CAWG 1.2

July 29, 2025

Last edited: *August 11 2025*

Author: Andrew Dworschak - Yakoa

Contributor: Alex Tweeddale - cheqd

This document proposes a set of interdependent pull requests (PRs) for the CAWG 1.2 spec. This set of PRs encapsulates functionality for Verifiable Presentations (VPs), Persistent IDs and Delegation (identity hooks), and mDoc Credentials (mDL). These implementations can be found in PR2, PR3, and PR5 respectively.

To accomplish this new functionality, we also propose several PRs restructuring the existing CAWG spec (PR1) and reinterpreting CAWG credentials (PR4).

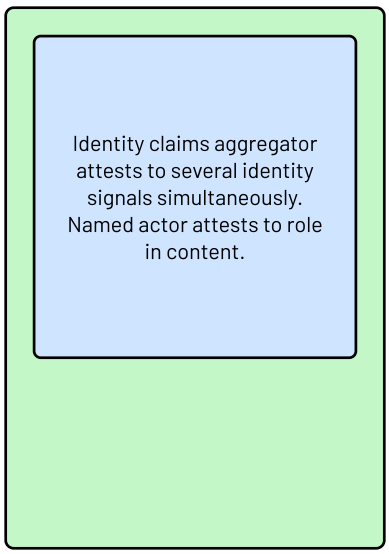
Each of the sections below describes the PRs, including sections for their dependencies with the other PRs, the list of changes & reasoning for those changes, potential issues and open questions, and example credentials resulting from the PR.

These changes are not yet written in formal spec language, but the idea and implementation should be clear enough to outline the required changes.

Once reviewed and agreed upon as a plan of action, these changes can be written into the formal spec and raised as PRs in line with their interdependencies.

What this unlocks

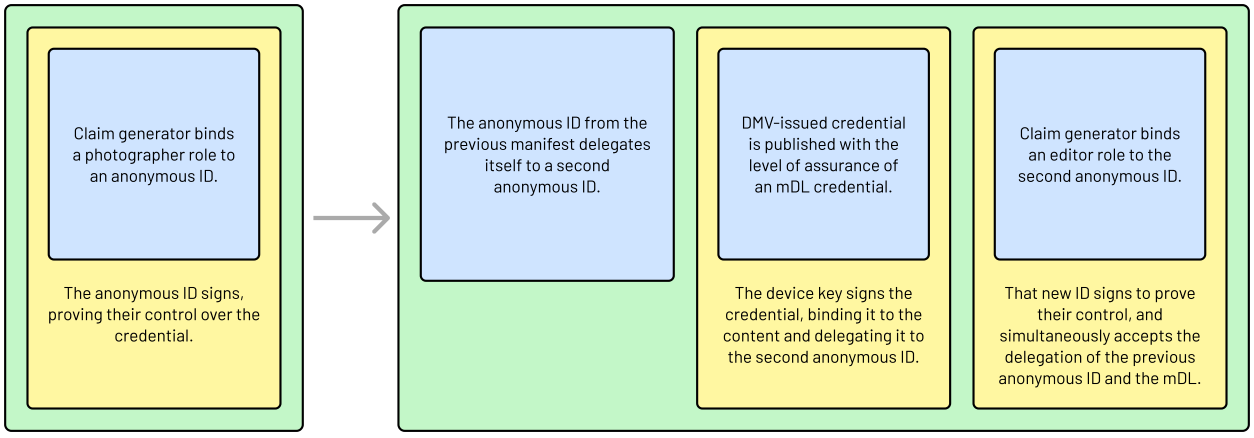
The types of statements that can be made in the existing CAWG spec are limited:



Resulting statement: An identity claims aggregator simultaneously made several statements about a named actor's identity, while named actor self-certified their role & involvement in a content's creation.

This statement merits limited trust, has no ability to support downstream actions, and relies on self-certification of many of the key elements like the role of the creator.

With the new additions proposed in this documents, we can make much more powerful statements about identity, augmenting the trust provided by the tamper-proof nature of the identity assertion:



Resulting statement: The photographer and subsequent editor of a piece of content was a single named actor, who has a DMV-certified credential backed by

the proven mDL standard.

This statement can still be communicated clearly to a content consumer, but the expression of identity is much more flexible, can evolve with time, and benefits from the trust afforded by parties like the claim generator and other identity standards like mDL.

▼ PR1: Reorganization of Verifiable Credential (VC)

Dependencies: None

The structure of the verifiable credential as it stands today in CAWG 1.1 makes it challenging to extend to accommodate other standards like verifiable presentations or mDL. It also potentially introduces compatibility issues with other identity providers in the ecosystem. I propose restructuring the structure of the credential:

- The current VC structure includes information in the `credentialSubject` field that is not actually related to the subject of the credential. This can be confusing, but also limits the extensibility of the standard. Namely, most of the `c2paAsset` field does not consist of information about the `credentailSubject`.
- A result of this blend of `credentialSubject` information leads to some fields that the `issuer` is assumed to attest to (such as the attributes of the `verifiedIdentities` field), and other information that the `issuer` does not attest to (most of the `c2paAsset` field, such as the `role`, and some of the `referenced_assertions`)
- It's unclear whether the `sig_type` field is necessary in the `c2paAsset` data. It currently complicates the structure a lot because it lives in the `c2paAsset` data which is mostly composed of self-attested data, but is clearly something the issuer is meant to attest to. We already have a mandatory `type` of `IdentityClaimsAggregationCredential`, so the `sig_type` seems redundant. If we can remove it, then it removes some complication around the presence of multiple signatures, as one would have in a VP model.
- What's currently expressed in `c2paAsset` feels much more like `termsOfUse` than it does a `credentialSubject` - in the sense that the credential is being issued for the purpose of binding a `credentialSubject` to a `c2paAsset`. In this sense, the

statement that is being made is akin to “a credential is being issued to this subject, but is only meant for use by the creator (`role`) of this content (hard binding `referenced_assertion`) to attest to the accuracy of this data (other `referenced_assertions`).

- I propose that the `c2paAsset` field be moved to `termsOfUse` outside the `credentialSubject` field and that it get mandatory type `C2PAAssetBinding`. That way, all the of the `c2paAsset` data can be flattened into the `termsOfUse` field accordingly.
- This change also allows us to flatten the `verifiedIdentities` field in the `credentialSubject` rather than wrapping it as we have today. This more closely matches other VC implementations and doesn’t sacrifice the ability to include multiple verified identities since the `credentialSubject` field supports array data. This will be very helpful down the line e.g. when interpreting mDL credentials in this format. It also removes confusion around self-signed credentials (e.g. identity hooks) where it’s not quite accurate to call it a “verified identity”.
- When it comes to the VP model, the key benefit from this change is that `termsOfUse` is an allowable field in both a VC and a VP, whereas `credentialSubject` is only permissible in a VC. This means that we can significantly standardize and streamline the binding of a VC to a C2PA asset using a VP, which is the key workflow we need to enable in PR2, the VP support pull request.

