Blockcerts V3 Proposal

a white paper from Rebooting the Web of Trust IX

by Anthony Ronning aronning@learningmachine.com (Learning Machine)
and Wong Wai Chung waichung@nextid.com (NextID)

ABSTRACT

As the standards around Verifiable Credentials are starting to take form, different flavors of "verifiable credentials-like" data structures need to make necessary changes to leverage on the rulesets outlined and constantly reviewed by knowledgeable communities such as the W3C. The purpose of this paper is to identify all of the changes needed for Blockcerts to comply with the Verifiable Credentials (VCs) and Decentralized Identifiers (DIDs) standards and to expand upon the additional benefits of using a blockchain in combination with Verifiable Credentials. This paper is meant to act as an explainer in which a formal specification can be created.

This paper proposes multiple implementation options for several properties. The intention is that we can engage the Blockcerts / Verifiable Credential communities and see what fits best.

VERIFIABLE CREDENTIAL SCHEMA

Verifiable Credentials are a data model that is defined and published as a W3C Recommendation. It seeks to represent the same information as a physical credential while also being tamper-evident and more trustworthy. Verifiable Credentials address future considerations in our societies, which are becoming increasingly digitalized, including (but not limited to) privacy-preserving goals.

Sponsors for the Rebooting the Web of Trust IX Design Workshop

An example of a minimally viable Verifiable Credential can be seen below:


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/openbadges/v2"
  ],
  "id": "https://example.org/beths-robotics-badge.json",
  "type": ["VerifiableCredential", "OpenBadgesV2"],
  "issuer": "https://example.org/organization.json",
  "issuanceDate": "2016-12-31T23:59:59Z",
  "credentialSubject": {
    "id": "https://example.org/recipient-id.json",
    "roboticsForBeginners": {
      "id": "https://example.org/organization.json",
      "name": [{
        "value": "Awesome Robotics Badge",
        "lang": "en",
        "description": "For doing awesome things with robots that people think is
pretty great.",
      }]
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://example.edu/issuers/keys/1",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
      sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUc
      X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DGlcTwLtj
      PAYuNzVBAh4vGHSrQyHUdBBPM"
  }
}
```


OPEN BADGES & BLOCKCERTS SCHEMA

Open Badges

Currently, Blockcerts is an Extension to Open Badges, which is a specification and open technical standard originally developed by the Mozilla Foundation. Open Badges is widely adopted by Universities and Microcredential platforms as a way to issue achievements that recipients can hold and collect in "backpacks". The benefit of using a blockchain as an extension to Open Badges is to provide immutability and proof of existence.

An example of a standard Open Badge can be seen below:


```
{
  "@context": "https://w3id.org/openbadges/v2",
  "type": "Assertion",
  "id": "https://example.org/beths-robotics-badge.json",
  "recipient": {
    "type": "email",
    "hashed": true,
    "salt": "deadsea",
    "identity":
      "sha256$c7ef86405ba71b85acd8e2e95166c4b111448089f2e1599f42fe1bba46e865c5"
  },
  "issuedOn": "2016-12-31T23:59:59Z",
  "badge": {
    "id": "https://example.org/robotics-badge.json",
    "type": "BadgeClass",
    "name": "Awesome Robotics Badge",
    "description": "For doing awesome things with robots that people think is
pretty great.",
    "image": "https://example.org/robotics-badge.png",
    "criteria": "https://example.org/robotics-badge.html",
    "issuer": {
      "type": "Profile",
      "id": "https://example.org/organization.json",
      "name": "An Example Badge Issuer",
      "image": "https://example.org/logo.png",
      "url": "https://example.org",
      "email": "steved@example.org",
    }
  },
  "verification": {
    "type": "hosted"
  }
}
```


An Open Badge can be separated into three parts: the assertion, the badge, and the issuer.

Assertion:


```
{
  "@context": "https://w3id.org/openbadges/v2",
  "type": "Assertion",
  "id": "https://example.org/beths-robotics-badge.json",
  "recipient": {
    "type": "email",
    "hashed": true,
```

```
    "salt": "deadsea",
    "identity":
      "sha256$c7ef86405ba71b85acd8e2e95166c4b111448089f2e1599f42fe1bba46e865c5"
  },
  "image": "https://example.org/beths-robot-badge.png",
  "evidence": "https://example.org/beths-robot-work.html",
  "issuedOn": "2016-12-31T23:59:59Z",
  "badge": "https://example.org/robotics-badge.json",
  "verification": {
    "type": "hosted"
  }
}
```

`Assertion.badge` resolves into the below:

Badge:

```
{
    "@context": "https://w3id.org/openbadges/v2",
    "type": "BadgeClass",
    "id": "https://example.org/robotics-badge.json",
    "type": "BadgeClass",
    "name": "Awesome Robotics Badge",
    "description": "For doing awesome things with robots that people think is
pretty great.",
    "image": "https://example.org/robotics-badge.png",
    "criteria": "https://example.org/robotics-badge.html",
    "issuer": "https://example.org/organization.json",
}
```

`Assertion.badge.issuer` resolves into the below:

Issuer:

```
{
    "@context": "https://w3id.org/openbadges/v2",
    "type": "Profile",
    "id": "https://example.org/organization.json",
    "name": "An Example Badge Issuer",
    "image": "https://example.org/logo.png",
    "url": "https://example.org",
    "email": "steved@example.org",
}
```

Blockcerts

Blockcerts follows this model as well, but with additional fields that allow it to be anchored by a blockchain.

An example of a Blockcerts can be seen below:


```
{
  "@context": [
    "https://w3id.org/openbadges/v2",
    "https://w3id.org/blockcerts/v2.1"
  ],
  "type": "Assertion",
  "id": "urn:uuid:bbba8553-8ec1-445f-82c9-a57251dd731c",
  "badge": {
    "id": "urn:uuid:82a4c9f2-3588-457b-80ea-da695571b8fc",
    "type": "BadgeClass",
    "name": "Certificate of Accomplishment",
    "image": "...",
    "description": "Lorem ipsum dolor sit amet, mei docendi concludaturque ad, cu
nec partem graece. Est aperiam consetetur cu, expetenda moderatius neglegentur ei
nam, suas dolor laudem eam an.",
    "criteria": {
      "narrative": "Nibh iriure ei nam, modo ridens neglegentur mel eu. At his cibo
mucius."
    },
    "issuer": {
      "id": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json",
      "type": "Profile",
      "name": "University of Learning",
      "url": "https://www.issuer.org",
      "email": "contact@issuer.org",
      "revocationList": "https://www.blockcerts.org/samples/2.0/revocation-list-
testnet.json",
      "image": "data:image/png;..."
    }
  },
  "recipient": {
    "hashed": false,
    "identity": "eularia@landroth.org",
    "type": "email"
  },
  "recipientProfile": {
    "type": [
      "RecipientProfile",
      "Extension"
    ],
    "publicKey": "ecdsa-koblitz-pubkey:mtr98kany9G1XYNU74pRnfBQmaCg2FZLmc",
```

```
    "name": "Eularia Landroth"
  },
  "issuedOn": "2017-06-29T14:58:57.461422+00:00",
  "verification": {
    "publicKey": "ecdsa-koblitz-pubkey:msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj",
    "type": [
      "MerkleProofVerification2017",
      "Extension"
    ]
  },
  "signature": {
    "type": [
      "MerkleProof2017",
      "Extension"
    ],
    "targetHash":
      "637ec732fa4b7b56f4c15a6a12680519a17a9e9eade09f5b424a48eb0e6f5ad0",
    "merkleRoot":
      "f029b45bb1a7b1f0b970f6de35344b73cccd16177b4c037acbc2541c7fc27078",
    "anchors": [
      {
        "sourceId":
          "d75b7a5bdb3d5244b753e6b84e987267cfa4ffa7a532a2ed49ad3848be1d82f8",
        "type": "BTCOpReturn",
        "chain": "bitcoinMainnet"
      }
    ],
    "proof": [
      {
        "right": "11174e220fe74de907d1107e2a357e41434123f2948fc6b946fbfd7e3e3eecd1"
      }
    ]
  }
}
```


Besides minor differences in layout/metadata between the example Blockcerts and the example Open Badge, the main differences in schema (i.e., in the Blockcerts extensions) are below.

RecipientProfile

Schema link

```
"recipientProfile": {
    "type": [
      "RecipientProfile",
      "Extension"
    ],
    "publicKey": "ecdsa-koblitz-pubkey:mtr98kany9G1XYNU74pRnfBQmaCg2FZLmc",
    "name": "Eularia Landroth"
  }
```

The `recipientProfile` allows for additional recipient information that can be used to make a strong claim of ownership over the credential. In addition to the `name` and `publicKey` properties in this example, there is an `id` field in this schema that is reserved for future uses of DIDs.

Verification

Schema link

