Coalition for Content Provenance and Authenticity

C2PA Implementation Guidance

1.0, 2021-12-21: Release

Table of Contents

Chapter 1. Introduction

1.1. Overview

The Coalition for Content Provenance and Authenticity (C2PA) has developed their technical specification for providing content provenance and authenticity. It is designed to enable global, opt-in, adoption of digital provenance techniques through the creation of a rich ecosystem of digital provenance enabled applications for a wide range of individuals and organizations while meeting appropriate security requirements.

The specification has been, and continues to be, informed by scenarios, workflows and requirements gathered from industry experts and partner organizations. However many of these requirements are not normative in nature or may differ between organizations or workflows - in those cases it is important to provide non-normative guidance to implementers - which is the goal of this document.

1.2. Scope

This guidance document describes non-normative technical aspects of the C2PA architecture including construction and consumption of the C2PA manifest and its components and the digital signature technology for enabling tamper-evidence as well as establishing trust. It will also address areas where implementers can extend the C2PA architecture and its ecosystem.

The C2PA also created their Guiding Principles that address areas such as respecting privacy and personal control of data with a critical eye toward potential abuse and misuse. This guidance document also will help implementers to understand how these concerns should be addressed in their implementations.

One area of guidance that is not covered in this document is that of User Experience, as that is covered in a [separate document].

Chapter 2. How to use this document

Rather than reading this document from beginning to end, it is recommended that as an implementer is first designing each aspect of their solution, they should review the relevant section of this document to ensure that they take advantage of the guidance provided.

Chapter 3. Architecture

3.1. Assertions

3.1.1. Description

Each of the actors in the system that creates or processes an asset will produce one or more assertions about when, where, and how the asset was originated or transformed. An assertion is labelled data which represents a statement made by an actor about an asset. The assertion's label is defined either by the C2PA specification or an external entity.

3.1.2. Encryption of Assertions

The set of assertions, their associated labels, and its serialization (i.e., CBOR or JSON-LD) are defined in the C2PA specification. In order to change any of these, such as the data/schema or its serialization, it is necessary to use a new label so that the new information can be clearly identified as different from the original.

A use case for creating variants of existing assertions would be to encrypt them to prevent access to those not possessing the necessary decryption key. This might be for privacy protection or the establishment of a more secure end-to-end workflow.

For example, if an implementation from `Litware.com` wished to encrypt the `std.exif` assertion to hide the information about the capture device, they could create the new assertion label as `com.litware.exif` or perhaps even `com.litware.encrypted-1` if they wanted to even hide what was encrypted.

NOTE

Items for future guidance

Guidance is required around which assertions should be chosen and when by different types of implementations.

3.2. Claim

NOTE

Items for future guidance

3.3. Manifest

3.3.1. General

A C2PA claim generator adds a new manifest to the existing asset's manifest store to reflect whatever changes it has made. If anything needs to be removed, then the specific assertions are redacted and the redaction reflected in the new manifest.

3.3.2. Frequency of Creation

The C2PA recommends that a manifest be created for an asset when a significant event in the lifecycle of the asset takes place, such as its initial creation or an "Export" operation from an editing tool. As the creation of a manifest is not a lightweight operation, due to the need to digitally sign the claim as well as potentially retrieve credentials from online services, it is recommended to do it as infrequently as possible. Additionally, the fewer the manifests, the easier validation will be.

3.3.3. Standard vs. Update Manifests

Normally, an asset's digital content will be modified between "significant events". As such, a standard manifest which includes hard bindings between that digital content and the manifest are provided. However, there are times in an asset's life where a change is required to the C2PA manifest but the digital content is not impacted. For example, the addition of some new assertion or even the redaction of an existing assertion. In those cases, an update manifest is used.

3.3.4. Manifest Repositories

The C2PA architecture supports manifest stores external to the asset they are associated with, in the form of a manifest repository. This is useful when working with a file format that does not support embedding (i.e., text or XML) or when storing the manifest store separately improves workflows (e.g., searching on manifest data in a CMS).

When using a cloud service to manage the manifest repository, and serving the manifest stores via http Link headers (as described in the C2PA specification), having the manifest store available at the same origin as the asset is recommended as it would reduce requirements on Cross-Origin Resource Sharing (CORS). However, it is worth noting that doing so does not provide any additional privacy or security protection of either the manifest store or the asset.