Possible issues and open questions:

- This very clearly breaks backward-compatibility. I think the current format is just too limiting and confusing as we move forward with into more complex and interesting workflows and that the change is justified, but I can understand why we may receive pushback as a result.
    - The way I’d imagine resolving this to as a way to not break everything is to continue (but deprecate) support for the existing `IdentityClaimsAggregationCredential` format, and introduce a new type for this credential, such as an `IdenittyClaimsAggregationCredentialV2`, or even something more generic to reflect the broader workflow that this restructuring enables.

- Minor point, but why is there a blend of camelCase and snake_case in the spec? There seems to be no discernible pattern - a mix of top-level and sub-fields have camelCase and the majority are in snake_case. Can this be standardized? If we're making a breaking change anyway, this would be a good time to sneak it in a change like this.

Example credentials:

Single verified identity


```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://schema.org",
    "https://cawg.io/identity/1.1/ica/context"
  ],
  "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
  "issuer": "did:web:identity-claims-aggregator.example.com",
  "validFrom": "2025-01-01T00:00:00Z",
  "validUntil": "2035-01-01T00:00:00Z",
  "credentialSubject": {
     "name": "First-Name Last-Name",
     "type": "cawg.document_verification",
     "provider": {
       "id": "https://example-id-verifier.com",
       "name": "Example ID Verifier",
     },
     "verifiedAt": "2024-07-26T22:30:15Z"
   },
   "termsOfUse": {
     "type": "C2PAAssetBinding",
     "referenced_assertions": [
     {
       "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
       "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
     },
```

```
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
      "hash": "G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
      "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
    }
   ],
   "role": "cawg.creator"
   }
}
```

Multiple verified identities

```
{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://schema.org",
  "https://cawg.io/identity/1.1/ica/context"
 ],
 "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
 "issuer": "did:web:identity-claims-aggregator.example.com",
 "validFrom": "2025-01-01T00:00:00Z",
 "validUntil": "2035-01-01T00:00:00Z",
 "credentialSubject": [
  {
       "type": "cawg.document_verification",
       "name": "First-Name Last-Name",
       "provider": {
        "id": "https://example-id-verifier.com",
        "name": "Example ID Verifier",
       },
       "verifiedAt": "2025-01-01T:00:00:00Z"
```

```
      },
      {
     "type": "cawg.social_media",
     "name": "My Account",
     "username": "username",
     "uri": "https://example-social-network.com/username",
     "provider": {
      "id": "https://example-social-network.com",
      "name": "Example Social Network"
     },
     "verifiedAt": "2025-01-01T:00:00:00Z"
    },
    ],
    "termsOfUse": {
     "type": "C2PAAssetBinding",
     "referenced_assertions": [
     {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
     },
     {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
      "hash": "G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
     },
     {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
      "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
     }
    ],
    "role": "cawg.creator"
    }
}
```

▼ PR2: Verifiable Presentation (VP) Support

Dependencies: PR1

Recognize VCs when wrapped in VPs, alongside the existing VC-only model:

- Continue to support the VC-only credential.
- If the VC includes an `id`, a `holder` matching that `id` can place additional information in the `termsOfUse` field of type `C2PAAssetBinding` via a VP, provided that there is no conflicting information between the `C2PAAssetBinding` of the VP and any VCs issued to the `id` of the `holder`.
    - `C2PAAssetBinding` info can be said to be conflicting if multiple separate hard binding assertions are referenced, if multiple distinct roles are listed, or similarly conflicting info is found in the `expected_claim_generator`, `expected_partial_claim`, or `expected_countersigners` fields.
- The VP must be signed by the `holder`. Failing to do so should trigger a validation error.
- With that said, it may be reasonable to bundle multiple VCs, or a set of `credentialSubject`s issued to different/no `id`s into a single VP. As a result, we should not necessarily trigger a validation error if the `holder` does not match with one of the `credentialSubject` `id`s. However, if the `holder` fails to match *all* of the `credentailSubject` `id`s, then a validation error may be acceptable.
- Notably, this means that a VC can choose to omit the `termsOfUse` field, relying entirely on the `holder` to establish a `C2PAAssetBinding` via a VP - this is a key enabler for interfacing with transposable credentials from the rest of the identity ecosystem. This is also why it's key for the format of the `credentialSubject` to match the expectation of the rest of the ecosystem, rather than being wrapped in a supplemental `verifiedIdentities` field, because it allows the `holder` of some VC that is not explicitly configured for CAWG to use it in their content credentials.
- This also means that, if multiple VCs or multiple `credentialSubject`s are present, and either the credentials are issued to multiple `id`s or choose to omit the

`id`, the updated `C2PAAssetBinding` can be selectively applied only to those `credentialSubject`s matching the `holder`.

Possible issues and open questions:

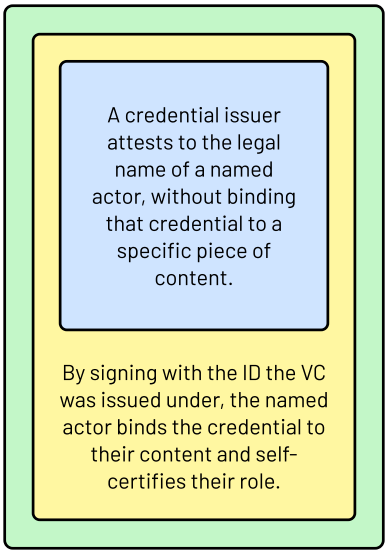
- It's not perfectly clear to me what would constitute conflicting information in the case of `expected_claim_generator`, `expected_partial_claim`, or `expected_countersigners`. This is mostly because I don't have a full view of the breadth of use cases that drove those fields to be created and the fields are not often referenced in CAWG meetings.
- A named actor may want to make statements about multiple identity signals that have been issued in VCs to separate `id`s. Under this framework, those statements cannot be easily linked together, because they do not share a common `holder` matching all of the `id`s. So, to properly express the `C2PAAssetBinding` of each of these credentials, they will often need to be placed in separate identity assertions.
    - To solve this problem, we need a way to link those statements together, showing that all identity signals originated from a common named actor. This is a statement of continuity, which relies on identity hooks to solve.
- Similarly, we need to to be very careful about which identity signals we interpret as belonging to the same person. For example, we could create an explicit rule (similar to what exists implicitly today) that an identity assertion can only describe information about a single named actor, as opposed to multiple individuals.
    - We would then need to be careful about attacks conflating one named actor's information with another. For example, a named actor could choose to group multiple unrelated VCs into a single VP (whether or not they can sign as a `holder` matching the identity of those VCs). For example, if Alice builds an identity assertion using only a VC, Bob could decide to intercept Alice's VC and bundle it with his own VC, wrapping both in a VP. He may not be able to sign with Alice's `id` (if she even has one), but could still make it look like he and Alice are the same person and, in that way, take credit for Alice's work. For that reason, while we can implement a rule that collocated `credentialSubject`s within a VC belong

to the same named actor, we should not assume the same for collocated VCs within a VP. This association should instead be established using a common `id` or via identity hooks where it applies.

