12)

3.15
modified media asset
media asset (3.2) that has been changed

3.16
assertion
data structure which represents a statement asserted by an *actor* (3.1) concerning the *media asset* (3.2)

Note 1 to entry: This data is a part of the *trust manifest* (3.19).

3.17
claim
digitally signed and tamper-evident data structure that references one or more *assertion* (3.16) by one or more *actors* (3.1), concerning a *media asset* (3.2) and the information necessary to represent the content_binding

Note 1 to entry: If any assertion was redacted, then a declaration to that effect is included. This data is a part of the *trust manifest* (3.19).

3.18
claim signature
digital signature on the *claim* (3.17) using the private key of an *actor* (3.1)

Note 1 to entry: The claim_signature is a part of the *trust manifest* (3.19).

3.19
trust manifest
set of information about the *media asset provenance* (3.13) of a *media asset* (3.3). A trust manifest is part of a *trust record* (3.23)

3.20
trust declaration
specific type of *trust manifest* (3.19) that, when present, is always first in the *trust record* (3.21). It represents the *actor* (3.1) that created the *media asset* (3.2) and contains only mandatory assertions

3.21
trust record
collection of one or more *trust manifest* (3.21) that can either be embedded into a *media asset* (3.2) or be external to its media asset

3.22
trust indicators
information derived from a combination of the *media asset* (3.2) and the *trust record* (3.21) that indicate a level of trustworthiness of a media asset in a given context

3.23
trust credential
the set of *trust indicators* (3.22) that are derived from a *media asset* (3.2) and its *trust record* (3.23)

3.24
trust profile
set of expressions that are used to evaluate each *trust indicators* (3.22) in a given *trust credential* (3.23) to indicate a level of trustworthiness for a given *media asset* (3.3)

3.25
trust report
result of evaluating a *trust credential* (3.23) against a *trust profile* (3.24)

3.26
authentic media asset
media asset (3.2) that is *verifiable* (3.27) or *trustworthy* (3.28) or both

3.27
verifiable
able to be checked

3.28
trustworthy
able to be relied on as being what it is asserted to be

3.29
media asset integrity
lack of corruption of a *media asset* (3.2)

3.30
registration
process of storing information (e.g. media asset, metadata or provenance) about a *media asset* (3.2), separate from the media asset itself

3.31
signer
actor (3.1) who digitally signs a *media asset* (3.2)

3.32
signing
process that establishes the relation between an *actor* (3.1) and a *media asset* (3.2) in a tamper-evident manner

3.33
intellectual property rights
IPR
exclusive right of the *actor* (3.1) to the intellectual work of their creation

3.34
anonymization
process of altering data in a media asset with the aim to protect the privacy of an *actor* (3.1) by obscuring identifiable features

3.35
obfuscation
process of altering data in a media asset with the aim to protect unauthorized access

3.36
JUMBF
universal format to embed any type of metadata in any box-based JPEG file format and means of reference to ISO/IEC 19566-5

4 JPEG Trust framework

4.1 Description

This document describes a framework for establishing trust in media that comply with the JPEG standards developed by ISO/IEC JTC1 SC 29 including ISO/IEC 10918-1 (JPEG 1), ISO/IEC 15444-1 (JPEG 2000), ISO/IEC 18477-1 (JPEG XT), ISO/IEC 18181-2 (JPEG XL), ISO/IEC 21122-1 (JPEG XS), ISO/IEC 19566-6 (JPEG 360), ISO/IEC 19566-7 (JLINK) and ISO/IEC 19566-8 (JPEG Snack)). The core components of the framework are built on the ISO/IEC 19566-5 (JPEG Universal Metadata Box Format, JUMBF), as described in 6.1. Therefore, the core principles can also be applied on other types of media that can embed JUMBF containers. The model for storing and accessing cryptographically verifiable information about a media asset is aligned with the technical aspects of the C2PA architecture.

NOTE For more information about the relationship between this document (JPEG Trust) and C2PA, see Annex B.

4.2 Overview

The JPEG Trust framework establishes that a media asset consists of the media asset content, media asset metadata, and a Trust Record. The Trust Record (4.3) is a tamper-evident unit consisting of a one or more Trust Manifests. The Trust Manifests contain a series of statements, called assertions, that cover areas such as asset creation, creation device details, authorship, edit actions, bindings to content and other information associated with the media asset.

A set of Trust Indicators (4.5) can be derived from the trust record as well as from other metadata or the media asset content. These Trust Indicators are gathered together into a Trust Credential (4.6), which can be used to assess the trustworthiness of a media asset in a given context through the use of a specified Trust Profile (4.7). The result of this evaluation is documented in a Trust Report (4.8).

The framework and its core components are illustrated in Figure 1.

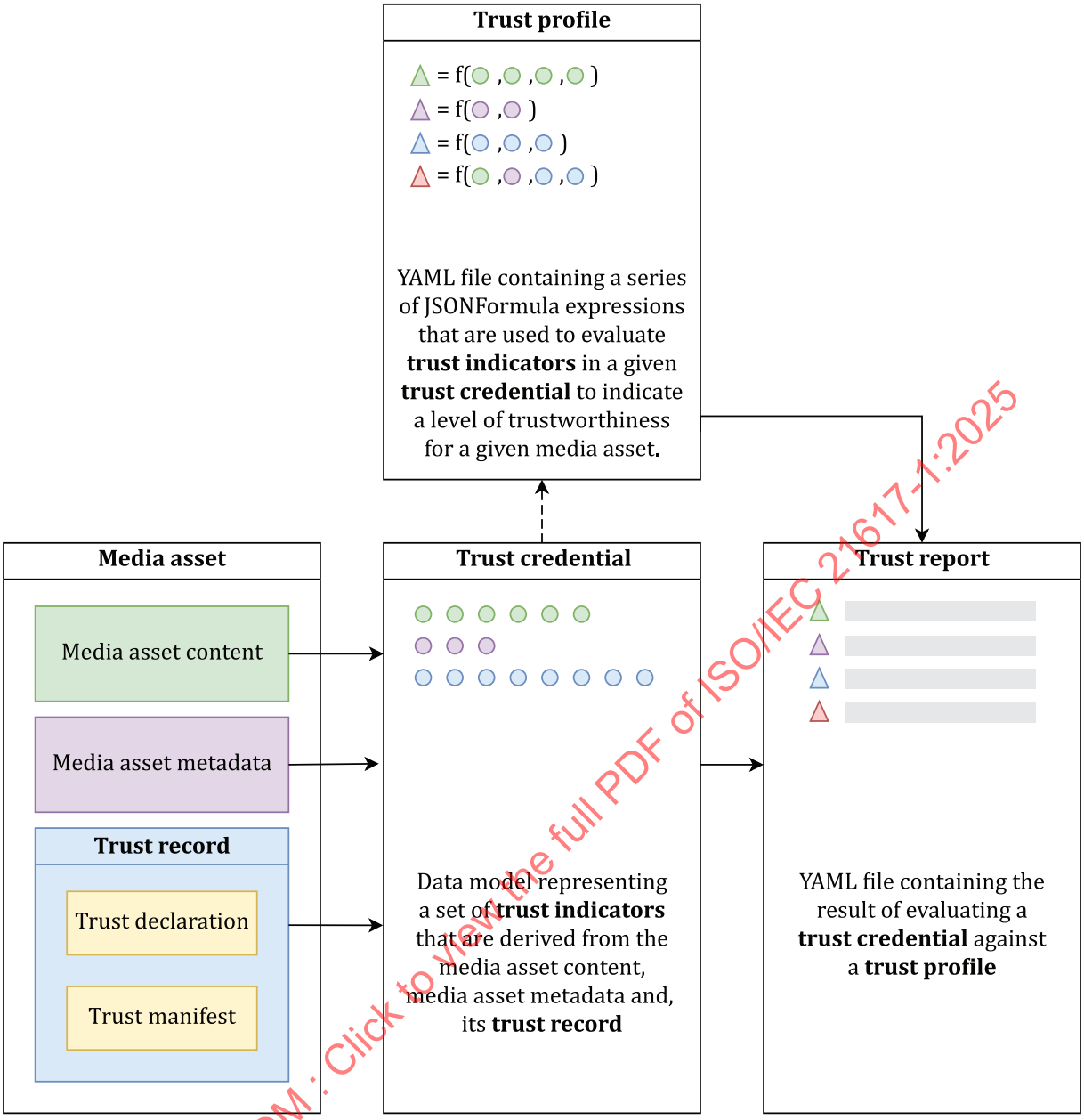

Figure 1 — Trust Framework

4.3 Trust Record

The Trust Record is a JUMBF superbox composed of a series of other JUMBF boxes and superboxes, each identified by their own UUID and label in their JUMBF Description box. The Trust Record Description box shall have a label of `c2pa`, a UUID of `0x63327061-0011-0010-8000-00AA00389B71` (`c2pa`) and shall contain one or more Trust Manifest superboxes (which may be a Trust Manifest or a Trust Declaration). The Trust Record may also contain JUMBF boxes and superboxes whose UUIDs are not defined in this document.

NOTE Allowing other boxes and superboxes enables custom extensions to this document (JPEG Trust) as well as enabling the addition of new boxes in future versions of this document without breaking compatibility.

A Trust Record (see Figure 2) shall contain at least one Trust Manifest. The set of Trust Manifests, as stored in the asset's Trust Record, represents its Media Asset Provenance.

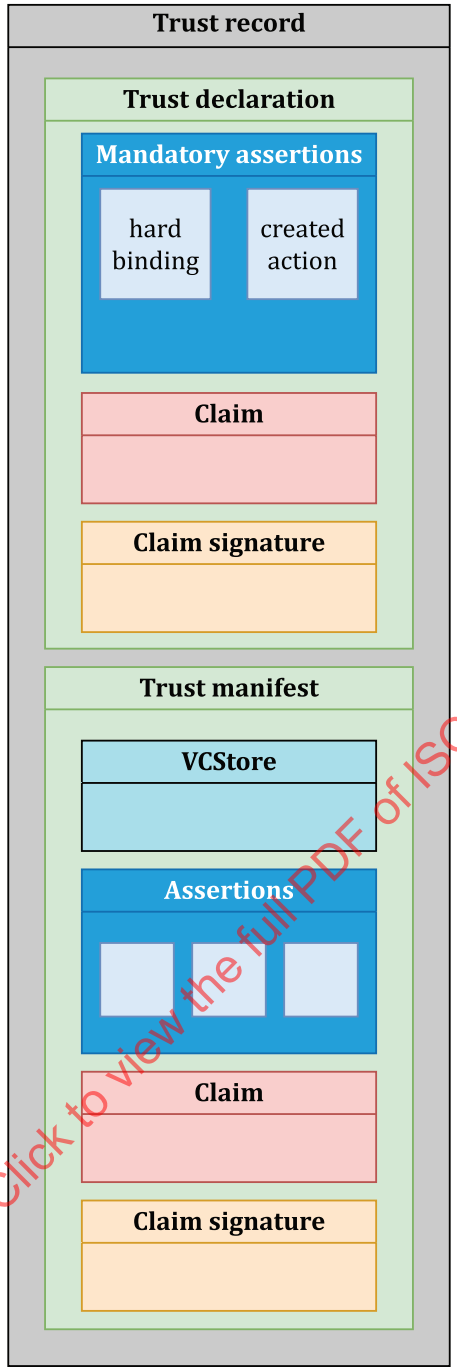

Figure 2 — A Trust Record

4.4 Trust Manifests

4.4.1 General

A Trust Manifest (see Figure 3) is the set of information about the media asset provenance, while a Trust Declaration is a special type of Trust Manifest with only mandatory assertions that always comes first in the Trust Record.