To mitigate risks to user privacy, we recommend manifest repository service providers to allow content creators to retain the ability to redact or remove content from the repository, or to determine which manifest may be publicly queried. For example, a content creator may want to include an information-rich manifest for internal cataloguing purposes, and an updated manifest with redacted information for public distribution.

3.3.5. Can an application remove an existing manifest store?

Completely removing an embedded manifest store from an asset is not recommended, unless the manifest store is being "externalized" - meaning that an embedded manifest store is replaced by a URI to an external location. This would be useful in scenarios where the size of assets is important but continues to support access to the asset's provenance.

NOTE Implementation of the C2PA specification is to be evaluated in the context of the current industry practices around stripping of metadata. Such action is encouraged by both MPEG and JPEG WGs. In addition many publishers including the NYT are stripping image metadata https://blog.imatag.com/state-of-image-metadata-in-news-sites-2019-update.

3.3.6. Can an application replace an existing manifest store?

Replacing an existing manifest store with a different manifest store is not recommended since doing so would completely change the provenance of an asset. However, there are some use cases where it could be appropriate. For example, a publisher may wish to remove all details about the capture and edit of an image and leave only their own publishing information.

3.3.7. Manifests for existing media

It may not always be possible (or practical) to embed a C2PA Manifest Store in an asset such as in the case of adding provenance information to assets that were created prior to the existence of C2PA. By creating an associated manifest repository for the asset and exposing its location via the methods described here, all assets can have provenance, no matter their age.

3.4. Ingredients

Ingredients are the key to the establishment of the provenance of an asset, by serving as a listing of what other assets went into the creation of the current asset. Each ingredient that is used can itself contain its provenance, thus creating a rich provenance for the current asset and all of its ingredients.

Each ingredient can either be documented as the `parentOf` the current asset or a `componentOf` that asset. The value of `parentOf` is used in the common case where one asset is opened in an editing application, modified, and then saved or exported as another asset. That original version is the `parentOf` the final (now current) asset. Alternatively, when one asset is created from a series of other assets (such as audio and video clips), those sources are identified as `componentOf` ingredients.

3.4.1. Redaction of Ingredient Assertions

NOTE *Items for future guidance*

3.5. Use of W3C Verifiable Credentials

The C2PA specification supports the inclusion of W3C Verifiable Credentials(VC) into a C2PA Manifest to represent a human or organisation that may be directly associated with an asset in some way, such as the author or publisher. VCs on their own do not infer an association; this is done via `credentials` links in the various places they are used, such as the `stds.schema-org.CreativeWork` assertion, or in Assertion Metadata. The actual VC's are stored as part of the "VC Store".

The core "identity" information is present in the VC's `credentialSubject` property, which is required but the VC may also contain any other property that the issuer of the VC wishes to include. All of these properties are then signed and the signing information included as the `proof` of the VC. However, as mentioned, in the VC data model specification (4.7), "there are multiple viable proof mechanisms, and this specification does not standardize nor recommend any single proof mechanism for use with verifiable credentials." Since there is no standardized approach

to validation of proofs, the C2PA specification does not require that VC’s be validated as part of the C2PA Manifest validation process.

Chapter 4. Guidance on the use of Content Bindings

4.1. Guidance on Hard Bindings

4.1.1. General

Every C2PA Manifest is required to have a hard binding to its associated digital asset. Use of hard bindings prevents collision-based attacks associated with soft bindings described below.

Selection of the specific hashing algorithm to use for a hard binding should be made based on the requirements of the workflow. In the absence of some compelling reason to do otherwise, it is recommended to use SHA-256.

4.1.2. Byte Range Bindings

The simplest type of hard binding that can be used to detect tampering is a cryptographic hashing algorithm over some or all of the bytes of an asset as described in the core specification. Traditionally, this type of binding is done over an inclusive list of byte ranges of the asset. However, a number of attacks on an inclusion list-based approach were identified and it was determined that they are prevented by the use of exclusions lists. These vulnerabilities would have allowed content to be added to an asset that altered the digital content without altering the hard bindings.

4.1.3. ISO BMFF Bindings

ISO BMFF-based assets utilize a well defined box structure. Accordingly, the provenance content binding for these assets needs to be expressed in terms of that box structure.

There are two approaches which were considered for integrating provenance into the ISO BMFF boxes - either by defining which boxes and/or box components should be included in the hash, or conversely, which should be excluded.