- Alex Tweeddale Open Questions:
  - If we allow the `holder` to carry out `C2PAAssetBinding` via the VP model, do we then still require the initial VCs to be of `type` / `IdentityClaimsAggregationCredential`, or can we allow the holder to bundle credentials from different sources into the VP with the `C2PAAssetBinding`?
  - E.g. can identity providers issue credentials directly to the holder, and the wallet/claims aggregator just applied the `C2PAAssetBinding` on the VP side? In my opinion, this would enable far more identity applications and wallets to provide Claims Aggregation capabilities, using their existing VC/VP signing flows.

Example credentials:

Single credential, bound by VP

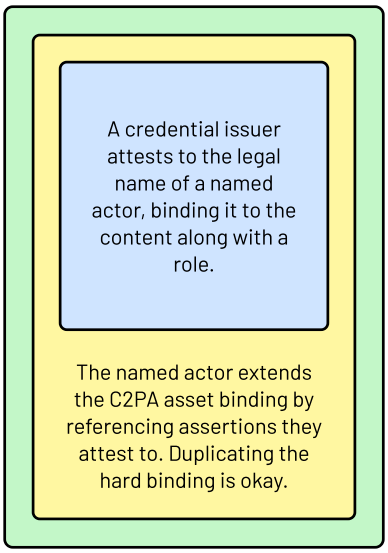



```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://cawg.io/identity/1.1/ica/context"
  ],
```

```
  "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
      "https://schema.org",
      "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:identity-claims-aggregator.example.com",
    "validFrom": "2025-01-01T00:00:00Z",
    "validUntil": "2035-01-01T00:00:00Z",
    "credentialSubject": {
      "id": "did:key:0xdeadbeef",
        "name": "First-Name Last-Name",
        "type": "cawg.document_verification",
        "provider": {
          "id": "https://example-id-verifier.com",
          "name": "Example ID Verifier",
        },
        "verifiedAt": "2024-07-26T22:30:15Z"
      }
  }],
  "termsOfUse": {
    "type": "C2PAAssetBinding",
    "referenced_assertions": [
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
      "hash": "G5hfJwYeWTIfIxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
    },
```

```
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
      "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
    }
   ],
   "role": "cawg.creator"
   }
}
```

Single credential, extended by VP



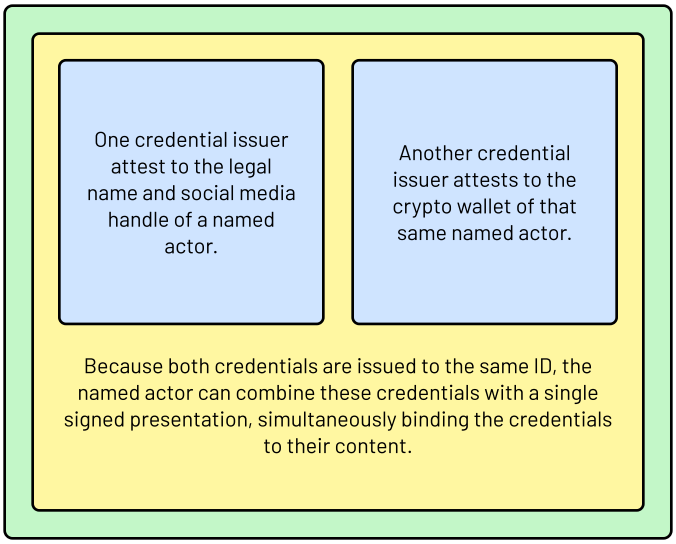
Provided that the `C2PAAssetBinding` *information does not conflict, there's no problem having duplicated information, nor is there a problem with including information in the VC that is not repeated in the VP, as long as the union of the information satisfies the requirements of the CAWG spec.*

```
{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://cawg.io/identity/1.1/ica/context"
 ],
 "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
```

```
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
      "https://schema.org",
      "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:identity-claims-aggregator.example.com",
    "validFrom": "2025-01-01T00:00:00Z",
    "validUntil": "2035-01-01T00:00:00Z",
    "credentialSubject": {
      "id": "did:key:0xdeadbeef",
        "name": "First-Name Last-Name",
        "type": "cawg.document_verification",
        "provider": {
          "id": "https://example-id-verifier.com",
          "name": "Example ID Verifier",
        },
        "verifiedAt": "2024-07-26T22:30:15Z"
      },
      "termsOfUse": {
        "type": "C2PAAssetBinding",
        "referenced_assertions": [
        {
          "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329
BF39FA1E4/c2pa.assertions/c2pa.hash.data",
          "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
        }
      ],
      "role": "cawg.creator"
      }
   }],
    "termsOfUse": {
      "type": "C2PAAssetBinding",
      "referenced_assertions": [
```

```
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
      "hash": "G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
      "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
    }
  ]
  }
}
```

Multiple credentials, bound by VP



Every VC subject being extended by a VP must match the `id` *of the* `holder` *to be associated with the* `C2PAAssetBinding` *of the VP. Non-matching or omitted* `id`*s are*

still acceptable, as long as the VC independently meets the criteria of the CAWG spec.


```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://cawg.io/identity/1.1/ica/context"
  ],
  "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [
     {
       "@context": [
         "https://www.w3.org/ns/credentials/v2",
         "https://schema.org",
         "https://cawg.io/identity/1.1/ica/context"
        ],
        "type": ["VerifiableCredential", "IdentityClaimsAggregationCredentia
l"],
        "issuer": "did:web:identity-claims-aggregator.example.com",
        "validFrom": "2025-01-01T00:00:00Z",
        "validUntil": "2035-01-01T00:00:00Z",
        "credentialSubject": [
         {
           "id": "did:key:0xdeadbeef",
              "type": "cawg.document_verification",
              "name": "First-Name Last-Name",
              "provider": {
                "id": "https://example-id-verifier.com",
                "name": "Example ID Verifier",
              },
              "verifiedAt": "2025-01-01T:00:00:00Z"
            },
            {
           "id": "did:key:0xdeadbeef",
           "type": "cawg.social_media",
```

```
          "name": "My Account",
          "username": "username",
          "uri": "https://example-social-network.com/username",
          "provider": {
            "id": "https://example-social-network.com",
            "name": "Example Social Network"
          },
          "verifiedAt": "2025-01-01T:00:00:00Z"
        },
        ]
      },
      {
        "@context": [
          "https://www.w3.org/ns/credentials/v2",
          "https://schema.org",
          "https://cawg.io/identity/1.1/ica/context"
        ],
          "type": ["VerifiableCredential", "IdentityClaimsAggregationCredenti
al"],
        "issuer": "did:web:another-identity-claims-aggregator.example.co
m",
        "validFrom": "2025-01-01T00:00:00Z",
        "validUntil": "2035-01-01T00:00:00Z",
        "credentialSubject": {
          "id": "did:key:0xdeadbeef",
      "type": "cawg.crypto_wallet",
      "username": "username",
      "uri": "https://example-crypto-wallet.com/username",
      "provider": {
        "id": "https://example-crypto-wallet.com",
        "name": "Example Crypto Wallet"
      },
      "verifiedAt": "2025-01-01T00:00:00Z"
    }
      }
  ],
```

```
  "termsOfUse": {
    "type": "C2PAAssetBinding",
    "referenced_assertions": [
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
      "hash": "G5hfJwYeWTIfIxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
      "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
    }
  ],
    "role": "cawg.creator"
  }
}
```

