slalom

C2PA

Coalition for Content Provenance and Authenticity

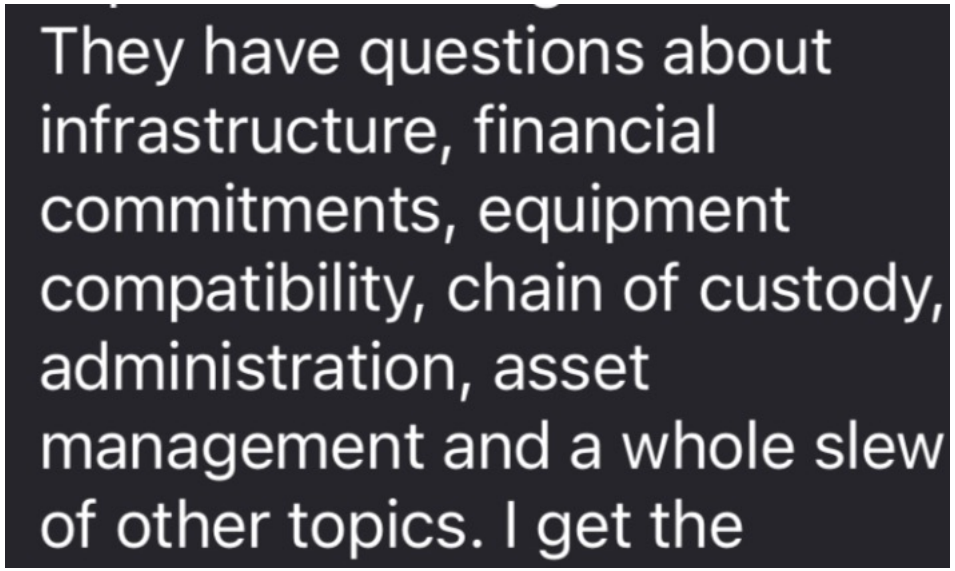
Today's agenda

- Introduction
- C2PA Architecture
- Specifications
- Technical Specification
- Signing Manifests

Client Asks

Internal – Delete Later

- Infrastructure – Can we go S3/serverless/Step Functions?
- Financials (One Time Cost) – Cost of processing 1000 images/5 minutes videos, Certificates with validity dates
- Equipment – We need adobe photoshop/truepic to enable appending metadata. No special hardware equipment. Using cloud services we can build pipelines to handle information. Question – Do we need photoshop to enable metadata writing for C2PA
- Chain of custody – Tools in chain during edit need to be c2pa compliant.
- Asset management - Tagging
- Administration – Read/Write/Consumer



What is C2PA, Why C2PA?

