**[Verifiable Credentials JS Library](https://github.com/digitalbazaar/vc" \l "verifiable-credentials-js-library-digitalbazaarvc)*[(@digitalbazaar/vc)](https://github.com/digitalbazaar/vc" \l "verifiable-credentials-js-library-digitalbazaarvc)***

A Javascript library for issuing and verifying Verifiable Credentials.

[**Table of Contents**](https://github.com/digitalbazaar/vc#table-of-contents)

* [Security](https://github.com/digitalbazaar/vc#security)
* [Background](https://github.com/digitalbazaar/vc#background)
* [Install](https://github.com/digitalbazaar/vc#install)
* [Usage](https://github.com/digitalbazaar/vc#usage)
* [Testing](https://github.com/digitalbazaar/vc#testing)
* [Contribute](https://github.com/digitalbazaar/vc#contribute)
* [Commercial Support](https://github.com/digitalbazaar/vc#commercial-support)
* [License](https://github.com/digitalbazaar/vc#license)

[**Security**](https://github.com/digitalbazaar/vc#security)

As with most security- and cryptography-related tools, the overall security of your system will largely depend on your design decisions (which key types you will use, where you'll store the private keys, what you put into your credentials, and so on.)

[**Background**](https://github.com/digitalbazaar/vc#background)

This library is a Javascript (Node.js and browser) implementation of the [Verifiable Credentials Data Model 1.0](https://w3c.github.io/vc-data-model/) specification (the JWT serialization is not currently supported).

It allows you to perform the following basic operations:

1. Signing (issuing) a Verifiable Credential (VC).
2. Creating a Verifiable Presentation (VP), signed or unsigned
3. Verifying a VP
4. Verifying a standalone VC

**Pre-requisites:** Usage of this library assumes you have the ability to do the following:

* [Generate LD key pairs and signature suites](https://github.com/digitalbazaar/vc/blob/main/BACKGROUND.md#generating-keys-and-suites)
* Publish the corresponding public keys somewhere that is accessible to the verifier.
* Make sure your custom @contexts, verification methods (such as public keys) and their corresponding controller documents, and any other resolvable objects, are reachable via a documentLoader.

[**Install**](https://github.com/digitalbazaar/vc#install)

* Browsers and Node.js 14+ are supported.

To install from NPM:

npm install @digitalbazaar/vc

To install locally (for development):

git clone https://github.com/digitalbazaar/vc.git

cd vc

npm install

[**Usage**](https://github.com/digitalbazaar/vc#usage)

[**Setting up a signature suite**](https://github.com/digitalbazaar/vc#setting-up-a-signature-suite)

For signing, when setting up a signature suite, you will need to pass in a key pair containing a private key.

import vc from '@digitalbazaar/vc';

// Required to set up a suite instance with private key

import {Ed25519VerificationKey2020} from

'@digitalbazaar/ed25519-verification-key-2020';

import {Ed25519Signature2020} from '@digitalbazaar/ed25519-signature-2020';

const keyPair = await Ed25519VerificationKey2020.generate();

const suite = new Ed25519Signature2020({key: keyPair});

[**Issuing a Verifiable Credential**](https://github.com/digitalbazaar/vc#issuing-a-verifiable-credential)

Pre-requisites:

* You have a private key (with id and controller) and corresponding suite
* If you're using a custom @context, make sure it's resolvable
* (Recommended) You have a strategy for where to publish your Controller Document and Public Key

const vc = require('@digitalbazaar/vc');

// Sample unsigned credential

const credential = {

"@context": [

"https://www.w3.org/2018/credentials/v1",

"https://www.w3.org/2018/credentials/examples/v1"

],

"id": "https://example.com/credentials/1872",

"type": ["VerifiableCredential", "AlumniCredential"],

"issuer": "https://example.edu/issuers/565049",

"issuanceDate": "2010-01-01T19:23:24Z",

"credentialSubject": {

"id": "did:example:ebfeb1f712ebc6f1c276e12ec21",

"alumniOf": "Example University"

}

};

const signedVC = await vc.issue({credential, suite, documentLoader});

console.log(JSON.stringify(signedVC, null, 2));

[**Creating a Verifiable Presentation**](https://github.com/digitalbazaar/vc#creating-a-verifiable-presentation)

Pre-requisites:

* You have the requisite private keys (with id and controller) and corresponding suites
* If you're using a custom @context, make sure it's resolvable
* (Recommended) You have a strategy for where to publish your Controller Documents and Public Keys

[**Creating an unsigned presentation**](https://github.com/digitalbazaar/vc#creating-an-unsigned-presentation)

To create a presentation out of one or more verifiable credentials, you can use the createPresentation() convenience function. Alternatively, you can create the presentation object manually (don't forget to set the @context and type properties).