```
"verification": {
    "publicKey": "ecdsa-koblitz-pubkey:msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj",
    "type": [
      "MerkleProofVerification2017",
      "Extension"
    ]
  }
```

In this example, `verification` is an Open Badge `VerificationObject` with a `MerkleProofVerification2017` extension to allow for the `publicKey` of the issuer. It is used during the verification step of Blockcerts to ensure that the issuer's public key matches the public key that creates the blockchain transaction with this credential.

Signature

Schema link

```
"signature": {
    "type": [
      "MerkleProof2017",
      "Extension"
    ],
    "targetHash":
      "637ec732fa4b7b56f4c15a6a12680519a17a9e9eade09f5b424a48eb0e6f5ad0",
    "merkleRoot":
      "f029b45bb1a7b1f0b970f6de35344b73cccd16177b4c037acbc2541c7fc27078",
```

```
    "anchors": [
      {
        "sourceId":
          "d75b7a5bdb3d5244b753e6b84e987267cfa4ffa7a532a2ed49ad3848be1d82f8",
        "type": "BTCOpReturn",
        "chain": "bitcoinMainnet"
      }
    ],
    "proof": [
      {
        "right": "11174e220fe74de907d1107e2a357e41434123f2948fc6b946fbfd7e3e3eecd1"
      }
    ]
}
```

The `signature` property goes through all of the Merkle proofs required to validate a hash against a Merkle root hash on a blockchain. For more information on this procedure, visit the MerkleProof2017 spec.

Issuer

Most of the properties in `issuer` come directly from the Open Badges spec. An example of a Blockcerts' "Issuer Profile" can be seen below:

```
{
  "@context": [
    "https://w3id.org/openbadges/v2",
    "https://w3id.org/blockcerts/v2"
  ],
  "type": "Profile",
  "id": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json",
  "name": "University of Learning",
  "url": "https://www.issuer.org",
  "introductionURL": "https://www.issuer.org/intro/",
  "publicKey": [
    {
      "id": "ecdsa-koblitz-pubkey:msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj",
      "created": "2017-06-29T14:48:03.814936+00:00"
    }
  ],
  "revocationList": "https://www.blockcerts.org/samples/2.0/revocation-list-
testnet.json",
  "image": "...",
  "email": "contact@issuer.org"
}
```

When verifying a Blockcert, the Issuer is checked to ensure that its public key anchored the Blockcert to the blockchain. After this check, the `revocationList` is checked to ensure that the issuer has not revoked their credential.

IntroductionURL

`IntroductionURL` is a field that was added to Blockcerts which was not present in Open Badge Issuer schema. It's used for a client (e.g., Blockcerts Wallet) to do a POST API call to transmit their public key to the issuer, so that they can include the key in the `RecipientProfile` of a Blockcerts.

More information about the exact schema being used for Blockcerts can be found here, with general information here.

This URL-based "Issuer Profile" will be improved by using DIDs for issuers. More on this in Issue Profile.

BLOCKCERTS AS VC IMPLEMENTATION

Focusing on the Blockcerts specific additions to Open Badges (`recipientProfile`, `verification`, and `signature`), we can make the following mappings to a Verifiable Credential (VC).

recipientProfile

The Blockcerts `recipientProfile` could essentially be replaced with `credentialSubject.id` and `credentialSubject.name`.

```
"recipientProfile": {
    "type": [
      "RecipientProfile",
      "Extension"
    ],
    "publicKey": "ecdsa-koblitz-pubkey:mtr98kany9G1XYNU74pRnfBQmaCg2FZLmc",
    "name": "Eularia Landroth"
  }
```

can become:

```
  "credentialSubject": {
    "id": "ecdsa-koblitz-pubkey:mtr98kany9G1XYNU74pRnfBQmaCg2FZLmc",
    "name": "Eularia Landroth",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  }
```

Blockcerts currently uses a `ecdsa-koblitz-pubkey` for the recipient. Since this is a valid URI, it can be also be used for Verifiable Credentials. Ideally, a DID is used instead, for better support throughout the VC/DID ecosystem.

`verification`

As specified in "BC V2 Schema & Examples" above, Blockcerts `verification` is used to verify that the public key of the issuer matches the public key used to issue the transaction to a blockchain.

This might be considered a redundant property since a Verifiable Credential has a `proof` property that verifies immutability as well, the "issuer profile" already specifies their keys used for issuing, and the key used for issuing will be known when resolving the blockchain transaction.

Unless there's a strong reason to keep this property in Blockcerts as it moves to the VC schema, we suggest removing `verification` in V3.

Signature / Proof Proposal

Verifiable Credentials require a `proof` property, which is used for to verify the immutability of a VC and to prove that a certain issuer signed the VC. Before VC, Blockcerts used `signature` to prove immutability. What role does `signature` provide if we are already required to implement `proof`?

Time stamping is an important property that `proof` methods do not provide along with typical signing keys. A `created` date could be applied to `proof`, but since that can be created with any date, we cannot prove it existed at a certain time. Using a blockchain can be beneficial here, as it proves that the document existed with a high degree of certainty at the time of the transaction (collisions technically can still occur due to hashing, though improvable).

To be Verifiable Credential compliant, we need to use a different signature proof. Currently, `MerkleProof2019` is being spec'd out and will be compliant with VCs.

While multiple signatures are allowable in a VC, the Blockcerts spec should only specify that a blockchain proof is required. There may be benefits to supplying both an RSA signature (as an example) and a `MerkleProof2019` signature so that there may be better interoperability for verifiers that might not support `MerkleProof2019` yet.

Example:

```
...
[Cert Data Hash]
...
  "proof": {
      "type": "MerkleProof2019",
      "creator": "did:example:abcdefghij0123456789",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42",
      "proofValue":
"z76WGJzY2rXtSiZ8BDwU4VgcLqcMEm2dXdgVVS1QCZQUptZ5P8n5YCcnbuMUASYhVNihae7m8VeYvfViYf
2KqTMVEH1B"
    }
```

Note: in the new `MerkleProof2019`, `proofValue` is a CBOR encoding of the JSON that would have been present in `MerkleProof2017`.

The above is decoded to:

```
{
    "merkleRoot":
      "3c9ee831b8705f2fbe09f8b3a92247eed88cdc90418c024924be668fdc92e781",
    "targetHash":
      "c65c6184e3d5a945ddb5437e93ea312411fd33aa1def22b0746d6ecd4aa30f20",
    "path": [{
      "right": "51b4e22ed024ec7f38dc68b0bf78c87eda525ab0896b75d2064bdb9fc60b2698"
    }, {
      "right": "61c56cca660b2e616d0bd62775e728f50275ae44adf12d1bfb9b9c507a14766b"
    }],
    "anchors": [{
      "sourceId":
        "582733d7cef8035d87cecc9ebbe13b3a2f6cc52583fbcd2b9709f20a6b8b56b3",
      "type": "BTCOpReturn"
    }]
}
```

Issuer Key Revocations

In addition to proof of existence, using a Blockchain can create additional benefits when factoring in revocation use cases. Consider a case where a non-blockchain based VC was signed with an RSA key. The Signature Proof has a `createdDate` associated with the signature, but we cannot prove that date is correct in actuality, only that the person or process signing the key claimed that as the time they signed.

In most cases, the issuer signing with keys they own should be signing with the correct time. However, in

situations where the signing key was stolen, the thief might want to issue a credential in the past to make it appear as though they, for instance, graduated with a degree at a college when they first started issuing VCs.

The college realizes their key was stolen or compromised, or they simply practice good key rotation hygiene. In any of these cases, the true issuer now revokes that key with an expiration date set to a day before the known theft.

Since credential dates can not be trusted, we can not determine which credentials fall within the `createdDate` & `revocationDate` range for a given key. Every single credential issued with a key that was stolen NEEDS to fail (or at least to warn) for signing key verification problems during the credential verification process. One bad credential issued with a stolen key can affect the status of every single recipient that received a credential from that issuer with that specific signing key. This is not satisfactory when it comes to life-long credentials.

Therefore, by utilizing the trusted timestamps of a blockchain, we can calculate the true issuance date and determine that if an issuer revoked/expired a signing key for a specific date, every credential that has an anchor on a blockchain before that date is unaffected by key revocations.

Additional Fields

In addition to the existing fields specified above, there are several fields in Blockcerts V2 that have had non-standard support in the ecosystem and so could benefit from being standardized.

`display`

In Blockcerts V2, we've been unofficially building a lot of support for `displayHtml`. This has occurred in mobile wallets, verifiers, etc. as well as third-party libraries.

In proposing changes for V3, it would be great if we can throw in official support for displays. Extending past `displayHtml`, we should allow support for any type of display. Some may not want to use `html` but instead use `pdf`, an `image`, etc.

Option 1

The schema can simply use `type` and `data` properties.

Example:

```
"display": {
    "type": "html",
    "data": "<p>hello world</p>"
}
```

Option 2

Alternatively, we can use DATA URLs instead.