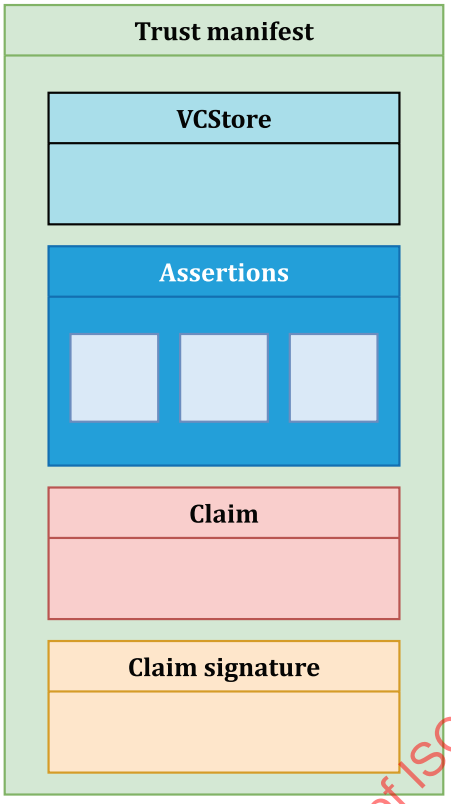

Figure 3 — A Trust Manifest

4.4.2 Components of a Trust Manifest

Each Trust Manifest may contain a Verifiable Credentials (VC) Store, Assertions, a Claim, and a Claim Signature.

Assertions (5.2) are statements about the media asset provenance of a given media asset; such as asset creation, capture device details, authorship, edit actions, and bindings to content. Assertions are wrapped up with additional information into a digitally signed entity called a Claim.

The W3C Verifiable Credentials[14] of individual actors that are involved in the creation of the Assertions can be added to the VCStore of a Trust Record to provide additional Trust Indicators concerning actors.

Together, these Assertions, Claims, Verifiable Credentials and Signatures are all bound together into the Trust Manifest by a hardware or software component called a Claim Generator.

4.4.3 Details of a Trust Manifest

The Trust Manifest is a JUMBF superbox composed of a series of other JUMBF boxes and superboxes, each identified by their own UUID and label in their JUMBF Description box. The UUID for each Trust Manifest shall be either `0x63326D61-0011-0010-8000-00AA00389B71` (`c2ma`), `0x6332636D-0011-0010-8000-00AA00389B71` (`c2cm`), `0x6332756D-0011-0010-8000-00AA00389B71` (`c2um`), or `0x63326D64-0011-0010-8000-00AA00389B71` (`c2md`) depending on the type of manifest. In order to enable uniquely identifying each Trust Manifest, they shall be labelled with a RFC 4122 (UUID) optionally proceeded by an identifier of the claim generator and a `:`.

EXAMPLE A label for the fictitious ACME claim generator might look like `acme:urn:uuid:F9168C5E-CEB2-4FAA-B6BF-329BF39FA1E4`.

NOTE More information about the JUMBF structure of Trust Manifests can be found in the C2PA specification.[6]

4.4.4 Trust Declaration

A Trust Declaration is a specific type of Trust Manifest (which is specified using the `UUID: 0x63326D64-0011-0010-8000-00AA00389B71` (c2md)) that shall only be defined during creation of a media asset and shall only contain a specific set of mandatory assertions, as well as the Claim and Claim Signature. The *Requestable* and *Label Present* toggles shall both be set in the JUMBF Description box of the Trust Manifest's JUMBF superbox.

These mandatory Assertions are:

— exactly one hard binding to content assertion

  — either a c2pa.hash.data, c2pa.hash.boxes, or c2pa.hash.bmff.v2 based on the type of asset and version for which the manifest is destined.

— exactly one actions assertion

  — consisting of a single `c2pa.created` action that specifies the time of creation and a `digitalSourceType` field whose values indicate if it was created by a camera, software tool or generative AI.

NOTE The `digitalSourceType` field would have the value https://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia when the media asset is created from scratch by generative AI. When a generative AI uses existing media assets in addition to newly created ones, then the value of https://cv.iptc.org/newscodes/digitalsourcetype/compositedWithtrainedAlgorithmicMedia is used.

If a specific workflow requires additional assertions, those requirements can be reflected through the use of a Trust Profile (4.7) that would be used to evaluate the Trust Credential (4.6) of the media assets. For example, a workflow might require that a media asset contain the date, time and location when it is created.

4.5 Trust Indicators

Trust Indicators are parameters that may be used to assess the trustworthiness of a media asset in a given context. Trust Indicators are derived from one or more of the following sources: media asset content, the trust record, and the media asset metadata. Each of these is represented as a separate "section" of the Trust Credential, along with any additional indicators that a given workflow may produce. Trust Indicators are expressed in the Trust Credential as specified in 4.6.

NOTE While some Trust Indicators are declared in this document, new Trust Indicators can be defined at any time to suit specific workflows.

4.6 Trust Credential

4.6.1 General

The Trust Credential consists of Trust Indicators extracted from the two portions of the media asset, namely metadata (including both Trust Record and other metadata) and content.

A Trust Credential should be serialized into a JSON-LD object, with fields representing groupings of the Trust Indicators. Additionally, this document describes how to map well known serializations to JSON-LD for the purposes of consistent processing by a validator.

NOTE Other serializations, whether ephemeral or persistent, are possible, but are not specified in this document.

The Trust Credential's JSON-LD serialization shall contain an JSON-LD standard `@context` field (such as Figure 4) whose value shall be either an object or an array listing terms (also known as namespaces) that are used in the Trust Credential. In the case of an object, the terms shall be listed as key-value pairs, where the key is the term name and the value is the URI of the term. In the case of an array, only the URI is required since it shall apply to all terms not otherwise identified by a specific term.

```
{
  "@context": [
    "https://jpeg.org/jpegtrust/"
  ]
}
```

Figure 4 — Example of a JSON-LD `@context` field

Since a `@context` element can appear inside of any object in JSON-LD, it is possible to have custom values, and their associated `@context` elements in multiple places throughout a single JSON-LD document, where the terms are localized to that specific object, such as in Figure 6.

4.6.2 For media asset content

A Trust Credential may contain a field named `content` whose value is an object containing information that represents the Trust Indicators derived from the media asset content. There are currently no pre-defined fields of the `content` object, but workflow-centric Trust Indicators may be added to the `content` object along with their associated `@context`. For example, Figure 5 shows a `content` object that contains information about the image, such as its width and height.

```
"content": {
  "@context": {
    "myimg": "http://example.com/images/",
    "myalgos": "http://example.com/algorithms"
  },
  "myimg:width": 612,
  "myimg:height": 792,
  "myalgos:aigc_probability": 0.95
},
```

Figure 5 — Custom content Trust Indicators

4.6.3 For Trust Manifests

4.6.3.1 General

As described in 4.4, a Trust Manifest consists of a set of Assertions, Claims, and Signatures (and possibly other things) that are bound together into a single entity. There are two types of Trust Manifests: Trust Declarations and standard Trust Manifests. If a Trust Declaration is present, its Trust Indicators shall be represented as fields of the `declaration` object. For any Trust Manifests present, there shall be an object in the `manifests` array, which is the value of the `manifests` field of the Trust Credential. Each of those objects shall contain a list of Trust Indicators derived from the Trust Manifest.

NOTE It is possible to produce a Trust Credential for a media asset that contains neither a Trust Declaration nor any Trust Manifests. This is a valid state and the lack of a Trust Declaration or any Trust Manifests serves as a valid trust indicator (4.5).

4.6.3.2 JSON-LD serialized assertions

For any Assertion which is serialized into JSON-LD that will be incorporated into the Trust Credential, the Trust Indicator shall be described using the same JSON-LD object. An example `stds.iptc` assertion is shown in Figure 6.

```
"stds.iptc": {
  "@context" : {
    "Iptc4xmpExt": "http://iptc.org/std/Iptc4xmpExt/2008-02-29/",
    "dc" : "http://purl.org/dc/elements/1.1/",
  "dc:creator": [ "Julie Smith" ],
  "Iptc4xmpExt:LocationCreated": {
    "Iptc4xmpExt:City": "Beijing, China"
  }
}
```

Figure 6 — A JSON-LD serialized assertion

NOTE The various terms and context namespaces in Figure 6 are defined as part of IPTC Photo Metadata Standard.[12]

4.6.3.3 CBOR serialized assertions

For each Assertion, which is serialized into CBOR, that will be incorporated into the Trust Credential, the Trust Indicator shall be described as the same key name in the CBOR map and its value type shall be determined by Table 1:

Table 1 — Mapping from CBOR to JSON-LD

CBOR Type(s)	JSON-LD Type
integer, unsigned integer	unsigned number
negative integer	integer
byte string	string (base64 encoded)
UTF-8 string	string
array	array
map	object
False, True	boolean
Null	null
half-precision float, single-precision float, double-precision float	float
date-time	string (as defined in ISO 8601-1)

Since CBOR allows map keys of any type, whereas JSON-LD only allows strings as keys in object values, CBOR maps with keys other than UTF-8 strings shall have those keys converted to UTF-8 strings. An example of a CBOR serialized assertion is shown in Figure 7, and its equivalent JSON-LD representation is shown in Figure 8.

```
{
  "actions": [
    {
      "action": "c2pa.cropped",
      "when": 0("2020-02-11T09:30:00Z")
    },
    {
      "action": "c2pa.filtered",
      "when": 0("2020-02-11T09:00:00Z"),
      "actors": [
        {
          "credentials": [
            {
              "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.credentials/Joe_Bloggs",
              "alg": "sha256",
              "hash": b64'hoOspQQ1lFTy/4Tp8Epx670E5QW5NwkNR+2b30KFXug='
            }
          ]
        }
      ],
    },
  ],
  "metadata": {
    "reviewRatings": [
      {
        "value": 1,
        "explanation": "Content bindings did not validate"
      }
    ]
  }
}
```

Figure 7 — CBOR Diagnostics for an Actions assertion

```
{
  "actions": [
    {
      "action": "c2pa.cropped",
      "when": "2020-02-11T09:30:00Z"
    },
    {
      "action": "c2pa.filtered",
      "when": "2020-02-11T09:00:00Z",
      "actors": [
        {
          "credentials": [
            {
              "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.credentials/Joe_Bloggs",
              "alg": "sha256",
              "hash": "hoOspQQ1lFTy/4Tp8Epx670E5QW5NwkNR+2b30KFXug="
            }
          ]
        }
      ],
    },
  ],
  "metadata": {
    "reviewRatings": [
      {
        "value": 1,
        "explanation": "Content bindings did not validate"
      }
    ]
  }
}
```


Figure 8 — JSON-LD representation of Figure 7

4.6.4 Claim

Since a Claim is a special type of Assertion, it shall be serialized in the same manner as a CBOR serialized assertion (4.6.3.3). An example is found in Figure 9.