A number of attacks on an inclusion list-based approach were identified and it was determined that they are prevented by the use of exclusions lists. These vulnerabilities would have allowed content to be added to an ISO BMFF asset that altered the audio/video presentation without altering the hard binding. This is why C2PA ISO BMFF hard bindings are integrated with the box structure using Exclusion Lists.

4.1.3.1. ISO BMFF Binding Exclusion Lists

Typically ISO BMFF content need only include the mandatory exclusions required by the C2PA specification. These should be sufficient for the most common brands registered with the MP4 Registry Authority. However, careful consideration should be given to determine if addition boxes should be included for your ISO BMFF brand of asset.

Do not add a box to the exclusion list if:

- the modification of the box can alter the audio-video presentation in any way.

- the box declares content external to the asset which you wish to be tamper evident.

4.1.4. Hashing Assertions

4.2. Guidance on use of Soft Bindings

4.2.1. General

Asset metadata (including any manifest present) may be routinely removed or corrupted by legacy or non-C2PA capable platforms during distribution. This is common, for example, on social media platforms that display asset renditions (e.g. altering the resolution, form factor or quality of the digital content) that do not have the appropriate C2PA Manifests declaring those modifications. Whilst these renditions may not create user perceptible change, they nevertheless change the underlying binary representation of the digital content.

Soft bindings provide a means for identifying manifests that have become 'decoupled' from their associated assets in these circumstances.

Examples of soft bindings are content fingerprints (such as perceptual hashes) computed from the digital content, or watermarks embedded within the digital content.

4.2.2. Manifest Repositories

Consider a data store or repository into which manifests may be stored. A content creator may, at the time of publishing an asset, opt in to the additional storage of that asset's manifest into the manifest repository. For this workflow, the manifest contains at least one soft binding - for example, a perceptual hash of the digital content.

Soft bindings may be used to identify manifests that have become decoupled from their associated assets. When a consumer encounters an asset with no manifest, but would like information on the asset's provenance, they may compute a soft binding and use it to query the manifest repository. The manifest repository would return any manifests that match that soft binding, for subsequent validation.

An alternative application of soft bindings is to mitigate the threat whereby an attacker substitutes the manifest within an asset with another valid manifest in an attempt to explain that asset with false provenance. In circumstances where a consumer wishes for further information on the asset's provenance, a similar query may be made using the soft binding to return alternative manifests within the repository for the consumer's consideration. Information within the returned manifests (such as timestamps or digital signatures) may inform subsequent trust decisions made by the consumer on that asset.

4.2.3. Illustrative Scenarios for the use of Soft Bindings

1. Recovery from stripping of metadata

   Alice is a photojournalist, and captures a photo of an important event, editing it to enhance visibility of some of the content. Alice's camera device is C2PA capable, as is her image editing tool, and so a C2PA Manifest is added

to document the capture and editing of her photo. The C2PA Manifest, added by the editing tool, is signed by Alice's personal key. Bob works for Acme Corp, a news publisher, who wishes to license Alice's photo for their publication. Bob decides to trust the content from Alice due to the presence of a C2PA Manifest documenting its provenance. Bob incorporates the photo into a composed image for his publication, using a C2PA capable editing tool. The C2PA Manifest is signed by Acme Corp. A soft binding assertion is computed by Acme Corp and added to the manifest prior to signing. A copy of the manifest is stored within a manifest repository maintained by a consortium of news providers. Bob publishes the photo and it is soon redistributed around social media.

Charlie is a news consumer and member of the general public. Charlie views a rendition of the photo on social media. The social media platform is not C2PA capable and no C2PA Manifest is contained within the rendition. Furthermore, the rendition of the photo has different resolution / form factor and changed by the social media platform.

Charlie wants to know about the provenance of the photo, since it documents an important event. Charlie right-clicks and submits the photo via a browser plug-in to a look-up service operated by a federation of news organisations, of which Acme Corp is a member. Charlie's browser software computes the soft binding of the photo and send it to that service. Charlie is directed to a web page generated by the service showing matching assets. Charlie visually verifies that the retrieved asset matches the photo he is interested in, and views the manifest. Charlie uses the information in the manifest to help make an informed trust decision based on the provenance of the photo.

2. Recovery from adversarial substitution of manifest

   Alice is a citizen journalist, and captures a video of major civil unrest using a C2PA capable device, and edits it using C2PA capable editing software. After signing the manifest in her video, a copy of the manifest is stored within a manifest repository maintained by a consortium of news providers. The manifest contains both a hard and a soft binding. Several years pass.