▼ PR3: Persistent IDs and Delegation (i.e. Identity Hooks)

Dependencies: PR1, PR2

Support statements of continuity from a named actor:

- **Persistent ids:** If the same `credentialSubject` `id` is spotted across multiple assertions (with an associated signature, often as the `holder` of a VP but sometimes as the `issuer` of a self-signed VC), those `id`s are used to show the continuity of statements from that `id`.
    - However, this is not always possible since VCs can be issued to non-matching `id`s, or and certain `id`s can have signing limitations that

prevent them from being used in other content credential contexts. This is where the concept of delegation becomes important.

- **Delegation** is a 2-step process where one identity takes responsibility for additional statements made by a named actor, and those statements consent to the delegation of their credential. It's a 2-way street, and both ways are important to capture.
- Add a new field to a `C2PAAssetBinding` named `delegatedTo`. This field references an `id`. When the `C2PAAssetBinding` includes `delegatedTo`, it attests that the named actor referenced as the `credentialSubject` also controls the `id` being delegated to. In that way, it delegates further actions relating to the `credentialSubject` to the referenced `id`.
- That first action does not demonstrate that the referenced `id` takes responsibility for the credential, which requires a separate identity assertion. That separate assertion must either be VC with the `delegatedTo` id as the `issuer` or a VP with the `delegatedTo` id as the `holder`, wherein the identity assertion with the delegation is included as a `referenced_assertion`. This unambiguously shows that the `delegatedTo` id attested to and accepted the delegated control.
    - An `id` matching `delegatedTo` must be at least one `credentialSubject` in the identity assertion where it accepts the delegated control.
    - Notably, this allows new identity signals to be issued as well as an acceptance of delegated control in one identity assertion. It's totally reasonable and possible for multiple VCs to be present in that same assertion, or other identity signals to be present in the same credential with an `id` matching `delegatedTo`.
- In effect, any `id` of a `credentialSubject` can receive delegation. The delegation mechanism is simply a clean way to express the relationship between multiple claims, and for other assertions unrelated to identity (e.g. rights assertions) to bind themselves to identity statements.
- There's no obligation for the `id` being `delegatedTo` to have existing meaning in the namespace. It's totally reasonable to do a key rotation of sorts by delegating authority to a new `id` that is not yet present in the manifest, and

then to subsequently (perhaps even days or weeks later) create a new assertion accepting that delegated authority.

Possible issues and open questions:

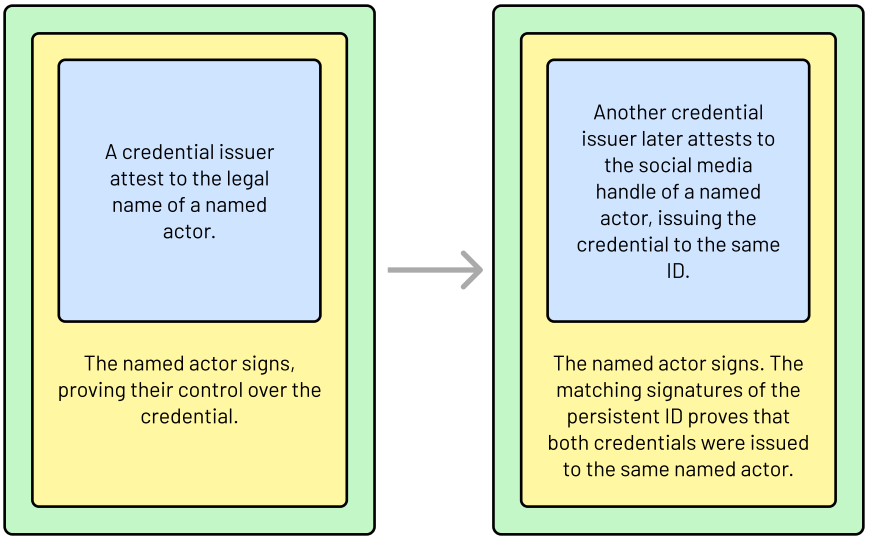
- There's a difference between persistent ids and delegation. In the case of delegation, trust flows only one way (not an issue for persistent ids): the `id` being `delegatedTo` may be willing to accept the delegation of authority by another `id`, but may not be willing for that `id` to make future statements on its behalf.
    - For this reason, we should throw a validation error if an `id` continues to make persistent statements that break the chain of delegation.
    - There may be valid reasons for a persistent `id` to make new statements, such as statements revealing additional identity information about its `id` namespace, but any such assertions should chain back up to the delegation head.
- CAWG Identity Assertion 5.1.1 specifies "For each assertion listed, an assertion with the same url, alg, and hash values MUST also be listed in the created_assertions, gathered_assertions, or assertions field of the C2PA claim in which the identity assertion appears."
    - This has a few limitations that really compromise the value of persistent IDs and delegation:
    - If someone can't reference an assertion from a previous manifest, delegation can only happen at a point in time, and when a payload accepts delegation, it can't reference all of the assertions from previous manifests (ingredients, update manifests) signed by that persistent `id`, raising questions over whether it really consented to the full delegation.
    - You can never create an identity assertion on an Update Manifest, which doesn't include a hard binding. That means that identity assertions can only ever be made and updated on manifests where the underlying content changed and significantly limits creators during the publishing flow.
    - Aside from a potential increase in the complexity of interpreting claims, I don't see any reason to keep the restriction listed in 5.1.1. Instead, I

propose that the restriction be relaxed as follows:

- “For each assertion listed, an assertion with the same url, alg, and hash values MUST also be listed in the created_assertions, gathered_assertions, or assertions field of a C2PA claim from the C2PA Manifest Store in which the identity assertion appears.”

