Capability-based Delegated Authority using Verifiable Credentials for accessing a Service

The following describes an idea from Kyle Den Hartog, spurred from contributors in the W3C Credentials Community Group, W3C Verifiable Claims Working Group, and the Decentralized Identity Foundation (DIF), about a Verifiable Credentials approach to implementing a capability-based delegated authority mechanism for accessing a Service - e.g. a website. The approach allows a Service to express the capabilities it offers to registered clients, and allows the client (person, organization) to delegate use of the capabilities to others associated with the entity.

Service Registration

[Note: “organization” in the following could be replaced with “person” or “person or organization” throughout and it all still works. Also, “user” could be replaced with “relying party” or “client” when user of the delegated authority is e.g. another service.][1]

An organization (somehow - various mechanisms can be used) enrolls with a Service resulting in the creation of an account - just as many Web Services operate today. During that process the organization shares a DID it controls with the Service and that DID is added by the Service to a “user” whitelist representing the account. The Service delivers back to the organization a list of capabilities that the organization is permitted to use. Capabilities are like "roles" in use on Web Services today. A capability will have a set of actions associated with it that are well known to the Service’s community. There is no master list of capabilities across all Services. A capability maps within the Service to the set of permissions available to a user presenting that capability for a session. For example the Patent Service would use labels like “Patent Attorney”, “Inventor”, “Licencee”, etc. The capabilities define the features the Service offers to registered entities. The mechanism to deliver the list is not important. It could be delivered through the use of a Verifiable Credential, but does not have to be.

Based on the existence of the account and the list of capabilities, the organization issues Verifiable Credentials to individual associates (staff members, contractors, lawyers, etc.) - anyone to whom the organization wants to delegate access to the Service. The Verifiable Credential MUST be issued using the same DID used to register with the Service.

---

[1] Interestingly, if the entity enrolling is a person, they would issue a claim to themselves to support the login process. That claim would be issued by the enrolled DID, which is necessary and sufficient to proof identity for access to the Service.

The Verifiable Credential should (ideally) use a standard schema, such as:

- ServiceName
- IssuerName
- IsDelegatable
- Capability
- DelegateID
- DelegateFirstName
- DelegateLastName

Note that the Service need only know the Capability - the rest of the items are optional, with their purpose defined in the sections following. The Capability must be one of the Capabilities the Service gave to the Organization.

If the “IsDelegatable” field is “True”, a delegated user can further independently delegate the capability to others. For example, a lawyer could delegate their capability to a paralegal by issuing a comparable Verified Credential to that person. This delegation can continue indefinitely. Verifying the delegations is described in the next section. If the value of the “Delegatable” field is “False”, the issuing Service would reject attempted logins from chained delegates.

An alternative to chained delegation might be to define an introduction model where there is only one level of delegation (organization to delegate), but an existing delegate can introduce to the organization an entity to whom the existing delegate wants to delegate. The organization can then provide delegation directly to that entity and track what information they need about the relationship between the existing and new delegate.

User Login

A user tries to log into the Service and is challenged to prove they have a Delegated Authority Verifiable Credential. The user responds with a Proof based on the Verifiable Credential issued from the organization. Note that the user may have many, many such Verifiable Credentials in their wallet, so they use the combination of ServiceName, IssuerName and Capability to decide the specific Credential to use for this session. Consider for example a lawyer that uses government sites to act on behalf of their clients. Each client would issue a Verifiable Credential to the lawyer for each site, and the lawyer would select a specific client’s Verifiable Credential for the specific site to which they are connecting to carry out a transaction.

On receipt of a Verifiable Credential, the Service looks up the DID of the Verifiable Credential Issuer against its list of user accounts (DIDs) to determine who the user is - e.g. what identity within the Service the user is representing. If the Service does not find the DID of the Issuer, they request a Verifiable Credential from the entity associated with that DID and repeat the process until either the the request for a Verifiable Credential fails (the user does not have a delegation) or the DID of the registered entity is found. This recursive process supports a chain

of delegations as described in the previous section. Note that if the initial DID issuing the Capability (delivered as Verifiable Credentials) is not found to be registered with the Service (e.g. not on the whitelist), authentication fails, and access is not given.

The Service also verifies that the Capability the user has presented in the Proof is valid for that DID. If all checks out, the user is given access with the Capability provided, and can act on behalf of the delegating DID (organization, person).

The additional fields in the Verifiable Credential are optional for the Service, intended for internal auditing purposes - the ID (as the Delegating entity knows the user), First Name and Last Name can be used. Other fields in the Verifiable Credential might also be helpful - e.g. a “NotifyIssuerURL” that if set could be a webhook used for the Service to notify the Issuer that someone logged in to the Service as its delegate. The handling of the ID when there is a delegation chain would be up to the Entities - does the ID stay the same for all delegates in the chain or is there an approach to have unique IDs for each.

Resulting Improvements Over Current Methods

This implementation has multiple improvements over the current approach commonly used today - an account per user on the Service, with each account linked to its associated organization and having a role defining the authority of the account. Those improvements include:

- The Service has a far simpler account management model - a single account for an entity vs. an account per user of an entity.
- The Service need only implement at most a very simple account management interface. The interface would not deal at all with entity users and roles, or with users linked to multiple entities.
- An entity implements Delegation of Authority within its own system, and across all Services using this model. The entity account admins do not have to log into each Service to, for example, delete an account of a departed associate.
- The Verifiable Credentials of associates can be revoked by the entity.
- Multiple Services wanting to share the same accounts can use the same model with a central authentication module used by the Services. This may be used as implementations of BC Government’s WebADE and Government of Canada’s Accounts system.