   Mallory wishes to use Alice's video to substantiate his story about a recent civil unrest. Mallory strips the C2PA Manifest from Alice's video and substitutes his own manifest. The manifest is signed and the video asset is distributed online.

   Bob is a news producer who receives Mallory's video. Bob suspects the video is fake news. Bob computes a soft binding of the video and submits it to a provenance service of which his organization is a member. The service retrieves the manifest associated with Alice's video. Bob visually verifies the retrieved manifest matches the video (it includes a thumbnail) and validates the manifest. Bob notices that Alice's manifest contains signed assertions with a timestamp earlier than those of Mallory's video. Bob uses knowledge of this previous existing manifest to help make a more informed trust decision on whether to trust the provenance of Mallory's video. Bob concludes not to trust Mallory's video, since an earlier manifest explains the video with an alternative provenance trail.

3. Preserving provenance through non-C2PA capable toolchains

   Acme Corp maintains a content production pipeline where some stages are non-C2PA capable. Bob receives a photograph from freelance photographer Alice, containing a valid C2PA Manifest. Bob's software inserts into the digital content of the image a watermark containing a unique identifier, and records the unique identifier as a soft

binding assertion in the manifest. The manifest is placed within a manifest repository maintained internally by Acme Corp. The content passes through legacy content production processes that strip the C2PA Manifest. The final stages of production are C2PA compliant. The watermark is read from the image and submitted as a query to the manifest repository run internally by Acme Corp. After passing through the non-C2PA tools, the manifest is automatically matched using the embedded watermark, and is included as an ingredient in a new manifest, which documents that actions have been performed on the content prior to entering the C2PA capable final stage of the pipeline. The manifest is embedded into the asset in the usual way. The content is published and provenance of the image may be traced back to Alice by end consumers of the content.

4.2.4. Implementation guidance

Soft bindings are not guaranteed to be exact, and so care should be taken in their use. Consider perceptual hashing; a common form of soft binding algorithm. By design, multiple renditions of the same digital content may generate the same soft binding. However different digital content, or renditions thereof, may generate the same soft binding either in error or due to attacks on the hashing function (for example, adversarial attacks on machine learning models). Therefore we make the following design recommendation on the implementation of soft bindings in C2PA.

1. Soft bindings must not be substituted for hard bindings in order to bind claims within a manifest.
2. The matches made using a soft binding must be interactively verified via human-in-the-loop checking. For example, a thumbnail of an image stored within the manifest might be displayed to aid visual verification of the match made using a soft binding.
3. Hard bindings (cryptographic hashes) may be used as an alternative to soft bindings, to query manifests within a manifest repository. However this method will fail if the digital content has been modified (for example, is an asset rendition).
4. We recommend that services provided for the lookup of manifests using hard or soft bindings advertise the types of binding that may be used as query, using the unique identifier of that binding (per the hard or soft binding registry).
5. We recommend that claim generators that add soft binding assertions to an asset’s manifest do so as an opt-in addition and not make it mandatory.

4.2.5. Trust and Privacy Considerations

To mitigate risks to user privacy, we recommend that the consumer should be informed explicitly (for example, via opt in) to the querying of the manifest repository. For example, a consumer may interactively initiate a query for an asset containing no manifest in order to recover provenance information about that asset.

We also recommend that content creators be informed of the trade offs involved in using manifest repositories that allow for asset link-up with soft bindings; that is, on the one hand, identifying manifests that have become 'decoupled' from their associated assets, while on the other hand, privacy risks that may result from a soft binding link-up to an earlier manifest with, for example, redacted information.

To mitigate risks to user privacy and to preserve bandwidth, we recommend that the soft binding used to query the

manifest repository is computed on the client side to avoid transmission of query asset to the lookup service.

It is unlikely that a single centralized manifest repository will emerge for all content. Rather it is anticipated that decentralized model will evolve in which multiple federated manifest repositories might emerge for different industry verticals, for example a coalition of news broadcasters might maintain a federated service for soft binding lookup based upon their own manifest repositories.

To promote the interoperability of independent services that query manifest repositories, we recommend that a standard communication protocol be established for clients to send queries to the soft binding lookup services and for returning manifests to clients.