In addition, the following two fields shall be present in all Claim objects:

`signature_status` A field whose value is a single C2PA status code[6] determined from validation of the signature.

`assertion_status` A field whose value is an object, where each field is the label of the Assertion and its value is the C2PA status code[6] determined from validation.

For the active Trust Manifest, the following additional field shall be present in the Claim object:

`content_status` A field whose value is a single C2PA status code[6] determined from validation of the content bindings.

NOTE This is present only in the active Trust Manifest, since that is the only one that can be used to check the validity of the content bindings.

```
"claim": {
  "alg" : "sha256",
  "claim_generator": "Joe's Photo Editor/2.0 (Windows 10)",
  "claim_generator_info" : [
    {
    "name": "Joe's Photo Editor",
    "version": "2.0",
    "schema.org.SoftwareApplication.operatingSystem": "Windows 10"
    }
  ],
  "signature" : "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.signature",
  "dc:format": "image/jpeg",
  "signature_status": "claimSignature.validated",
  "assertion_status": {
    "c2pa.thumbnail.claim.jpeg": "assertion.hashedURI.match",
    "c2pa.hash.data": "assertion.hashedURI.match",
    "c2pa.actions": "assertion.hashedURI.match",
  },
  "content_status": "assertion.dataHash.match"
},
```

Figure 9 — A JSON-LD serialized claim

4.6.5 For claim signature

The JSON-LD representation of the X.509 certificate from the claim signature is based on a logical mapping of its ASN.1 serialization as defined in IETF RFC 5280 to JSON-LD. An example certificate is in Figure 10. Additional mappings, such as the mapping of the distinguished name (as defined in IETF RFC 4514) to JSON-LD shall also be done in the most logical fashion possible.

NOTE An X.509 certificate can contain all sorts of information, but only the information that is relevant to the trust framework would be included in the JSON-LD representation.

```
"signature": {
  "signature_algorithm": "sha1WithRSAEncryption",
  "subject": {
    "CN": "Mike Smith",
    "MAIL": "mike@smith.com",
    "O" : "Smith Inc.",
    "C" : "US"
  },
  "issuer": {
    "CN": "Alice's Trust Services",
    "O" : "Alice Corporation",
    "C" : "US"
  },
  "validity": {
    "not_before": "2015-12-03T12:19:06",
    "not_after": "2023-12-31T20:17:50.245Z"
  },
}
```

Figure 10 — Representation of an X.509 Certificate

An example of Trust Profile showing how to test some of these values can be found in 4.7.6.4.

4.6.6 For verifiable credentials

Since W3C verifiable credentials, found in a Trust Record, are serialized into JSON-LD, this Trust Indicator, when present, shall be described using the same JSON-LD object and placed into an array which is the value of the `credentials` field of the trust credential. An example is shown in Figure 11.

```
"credentials": [
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "http://schema.org"
  ],
  "type": [
    "VerifiableCredential",
    "NPPACredential"
  ],
  "issuer": "https://nppa.org/",
  "credentialSubject": {
    "id": "did:nppa:eb1bb9934d9896a374c384521410c7f14",
    "name": "John Doe",
    "memberOf": "https://nppa.org/"
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2021-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod":
"did:nppa:eb1bb9934d9896a374c384521410c7f14#_Qq0UL2Fq651Q0Fjd6TvnYE-
faHiOpRlPVQcY_-tA4A",
    "jws": "eyJhbGciOiJQUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19
      DJBMvvFAIC00nSGB6Tn0XKbbF9XrsaJZREWvR2aONYTQQxnyXirtXnlewJMB
      Bn2h9hfcGZrvnC1b6PgWmukzMJ1IiH1dWgnDIS81BH-IxXnPkbuYDeySorc4
      QU9MJxdVkY5EL4HYbcIfwKi6X4LBQ2_ZHZIu1jdqLcRZqHcsDF5KKylKc1TH
      n5VRWy5WhYg_gBnyWny8E6Qkrze53MR7OuAmmNJ1m1nN8SxDrG6a08L78J0-
      Fbas5OjAQz3c17GY8mVuDPOBIOVjMEghBlgl3nOi1ysxbRGhHLEK4s0KKbeR
      ogZdgt1DkQxDFxxn41QWDw_mmMCjs9qxg0zcZzqEJw"
  }
}
],
```

Figure 11 — Representation of a Verifiable Credential

4.6.7 For media asset metadata

To represent various types of media asset metadata, it is necessary to serialize the information into JSON-LD, for inclusion in the Trust Credentials structure. The data, if present, shall be the value of the `metadata` field of the Trust Credential.

Since both Exif[7] and IPTC[12] metadata can be expressed in XMP, it is recommended that these are first incorporated into the XMP[5]. Then the XMP data shall be serialized according to the rules of the JSON-LD serialization of XMP.

The JSON-LD object may also contain other properties, from other metadata schemas contained within the media asset metadata, provided that they are serialized as JSON-LD.

The `@context` property within the JSON-LD object shall be present, as it is required to provide context & namespaces for all used metadata schemas.

An example can be found in Figure 12.

```
"metadata" : {
"@context" : {
"exif": "http://ns.adobe.com/exif/1.0/",
"exifEX": "http://cipa.jp/exif/2.32/",
"tiff": "http://ns.adobe.com/tiff/1.0/",
"Iptc4xmpCore": "http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/",
"Iptc4xmpExt": "http://iptc.org/std/Iptc4xmpExt/2008-02-29/",
},
"Iptc4xmpExt:DigitalSourceType":
"https://cv.iptc.org/newscodes/digitalsourcetype/digitalCapture",
"Iptc4xmpExt:LocationCreated": {
  "Iptc4xmpExt:City": "San Francisco"
},
"Iptc4xmpExt:PersonInImage": [
  "Erika Fictional"
],
"Iptc4xmpCore:AltTextAccessibility": "Photo of Erika Fictional standing
in front of the Golden Gate Bridge at sunset.",
"exif:GPSVersionID": "2.2.0.0",
"exif:GPSLatitude": "39,21.102N",
"exif:GPSLongitude": "74,26.5737W",
"exif:GPSAltitudeRef": 0,
"exif:GPSAltitude": "100963/29890",
"exif:GPSTimeStamp": "2019-09-22T18:22:57Z",
"exif:GPSSpeedRef": "K",
"exif:GPSSpeed": "4009/161323",
"exif:GPSImgDirectionRef": "T",
"exif:GPSImgDirection": "296140/911",
"exif:GPSDestBearingRef": "T",
"exif:GPSDestBearing": "296140/911",
"exif:GPSHPositioningError": "13244/2207",
"exif:ExposureTime": "1/100",
"exif:FNumber": 4.0,
"exif:ColorSpace": 1,
"exif:DigitalZoomRatio": 2.0,
"tiff:Make": "CameraCompany",
"tiff:Model": "Shooter S1",
"exifEX:LensMake": "CameraCompany",
"exifEX:LensModel": "17.0-35.0 mm",
"exifEX:LensSpecification": { "@list": [ 1.55, 4.2, 1.6, 2.4 ] }
}
```

Figure 12 — Example of JSON-LD serialized XMP

NOTE The various terms and context namespaces in Figure 12 are defined as part of IPTC Photo Metadata Standard[12] and Exif[7].

4.6.8 Example Trust Credentials

4.6.8.1 Camera

An example Trust Credential for a JPEG image from a camera might look like Figure 13:

```
{
"@context": [
"http://jpeg.org/jpegtrust/"
],
"declaration" : {
  "claim": {
"alg" : "sha256",
"claim_generator_info" : [
  {
"name": "Joe's Camera App"
  }
],
"signature" : "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.signature",
"dc:format": "image/jpeg",
"signature_status": "claimSignature.validated",
"assertion_status": {
"c2pa.hash.data": "assertion.hashedURI.match",
"c2pa.actions": "assertion.hashedURI.match"
  }
  },
  "assertions": {
"c2pa.hash.data": {
  "alg" : "sha256",
  "pad" : "",
  "hash": "Auxjtmax46cC2N3Y9aFmBO9Jfay8LEwJWzBUtZ0sUM8gA=",
  "name": "JUMBF manifest",
  "exclusions": [
{
  "start": 9960,
  "length": 4213
}
  ]
},
"c2pa.actions":  {
  "actions": [
{
  "action": "c2pa.created",
  "when": "2023-02-11T09:00:00Z",
  "softwareAgent": {
  "name": "Joe's Camera App"
  },
  "digitalSourceType":
"https://cv.iptc.org/newscodes/digitalsourcetype/digitalCapture"
}
  ]
}
  }
},
"manifests": [
  {
"claim": {
  "alg" : "sha256",
  "claim_generator_info" : [
```

```
{
  "name": "Joe's Camera App"
}
  ],
  "signature" : "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.signature",
  "dc:format": "image/jpeg",
  "signature_status": "claimSignature.validated",
  "assertion_status": {
"stds.exif": "assertion.hashedURI.match",
"c2pa.hash.data": "assertion.hashedURI.match",
"stds.iptc": "assertion.hashedURI.match"
  },
  "content_status": "assertion.dataHash.match"
},
"assertions": {
  "stds.exif": {
"@context" : {
  "exif": "http://ns.adobe.com/exif/1.0/"
},
"exif:GPSVersionID": "2.2.0.0",
"exif:GPSLatitude": "39.918522630419034",
"exif:GPSLongitude": "116.39726729279847",
"exif:GPSAltitudeRef": 0,
"exif:GPSAltitude": "100963/29890",
"exif:GPSTimeStamp": "2019-09-22T18:22:57Z",
"exif:GPSSpeedRef": "K",
"exif:GPSSpeed": "4009/161323",
"exif:GPSImgDirectionRef": "T",
"exif:GPSImgDirection": "296140/911",
"exif:GPSDestBearingRef": "T",
"exif:GPSDestBearing": "296140/911",
"exif:GPSHPositioningError": "13244/2207",
"exif:ExposureTime": "1/100",
"exif:FNumber": 4.0,
"exif:ColorSpace": 1,
"exif:DigitalZoomRatio": 2.0
  },
  "stds.iptc": {
"@context" : {
"Iptc4xmpExt": "http://iptc.org/std/Iptc4xmpExt/2008-02-29/",
"dc" : "http://purl.org/dc/elements/1.1/",
  },
  "dc:creator": [ "Julie Smith" ],
  "Iptc4xmpExt:LocationCreated": {
"Iptc4xmpExt:City": "Beijing, China"
  }
  }
}
  }
],
"content": {
}
  }
```