```
"display": "data:text/html,%3Ch1%3EHello%2C%20World!%3C%2Fh1%3E"
```

In the beginning, the official Blockcerts Universal Verifier might only support HTML officially, but it would allow others to create valid Blockcerts with different display types. The official verifier should fall back to a default display when it does not understand a Data URL, like it does today when `displayHtml` is missing.

metadata

Similar to the issues with `display`/`displayHtml`, we do not currently have an official standard around the use of the `metadatajson` property. Nonetheless, we have unofficial support in both the mobile app and verifier for displaying some metadata information to the viewer.

Option 1

We could add this to the standard, but allow for different types of metadata format, such as `XSD` as an example. This would allow issuers to take advantage of different formats and continue to support them officially in some of the Blockcerts ecosystems.

The implementation could be similar to `display`, adding `type` and `data`:

```
"metadata": {
    "type": "json",
    "data": "{\"test\": true}"
}
```

Option 2

We could remove `metadatajson` completely. Metadata could instead be grabbed by the `credentialSubject` field. The VC spec does not have requirements about the types of information that can be mentioned in `credentialSubject`; because this is where the "holder" and "subject" properties live, it makes sense that any sort of additional metadata would live here. This will be consistent and interoperable with other Verifiable Credentials that are not Blockcerts and with other Verifiable Credential wallets.

Option 3

We could leave it as is. It is not a requirement that we change this at all. Going with option 2 would remove the need to have some possibly duplicate information, but leaving as is would allow issuers to make explicit the metadata information they want to be displayed to a user and parsed by systems.

Note that leaving `metadataJson` or changing it to `metadata` will more than likely be specific to Blockcerts and not understood by the wider VC ecosystem.

The recommended approach would be to just pull additional metadata information from `credentialSubject` (option 2).

EXAMPLE BLOCKCERTS V3

Credential:


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/blockcerts/v3"
  ],
  "id": "urn:uuid:bbba8553-8ec1-445f-82c9-a57251dd731c",
  "type": ["VerifiableCredential", "BlockcertsCredential"],
  "issuer": "did:example:23adb1f712ebc6f1c276eba4dfa",
  "issuanceDate": "2010-01-01T19:73:24Z",
  "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "holds": {
           "id": "https://example.com/badgeclasses/123",
          "type": "BadgeClass",
          "name": "Certificate of Accomplishment",
          "image": "...",
          "description": "A badge describing great accomplishments",
          "criteria": {
              "narrative": "Perform tasks of valor and wit."
          },
          "issuer": {
              "type": "Profile",
              "id": "did:example:23adb1f712ebc6f1c276eba4dfa",
              "name": "Example Issuer",
              "url": "http://example.com",
              "email": "test@example.com"
          }
      }
  },
  "evidence": {
    "id": "https://example.org/portfolios/25",
    "name": "Bob's Portfolio",
    "narrative": "Bob worked hard to develop a good portfolio",
    "genre": "ePortfolio"
  },
```

```
  "metadata": {
    "type": "json",
    "data": "{\"class\": \"2019\"}"
  },
  "display": {
    "type": "html",
    "data": "<p>This subject has received this Certificate of Accomplishment</p>"
  },
  "verification": {
    "type": [
      "MerkleProofVerification2017",
      "Extension"
    ],
    "publicKey": "ecdsa-koblitz-pubkey:1AwdUWQzJgfDDjeKtpPzMfYMHejFBrxZfo"
  },
  "proof": {
      "type": "MerkleProof2019",
      "creator": "did:example:abcdefghij0123456789",
      "created": "2017-09-23T20:21:34Z",
      "domain": "example.org",
      "nonce": "2bbgh3dgjg2302d-d2b3gi423d42",
      "proofValue":

"z76WGJzY2rXtSiZ8BDwU4VgcLqcMEm2dXdgVVS1QCZQUptZ5P8n5YCcnbuMUASYhVNihae7m8VeYvfViYf
2KqTMVEH1B"
  }
}
```

There were several options outlined above. This specific example used `metadata` as an object, `display` as an object, `holds` and `evidence` for OB-like `credentialSubject` types, and `MerkleProof2019` as the new `signature/proof`. This is just an example and not necessarily a "recommended" route for Blockcerts V3.

ISSUER PROFILE

As an Open Badge extension (and while VCs & DIDs were still getting standardized), we had a requirement that Issuer Profiles had to be resolvable via `https` for information such as public key, revocation lists, and even metadata such as name and image to be gathered when verifying a certificate. By moving to a Verifiable Credential standard that is capable of utilizing Decentralized Identifiers, we no longer have to rely upon URLs inside certificates for this information.

The Verifiable Credential spec does not require that VCs be issued using DIDs, so it's proposed that we do not make this a requirement, simply a new option. We may want to continue supporting URL-based issuer profiles as well, though we might need to make some changes to support specific key links.