To create a verifiable presentation with a custom @context field use a [custom documentLoader](https://github.com/digitalbazaar/vc#custom-documentLoader)

const verifiableCredential = [vc1, vc2]; // either array or single object

// optional `id` and `holder`

const id = 'ebc6f1c2';

const holder = 'did:ex:12345';

const presentation = vc.createPresentation({

verifiableCredential, id, holder

});

console.log(JSON.stringify(presentation, null, 2));

// ->

{

"@context": [

"https://www.w3.org/2018/credentials/v1"

],

"type": [

"VerifiablePresentation"

],

"id": "ebc6f1c2",

"holder": "did:ex:12345",

"verifiableCredential": [

// vc1:

{

"@context": [

"https://www.w3.org/2018/credentials/v1",

"https://www.w3.org/2018/credentials/examples/v1"

],

"id": "http://example.edu/credentials/1872",

"type": [

"VerifiableCredential",

"AlumniCredential"

],

"issuer": "https://example.edu/issuers/565049",

"issuanceDate": "2010-01-01T19:23:24Z",

"credentialSubject": {

"id": "did:example:ebfeb1f712ebc6f1c276e12ec21",

"alumniOf": "<span lang=\"en\">Example University</span>"

},

"proof": {

"type": "Ed25519Signature2018",

"created": "2020-02-03T17:23:49Z",

"jws": "eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..AUQ3AJ23WM5vMOWNtYKuqZBekRAOUibOMH9XuvOd39my1sO-X9R4QyAXLD2ospssLvIuwmQVhJa-F0xMOnkvBg",

"proofPurpose": "assertionMethod",

"verificationMethod": "https://example.edu/issuers/keys/1"

}

},

// vc2 goes here ...

]

}

Note that this creates an *unsigned* presentation (which may be valid for some use cases).

[**Custom documentLoader**](https://github.com/digitalbazaar/vc#custom-documentloader)

Pre-requisites:

* You have an existing valid JSON-LD @context.
* Your custom context is resolvable at an address.

// jsonld-signatures has a secure context loader

// by requiring this first you ensure security

// contexts are loaded from jsonld-signatures

// and not an insecure source.

const {extendContextLoader} = require('jsonld-signatures');

const vc = require('@digitalbazaar/vc');

// @digitalbazaar/vc exports its own secure documentLoader.

const {defaultDocumentLoader} = vc;

// a valid json-ld @context.

const myCustomContext = require('./myCustomContext');

const documentLoader = extendContextLoader(async url => {

if(url === 'did:test:context:foo') {

return {

contextUrl: null,

documentUrl: url,

document: myCustomContext

};

}

return defaultDocumentLoader(url);

});

// you can now use your custom documentLoader

// with multiple vc methods such as:

const vp = await vc.signPresentation({

presentation, suite, challenge, documentLoader

});

// or

const signedVC = await vc.issue({credential, suite, documentLoader});

// or

const result = await vc.verifyCredential({credential: signedVC, suite, documentLoader});

[**Signing the Presentation**](https://github.com/digitalbazaar/vc#signing-the-presentation)

Once you've created the presentation (either via createPresentation() or manually), you can sign it using signPresentation():

const vp = await vc.signPresentation({

presentation, suite, challenge, documentLoader

});

console.log(JSON.stringify(vp, null, 2));

// ->

{

"@context": [

"https://www.w3.org/2018/credentials/v1"

],

"type": [

"VerifiablePresentation"

],

"verifiableCredential": [

{

"@context": [

"https://www.w3.org/2018/credentials/v1",

"https://www.w3.org/2018/credentials/examples/v1"

],

"id": "http://example.edu/credentials/1872",

"type": [

"VerifiableCredential",

"AlumniCredential"