Components involved, Systems

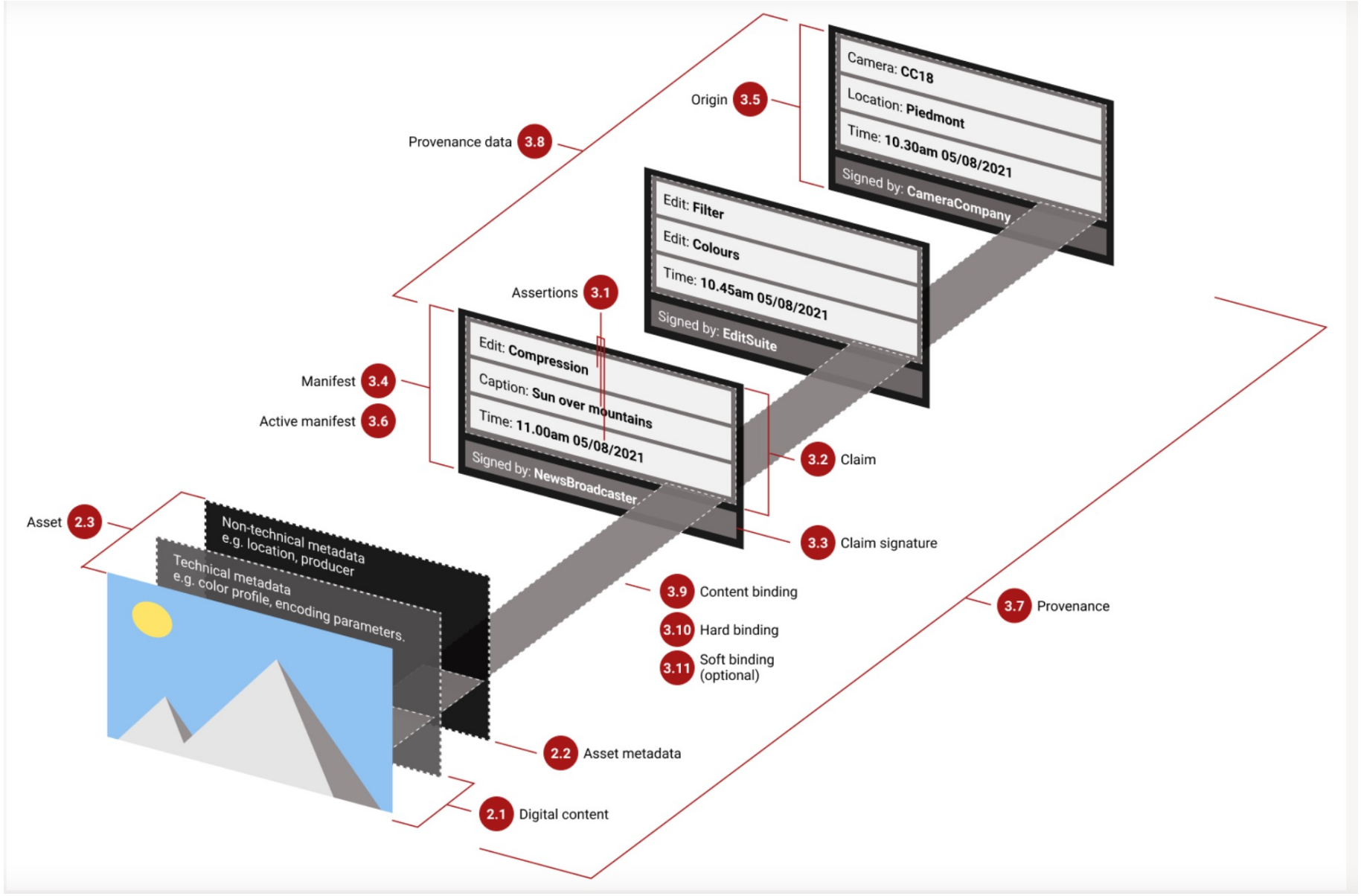
Is C2PA right for you?

Slalom View Point

Use cases

Time and skills required?

C2PA Architecture



Key Concepts

- **Asset**: A file or stream of data containing *digital content*, asset metadata and optionally, a *C2PA manifest*.
- **Composed asset**: An *asset* created by building up a collection of multiple parts or fragments of digital content (referred to as ingredients) from one or more other assets. For example, a composed asset could be an image (image A) with another image (image B) imported and super-imposed on top of it. In this example, image B is referred to as an *ingredient*.
- **C2PA manifest**: The set of information about the provenance of an asset based on the combination of one or more *assertions* (including content bindings), a single *claim*, and a *claim signature*. A *C2PA manifest* is part of a *C2PA manifest store*.
- **C2PA manifest store**: A collection of *C2PA manifests* that can either be embedded into an asset or be external to it.
- **Active manifest**: The last manifest in the list of *C2PA manifests* inside of a *C2PA manifest store* which is the one with the set of *content bindings* that are able to be validated.
- **Assertion**: A data structure which represents a statement asserted by an *actor* concerning the *asset*. This data is part of the *C2PA manifest*. For a list of standard C2PA assertions, see C2PA 1.1 Technical Specification.
- **Ingredient**: Part of a *composed asset*, such as an image superimposed on top of another image.

slalom

Which tool is right for you?

Implementation	JavaScript SDK	C2PA Tool	Rust SDK
Display C2PA data on your site or app	✓	✓	✓
Link C2PA data displayed on your site to Verify	✓	✓	✓
Write C2PA data into files		✓	✓
Quickly create and inspect C2PA data		✓	✓
Customize displaying and creating C2PA data			✓
Deploy on Web, mobile, and desktop			✓

Explainer and Guidance for Implementers

- 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.
- 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.
- **Recommend to create manifest for an asset as infrequently as possible**. This is not lightweight operation (digital sign the claim, retrieve credentials from online service). Validation will also be easier.
- One can update manifest if digital content is not impacted for example updating of assertions.
- **Replacing an existing manifest store with a different manifest store is not recommended** since doing so would completely change the provenance of an asset.
- **Manifests for existing media** - keeping the C2PA Manifests externally to the asset is an acceptable model for providing providence to assets.
- 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.
- For hard bindings recommended to use SHA-256

Explainer and Guidance for Implementers and Security

- C2PA prescribes cryptographic algorithms permitted for hashing (or message digest), and digital signatures both of manifests and of signing credentials.For hashing, C2PA recommends using SHA2-256
- C2PA strongly recommends that claim generators retrieve and attach time-stamps and credential freshness information at signing time
- Both Claim and Claim Signature are digitally signed. Manifest is created using digital signatures.
- **C2PA doesn't recommend to over ride manifests or remove manifests altogether**.