Figure 13 — Example Trust Credential from a camera

4.6.8.2 Generative AI

An example Trust Credential for a JPEG image created by a generative AI might look like Figure 14:

```
{
"@context": [
"http://jpeg.org/jpegtrust/"
],
"declaration" :  {
  "claim": {
"alg" : "sha256",
"claim_generator": "Joe's Photo Editor/2.0 (Windows 10)",
"claim_generator_info" : [
  {
"name": "Joe's Photo Editor",
"version": "2.0",
"schema.org.SoftwareApplication.operatingSystem": "Windows 10"
  }
],
"signature" : "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.signature",
"dc:format": "image/png",
"signature_status": "claimSignature.validated",
"assertion_status": {
"c2pa.hash.data": "assertion.hashedURI.match",
"c2pa.actions": "assertion.hashedURI.match"
  },
  "content_status": "assertion.dataHash.match"
  },
  "assertions": {
"c2pa.hash.data": {
  "alg" : "sha256",
  "pad" : "",
  "hash": "Auxjtmax46cC2N3Y9aFmBO9Jfay8LEwJWZBUtZ0sUM8gA=",
  "name": "JUMBF manifest",
  "exclusions": [
{
  "start": 9960,
  "length": 4213
}
  ]
},
"c2pa.actions":  {
  "actions": [
{
  "action": "c2pa.created",
  "when": "2023-02-11T09:00:00Z",
  "softwareAgent": {
  "name": "Joe's Photo Editor",
  "version": "2.0",
  "schema.org.SoftwareApplication.operatingSystem": "Windows 10"
  },
  "digitalSourceType":
"http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia"
}
  ]
}
  }
},
  "content": {
}
}
```


Figure 14 — Example Trust Credential from a Generative AI

4.6.8.3 No Trust Manifests

An example Trust Credential for a JPEG image without any trust manifests might look like Figure 15:

```
{
"@context": [
"http://jpeg.org/jpegtrust/"
],
"content": {

}
}
```

Figure 15 — Example Trust Credential without any trust manifests

4.7 Trust Profile

4.7.1 General

A Trust Profile enables the generation of a Trust Report from a Trust Credential. A Trust Profile contains a block of information about the Trust Profile and a set of statements that are evaluated against a Trust Credential. Each statement has an expression/formula that takes one or more trust indicators as an input and produces a single output value. The Trust Profile is expressed in YAML[15] and the statement expressions/formulas are expressed as json-formula.[13]

4.7.2 Trust Profile information

The Trust Profile information block provides additional information about the specific Trust Profile. This information is signalled in the Trust Profile as a dictionary named `metadata` that can contain the keys listed in Table 2.

Table 2 — Trust Profile metadata

Key	Value	Type	Mandatory
name	Name of the Trust Profile.	`String`	yes
issuer	Name of the issuer of the Trust Profile.	`String`	yes
date	Date when the Trust Profile was issued.	date (as defined in ISO 8601-1)	yes
version	Version number of the Trust Profile.	`String` (SemVer formatted)	yes
language	Default of the text in the Trust Profile for non-multilingual fields.	BCP-47 language tag	no

When a field is expressed in multiple languages, these multilingual fields use a dictionary with IETF BCP 47 language tags as key and the text in the corresponding language as value as in Figure 16. These types are indicated as `ML_Dict`.

```
field_name:
  en: Text in English.
  es: Text in Spanish.
  fr: Text in French.
```

Figure 16 — Example of multilingual fields

4.7.3 Statements

The metadata is followed by a list of statements represented as dictionaries with predefined fields. Statements can be grouped in sections by inserting YAML document separators (triple-dash separator ---). The first optional statement of a section of grouped statements may be used to specify a section title and an optional section description. These statements are referred to as section statements. Table 3 gives

an overview of the structure of section statements. All other statements are referred to as expression statements.

Table 3 — Structure of section statements

Key	Value	Type	Mandatory
id	Identifier that uniquely identifies the statement/section within the Trust Profile.	`String [a-zA-Z0-9_.-]*`	yes
description	A human readable description of the statement/section that will not be taken over in associated Trust Reports.	`String`	no
title	Section title that will be copied in the Trust Reports.	`String | ML Dict`	yes
report_text	Text that provides additional information about the section and is copied in the Trust Reports.	`String | ML Dict`	no

Table 4 gives an overview of the structure of expression statements.

Table 4 — Structure of expression statements

Key	Value	Type	Mandatory
id	Identifier that uniquely identifies the statement within the Trust Profile.	`String [a-zA-Z0-9_.-]*`	yes
description	A human readable description of the statement that will not be taken over in associated Trust Reports.	`String | ML Dict`	no
expression	Expression/formula to be evaluated against the Trust Credential.	json-formula	yes
report_text	Text that goes in the associated Trust Report to explain the output value of this statement to an end user.	`String | ML Dict`	no

4.7.4 Expressions

Statements contain expressions that take one or more Trust Indicators as an input and produce a single value as an output. The output value can be binary, numeric (floating point), textual (string), or a URI reference which can be internal (JUMBF reference) or external (URL).

4.7.5 Predefined statement IDs

Some statement IDs are defined by this document and are used to signal an expression with a particular purpose. Table 5 gives an overview of these predefined IDs.

Since this document does not associate any meaning with the values of the predefined IDs, the Trust Profile should assign meanings to these values.

Table 5 — Predefined statement IDs

ID	Purpose	Type	Mandatory
jpt.profile_compliance	Expression that produces a binary output that signals the overall compliance of a media asset with this Trust Profile.	`Boolean`	no
jpt.profile_score	Expression that produces a score between 0 and 1 that signals the overall compliance of a media asset with this Trust Profile.	`Float [0…1]`	no
jpt.profile_label	Expression that produces a textual label to indicate the trustworthiness of a media asset according to this Trust Profile.	`String [a-zA-Z0-9_.-]*`	no
jpt.profile_summary	Expression that produces a textual summary that describes the result of this Trust Profile for a given media asset.	`String`	no

4.7.6 Examples

4.7.6.1 Camera

An example Trust Profile for a JPEG image from a camera might look like Figure 17:

```
# Example Trust Profile for evaluating camera_credential.json
---

metadata:
  name: Experimental Camera Profile
  issuer: JPEG Trust Committee
  date: 2023-09-28T13:19:36.705Z
  version: 1.0.0
  language: en

profile_metadata:
  something: value

---
# Section 1

-
  description: Section 1 - General Information
  title: General Information
  report_text: This section provides information about general stuff

-   # check for content modification
  # description provides additional context that does not go into the
report
  id: content
  description: content is unmodified
  expression: >
    @manifest[0].content_status == "assertion.dataHash.match"
  result_text:
    "true":
      en: This content has not been modified
      es: Translation in Spanish.
      zh: Translation in Simplified Chinese.
    "false":
     en: This content has been modified

---
# Section 2 - GPS & Location information

-   # check the GPS location
  id: gps
  description: GPS location approximate to China's
  expression: >
    (manifests[0].assertions."stds.exif"."exif:GPSLatitude" > 20) &&
      (manifests[0].assertions."stds.exif"."exif:GPSLatitude" < 50) &&
      (manifests[0].assertions."stds.exif"."exif:GPSLongitude" > 80) &&
      (manifests[0].assertions."stds.exif"."exif:GPSLongitude" < 120)
  report_text:
    en: The GPS information shows that the image was taken inside of
China
    fr: Translation in French.
    zh: Translation in Simplified Chinese.


-   # check the human entered value for the city
  id: city
  description: name of the city contains "China"
  expression: >
    contains(manifests[0].assertions."stds.iptc"
                                    ."Iptc4xmpExt:LocationCreated"
                                    ."Iptc4xmpExt:City"
            , 'China')
  report_text:
    en: The city is in China
    fr: Translation in French
    zh: Translation in Chinese
    jp: Translation in Japanese
```

Figure 17 — Example Trust Profile for a camera

4.7.6.2 Generative AI

An example Trust Profile for a JPEG image created by generative AI might look like Figure 18:

```
# Example Trust Profile for evaluating genai_credential.json
---

metadata:
  name: Experimental Generative AI Profile
  issuer: JPEG Trust Committee
  date: 2023-10-31T00:16:40.346Z
  version: 1.0.0
  language: en

profile_metadata:
  something: value

---
# Section 1

-
  description: Section 1 - General Information
  title: General Information
  report_text: This section provides information about general stuff

-   # check for content modification
  # description provides additional context that does not go into the
report
  id: content
  description: content is unmodified
  expression: >
      @declaration.content_status == "assertion.dataHash.match"
  result_text:
    "true":
      en: This content has not been modified
      es: Translation in Spanish
      zh: Translation in Simplified Chinese
    "false":
      en: This content has been modified

---
# Section 2 - Is produced by Generative AI (AIGC)?

-   # Is Generative AI?
    # Checks for 'trainedAlgorithmicMedia' which includes both regular
and composited flavors.
  id: aigc
  description: Is AIGC?
  expression: >
    contains(declaration.assertions."c2pa.actions".actions[0].digitalSour
ceType,
             'trainedAlgorithmicMedia')
  result_text:
```

```
    "true":
      en: This media asset was produced by generative AI
      de: Translation in German
      zh: Translation in Simplified Chinese
    "false":
      en: This media asset was not produced by generative AI

-   # See if the asset has been modified after creation
  id: declaration_only
  description: is there a declaration and no manifests present?
  expression:  contains(keys(@), 'declaration') && !contains(keys(@),
'manifest')
  result_text:
    "true":
      en: No modifications took place after it was created
    "false":
      en: This media asset was modified after creation, but with full
provenance

---
# Section 3 - Compliance

- # check compliance
  id: jpt:profile_compliance
  description: is the asset compliant with this profile?
  expression: >
    @profile.aigc && @profile.declaration_only
  result_text:
    "true":
      en: This media asset is compliant with this profile.
    "false":
      en: This media asset is not compliant with this profile.
```