Issuer Profile as a DID

There are a few things we may need outside of a normal DID resolution to provide the same UX we have today in Blockcerts V2.

Here's an example of a DID when resolved:

```
{
  "@context": "https://www.w3.org/2019/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKeys": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "service": [{
    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

One of the main requirements to verify the integrity of a certificate and to prove that a certain issuer did indeed issue it is to look at the `#key` field in the `publicKeys` property of a DID.

One can link a signing key by referencing the DID and property: `did:example:123456789abcdefghi#keys-1`.

However, at minimum, there are a few other things we need to carry over to a DID profile.

- Name
- URL
- introductionURL
- revocationList
- image
- email

We are proposing a service endpoint for `BlockcertsIssuer` that contains some of this metadata, as well as a `BlockcertsRevocation` URL to handle certificate revocations.

BlockcertsIssuerService

```
"service": [{
  "id": "did:example:123456789abcdefghi#BlockcertsIssuer",
  "type": "BlockcertsIssuerService",
  "name": "University of Example,
  "URL": "https://example.com",
  "imageURL": "https://example.com/img.png",
  "email": "test@example.com",
  "serviceEndpoint": "https://example.com/introductionURL"
}]
```

When resolving the issuer DID `did:example:123456789abcdefghi`, we can then look for types that relate to `BlockcertsIssuerService` in order to check for metadata and to allow recipients to post their own DIDs/public keys to the issuer, much like we do today for URL-based Issuer Profiles.

BlockcertsRevocationService

It's possible that we can include `BlockcertsRevocationService` as a field in the `BlockcertsIssuerService` instead of a standalone ServiceEndpoint. We may need to research best practice among ServiceEndpoints. If we were to separate it into its own field, it could look like the below.

Option 1: Check revocation for a single ID at a time.

```
"service": [{
  "id": "did:example:123456789abcdefghi#BlockcertsRevocation",
  "type": "BlockcertsRevocationService",
  "serviceEndpoint": "https://example.com/revocationEndpoint"
}]
```

If an issuer wants the ability to revoke any of their issued certificates, they can add `BlockcertsRevocationService` to their list of DID services.

This will allow a Blockcerts verifier to check the revocation status of a certificate by making a `GET` call to `https://example.com/revocationEndpoint/{certId}`.

Option 2 Check a revocation list for the ID.

An alternative method for creating this `BlockcertsRevocationService` would be to make it a `BlockcertsRevocationListService` instead, similar to how we do `revocationList` today.

```
"service": [{
  "id": "did:example:123456789abcdefghi#BlockcertsRevocationList",
  "type": "BlockcertsRevocationListService",
  "serviceEndpoint": "https://example.com/revocationListEndpoint"
}]
```

There are pros and cons to both options.

Option 1 allows a verifier to only obtain and see the revocation status of a single certificate. A verifier would not be able to see the revocation status of certificates that they do not possess. A standard practice for making UUID-based certificate IDs should prohibit verifiers from guessing another certificate ID.

However, option 1 allows the issuer to see, log, monitor (etc.) a specific certificate. They would be able to see what IP address, origin, etc. was trying to verify a specific individual's certificate and then infer certain things.

Option 2 pulls an entire list of revocation events, which means that it does not reveal to the issuer who is getting verified, but it does reveal to verifiers every revocation event they have ever made and why. In the case of a large revocation list, the verifier may have to wait for all of the revocations to get processed and retrieved.

There has not been a very good consensus yet on what method of revocation/status lists should be used for Verifiable Credentials, and thus there are no standards yet. Ideally, there is a generic RevocationServiceEndpoint for all Verifiable Credential revocations. In the meantime, we suggest that we label this as a Blockcerts-specific revocation endpoint.

Instead of the Blockcerts standard picking one of these two options, we may support both and allow issuers to decide for themselves which makes better sense for their organization.

Example

Here is an example of what an issuer DID might look like when resolved, picking option 2 as an example for revocations:

```
{
  "@context": "https://www.w3.org/2019/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKeys": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "service": [{
    "id": "did:example:123456789abcdefghi#BlockcertsIssuer",
    "type": "BlockcertsIssuerService",
    "name": "University of Example,
    "URL": "https://example.com",
    "imageURL": "https://example.com/img.png",
    "email": "test@example.com",
    "serviceEndpoint": "https://example.com/introductionURL"
  }, {
```