Example credentials:

Multiple credentials, issued to same `id` and signed by same `holder`



In this example, we can correlate two credentials as being issued by the same named actor, because they sign a VP as the `holder` of the common `id` between the two credentials. These credentials could even be issued across multiple manifests on a piece of content.

If either of these credentials were not signed by a common `holder`, or either VC was not issued to an `id` matching the `holder`, the credentials could not be assumed to represent a consistent named actor.

```
{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://cawg.io/identity/1.1/ica/context"
 ],
 "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
```

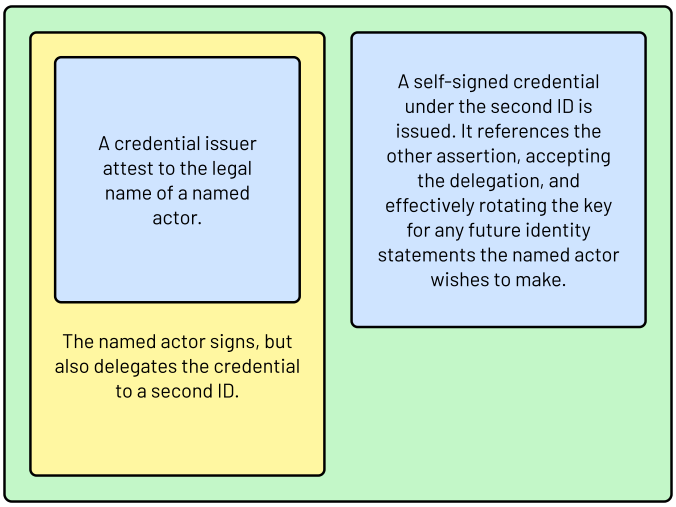
```
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
      "https://schema.org",
      "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:identity-claims-aggregator.example.com",
    "validFrom": "2025-01-01T00:00:00Z",
    "validUntil": "2035-01-01T00:00:00Z",
    "credentialSubject": {
      "id": "did:key:0xdeadbeef",
        "name": "First-Name Last-Name",
        "type": "cawg.document_verification",
        "provider": {
          "id": "https://example-id-verifier.com",
          "name": "Example ID Verifier",
        },
        "verifiedAt": "2024-07-26T22:30:15Z"
      },
      "termsOfUse": {
        "type": "C2PAAssetBinding",
        "referenced_assertions": [
        {
          "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329
BF39FA1E4/c2pa.assertions/c2pa.hash.data",
          "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
        }
      ],
      "role": "cawg.creator"
      }
   }]
}

{
```

```
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
      "https://schema.org",
      "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:identity-claims-aggregator.example.com",
    "validFrom": "2025-01-01T00:00:00Z",
    "validUntil": "2035-01-01T00:00:00Z",
    "credentialSubject": {
      "id": "did:key:0xdeadbeef",
      "type": "cawg.social_media",
      "name": "My Account",
      "username": "username",
      "uri": "https://example-social-network.com/username",
      "provider": {
        "id": "https://example-social-network.com",
        "name": "Example Social Network"
      },
      "verifiedAt": "2025-01-01T:00:00:00Z"
    },
      "termsOfUse": {
        "type": "C2PAAssetBinding",
        "referenced_assertions": [
        {
          "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329
BF39FA1E4/c2pa.assertions/c2pa.hash.data",
          "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
        }
```

```
      ],
      "role": "cawg.creator"
      }
   }]
}
```

Delegation of authority from one `id` to another



This is the minimal set of information for 0xdeadbeef to delegate control of their credential to 0xfeedface. First, we see 0xdeadbeef `delegateTo` 0xfeedface in the `C2PAAssetBinding`, thereby consenting to the delegated control. Then, 0xfeedface self-signs a credential with the delegating assertion as a `referenced_assertion`, thereby accepting the delegation.

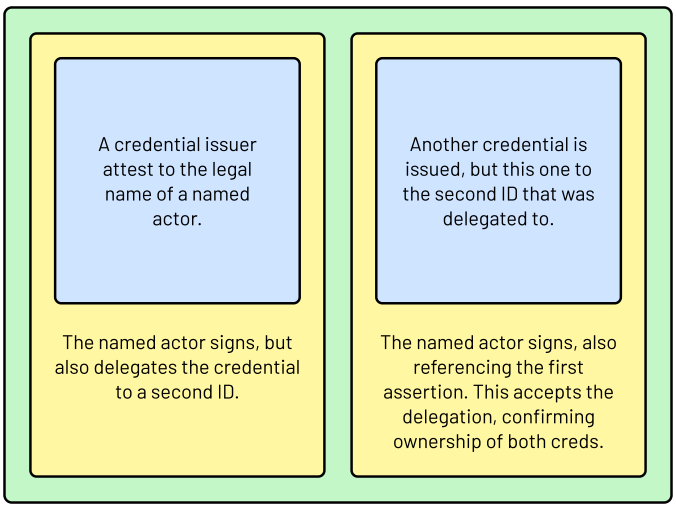
```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://cawg.io/identity/1.1/ica/context"
  ],
  "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
```

```
      "https://schema.org",
      "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:identity-claims-aggregator.example.com",
    "validFrom": "2025-01-01T00:00:00Z",
    "validUntil": "2035-01-01T00:00:00Z",
    "credentialSubject": {
      "id": "did:key:0xdeadbeef",
       "name": "First-Name Last-Name",
       "type": "cawg.document_verification",
       "provider": {
         "id": "https://example-id-verifier.com",
         "name": "Example ID Verifier",
       },
       "verifiedAt": "2024-07-26T22:30:15Z"
     }
  }],
  "termsOfUse": {
    "type": "C2PAAssetBinding",
    "referenced_assertions": [
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
    }
  ],
  "role": "cawg.creator",
  "delegatedTo": "did:key:0xfeedface"
  }
}

{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://schema.org",
```

```
    "https://cawg.io/identity/1.1/ica/context"
  ],
  "type": ["VerifiableCredential", "SelfSignedIdentityCredential"],
  "issuer": "did:key:0xfeedface",
  "validFrom": "2025-01-01T00:00:00Z",
  "validUntil": "2035-01-01T00:00:00Z",
  "credentialSubject": {
     "id": "0xfeedface",
  },
  "termsOfUse": {
     "type": "C2PAAssetBinding",
     "referenced_assertions": [
     {
       "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
       "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
     },
     {
       "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/cawg.identity",
       "hash": "G5hfJwYeWTIfIxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
     }
    ]
    }
}
```