],

"issuer": "https://example.edu/issuers/565049",

"issuanceDate": "2010-01-01T19:23:24Z",

"credentialSubject": {

"id": "did:example:ebfeb1f712ebc6f1c276e12ec21",

"alumniOf": "<span lang=\"en\">Example University</span>"

},

"proof": {

"type": "Ed25519Signature2018",

"created": "2020-02-03T17:23:49Z",

"jws": "eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..AUQ3AJ23WM5vMOWNtYKuqZBekRAOUibOMH9XuvOd39my1sO-X9R4QyAXLD2ospssLvIuwmQVhJa-F0xMOnkvBg",

"proofPurpose": "assertionMethod",

"verificationMethod": "https://example.edu/issuers/keys/1"

}

}

],

"id": "ebc6f1c2",

"holder": "did:ex:holder123",

"proof": {

"type": "Ed25519Signature2018",

"created": "2019-02-03T17:23:49Z",

"challenge": "12ec21",

"jws": "eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..ZO4Lkq8-fOruE4oUvuMaxepGX-vLD2gPyNIsz-iA7X0tzC3\_96djaBYDxxl6wD1xKrx0h60NjI9i9p\_MxoXkDQ",

"proofPurpose": "authentication",

"verificationMethod": "https://example.edu/issuers/keys/1"

}

}

[**Verifying a Verifiable Presentation**](https://github.com/digitalbazaar/vc#verifying-a-verifiable-presentation)

Pre-requisites:

* Your custom @contexts, verification methods (like public keys) and their corresponding controller documents are reachable via a documentLoader.

To verify a verifiable presentation:

// challenge has been received from the requesting party - see 'challenge'

// section below

const result = await vc.verify({presentation, challenge, suite, documentLoader});

// {valid: true}

By default, verify() will throw an error if the proof section is missing. To verify an unsigned presentation, you must set the unsignedPresentation flag:

const result = await vc.verify({

presentation, suite, documentLoader, unsignedPresentation: true

});

// {valid: true}

[**challenge parameter**](https://github.com/digitalbazaar/vc#challenge-parameter)

Verifiable Presentations are typically used for authentication purposes. A challenge param (similar to a nonce in OAuth2/OpenID Connect) is provided by the party that's receiving the VP, and serves to prevent presentation replay attacks. The workflow is:

1. Receiving party asks for the VerifiablePresentation, and provides a challenge parameter.
2. The client code creating the VP passes in that challenge (from the requesting party), and it gets included in the VP.
3. The client code passes the VP to the receiving party, which then checks to make sure the challenge is the same as the one it provided in the request in 1).

[**Verifying a Verifiable Credential**](https://github.com/digitalbazaar/vc#verifying-a-verifiable-credential)

For most situations, Verifiable Credentials will be wrapped in a Verifiable Presentation and the entire VP should be verified. However, this library provides a utility function to verify a Verifiable Credential on its own.

Pre-requisites:

* Your custom @contexts, verification methods (like public keys) and their corresponding controller documents are reachable via a documentLoader.

To verify a verifiable credential:

const result = await vc.verifyCredential({credential, suite, documentLoader});

// {valid: true}

To verify a verifiable credential with a custom @context field use a [custom documentLoader](https://github.com/digitalbazaar/vc#custom-documentLoader)

[**CLI**](https://github.com/digitalbazaar/vc#cli)

To use on the command line, see [vc-js-cli](https://github.com/digitalbazaar/vc-js-cli).

[**Testing**](https://github.com/digitalbazaar/vc#testing)

To run Mocha tests:

npm run test-node

To run Karma (in-browser) tests:

npm run test-karma

[**Contribute**](https://github.com/digitalbazaar/vc#contribute)

See [the contribute file](https://github.com/digitalbazaar/bedrock/blob/master/CONTRIBUTING.md)!

PRs accepted.

Note: If editing the Readme, please conform to the [standard-readme](https://github.com/RichardLitt/standard-readme) specification.

[**Commercial Support**](https://github.com/digitalbazaar/vc#commercial-support)

Commercial support for this library is available upon request from Digital Bazaar: [support@digitalbazaar.com](mailto:support@digitalbazaar.com)

[**License**](https://github.com/digitalbazaar/vc#license)

[New BSD License (3-clause)](https://github.com/digitalbazaar/vc/blob/main/LICENSE) © Digital Bazaar