References

https://c2pa.org/specifications/specifications/1.2/explainer/Explainer.html#_what_does_provenance_mean_in_the_c2pa_specifications

https://c2pa.org/specifications/specifications/1.2/guidance/Guidance.html#_introduction

https://c2pa.org/specifications/specifications/1.2/explainer/Explainer.html#_what_does_provenance_mean_in_the_c2pa_specifications

https://c2pa.org/specifications/specifications/1.2/guidance/Guidance.html#_introduction

Adding a Manifest

```rust
use c2pa::{
    assertions::User,
    create_signer,
    Manifest,
    SigningAlg,
};

use std::path::PathBuf;
use tempfile::tempdir;

let mut manifest = Manifest::new("my_app".to_owned());
manifest.add_assertion(&User::new("org.contentauth.mylabel", r#"{"my_tag":"Anything I want"}"#))

let source = PathBuf::from("tests/fixtures/C.jpg");
let dir = tempdir()?;
let dest = dir.path().join("test_file.jpg");

// Create a ps256 signer using certs and key files
let signcert_path = "tests/fixtures/certs/ps256.pub";
let pkey_path = "tests/fixtures/certs/ps256.pem";
let signer = create_signer::from_files(signcert_path, pkey_path, SigningAlg::Ps256, None)?;

// embed a manifest using the signer
manifest.embed(&source, &dest, &*signer)?;
```

https://docs.rs/c2pa/latest/c2pa/index.html

Reading Manifest Store

```rust
use c2pa::{assertions::Actions, ManifestStore};

let manifest_store = ManifestStore::from_file("tests/fixtures/C.jpg")?;
println!("{}", manifest_store);

if let Some(manifest) = manifest_store.get_active() {
    let actions: Actions = manifest.find_assertion(Actions::LABEL)?;
    for action in actions.actions {
        println!("{}\n", action.action());
    }
}
```

https://docs.rs/c2pa/latest/c2pa/index.html

Introduction

slalom

- https://opensource.contentauthenticity.org/docs/introduction/ - Do use all 3 tools to check capability

- https://c2pa.org - done

- Manifest (C2PA manifest) consists of Provenance (historical) data.

Tools

- https://opensource.contentauthenticity.org - done

Javascript SDK

- Use Javascript SDK to display manifest data on a website or web application.
- Link manifest data on your site to verify.

C2PA command-line tool

- Write C2PA data to supported file format
- Read json manifest data.

Rusk SDK

- Embed Manifest in certain file format
- Parse and validate manifest
- Create and sign manifest

Specifications

- https://c2pa.org/specifications/specifications/1.2/index.html

Manifest Definition File - Schema

- https://opensource.contentauthenticity.org/docs/c2patool#manifest-definition-file

- Example file - https://opensource.contentauthenticity.org/docs/c2patool#example-manifest-definition-file

Generate Manifest

- Using JUMBF (JPEG universal metadata box format) This works for JPEG and PDF
- Manifest consists of – Claim, Claim Signature, C2PA Assertion store
- https://c2pa.org/specifications/specifications/1.0/specs/C2PA_Specification.html#_manifests – Tells what is included in manifest
- https://opensource.contentauthenticity.org/docs/c2patool#appendix-creating-and-using-an-x509-certificate – Tells create manifest
- https://github.com/contentauth/c2pa-rs - June 2022

Types of Manifest

- https://c2pa.org/specifications/specifications/1.0/specs/C2PA_Specification.html#_types_of_manifests

Technical Specification

slalom

- https://c2pa.org/specifications/specifications/1.1/specs/C2PA_Specification.html#_introduction

- Cargo is Rust Package Manager.
- In Rust, a library or executable program is called a *crate*. Crates are compiled using the Rust compiler, rustc
- Cargo.toml is about describing your dependencies in a broad sense, and is written by you.
- Cargo.lock contains exact information about your dependencies. It is maintained by Cargo and should not be manually edited.

C2PA command line

- Display Manifest data - c2patool sample/C.jpg

Manifest Definition File

- https://opensource.contentauthenticity.org/docs/c2patool/#manifest-definition-file

Signing Manifests

- https://opensource.contentauthenticity.org/docs/signing-manifests/

GitHub

slalom

- https://github.com/contentauth/c2patool
- https://github.com/contentauth/c2patool/tree/main/sample
- https://github.com/numbersprotocol/pyc2pa (Python implementation)
- https://verify-beta.contentauthenticity.org/inspect
- https://download.blender.org/demo/movies/BBB/

Thank you!

slalom