Figure 18 — Example Trust Profile for generative AI

4.7.6.3 No manifests

An example Trust Profile for a JPEG image without any trust manifests might look like Figure 19:

```
# Example Trust Profile for evaluating no_manifests_credential.json
---

metadata:
  name: Experimental No Manifests Profile
  issuer: JPEG Trust Committee
  date: 2023-11-02T01:03:16.443Z
  version: 1.0.0
  language: en

profile_metadata:
  something: value

---
# Section 1

-
  description: Section 1 - General Information
  title: General Information
  report_text: This section provides information about general stuff

-   # check for manifests
  id: manifests
  description: does the asset contain any manifests
  expression: not_null(@.manifest) || not_null(@.declaration)
  result_text:
    "true":
      en: This media asset has one or more trust manifests
      de: Translation in German
      zh: Translation in Simplified Chinese
    "false":
      en: No trust manifests found in this media asset

- # check compliance
  id: jpt:profile_compliance
  description: is the asset compliant with this profile?
  expression: >
    @profile.manifests
  result_text:
    "true":
      en: This media asset is compliant with this profile.
    "false":
      en: This media asset is not compliant with this profile.
```

Figure 19 — Example Trust Profile without any trust manifests

4.7.6.4 Certificate validation

An example Trust Profile to checking the information about the X.509 certificate from the Claim Signature might look like Figure 20:

```
# Example Trust Profile for evaluating no_manifests_credential.json
---

metadata:
  name: Experimental Signature Validation Profile
  issuer: JPEG Trust Committee
  date: 2023-09-28T13:19:36.705Z
  version: 1.0.0
  language: en

profile_metadata:
  something: value

---
# Section 1

-
  description: Section 1 - Date checks
  title: Validity checking based on dates
  report_text: This section checks the validity dates

-   # check for `not_before`
  id: not_before
  description: check the current date is after `not_before`
  expression: datetime(signature.validity.not_before) < now()
  result_text:
    "true":
      en: The `not_before` date is prior to right now
      fr: Translation in French
      zh: Translation in Simplified Chinese
    "false":
      en: The certificate is not yet valid

-   # check for `not_after`
  id: not_after
  description: check the current date is before `not_after`
  expression: datetime(signature.validity.not_after) > now()
  result_text:
    "true":
      en: The certificate is currently valid
    "false":
      en: The certificate is no longer valid
      fr: Translation in French
      zh: Translation in Simplified Chinese

---
```

```
# Section 2

-
  description: Section 2 - Check the CA/issuer
  title: Make sure the issuer is on our Trust List
  report_text: This section checks for valid issuers
-   # check the issuer against known ones
  id: issuer
  description: see if the issuer is on our trust list
  expression: >
    (signature.issuer.CN="Alice's Trust Services" &&
signature.issuer.C="US")
    || (signature.issuer.CN="ЦСК НБУ" && signature.issuer.C="UA")
  result_text:
    "true":
      en: The issuer is trusted
      fr: Translation in French
      zh: Translation in Simplified Chinese
    "false":
      en: The issuer is not trusted
```

Figure 20 — Example Trust Profile for an X.509 certificate

4.8 Trust Report

4.8.1 General

A Trust Report is produced from the combination of a Trust Profile and a Trust Credential, thereby documenting the result of evaluating the Trust Credential against the Trust Profile. The evaluation helps to indicate a level of trustworthiness for a given media asset. A Trust Indicator presented in the Trust Credential (extracted from the media asset) passes the evaluations if it satisfies the Trust Indicator requirement listed in the Trust Profile.

A new Trust Manifest can be generated based on the Trust Report. The new Trust Manifest can be added to update the Trust Record of the media asset to further enhance the trustworthiness of the media asset.

A Trust Report contains a block of information, which is extracted from and identifies the associated Trust Profile, and a list of statements that are the results of evaluating the statements in the associated Trust Profile against a given media asset's Trust Credential.

The structure of statements in the Trust Report are provided in Table 6.

Table 6 — Structure of Trust Report statements

Key	Value	Type	Mandatory
id	Identifier of the specific statement in the associated Trust Profile.	`String` [a-zA-Z0-9_.-]*	yes
title	Title of section statements.	`String`	no
report_text	Text generated by the associated statement in the Trust Profile.	`String` | `ML Dict`	yes
value	The raw output value of the expression of the associated statement in the Trust Profile.	`Boolean` | `Float` `[0…1]`, `String`	yes (except for section statements)

4.8.2 Examples

4.8.2.1 Camera

An example Trust Report, which is the result of processing the example Trust Credential (Figure 13) for the JPEG image from a camera, using the example Trust Profile (Figure 17) might look like Figure 21:

```
metadata:
  name: Experimental Camera Profile
  issuer: JPEG Trust Committee
  date: 2023-09-28T13:19:36.705Z
  version: 1.1.0
  language: en  # this report has been generated in English
  # other online or offline available signed copies of this report in
other languages
  related_reports: # alternative languages, previously generated reports,
alternate versions, ...
    fr: https://jpeg.org/trustreports/exp-fr.yaml
    es: self#jumbf=exp-es.yaml

---
# Section 1

# section info, English strings extracted, except for description
-
  title: Example section title
  report_text: This is an example of a section description.

-
  # 'jpegtrust.compliance' is a reserved ID that can (optionally) be used
to signal the result of an overall binary profile compliance test
  id: jpt.profile_compliance
  report_text: This media asset passed the compliance test of this
profile.
  value: True

-
  # 'jpegtrust.score' is a reserved ID that can (optionally) be used to
signal the result of an overall binary profile compliance test
  id: jpt.profile_score
  report_text: This media asset scored a 0.8 compliance score according
to this profile.
  value: 0.8

-
  id: content
  # Need for multilingual reports probably exists, maybe we should
optionally allow the following (in addition/alternative to the references
as suggested above):
  report_text:
    en: This content has not been modified
    es: Translation in Spanish
  value: False
```

Figure 21 — Example Trust Report for a camera

4.8.2.2 Generative AI

An example Trust Report, which is the result of processing the example Trust Credential (Figure 14) for the JPEG image created by generative AI, using the example Trust Profile (Figure 18) might look like Figure 22:

```
metadata:
  name: Experimental Generative AI Profile
  issuer: JPEG Trust Committee
  date: 2023-10-31T20:23:30.668Z
  version: 1.1.0
  language: en  # this report has been generated in English

---
# Section 1

# section info, English strings extracted, except for description
-
  title: Example section title
  report_text: This is an example of a section description.

-
  # 'jpt.profile_compliance' is a reserved ID that can (optionally) be
used to signal the result of an overall binary profile compliance test
  id: jpt.profile_compliance
  report_text: This media asset is compliant with this profile.
  value: True

-
  # Need for multilingual reports probably exists, maybe we should
optionally allow the following (in addition/alternative to the references
as suggested above):
  id: content
  report_text:
    en: This content has not been modified
    es: Translation in Spanish
  value: True

-
  # Is Generative AI?
  id: aigc
  report_text:
      en: This media asset was produced by generative AI
      de: Translation in German
      zh: Translation in Simplified Chinese
  value: True

-
  # Any modifications?
  id: declaration_only
  report_text:
      en: No modifications took place after it was created
  value: True
```

Figure 22 — Example Trust Report for generative AI

4.8.2.3 No manifests

An example Trust Report, which is the result of processing the example Trust Credential (Figure 15) for the JPEG image without any Trust Manifests, using the example Trust Profile (Figure 19) might look like Figure 23:

```
metadata:
  name: Experimental No Manifests Profile
  issuer: JPEG Trust Committee
  date: 2023-09-28T13:19:36.705Z
  version: 1.1.0
  language: en  # this report has been generated in English

---
# Section 1

# section info, English strings extracted, except for description
-
  title: Example section title
  report_text: This is an example of a section description.

-
  # 'jpt.profile_compliance' is a reserved ID that can (optionally) be
used to signal the result of an overall binary profile compliance test
  id: jpt.profile_compliance
  report_text: This media asset is not compliant with this profile.
  value: False

-
  id: manifests
  report_text:
    en: No trust manifests found in this media asset
  value: False
```

Figure 23 — Example Trust Report without any Trust Manifests

4.8.3 Trust Report generation procedure

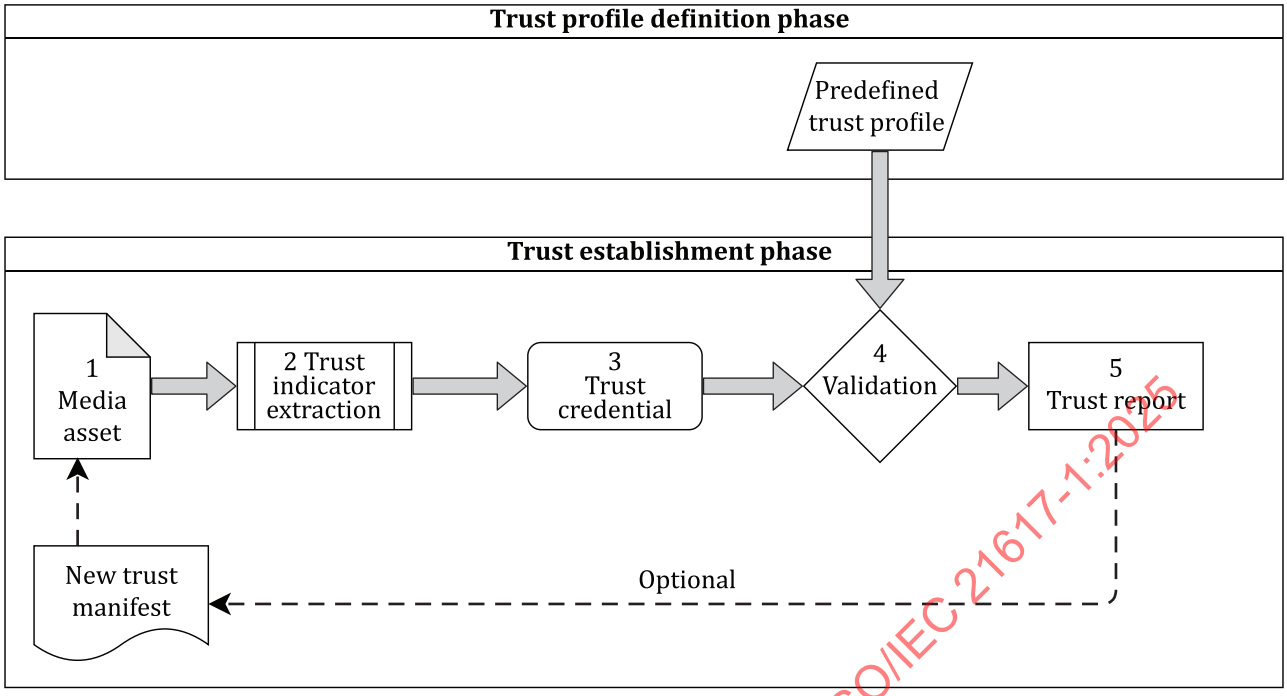

Figure 24 — Trust Report Generation Procedure

The Trust Report generation procedure is illustrated in Figure 24, and the detailed procedure is as follows:

— The Trust Profile definition phase involves the creation of one or more Trust Profiles, which may be created by governments, media platforms, users, etc. It can happen at any time prior to the trust establishment phase;

— The trust establishment phase starts with media assets, consisting of the media asset content and metadata (including both Trust Record and media asset metadata);

— The media asset (1) is passed to an extraction algorithm (2), which extracts Trust Indicators (4.5) from the metadata (including both Trust Record and other metadata) and content of the media asset;

— The extracted Trust Indicators are then used to generate the Trust Credential (3) for this media asset;

— Next, the Trust Credential is sent to the verification process (4) along with the predefined Trust Profile. Using that Trust Profile, the verification algorithm checks whether Trust Indicators in the Trust Credential satisfy the Trust Indicators' requirements in the predefined Trust Profile;

— The verification algorithm then generates a Trust Report (5), which documents the result of the evaluation of a Trust Credential against a Trust Profile;

— Afterwards, a new Trust Manifest may be generated based on the Trust Report. The Trust Record in the media asset would then be updated by adding in the new Trust Manifest.

5 Media asset life cycle annotations

5.1 Overview

This document establishes media asset life cycle annotations to facilitate the production and distribution and consumption of media assets in a trustworthy manner. This is based on the use of JUMBF in which to serialize the various annotations (called assertions) into many different media asset formats. The use of

these annotations is expected to improve trust in media consumption by expressing the provenance of a media asset throughout its lifecycle while ensuring media integrity and authenticity.

Reliable association of the media lifecycle annotations with the corresponding media asset is essential for ensuring that no information is tampered with. This enables detection capabilities in the events of alteration of the media asset. The document specifies that to achieve a wide adoption of such an annotation ecosystem, interoperability is essential.

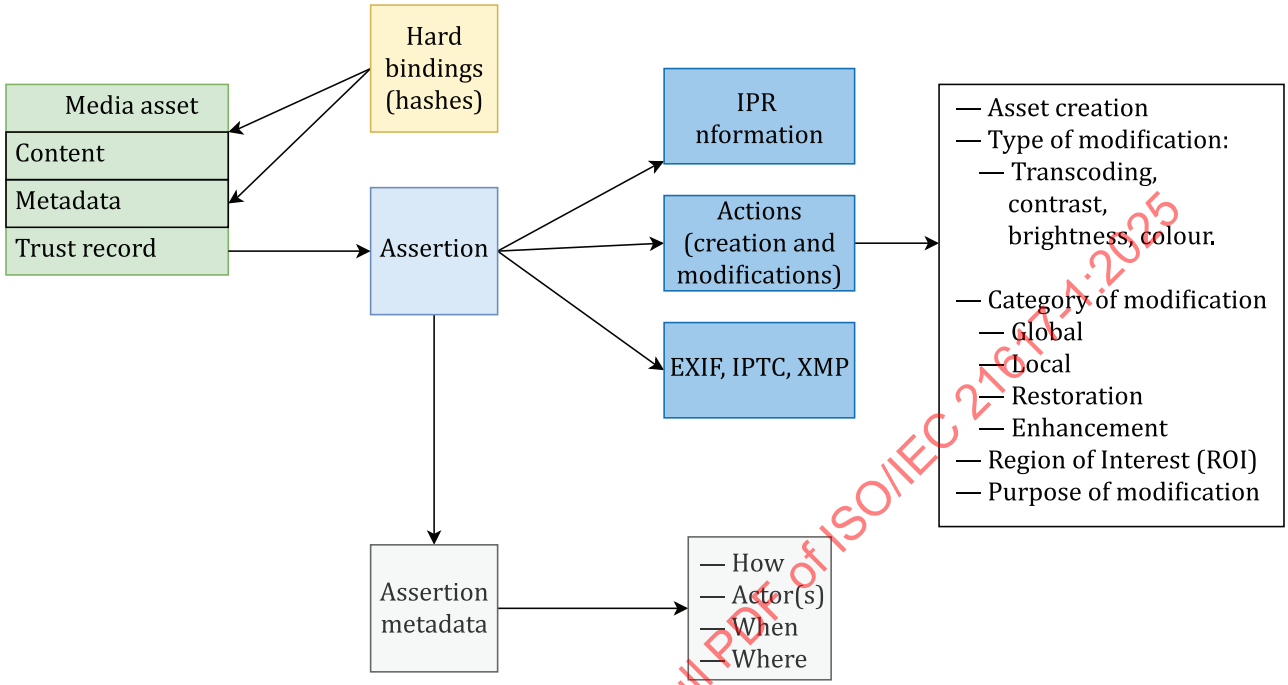

Figure 25 — JPEG Trust Assertions Diagram

In addition, it identifies that media assets consist of at least two parts — content and metadata. Figure 25 provides an overview of the media life cycle annotations, represented in the trust record aspects of the media asset metadata.

5.2 Assertions

5.2.1 Description

In this framework, an annotation for representing information about the provenance of a media asset is called an assertion. An assertion is labelled data representing a statement made by an actor about a media asset. Each of the actors in the system that creates or processes an asset should produce one or more assertions about when, where, and how the asset originated or was transformed. Actors may be human, thus adding human-generated information (i.e. copyright) or machines (software/hardware) providing the information they generated (i.e. camera type or lens details). An assertion may also include information about the actors themselves.

5.2.2 IPR information

5.2.2.1 General

Intellectual Property Right (IPR) information is aimed at any actor who may wish to:

— share their media asset through web services or other means; or

— find if a media asset is being used unlawfully;

— find if a media asset can be used lawfully.

While there might be multiple levels of information that can be included in this assertion, the following basic information is proposed as a minimum:

— ownership of the media asset by single or multiple entities;

— genre or type of media asset, for example, artistic, style, journalistic, product or other genre(s).

In specific cases where assets are created through digitization of physical assets, clarifications are expected about the copyright ownership whether and how the ownerships are shared among the original owner of the asset and the new owner of the asset.

5.2.2.2 Creative work

Depending on the type of information that is desired to be represented, either a Creative Work assertion[19] or a IPTC Photo and Video Metadata assertion[19] can be used to represent IPR information.

An example Creative Work assertion for a media asset might look like Figure 26:

```
{
  "@context": [
    "http://schema.org/",
    {
      "credential": null
    }
  ],
  "@type": "CreativeWork",
  "datePublished": "2021-05-20T23:02:36+00:00",
  "publisher": {
    "name": "Example Corp. News",
    "publishingPrinciples": "https://www.example.com/news/publishing-
principles",
    "logo": "https://www.example.com/news/images/example_news_logo.png",
    "parentOrganization": {
      "name": "Example Corp",
      "legalName": "The Example Corporation"
    }
  },
  "url": "https://www.example.com/news/stories/22107423-f7ba-4c4f-a015-
d952ae7e3e59",
  "identifier": "22107423-f7ba-4c4f-a015-d952ae7e3e59",
  "producer": {
    "name": "Joe Bloggs",
    "credential": [
      {
        "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.credentials/Joe_Bloggs",
        "alg": "sha256",
        "hash": "Auxjtmax46cC2N3Y9aFmBO9Jfay8LEwJWzBUtZ0sUM8gA"
      }
    ]
  },
  "copyrightHolder": {
    "name": "Example Corp. News",
    "legalName": "Example Corporation News"
  },
  "copyrightYear": 2023,
  "copyrightNotice": "Copyright © 2023 Example Corp. News"
}
```


Figure 26 — A Creative Work assertion

5.2.2.3 Training and data mining

An actor may wish to provide information about whether the media asset may be used as part of a data mining or AI/ML training workflow. For example, the actor may wish to assert:

— whether media asset's use is allowed or not allowed for training other machine learning models;

— whether media asset's use is allowed but in a constrained manner where permission is not unconditionally granted for this usage. In this case consumers wish to contact the right holder;

— whether AI generated media asset used only copyright free media assets, obtained all necessary usage permission or a combination of both during the training of the machine learning models that was used to produce the current media asset.

For the first two items in the above-mentioned list, this assertion follows the C2PA Training and Data Mining assertion[19] structure.

5.2.3 Using existing metadata standards

5.2.3.1 Exif

The Exif Information assertion[19] can be used to ensure that Exif[7] information, for example about the capture device or the location where an asset was created, is added to the asset in a way that can be validated cryptographically.

5.2.3.2 IPTC

Additionally, the `stds.iptc` assertion[19] can be used to provided addition location information (such as IPTC location information) in addition to IPR information such as administrative and rights information.

5.2.4 Actions

5.2.4.1 Description

This assertion records an array of actions that are carried out during the creation or subsequent modification of the media asset. While the creation refers to the activity relating to the media asset source, a modification involves any type of change to the media asset including the media asset content and media asset content.

5.2.4.2 Type of modification

Each action element within the Actions assertion provides information on the type of modification performed on a media asset including but limited to transcoding, contrast, brightness, colour temperature, adding annotations, etc. The current version of the Actions assertion follows the structure defined by C2PA Actions.[6]

5.2.4.3 Region of interest (ROI)

For each action, it is possible to identify a region of interest (ROI) within the media asset. This region can either be defined spatially (e.g. a bounding box) or temporally (e.g. time). The ROI could be used for identifying a modification region, target region for privacy perseverance or any other reasons. The details of how this is represented is defined by the C2PA region of interest (ROI) metadata.[6]

5.2.4.4 Purpose of modification(s)

Each action may also provide a `reason` field that provides a free text description of the purpose of the modification. This field is optional and may be used to provide additional information about the modification.