Trust in the lookup process is derived from trust in the integrity of the manifest repository. It may be desirable to use a decentralized, immutable data technology, such as a distributed ledger or blockchain, to underwrite the integrity of the manifest repository.

Chapter 5. Trust

5.1. Cryptography

C2PA prescribes cryptographic algorithms permitted for hashing (or message digest), and digital signatures both of manifests and of signing credentials. This list contains algorithms instantiated at multiple different security levels. Unless there are particular application needs or policy requirements that call for a higher security level, C2PA recommends using the following algorithms, key types, and key sizes.

For hashing, C2PA recommends using SHA2-256. Standalone hashes are used in multiple places, including hard bindings of content, lists of assertions in claims, and in `hashed-uri` and `hashed-uri-ext` structures.

When generating a key pair for certification in a signing credential, C2PA recommends an elliptic curve cryptography (ECC) key pair on the P-256 elliptic curve or the X25519 elliptic curve. If using ECC is not desired, as an alternate, C2PA recommends an RSA key pair with a modulus length of 2048 bits.

When choosing a signature algorithm for signing manifests, C2PA recommends: * `ES256` (ECDSA with SHA-256) when using an ECC key pair on the P-256, P-384, or P-521 elliptic curves, * `EdDSA` (Ed25519) when using an ECC key pair on the X25519 elliptic curve, or * `PS256` (RSASSA-PSS using SHA-256 and MGF1 with SHA-256) when using an RSA key pair.

5.2. Digital Signatures

5.2.1. Revocation Information and Time-stamps

C2PA strongly recommends that claim generators retrieve and attach time-stamps and credential freshness information at signing time. This information should be added into the COSE signature as described in the specification.

NOTE Attaching a time-stamp and freshness information to the signature allows validators to conclude the manifest is still valid a) even if the signing credential has since expired or was revoked after signing time and b) without the need of an online query.

5.2.2. Protecting claim signing keys

In practice, C2PA claim signing keys will be issued to systems that perform claim signing operations. These systems may make these operations available to end users and/or be deployed to user-owned platforms (e.g., mobile phones). Issuance or disclosure of claim signing keys to malicious actors enables attackers to create claim signatures on arbitrary assets using the compromised identity. The resulting manifests are valid in terms of the C2PA specification, but effectively allow for spoofing provenance.

It is therefore important that systems that manage C2PA claim signing keys adhere to security and key management

best practices. This includes leveraging platform-specific features (e.g., hardware security modules and cloud key management services), minimizing key reuse, and revoking keys when compromise is suspected. For more information on key management, see the NIST Key Management Guidelines.

5.2.2.1. Securing claim generation and signing operations

Some C2PA claim generation and signing systems may be exposed to untrusted users. Exploitation or misuse of these systems may allow attackers to create claim signatures on arbitrary assets using identities provided by the system. The resulting manifests are valid in terms of the C2PA specification, but effectively allow for spoofing provenance. The impact of such an attack may be amplified if identities are shared between users, and/or if the attack goes undetected for an extended period of time.

C2PA claim generation and signing systems should consider industry best practices for information security, secure development and operation, and anti-abuse practices, including leveraging available platform-specific features for deployment (e.g., Android SafetyNet, Apple DeviceCheck and AppAttest).

5.2.3. Verifying Suitability of Signing Credentials

If possible, a claim generator should attempt to validate that the signing credential used is suitable for the expected audience of validators for the asset. C2PA currently only supports X.509 certificates as signing credentials. A claim generator should be configured with the same trust anchor list and Extended Key Usage (EKU) list as the validators. Then, at signing time, the claim generator should:

1. Validate that the certificate fulfills all the requirements for C2PA signing credentials as described Certificate Profile section of the Trust Model chapter.
2. Validate that a chain of trust can be computed from the signing certificate to an entry in the trust anchor list using the contents of the `x5chain` COSE header, following the procedure in RFC 5280 section 6. This header will include both the signing certificate and any intermediate certification authorities required to build a trust chain to the trust anchor.
3. Validate that the signing certificate is valid for one of the Extended Key Usages listed.

If the claim generator is not configured with the trust anchor list and EKU list as the validators, it should not attempt any validation and sign with the provided credential.

If it is so configured and validation fails, the claim generator should warn its user with an explanation of the problem, but should allow the user to choose to proceed with signing. Users may choose to sign in situations where they know their audience has a different trust anchor or EKU list configuration.