Multiple credentials, bound by delegation



Here, in addition to accepting 0xdeadbeef's delegation of control, 0xfeedface adds an additional identity signal about their social media profile.


```
{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://cawg.io/identity/1.1/ica/context"
 ],
 "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
 "holder": "did:key:0xdeadbeef",
 "verifiableCredential": [{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://schema.org",
    "https://cawg.io/identity/1.1/ica/context"
   ],
   "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
   "issuer": "did:web:identity-claims-aggregator.example.com",
   "validFrom": "2025-01-01T00:00:00Z",
   "validUntil": "2035-01-01T00:00:00Z",
   "credentialSubject": {
    "id": "did:key:0xdeadbeef",
     "name": "First-Name Last-Name",
     "type": "cawg.document_verification",
```

```
        "provider": {
          "id": "https://example-id-verifier.com",
          "name": "Example ID Verifier",
        },
        "verifiedAt": "2024-07-26T22:30:15Z"
      }
   }],
   "termsOfUse": {
    "type": "C2PAAssetBinding",
    "referenced_assertions": [
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
    }
   ],
   "role": "cawg.creator",
   "delegatedTo": "did:key:0xfeedface"
   }
}

{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://www.w3.org/ns/credentials/examples/v2"
 ],
 "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
 "holder": "did:key:0xfeedface",
 "verifiableCredential": [{
  "@context": [
     "https://www.w3.org/ns/credentials/v2",
     "https://schema.org",
     "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:identity-claims-aggregator.example.com",
```

```
      "validFrom": "2025-01-01T00:00:00Z",
      "validUntil": "2035-01-01T00:00:00Z",
      "credentialSubject": {
        "id": "0xfeedface",
        "type": "cawg.social_media",
        "name": "My Account",
        "username": "username",
        "uri": "https://example-social-network.com/username",
        "provider": {
          "id": "https://example-social-network.com",
          "name": "Example Social Network"
        },
        "verifiedAt": "2025-01-01T:00:00:00Z"
      }
    }],
    "termsOfUse": {
      "type": "C2PAAssetBinding",
      "referenced_assertions": [
      {
        "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
        "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
      },
      {
        "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/cawg.identity",
        "hash": "G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
      }
    ]
    }
}
```


▼ PR4: Claim Generator Attestations

Dependencies: PR2, PR3 (not a true dependency, but much more powerful with PR3 included)

Specify which attestations are being made by a claimed generator when an identity assertion involves a claim generator.

- The claim generator can confer trust to credentials in a way others cannot:
    - The claim generator has a direct line of communication with named actors in certain `role`s (e.g. creators). While an identity aggregator generally *cannot* attest to the `role` a named actor played a piece of content, the claim generator *can*.
    - With that direct line of communication, a claim generator can place self-signing credentials directly in the custody of those named actors. That way, even a completely anonymized `id` can be bound to a trusted `role` and make trusted attestations about their `referenced_assertions`.
    - The claim generator can attest to certain identity attributes of an authenticated user, such as their login email address or their in-platform username. These attributes merit a high degree of trust when attested to by a claim generator.
- All info included in a VC by the claim generator is assumed to be attested to by the claim generator - this includes the full `C2PAAssetBinding`, including the `role` of the named actor and all `referenced_assertions`.
- This additional trust can be inferred from the credential by recognizing that the `issuer` of the VC matches the signature of the enveloping manifest.

Possible issues and open questions:

- This raises the question of why not to strengthen the assumption of attestation on behalf of identity aggregators? PR2 gives them an unambiguous way to separate information attested to by the holder vs. information attested to by the aggregator (by placing the self-attested `C2PAAssetBinding` into the `holder`-signed VP as opposed to the `issuer`-signed VC), so why not just encourage that behavior? There's nothing to stop an identity aggregator from wrapping the VC in an ephemeral key and signing as the holder, and this allows an advanced identity aggregator to attest to `C2PAAssetBinding` info they otherwise couldn't.
- There are several ways a claim generator could conceivably indicate their attestation to the properties of a named actor - by placing the identity

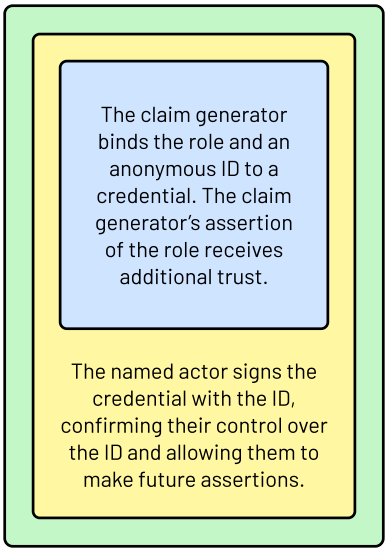
assertion as a `created_assertion` or by signing the VC with a certificate matching the manifest. I prefer the 2nd option, which means keeping the identity assertion as a `gathered_assertion`. This allows the claim generator to attest to certain information in the `C2PAAssetBinding` while still allowing the creator to add self-attested information via envelopment of the VC in a `holder`-signed VP.

    - This doesn't imply that a creator needs to manually sign VPs - the entire process can still be managed by a claim generator or an identity aggregator, but it provides a clear demarcation of who attests to which information
    - *Alex Tweeddale notes:* I see a clear future goal as having custodied (via claims aggregator) and non-custodied (via identity wallet) signing of a C2PAsset. Most users would feasibly use a claims aggregator for simplicity, but the specification should be flexible enough to support self-signing as well.

- There may be challenges in associating a claim generator signature on the claim with their signature on an identity assertion due to the differences in X.509s vs. DID methods. These challenges are likely solvable, but in the worst case, the claim generator can also indicate their attestation via a `created_assertion`. Next step is to look into how these signatures can be formed.

Example credentials:

Issuance of a basic credential:

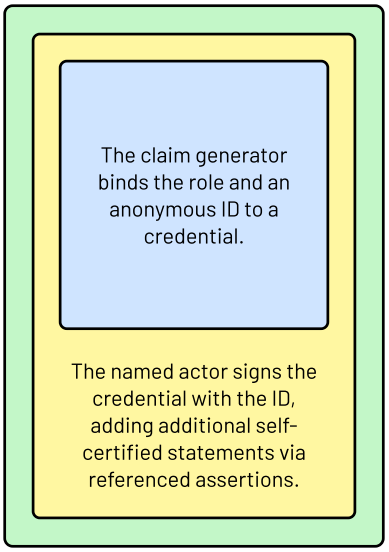


A claim generator can often attest to the role of a creator in a piece of content, and give that creator a self-signing `id` *by which to make future statements about their work. By signing a verifiable presentation with that key, the creator is then able to use it in future attestations via persistent IDs.*

```
{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://cawg.io/identity/1.1/ica/context"
 ],
 "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
 "holder": "did:key:0xdeadbeef",
 "verifiableCredential": [{
    "@context": [
     "https://www.w3.org/ns/credentials/v2",
     "https://schema.org",
     "https://cawg.io/identity/1.1/ica/context"
    ],
    "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
    "issuer": "did:web:claim-generator.example.com",
    "validFrom": "2025-01-01T00:00:00Z",
    "validUntil": "2035-01-01T00:00:00Z",
    "credentialSubject": {
```

```
        "id": "did:key:0xdeadbeef"
      },
      "termsOfUse": {
        "type": "C2PAAssetBinding",
        "referenced_assertions": [
        {
          "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329
BF39FA1E4/c2pa.assertions/c2pa.hash.data",
          "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
        }
      ],
      "role": "cawg.creator"
      }
   }]
}
```