The 'reason' field is explicitly defined in the `c2pa.actions` schema.[6]

5.2.4.5 How was it created

Since the actor performing a given action could be either a human or a machine, it is important to provide information about how the action was performed and what additional tools were used in the process. The information may contain (not limited to):

— a camera capture original media asset;

— a partly modified camera capture media asset;

— a composite of two or more media assets;

— a digitized version of physical media asset;

— a modified version of digitized media asset;

— a synthetically generated media asset including drawing;

— a generative AI media asset (using machine learning models, such as generative AI);

— a combination of some or all of the other items in this list.

This information is recorded using the IPTC Digital Source Type.[12]

5.2.4.6 Category of Modifications

The category of modification can be determined by examining the values of the Actions assertions, such as the `action` itself. Figure 27 shows a test that can be added to a Trust Profile to determine if the image was resized in some way.

```
-   # Resize type of action?
    # Checks the value of each action for one of the possible values
  id: resized
  description: Was the image resized or re-oriented
  expression: >
    contains(actions[*].action, "c2pa.cropped") ||
    contains(actions[*].action, "c2pa.resized") ||
    contains(actions[*].action, "c2pa.orientation")

  result_text:
    "true":
      en: This media asset was resized or re-oriented
      de: Translation in German
      es: Translation in Spanish
      jp: Translation in Japanese
    "false":
      en: This media asset did not change size or orientation
      zn: Translation in Simplified Chinese
```

Figure 27 — Trust Profile fragment for testing resize actions

5.2.5 Bindings (hashes)

5.2.5.1 General

In order to establish the integrity of the media asset content and media asset metadata, a series of assertions existing to establish both hard and soft bindings between the trust record and the media asset.

5.2.5.2 Hard bindings

The hard binding ensures integrity through established hashing algorithms, as further explained in 8.2, and is one of the mandatory assertions in any Trust Manifest and Trust Declaration. This assertion follows the C2PA Hard Binding structure with relevant hashing algorithms.

5.2.5.3 Soft bindings

Soft bindings refer to verifying content integrity even when some bits are changed due to processing of the media such as format changes or enhancements. This is typically managed by perceptual hashing or techniques such as digital fingerprinting and digital watermarking. The soft binding assertion[6] provides a list of predefined algorithms that can be used to generate such soft binding.

5.3 Assertion metadata

5.3.1 General

In various scenarios it is necessary or useful to provide additional information about an assertion, such as the actors, when (date and time) it was generated or other data that may help users to make informed decisions about the provenance or veracity of the assertion data.

5.3.2 Actors

An actor object references a particular person or organization. A valid actor object shall have either:

— an identifier field containing an identifier, or

— a credentials or credential field containing at least one hashed_uri to a W3C Verifiable Credential in the Credential Store that is associated with the actor, or

— both (in which case, the id field in the credentialSubject W3C Verifiable Credential object should match the identifier field of the actor).

5.3.3 When (date and time)

An ISO 8601-1 formatted date and time string may be included as the value of the `dateTime` field of the C2PA assertion metadata assertion.[6]

5.3.4 Extent of modification(s)

5.3.4.1 Description

This assertion is used to provide a means to signal the extent of modifications of this asset compared to a reference version of the asset by providing one or more of the following objective similarity metrics:

— Root mean square error (RMSE);

— Peak signal-to-noise ratio (PSNR).

NOTE Additional metrics such as SSIM,[17] FSIM[18] and UIQ[16] can also be used.

An extent of modification assertion shall have a label of `jpt.mod-extent`.

5.3.4.2 Metadata structure

The extent of modification is expressed in this CBOR-serialized assertion by signalling the existence of an objective metric, type of objective metric and an associated value. Table 7 gives an overview of the fields of the `mod-extent-map`.

Table 7 — Extent of Modification fields

ID	Purpose	Type	Mandatory
`compliance`	A binary output that signals the compliance/existence of an objective metric indicating extent of modification.	`Boolean`	no
`metric`	Expression that provides the information about the objective metric used to express the extent of modification.	`String`	no
`score`	Expression that produces a score that is produced by the objective metric.	`Float`	no

5.3.4.3 Schema

The schema for this type is defined by the `mod-extent-map` rule in Figure 28:

```
mod-extent-map = {
  ? "compliance" : bool,
  ? "metric" : tstr .size (1..max-tstr-length),
  ? "score" : float
}
```

Figure 28

6 Embedding and referencing

6.1 Use of JUMBF

In order to support many of the requirements of JPEG Trust, Trust Manifests need to be stored (serialized) into a structured binary data store that enables some specific functionality including:

— ability to store multiple manifests (e.g. parents and ingredients) in a single container;

— ability to refer to individual elements (both within and across manifests) via URIs;

— ability to clearly identify the parts of an element to be hashed;

— ability to store pre-defined data types (e.g. JSON and CBOR);

— ability to store arbitrary data formats (e.g. XML, JPEG).

In addition to supporting all of the requirements in the list above, the chosen container format (JUMBF, ISO/IEC 19566-5) is also natively supported by the JPEG standards developed by ISO/IEC JTC1 SC 29 and is compatible with the box-based model (ISO/IEC 14496-12) used by many common image and video file formats. Using JUMBF enables all the same benefits of other native box-based formats but with a few additional benefits, such as URI References, in addition to being able to work with the various JPEG standards as well as other formats.

NOTE 1 The JUMBF serialized format can be used when the Trust Records are stored separately from the asset, such as in a separate file or URI location.

A Trust Manifest consumer shall never process an assertion, assertion store, claim, claim signature or Trust Manifest that is not contained inside of a Trust Record. Additionally, when a Trust Manifest Consumer encounters a JUMBF box or superbox whose UUID it does not recognize, it shall skip over (and ignore) its contents.

NOTE 2 This means that the Trust Manifest Consumer can process private boxes that it knows about, but ignore ones of which it is unaware.

6.2 Embedding manifests into JPEG assets

6.2.1 Embedding manifests into JPEG 1 and JPEG XT

The Trust Record shall be embedded as the data contained in an APP11 Marker as defined in ISO/IEC 18477-3.

Since a single marker segment in JPEG 1 cannot be larger than 64K bytes, it is likely that multiple APP11 segments will be required, and they shall be constructed as per ISO/IEC 10918-1 and ISO/IEC 19566-5:2023, D.2. When writing multiple segments, they shall be written in sequential order, and they shall be contiguous (i.e. one segment immediately following the next).

NOTE The previous requirement is specific to JPEG Trust, as JPEG 1 allows the segments to be written in any order and does not require them to be contiguous.

6.2.2 Embedding manifests into JPEG XL

As described in ISO/IEC 18181-2:2024, Clause 4, JPEG XL supports two different formats for the data. It may use a box structure that is compatible with JPEG 2000 and JPEG XS or it may be a direct JPEG XL codestream without the box structure. A JPEG XL file that uses the box structure shall contain at most one JUMBF (`jumb`) superbox (ISO/IEC 18181-2:2024, 9.4) containing a Trust Record Description box (4.3), which contains the Trust Record as described in C2PA Technical Specification.[6] A JPEG XL file that is only a codestream is unable to include an embedded Trust Record.

Figure 29 shows an example of a Trust Manifest embedded in a JPEG XL image.

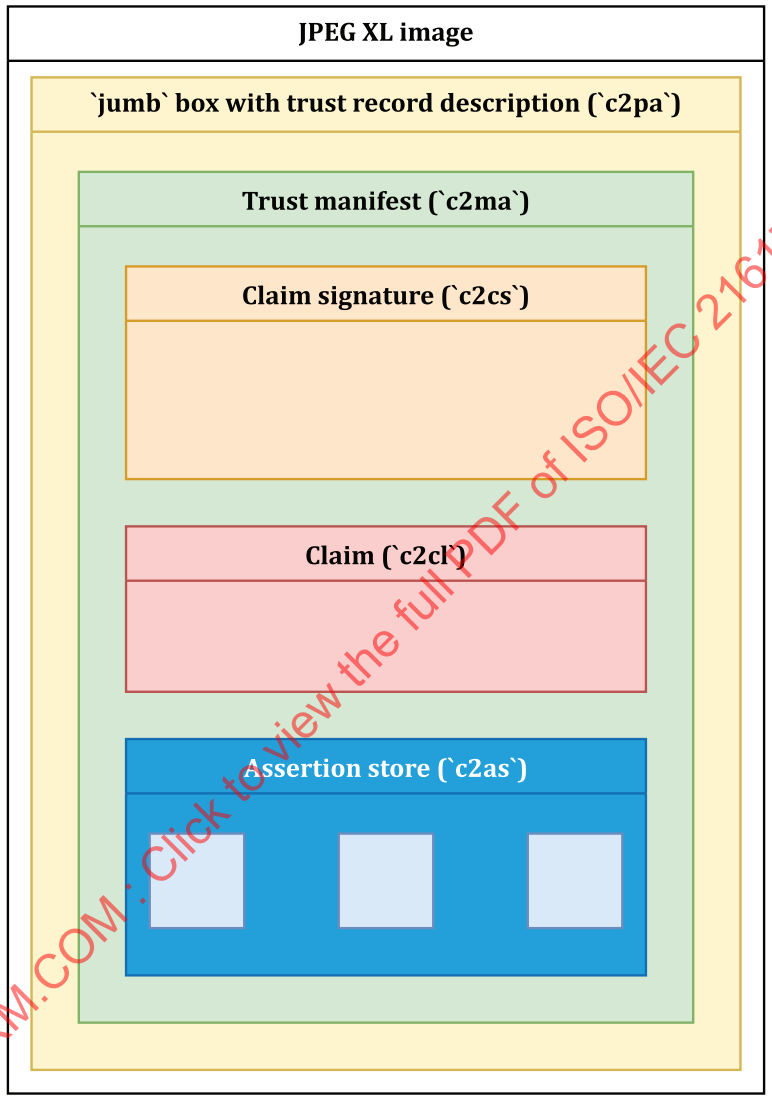

Figure 29 — A Trust Manifest embedded in a JPEG XL image

6.2.3 Embedding manifests into JPEG 2000

The JPEG 2000 file format, as codified in ISO/IEC 15444-1:2024, Annex I, provides a box-based container format for storing data similar to ISO/IEC 14496-12 (ISO BMFF). The Trust Record shall be embedded into a JPEG 2000 file (i.e. J2C, JP2 or JPX) as the data of a box whose TBox shall have a value of `jumb` as described in 6.2.2 and in ISO/IEC 18181-2:2024, 9.3.