5.2.4. Trust and Privacy Considerations

C2PA claim generation and signing systems should consider industry best practices for addressing common privacy concerns of its users.

5.3. Trust Model

5.3.1. Trust Lists

The C2PA does not mandate the use of any specific "list of certificates or CAs that can be used to verify the trustworthiness of the signer of a manifest". There exists a variety of complexities in choosing the membership for such a list, and implementers should understand them prior to the creation of their list. The C2PA will continue to improve their guidance in this matter as the ecosystem grows.

Although consumers should be able to modify the configuration of their validators, implementers should discourage consumers from adding or removing individual trust anchors. If appropriate for the application, implementers should provide users with a selection of lists they can choose from. Certain applications may have only one list of trust anchors, and others may have more than one list of trust anchors.

5.3.2. Use of the Private Credential Store (also known as the "address book")

The identity of signers will typically be established by identity providers through the use of trust lists, as there is no expectation that signers and validators will be directly known to one another. There are exceptions, however, where two users who do know one another will want to be able to validate assets signed by the other. For example, a journalist working for a well-known media organization may have a signing credential for their organization, but still wish to receive C2PA-protected assets from sources who are otherwise anonymous. The source’s validator may already trust the journalist’s credential based on its issuance from a trusted identity provider, but the source almost certainly will not have such a credential. In this case, the source can generate their own signing credential. Such a credential would not be trusted in general, as it is not issued by a recognized identity provider, but in this special case it can be communicated to the journalist, who can elect to add it to their private credential store.

The private credential store is only usable for the purpose of directly trusting assets signed by a credential that would not otherwise be accepted. Credentials in the private credential store cannot act as identity providers and issue credentials for others.

Implementers who want to provide this functionality should allow such a consumer to generate a credential with a simple user interface. In the case of X.509 certificates, this takes the form of a self-signed certificate. Implementers should follow the Certificate Profile in the specification when generating the certificate. Consumers can have the option to provide personally identifying information to be placed in the certificate if desired, but should not be required to do so; where such fields are required in the credential, the implementation can place non-identifying default values. Implementers should also allow the consumer to choose a validity period for the certificate after which it will become invalid, to suit the expected length of time the credential will be required. C2PA recommends a suggested default of one year.

Implementers should also consumers to import and export these credentials, but stress that importing such a credential should only be permitted if the consumer has personal knowledge the credential originates from the known source. Consumers should also be directed to remove credentials when assets signed with them no longer need to be validated, or the consumer learns the signing credential has been compromised or lost.

Implementers should enforce time validity periods on credentials in the private credential store, and either warn about or reject manifests signed by credentials that have expired. Implementers may provide functionality to time-stamp a single asset or otherwise mark it as requiring preservation, and continue to validate that particular manifest when needed.

Consumers should only accept credentials from others with whom they have an existing relationship and an out-of-band reason to believe the credential belongs to the intended subject.

Chapter 6. Validation

6.1. Validation security practices

Special care should be taken when implementing validators. Like other software that processes untrusted input, validators may be the target of memory safety attacks, parser attacks, request forgery attacks against adjacent systems (e.g., when retrieving remote content or decoupled manifests), information leaks (e.g., via OCSP queries), denial of service attacks, and so on. Thus, it is important that these validators adhere to secure development and operations practices associated with their respective execution environment.

A manifest consumer that is performing validation (e.g., a web browser) may detect and mitigate attempted compromise of C2PA manifests or even the complete removal of C2PA manifests. It is recommended that manifest consumers consider forthcoming C2PA User Experience guidance, retrieval of decoupled manifests via soft bindings when appropriate, and other forthcoming C2PA recommendations to mitigate the impact of these types of attacks.

6.2. Validation of Ingredient manifests

As described in the Validation section of the specification, "The validator may optionally recursively validate the ingredient's ingredients". To do so, the implementation resolves each ingredient's `url` field to find the next ingredient in the chain. It is possible that an infinite recursion situation could occur during this resolution process (whether constructed on purpose as a DoS attack or not). Implementations should be careful to check for such situations when performing this recursive resolution of ingredients.

6.3. Data validation

The C2PA Manifest consists of data that has been serialized into either CBOR or JSON-LD. The schemas (in CDDL and JSON Schema, respectively) have been published as part of the C2PA specification website. However, those schemas exist to help implementers create valid data to improve interoperability - they should not be used as part of the standard C2PA validation process.