Blending a claim generator-issued credential with self-signed attestations

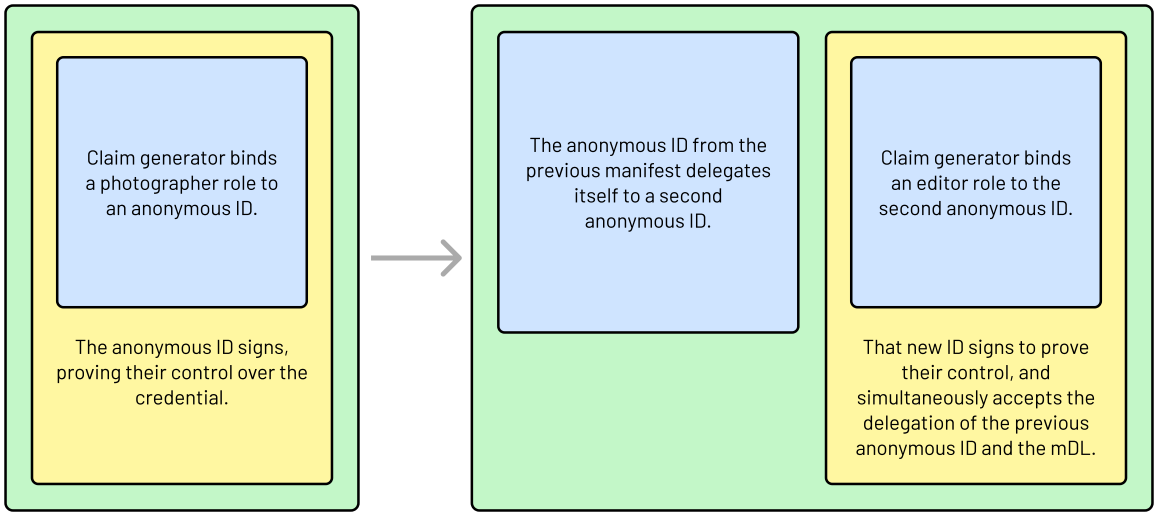


A VP can be used to clearly demarcate which parts of a credential are attested to by the claim generator and which are self-attested by the named actor. Here, a claim generator attests to the named actor's role in the content, and that named actor attests to additional assertions using that `id`*. The identity of the creator remains completely anonymous.*

```
{
 "@context": [
  "https://www.w3.org/ns/credentials/v2",
  "https://cawg.io/identity/1.1/ica/context"
 ],
 "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
 "holder": "did:key:0xdeadbeef",
 "verifiableCredential": [{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://schema.org",
    "https://cawg.io/identity/1.1/ica/context"
   ],
   "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
   "issuer": "did:web:claim-generator.example.com",
   "validFrom": "2025-01-01T00:00:00Z",
   "validUntil": "2035-01-01T00:00:00Z",
   "credentialSubject": {
    "id": "did:key:0xdeadbeef"
    },
    "termsOfUse": {
     "type": "C2PAAssetBinding",
     "referenced_assertions": [
     {
       "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329
BF39FA1E4/c2pa.assertions/c2pa.hash.data",
       "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
     }
    ],
    "role": "cawg.creator"
    }
  }],
  "termsOfUse": {
   "type": "C2PAAssetBinding",
   "referenced_assertions": [
```

```
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
      "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
      "hash": "G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
    },
    {
      "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
      "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
    }
  ]
  }
}
```

Demonstrating continued involvement along the content lifecycle



The trust conferred by a claim generator allows a named actor to bring trust to future statements they make along the content lifecycle. Here, as an example, they can add their attestation to new assertions via `referenced_assertions`*, even*

when those assertions live in a future manifest, and they can delegate their control to another `id` *which may have more meaning in the later manifest (perhaps that* `id` *was attested to by the later manifest's claim generator).*


```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://cawg.io/identity/1.1/ica/context"
  ],
  "type": ["VerifiablePresentation", "IdentityAssertionPresentation"],
  "holder": "did:key:0xdeadbeef",
  "verifiableCredential": [{
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://schema.org",
        "https://cawg.io/identity/1.1/ica/context"
      ],
      "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
      "issuer": "did:web:claim-generator.example.com",
      "validFrom": "2025-01-01T00:00:00Z",
      "validUntil": "2035-01-01T00:00:00Z",
      "credentialSubject": {
          "id": "0xdeadbeef"
        },
        "termsOfUse": {
          "type": "C2PAAssetBinding",
          "referenced_assertions": [
          {
            "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329
BF39FA1E4/c2pa.assertions/c2pa.hash.data",
            "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
          }
        ],
        "role": "cawg.creator"
        }
    }]
```

```
}

{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://schema.org",
    "https://cawg.io/identity/1.1/ica/context"
  ],
  "type": ["VerifiableCredential", "IdentityClaimsAggregationCredential"],
  "issuer": "did:key:0xdeadbeef",
  "validFrom": "2025-01-01T00:00:00Z",
  "validUntil": "2035-01-01T00:00:00Z",
  "credentialSubject": {
     "id": "0xdeadbeef"
    },
    "termsOfUse": {
     "type": "C2PAAssetBinding",
     "referenced_assertions": [
     {
       "url": "self#jumbf=c2pa/urn:uuid:E9168C5E-CEB2-4faa-B6BF-329BF
39FA1E3/c2pa.assertions/c2pa.hash.data",
       "hash": "V9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n9="
     },
     {
       "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.thumbnail.claim.jpeg",
       "hash": "G5hfJwYeWTlflxOhmfCO9xDAK52aKQ+YbKNhRZeq92c="
     },
     {
       "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4faa-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.ingredient.v2",
       "hash": "Yzag4o5jO4xPyfANVtw7ETlbFSWZNfeM78qbSi8Abkk="
     }
    ],
    "delegatedTo": "did:key:0xfeedface"
```

```
  }
}
```

