Quickstart

Suggest Edits

If you'd like to jump right into writing code, we have a quick start guide below. You should still read through the foundations of Verite to learn more about the overall architecture and how verifiable credentials can be applied in the real-world, but it's understandable to get excited and want to dive head-first into code.

This quick start is representative of the end-to-end flow of a person (subject) requesting a credential, an issuer issuing that credential, and a verifier verifying that credential. It is all encapsulated in a single function for readability and for quickly testing the referenced functions from the Verite library. However, it's important to understand that each step in the process will happen on different computers and in different environments.

For more detailed guides, please see theFor Developers section.

Installing Verite

The entire Verite library is housed in a single monorepo called verite . To install the library, simply run either:

```
npm i verite
```

or

```
yarn add verite
```

Example Code

```
ts import { randomDidKey, buildIssuer, composeFulfillmentFromAttestation, buildKycAmlManifest,
validateCredentialApplication, composeCredentialApplication, buildKycVerificationOffer, composePresentationSubmission,
validateVerificationSubmission, verifyVerifiableCredential, verifyVerifiablePresentation } from "verite" import { randomBytes }
from "crypto" import { v4 as uuidv4 } from "uuid"
```

```
// Simulation of key generation and storage // Each party will have their own keys and will need to maintain these const
subject = randomDidKey(randomBytes) const issuerDidKey = randomDidKey(randomBytes) const verifier =
randomDidKey(randomBytes)
```

```
// Issuer builds a manifest representing the type of credential (in this case a KYCAML credential) const manifest =
buildKycAmlManifest({ id: issuerDidKey.controller }) // The credential application is created and returned as a JWT const
application = await composeCredentialApplication(subject, manifest) // The decoded JWT is necessary when it comes time
to issue the verifiable presentation which will include this credential const decodedApplication = await
validateCredentialApplication(application) // The attestation is a standardized representation of the issuer const attestation =
{ type: "KYCAMLAttestation", process: "https://verite.id/definitions/processes/kycaml/0.0.1/usa", approvalDate: new
Date().toISOString() } const credentialType = "KYCAMLCredential"
```

```
// The issuer is created from the issuer key, and the credential is issued const issuer = buildIssuer(issuerDidKey.subject,
issuerDidKey.privateKey) const presentation = await composeFulfillmentFromAttestation( issuer, decodedApplication,
attestation, credentialType )
```

```
// As with the application, the verifiable presentation (which contains the credential) // is in JWT form and must be decoded
by the subject. This can be done in a mobile app // client or a web browser. const decoded = await
verifyVerifiablePresentation(presentation) // The verifiable credential is another JWT within the verifiable presentation and //
can be extracted like this: const vc = decoded.verifiableCredential[0] // The verifiable credential must then be decoded so
that the subject can request // verification const decodedVc = await verifyVerifiableCredential(vc.proof.jwt)
```

```
// The subject would make a request to the verifier's server to obtain the verification // offer. The code below must be
executed by the verifier, using the verifier's key. const offer = buildKycVerificationOffer( uuidv4(), verifier.subject,
"https://test.host/verify" )
```

```
// The subject can then create a submission is the full verification request which would // be sent to the verifier that uses the
offer created and supplied by the verifier const submission = await composePresentationSubmission( subject,
offer.body.presentation_definition, decodedVc )
```

```
// The verifier will take the submission and verify its authenticity. There is no response // from this function, but if it throws,
then the credential is invalid. await validateVerificationSubmission( submission, offer.body.presentation_definition )
console.log("Credential verified!")
```

Updated3 months ago

* Table of Contents * * Installing Verite * * Example Code