NOTE A C2PA Validator can choose to offer schema validation as an "extra", but the results of that validation would not effect the validity of the C2PA Manifest.

Chapter 7. Additional Guidance

7.1. Distributed Ledger Technology (DLT) and C2PA

Distributed Ledger Technologies (DLTs) enable multiple parties to collaborate to produce a tamper-evident, distributed data store.

DLT enables a ledger to be shared across a set of DLT nodes and synchronized between the DLT nodes using a consensus mechanism.

In such a distributed system, control is distributed among the persons or organizations participating in the operation of the system (ISO 22739:2020).

Data stored on a DLT is immutable; once committed to a DLT, data cannot be changed or deleted. The ordering of data stored within a DLT is also immutable.

C2PA manifests store data on asset provenance that in most cases should similarly be immutable. However redaction mechanisms exist to remove past assertion data from a manifest. For example, to ensure the privacy of a creator, removing identity data from the assertion store and updating the manifest to record the event of that redaction. Other circumstances that may involve redaction could be the removal of personally identifiable information (PII) to comply with relevant legislation on data protection. Whilst the C2PA redaction mechanism provides for the deletion of data, the prior existence of that data and the act of redaction will be visible to the manifest consumer.

For this reason we make a general recommendation that C2PA manifests should not be stored on DLTs, since the data immutability guarantees of DLTs prevent redaction of manifests stored within them.

DLTs may, however, be used to underwrite the integrity of a manifest repository containing C2PA manifests (for example a cloud database). For example, a hash of a manifest, or other cryptographic proof, may be stored immutably within a DLT. This may be used to prove that the manifest has not been altered, or deleted (non-repudiation).

We outline several possible ways that DLT may be used to implement or instantiate aspects of the C2PA specification:

1. Underwriting the integrity of manifest stores

   Consider the case of an external manifest store, where manifests might stored decoupled from the digital content they describe. Such a manifest repository may be used to store manifests and query those manifests via a lookup service using either a hard or a soft binding, for example to recover provenance for assets where manifests have been removed or corrupted.

   The user of such a manifest repository trusts the governance of that manifest repository operator not to manipulate or remove manifests stored within. In other words, trust is centralized within the provider of the manifest repository. A DLT may be used to store hashes of manifests as they are committed to the manifest repository to assure consumers of the integrity of that manifest repository without the need to trust the manifest

repository provider.

A related application of DLT related to soft binding is in the creation of a federated lookup service for soft bindings. It is unlikely that a single manifest repository will exist for all manifests resolvable via soft bindings; multiple such manifest repositories are likely to emerge for any given vertical (e.g. news journalism). A distributed key-value store on DLT may resolve a soft binding to a provider running a manifest repository, which may in turn run the query and return the relevant stored manifest.

NOTE	Given the high throughput and low latency requirements for storing manifests, it is advisable that a Layer 2 solution or other efficient consensus mechanism for DLT (such as proof of stake, or proof of history) is used to mitigate adverse cost or energy usage.

2. Decentralized and self-sovereign identity

   A separate use of DLT within the scope of C2PA might include the use of self-sovereign identity (SSI) schemes based upon DLT storage of DIDs. C2PA is agnostic to the provider of identity data and provides for the concept of an actor which is representable either via a simple identifier (such as a DID) or via a W3C Verifiable Credential (which could include a DID). In some use cases it may be preferable for users to create their own identity wallets rather than rely on a centralized or third party identity provider. In such cases DIDs stored on a DLT may provide a decentralized mechanism to ground trust in the public keys of SSI wallet holders.

3. Decentralized signing

   A smart contract is a computer program stored in a DLT system wherein the outcome of any execution of the program is recorded on the distributed ledger (ISO 22739:2020). Smart contracts may be configurable via a tokenized consensus mechanism. For example, a smart contract that may be upgraded or configured according to a vote by holders of a particular cryptographic token. Such contracts are referred to as ‘decentralized autonomous organizations’ or DAOs. Like regular programs, a DAO may be used to store and process data and even take payments for doing so.

   A DAO might be set up to sign claims autonomously, according to a certificate installed by its operators. This would provide a decentralized alternative to the claim signing services run by centralized organizations tied to particular geographies or legislative zones.