▼ PR5: mDL Compatibility

Dependencies: PR1, PR2

- mDocs are an excellent, widely-adopted example of an identity credential build atop verifiable presentations of verifiable credentials. However, mainstream implementation of the mDoc standard (including mDL) do not generally allow arbitrary plaintext information to be included in signed identities.
- For example, Apple's PassKit allows only a `descriptor` (which identity elements are being requested, such as name & age), a `merchantIdentifier` string, and a `nonce` to be included in the request. Other mainstream libraries face similar limitations.
- This is problematic because the signed credential needs to encapsulate the scope of the content being signed.
- That said, it is possible to leverage these mainstream implementations of secure identity credentials by overloading the `nonce`. Since the `nonce` is meant to prescribe a session challenge to ensure the proper scope of the credential anyway, this is not entirely outside the purpose of the `nonce`.
- To that end, the standard can be expanded to support mDoc credentials by binding data to a nonce as follows:

```
{
 "mDocCredential": {
  ← Verifiable presentation object conforming to mDoc standard →
  ...
  "nonce": "hashedc2paAssetData",
 },
 "termsOfUse": {
    /* Includes all usual C2PA asset data like the C2PAAssetBinding typ
e, referenced assertions, and role */
  ...
```

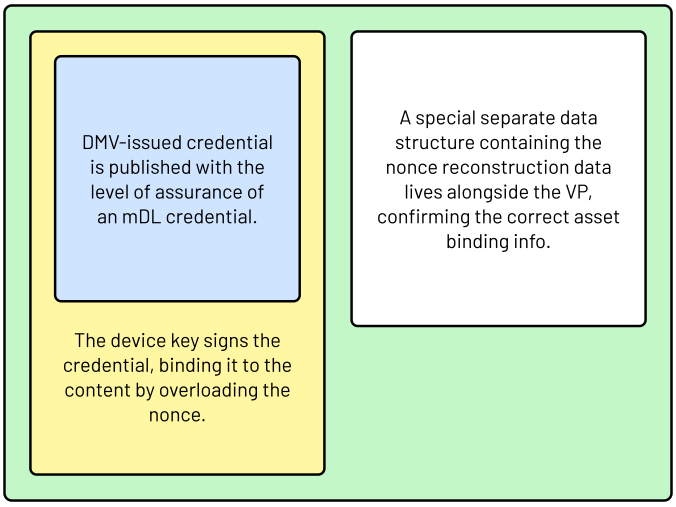
```
   /* Includes a random one-off salt making the resulting nonce rando
m and conforming to the spirit of a nonce */
   "salt": "randomuuid"
  }
  ,
  "nonceData": "hashedc2paAssetData"
}
```

Possible issues and open questions:

- There are many ways one could create the hash for the nonce. The method to hash the `C2PAAssetBinding` into a nonce can either be standardized or appropriately specified with a field that describes the hash method, in the same way that C2PA hashed URIs allow the specification of an `alg`.
- While this methodology can't be *blocked* from any implementation of mDocs, it does raise issues around *consent*. Namely, how can we be sure that the user was adequately informed that their shared credentials would be mounted permanently to a piece of content when the actual data bearing object is obscured away to this degree?

Example credentials:

mDL credential bound to a C2PA asset



The below example is somewhat simplified rather than enumerating all the data an mDL credential might enumerate in its credential (e.g. full session

transcript, revealing of hashed identity data), but the approach should be evident: we can overload the nonce as a way to sign over all the data in a VP we otherwise have no control over.

```
{
  "mDocCredential": {
    "@context": [
      "https://www.w3.org/ns/credentials/v2",
      "https://w3id.org/mdl/v1"          // typical mDL context
    ],
    "type": ["VerifiablePresentation", "mDL"],
    "id": "urn:uuid:e6cfa2d1-1234-4567-8901-abcdef012345",
    "docType": "org.iso.18013.5.1.mDL",
    "holder": "did:key:z6MksfYj8KX5vB8t7f9syh2mEcH3NWVEGXz3vR3hC
9qF",
    /* 32-byte SHA-256 of the canonicalised c2paAsset object shown below
*/
    "nonce": "khY+wzWPo1m5Eiyz5S0PmaoYuC0Xhe9kWpjU1vySk1A=",

    "verifiableCredential": [{
      "@context": [
        "https://www.w3.org/ns/credentials/v2",
        "https://w3id.org/mdl/v1"
      ],
      "type": ["VerifiableCredential", "mDL"],
      "id": "urn:uuid:01b7c653-1e9f-4e2d-9bd7-6ecf71af844a",
      "issuer": {
        "id": "did:web:dmv.ca.gov",
        "name": "California Department of Motor Vehicles"
      },
      "issuanceDate": "2025-07-20T00:00:00Z",
      "expirationDate": "2035-07-20T00:00:00Z",

      /* ---------- mDL data elements (namespace org.iso.18013.5.1) --------
-- */
      "credentialSubject": {
```

```
      "id": "did:key:z6MksfYj8KX5vB8t7f9syh2mEcH3NWVEGXz3vR3hC9q
F",
      "family_name": "DOE",
      "given_name": "JANE",
      "birth_date": "1990-05-15",
      "issue_date": "2025-07-20",
      "expiry_date": "2035-07-20",
      "issuing_country": "USA",
      "issuing_authority": "CA-DMV",
      "portrait": "
A..."   // truncated
    },

    /* Revocation list or status service per VC best-practice */
    "credentialStatus": {
      "id": "https://dmv.ca.gov/credentials/status/01b7c653-1e9f",
      "type": "CredentialStatusList2017"
    },

    "proof": {
      "type": "Ed25519Signature2018",
      "created": "2025-07-20T00:00:05Z",
      "verificationMethod": "did:web:dmv.ca.gov#key-1",
      "proofPurpose": "assertionMethod",
      "jws": "eyJhbGciOiJFZERTQSIs..."
    }
  }],

  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2025-07-20T00:00:07Z",
    "verificationMethod": "did:key:z6MksfYj8KX5vB8t7f9syh2mEcH3NWV
EGXz3vR3hC9qF#controller",
    "proofPurpose": "authentication",
    "jws": "eyJhbGciOiJFZERTQSIs..."
  }
```

```
  },

  "c2paAsset": {
    "type": "C2PAAssetBinding",
    "referenced_assertions": [
      {
        "url": "self#jumbf=c2pa/urn:uuid:F9168C5E-CEB2-4FAA-B6BF-329BF
39FA1E4/c2pa.assertions/c2pa.hash.data",
        "hash": "U9Gyz05tmpftkoEYP6XYNsMnUbnS/KcktAg2vv7n1n8="
      }
    ],
    "role": "cawg.creator",
    /* one-off salt mixed into the digest to preserve true nonce semantics */
    "salt": "1e1c9780-452d-420c-8c18-6e1b5adc94b8",
    /* identical to the `nonce` above to make the binding verifiable */
      "nonceData": "khY+wzWPo1m5Eiyz5S0PmaoYuC0Xhe9kWpjU1vySk1A
="
  },
}
```