The `jumb` box may appear anywhere in the file, though it is recommended to put it earlier in the file to improve retrieval. There shall be only one `jumb` box, containing a Trust Record Description box (4.3), in a JPEG 2000 file.

6.2.4 Embedding manifests into JPEG XS

As JPEG XS uses the same box format as JPEG 2000, the same approach as described in 6.2.3 shall be used.

The `jumb` box may appear anywhere in the file, though it is recommended to put it earlier in the file to improve retrieval. There shall be only one `jumb` box, containing a Trust Record Description box (4.3), in a JPEG XS file.

6.3 Embedding manifests into other asset types

JUMBF can be embedded into assets that are not JPEG, including other raster image formats, vector formats, audio and video formats and more. Details can be found in C2PA Technical Specification.[6]

6.4 External manifests

In some cases, it is not possible (or practical) to embed a Trust Record in an asset. In those cases, keeping the Manifests external to the asset is an acceptable model for providing providence to assets. The manifest should be stored in a location, referred to as a manifest repository, that is easily locatable by a manifest consumer working with the asset, such as by reference or URI. As the Trust Record is a JUMBF box, it shall be served with the JUMBF Media Type, `application/c2pa`.

Some common reasons to use an external manifest are:

— It is not practical, such as when the size of the Trust Record is larger than the asset's digital content;

— It is not appropriate, such as when it would modify an asset that should not be modified.

NOTE An example of this is creating a manifest for a pre-existing asset.

6.5 Embedding a reference to the active manifest

If the asset has embedded XMP (ISO 16684-1), it is recommended that the claim generator add a `dcterms:provenance` key to the XMP, the value (a URI reference) being where to locate the Active Manifest. The URI shall either be a JUMBF URI for an embedded Manifest or a standard URI for any non-embedded scenarios, whether stored remotely or locally on the same storage system as the asset itself.

An example JUMBF URI would be stored as shown in Figure 30:

```
<dcterms:provenance>
self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF39FA1E4
</dcterms:provenance>
```

Figure 30 — Example JUMBF URI

7 Identification of actors

7.1 Identity and actors

7.1.1 Verifiable credentials

7.1.1.1 General

Actors may wish to provide their W3C Verifiable Credentials to the claim generator as part of establishing provenance for an asset. Actors may be individuals, groups or organizations.

W3C Verifiable Credentials, as specified in W3C Verifiable Credentials Data Model, are used in this document to provide additional detail about the actors identified in assertions with more information, potentially providing additional trust signals. Although these W3C Verifiable Credentials can include proofs of their own

authenticity, they are not a mechanism for verifying that a particular actor authorized a claim, assertion or piece of metadata. Any validation or usage of the W3C Verifiable Credential is out of scope of this document.

EXAMPLE Conveying a W3C Verifiable Credential for the actor identified as the `author` in an assertion might link that author's ID with an email address, social media ID, or real name, or it might identify that actor as a member of a particular professional body or provide other qualifications relevant to the actor's involvement in the asset.

Any provided W3C Verifiable Credentials shall be compliant with the JSON-LD serialisation described in the W3C Verifiable Credentials Data Model.[14]

NOTE 1 JSON-LD serialization is mandated as it is the most commonly used of the three syntaxes presented in section 6 of the W3C Verifiable Credentials specification. It is also the one that aligns best with its extensibility model, which could be useful to some implementers.

An example of a compliant credential for an individual might be one issued by the National Press Photographers Association (NPPA), which links an identifier for a person to their name ("John Doe") and a statement about their membership of the NPPA. A W3C Verifiable Credential example is shown in Figure 31:

```
{
"@context": [
  "https://www.w3.org/2018/credentials/v1",
  "http://schema.org"
],
"type": [
  "VerifiableCredential",
  "NPPACredential"
],
"issuer": "https://nppa.org/",
"credentialSubject": {
  "id": "did:nppa:eb1bb9934d9896a374c384521410c7f14",
  "name": "John Doe",
  "memberOf": "https://nppa.org/"
},
"proof": {
  "type": "RsaSignature2018",
  "created": "2021-06-18T21:19:10Z",
  "proofPurpose": "assertionMethod",
  "verificationMethod":
"did:nppa:eb1bb9934d9896a374c384521410c7f14#_Qq0UL2Fq651Q0Fjd6TvnYE-
faHiOpRlPVQcY_-tA4A",
  "jws": "eyJhbGciOiJQUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19
DJBMvvFAIC00nSGB6Tn0XKbbF9XrsaJZREWvR2aONYTQQxnyXirtXnlewJMB
Bn2h9hfcGZrvnC1b6PgWmukzFJ1IiH1dWgnDIS81BH-IxXnPkbuYDeySorc4
QU9MJxdVkY5EL4HYbcIfwKj6X4LBQ2_ZHZIu1jdqLcRZqHcsDF5KKylKc1TH
n5VRWy5WhXg_gBnyWny8E6Qkrze53MR7OuAmmNJ1m1nN8SxDrG6a08L78J0-
Fbas5OjAQz3c17GY8mVuDPOBIOVjMEghBlgl3nOi1ysxbRGhHLEK4s0KKbeR
ogZdgtIDkQxDFxxn41QWDw_mmMCjs9qxg0zcZzqEJw"
}
  }
```

Figure 31 — Example W3C Verifiable Credential

A W3C Verifiable Credential used as part of this document shall contain only a single `credentialSubject` and that `credentialSubject` shall have an `id` value.

NOTE 2 Although Figure 31 and many examples in the W3C Verifiable Credentials Data Model specification use Decentralized Identifiers (DIDs) as the value of the `id` field, DIDs are not necessary. Specifically, W3C Verifiable Credentials do not depend on DIDs and DIDs do not depend on W3C Verifiable Credentials. DID-based URLs are just one way to express identifiers associated with subjects, issuers, holders, credential status lists, cryptographic keys, and other machine-readable information associated with a W3C Verifiable Credential.

7.1.1.2 VCStore

The set of W3C Verifiable Credentials in a JPEG Trust manifest are collected together into a logical construct that is referred to as the **VCStore** and it shall be stored as described in 6.1. Just as with the assertion store, the VCStore shall always be included/embedded in the JUMBF — it is not stored separately.

For each JPEG Trust manifest, there shall be no more than one VCStore associated with it. However, as an asset may have multiple JPEG Trust manifests associated with it, there may be multiple VCStores associated with an asset.

7.1.1.3 Using verifiable credentials

Some assertions, such as *Creative Work* and *Actions*, may contain references to Persons or Organisations which are responsible for various roles and responsibilities. An example of a creative work assertion is in Figure 32.

References to Actors are defined in C2PA Technical Specification.[19]

```
{
"@context": "http://schema.org/",
"@type": "CreativeWork",
"copyrightHolder": {
  "name": "Example Corp",
  "legalName": "The Example Corporation",
  "identifier": "https://www.example.com/",
  "credential": [
{
  "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-
329BF39FA1E4/c2pa.credentials/https://www.example.com/",
  "alg": "sha256",
  "hash": "Aux1tmax46cC2N3Y9aFmBO9Jfay8LEwJWzBUtZ0sUM8gA"
}
  ]
},
"copyrightYear": 2021,
"copyrightNotice": "Copyright © 2021 Example Corp."
}
```

Figure 32 — Referencing a VC from a CreativeWork assertion

7.1.1.4 Verifiable credential security considerations

In most W3C Verifiable Credential workflows, the information about the subject (e.g. the cryptographic keys) is fetched on demand at the time of validation. While that is an acceptable model, it does open up a possible attack vector by providing an attacker with an externally-visible signal about what the validator is validating. Therefore, this document also supports having the information captured and embedded at the time of signature. This not only prevents leakage, but also makes it very clear what data the signer is asserting about the credential's subject.

7.1.1.5 Redaction of verifiable credentials

Since a W3C Verifiable Credential can contain personally identifiable information, there may be workflows where it is necessary to remove/redact a W3C Verifiable Credential from the VCStore. To redact a W3C Verifiable Credential, a claim generator shall overwrite the contents of the credential in the W3C Credential Store with all zeroes, just as one would when redacting an assertion, however, doing only that will cause validation failures later on since the `hashed_uri` to the W3C Verifiable Credential will fail to resolve causing a validation failure. To fully redact a W3C Verifiable Credential, the associated (referencing) assertion would also need to be redacted.

8 Media asset content binding

8.1 General

Any media asset may contain a trust manifest which is bound to the media asset using a signed cryptographic hash. Trust manifests are validated by a process that tests the validity of the associated claim signature as well as the time stamp and any included credential revocation information.

A validator may also evaluate each of the assertions in the trust manifest, and those results become new trust indicators that are added to the full list of trust indicators (for a given asset) and compiled together into a trust credential. This trust credential may also then undergo evaluation and reporting in alignment with the requirements of a provided trust profile(s).

8.2 Cryptographic binding to content

A key aspect to a trust manifest is that is cryptographically bound to a uniquely identifiable media asset. This is accomplished through the use of standard cryptographic hashes, such as SHA2-256. These hashes also enable a validator to detect if the asset has been modified since the trust manifest was constructed.

Because media assets can be packaged in various well-known formats, that may differ in how the data contained therein is serialised — there are multiple approaches to how to perform and then store the results of the hashing algorithm used. For example, a JPEG 1 file could be hashed using a simple C2PA data hash assertion while a JPEG 2000 file could be hashed using a C2PA general box hash.[6] However, either could be used for the other format, and the results stored in the trust manifest.

8.3 Use of digital signatures

A trust manifest is signed by the claim generator of the manifest, using a certificate for a specific actor. This signature is used to not only ensure that the manifest has not been tampered with since it was created, but also to clearly identify (and enable validation of) the actor that claims to have created it. This is accomplished through the use of standard digital signatures algorithms and key types, such as RSA or ECDSA.

Because the majority of assertions in a trust manifest are serialized into CBOR, which enables native binary data, the signature is also serialized into CBOR, specifically COSE (IETF RFC 8152).[11] This enables the signature to be stored in the trust manifest in a compact and efficient manner that is also compatible with the rest of the assertions and claim in the manifest.

8.4 Validation

The determination of whether the Trust Record for a given media asset is valid is accomplished through the use of a the standard C2PA Manifest validation process.[6] The process works by first determining if the Claim Signature of the active trust manifest is valid, including validation of an associated time stamp and included credential revocation information. If there are no problems with the active trust manifest, then each of the assertions in the trust manifest is validated, including checking its associated hash provided in the Claim. If any of the assertions are invalid, then the entire trust manifest is considered invalid. The results of each of these validations are expressed as trust indicators which will then make their way out to the trust credential, as described in 4.6.