   Alternatively, or in addition, a DAO might be used to set up and manage a certificate trust list. Signing of claims is grounded in public key cryptography rooted in a trust list managed by a federation of partners. Since C2PA allows for the existence of multiple such trust lists, the DLT may be leveraged to manage the trust list via a tokenized governance system. This might be attractive to content creators and consumers wishing to utilize a decentralized governance for such a list, and in turn agency over the issuance and revocation of signing certificates

7.2. Digital Non-Fungible Tokens (NFTs)

Digital NFTs (hereafter, NFTs) are digital tokens that represent assets - most commonly, creative works. NFTs are created and traded on distributed ledgers (DLT).

NFTs represent assets via an indirection (linking) mechanism. A standard ERC-721 compliant NFT links via URI to a metadata file, that in turn links via URI to an asset. Commonly these URIs incorporate a hashed component, providing cryptographic proof for the uniqueness of the linked metadata, and the URI linking to the asset from that metadata also incorporates a hashed component. For example, the URIs are commonly content IDs (CIDs) on a distributed filesystem such as the InterPlanetary File System (IPFS).

The provenance of NFT ownership is recorded through the immutable transaction history on the DLT (i.e. who currently owns or has owned the NFT).

C2PA specifies a technology for describing the provenance of an asset's creation (who created it, what was done to it, etc.). This is distinct from the provenance of NFT ownership recorded by the DLT.

Much as with physical artwork, both the provenance of ownership and the provenance of creation, ascribe value to an NFT.

C2PA may add value to NFTs by attesting to the provenance of their linked asset's creation, and also leverage that provenance to mitigate the threat of that asset being misappropriated.

It is common for valuable NFTs to copied and placed on the market anew (re-minting) by someone other than their creator, in order to misappropriate and potentially to gain reward for another creator's work.

C2PA provides for the assertion of identity within a manifest, and is agnostic to the identity system used. One or more identity assertions may be used to encode the wallet address(es) identifying the creator on the DLT(s) they intend to mint the asset on.

When a consumer, or a marketplace, encounters an NFT with a C2PA manifest in it, they may verify that the NFT being minted or offered for sale by a particular DLT user (identified by their wallet address) matches the identifier signed into the C2PA asset.

Additional checks can be made that the C2PA manifest signed to include that wallet address is known to come from a public key maintained by the content creator. This and other checks, for example on the wallet originating the minting transaction, may be used to additionally prevent spoofing or ('sleep minting') of NFTs.

NFTs may still be misappropriated by stripping metadata including C2PA manifest from the asset, prior to minting it. This may be remediated through use of soft-bindings to recover a manifest from a manifest repository, as with general metadata stripping attacks.

7.3. Playback verification for audio/video content

NOTE

Items for future guidance

A useful thing to add would be the steps a client audio/video player should perform to verify an actual content file for MP4/fMP4. This is currently being tracked as issue #518 the C2PA Specification github repo.

7.4. Attribution, Rights and Licensing

The standard assertions defined for use in a C2PA Manifest include opportunities to add information about the attribution, rights and licenses of the associated asset.

The following table shows which assertions and the specific fields thereof can be used for which type of information.

Assertion	Attribution	Rights	Licensing
Creative Work (for all assets)	author, contributor, creator	copyrightHolder, copyrightNotice, copyrightYear	acquireLicense, license, usageInfo

Assertion	Attribution	Rights	Licensing
IPTC Photo Metadata (for images)	dc:creator, plus:ImageCreator	dc:rights, plus:CopyrightOwners, xmpRights:webStatement	plus:Licensor

Assertion	Attribution	Rights	Licensing
Exif information (for images)	dc:creator	dc:rights	

While adding this information to a C2PA Manifest via the standard assertions will provide a tamper-evident declaration of the information, it may also be important to include a duplicate of the information in their standard locations within assets as defined by as schema.org, IPTC or Exif standards. The reason for providing both versions is that currently existing solution won't look for the information in the C2PA Manifest. For example, providing values for `acquireLicense` and `license` in a *Creative Work* assertion will not invoke the Licensable badge in Google Images. The values would also need to be provided as structured data in the corresponding HTML page, as required by the specification of schema.org.

7.5. GDPR

NOTE

Items for future guidance

Another topic needs to be around GDPR and other related legal aspects. This is currently being tracked as issue #114 the C2PA Specification github repo.

7.6. Social Media Platforms

NOTE

Items for future guidance

Considering its potential reach and impact, specific guidance for implementations of C2PA technology in the context of social media will be developed, as discussed in issue #146 of the C2PA GitHub repo.