```
    "id": "did:example:123456789abcdefghi#BlockcertsRevocationList",
    "type": "BlockcertsRevocationListService",
    "serviceEndpoint": "https://example.com/revocationListEndpoint"
  }]
}
```

Issuer Profile as a URL in V3

Here is an example of an Issuer Profile in Blockcerts V2:

```
{
  "@context": [
    "https://w3id.org/openbadges/v2",
    "https://w3id.org/blockcerts/v2"
  ],
  "type": "Profile",
  "id": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json",
  "name": "University of Learning",
  "url": "https://www.issuer.org",
  "introductionURL": "https://www.issuer.org/intro/",
  "publicKey": [
    {
      "id": "ecdsa-koblitz-pubkey:msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj",
      "created": "2017-06-29T14:48:03.814936+00:00"
    }
  ],
  "revocationList": "https://www.blockcerts.org/samples/2.0/revocation-list-
testnet.json",
  "image": "...",
  "email": "contact@issuer.org"
}
```

Comparing that to the example issuer DID above, the only thing that needs to be changed is how `publicKey` is handled.

```
{
  "@context": [
    "https://w3id.org/openbadges/v2",
    "https://w3id.org/blockcerts/v2"
  ],
  "type": "Profile",
  "id": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json",
  "name": "University of Learning",
  "url": "https://www.issuer.org",
  "introductionURL": "https://www.issuer.org/intro/",
  "publicKey": [{
    "id": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json#keys-1",
    "type": "Ed25519VerificationKey2018",
```

```
    "controller": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "revocationList": "https://www.blockcerts.org/samples/2.0/revocation-list-
testnet.json",
  "image": "...",
  "email": "contact@issuer.org"
}
```

This will allow us to link directly to a specific key used for signing a Verifiable Credential, which is a standard way of finding the key, instead of using the current Blockcerts model of checking the blockchain issuing key against all public keys for which an issuer claims ownership.

Unfortunately, because `publicKey` has the same property name in V2, this change will make URL-based Issuer Profiles a bit tricky to deal with. Either we make URL-based Issuer Profiles incapable of handling both V2 and V3 Blockcerts or we allow both V2 and V3 public keys in there. For example:

```
  "publicKey": [
    {
      "id": "ecdsa-koblitz-pubkey:msBCHdwaQ7N2ypBYupkp6uNxtr9Pg76imj",
      "created": "2017-06-29T14:48:03.814936+00:00"
    },
    {
      "id": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json#keys-1",
      "type": "Ed25519VerificationKey2018",
      "controller": "https://www.blockcerts.org/samples/2.0/issuer-testnet.json",
      "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }
  ]
```

For V2, we may need to update verification to ignore cases where `id` is a `did:` or `http:`/`https:` URI. For V3, we may need to do the reverse or to ignore cases where `id` does not end with an `#`, as it should in the VC/DID model.

CONSIDERATIONS

Backwards Compatibility

All aspects of V2 shall continue to operate as it does today, such as accepting an organization as a recipient, displaying metadata and HTML, verifying a credential, etc.

Programmatic decisions as to how to accept an issuer, what information to display and how to display it, etc. shall be made based on the Blockcerts version outlined in `@context`:

```
"@context": [
    "https://w3id.org/openbadges/v2",
    "https://w3id.org/blockcerts/v2"
  ]
```

In cases where the issuer profile is a DID instead of a URL, universal DID resolvers shall exist throughout the ecosystem that will resolve it down to a profile, in which case Blockcerts version can then be checked.

Existing V2 / Open Badges extension support

While Blockcerts adopts the Verifiable Credential standard and moves off of the Open Badges standard directly, there may be a desire for organizations to continue issuing recognized Open Badges extensions. Blockcerts V2 credential creation and issuing can still be maintained via versioning. Any critical changes necessary can be made and published as a new Python Package under the V2 versioning (e.g., `v2.0.33`). This can be done out of good faith by the community, but we would make no guarantees for how long this might be done. We would love to invite anyone wishing to make critical update changes for V2 to make code contributions that we can merge into the official V2 branch that will be created.

Blockcerts V3 and embedded compliant Open Badges

It will be up to IMS Global and the Open Badges community to support Verifiable Credential-based Open Badges through the schema changes outlined in Open Badges are Verifiable Credentials and/or the official Open Badges verifiers. There are several ways to issue Open Badges like this through Blockcerts, which may include the introduction of `holds` into the standard or extracting a full badge from a Verifiable Credential. Note that issuing a full badge inside of a Verifiable Credential is not a standard/recommended way to use VCs, though it is technically still a VC.

Breaking Changes Summary

There are a few breaking changes that are necessary as we move to Verifiable Credentials and several optional things that may be breaking changes if we wish to implement them. As mentioned, V2 will continue to behave as is, but to issue V3 credentials and to support V2 and V3 URL-based issuer profiles, please review the changes below.

Issuer Profile

Existing Issuer Profiles could continue to operate but issuing V3 credentials may require a new Issuer Profile (either URL- or DID-based). As mentioned above in Issuer Profile, the `publicKey` property needs to change in V3. We have two options: support having a mix of V2 & V3 `publicKey` models in a single Issuer Profile or require that a V3 issuer have a seperate Issuer Profile.

For DID-based Issuer Profiles, it's understood that you would be creating a new issuer profile and would need to continue to maintain your URL-based profile for every credential you've issued (unless you reactively issue V3 for every V2 credential you've ever issued). DID issuer profiles will act similar in nature to URL-based Issuer Profiles

but under the DID-document schema model. See issuer profile as a did for a summary of these changes and options.

Data Model

The data model will change for V3, so if you've previously created a template using the open-source `cert-tools` project, you'd need to create a new one that is a valid Verifiable Credential. Please review blockcerts as VC implementation to see how V2 will map to V3 and some of the proposed options that Blockcerts V3 could use. We will more than likely have some sample V3 credentials in the `cert-tools` project as we start implementing V3.

If you wish, you could continue issuing a badge-like credential by utilizing `hold` & `evidence` as described in Open Badges are Verifiable Credentials to minimize the data changes required. Otherwise, you're free to create a new Verifiable Credential type.

There are a few optional changes that may break your existing templates as well. Please click the in-page links to jump to the summaries above:

- display
- metadata

These fields have not been standardized before, but we do have the opportunity to make these improvements while moving to a new major version number. If it interests you, please give any feedback you might have. It may be advisable to implement these features in the future to minimize the scope of work and breaking changes for V3.

Note: While these changes were identified as needed changes early on, there may still be other breaking changes that are made known during the development of Blockcerts V3.

SUMMARY

The current Blockcerts V2 standard and the Verifiable Credentials standard has a lot of similarities that easily map to each other in many ways. Blockcerts can achieve much of the same functionality and more by utilizing the Verifiable Credential and Decentralized Identifiers standards. There are many options for how to map specific properties, but in the end Blockchain Proofs, Issuer Profiles/Identities, Recipient Ownership, and the aspects of life-long credentials are better supported/standardized, giving options for doing so in a more decentralized way. From here, we incredibly value community feedback and support. Based on the conversations and decisions preferred by the community, we can work on an official specification for Blockcerts V3.

Additional Credits

Lead Author: Anthony Ronning aronning@learningmachine.com (Learning Machine)

Authors: Wong Wai Chung waichung@nextid.com (NextID)

Contributors: Matthieu Collé (@raiseandfall)

Sample APA Citation:

Chung, W., and Ronning, A. (2019). Blockcerts Proposal V3. *Rebooting the Web of Trust IX*. Retrieved from https://github.com/WebOfTrustInfo/rwot9-prague/blob/master/final-documents/BlockcertsV3.pdf.

About Rebooting the Web of Trust

This paper was produced as part of the Rebooting the Web of Trust IX design workshop. On September 3rd to 6th, 2019, over 60 tech visionaries came together in Prague, The Czech Republic to talk about the future of decentralized trust on the internet with the goal of writing at least 5 white papers and specs. This is one of them.

RWOT Board of Directors: Christopher Allen, Joe Andrieu, Kim Hamilton Duffy

Members of the Organizing Committee: Dan Burnett, Dmitri Zagidulin

Sponsors: Digital Contract Design (Gold), Protocol Labs (Silver), Digital Bazaar (Ongoing Sustaining), Jolocom

Community Sponsors: Blockchain Commons, Consensys, Learning Machine, Legendary Requirements

Workshop Credits: Christopher Allen (Founder, Co-Producer), Joe Andrieu (Co-Producer and Facilitator), and Shannon Appelcline (Editor-in-chief).

Thanks to our other contributors and sponsors!

Thanks also to Paralelní Polis and the Institute for CryptoAnarchy in Prague.

What's Next?

The design workshop and this paper are just starting points for Rebooting the Web of Trust. If you have any comments, thoughts, or expansions on this paper, please post them to our GitHub issues page:

https://github.com/WebOfTrustInfo/rwot9/issues

The tenth Rebooting the Web of Trust design workshop is scheduled for early 2020. If you'd like to be involved or would like to help sponsor the event, email:

rwot-leadership@googlegroups.com
