# Self-sovereign Credit Record

## Description

The *Self-sovereign credit record (SSCR)* is a new type of decentralized identity built using W3C standards for decentralized identifiers (DIDs) and verifiable credentials (VCs). The SSCR is intended to represent a borrower's unique global identity and financial record, storing various general-purpose and protocol-specific verifiable credentials based on the borrower’s on-chain activity, trusted off-chain data, peer endorsement, financial health metrics, and others.

The SSCR contains both hard information (facts such as credit score and history, debt-to-income ratio, bank account verification, and business financial indicators) and soft information (such as endorsement, community membership, and self-declared business plans) that are used in the credit risk assessment.

## Characteristics

The verifiable credentials, stored in SSCR, have the following characteristics:

- *Issued by:* Trusted third parties (credential issuers)
- *Issued to:* Borrower
- *Owned by:* Borrower
- *Presented to:* Growr protocol (Project smart contract)
- *Verified by:* Credential verification service

## Implementation

The unique identity address is generated using an SSI framework such as Rif Identity. The verifiable credentials are stored in secure decentralized storage, such as Rif Data Vault.

## Credential types

### KYC credential

This credential proves a successfully passed KYC process (including AML/CFT risk check). It could be mandatory for certain regulated lenders to provide funding.

- Type: Hard information
- Used to prove: Identity
- Issued by: Distributor / Traditional third-party identity verification service

### Financial data credential

This credential contains various financial data of a user, such as products and transaction history. It shows the financial habits of the user, as well as the trends in his cash flow.

- Type: Hard information
- Used to prove: Creditworthiness
- Issued by: Distributor / Account servicing financial institution / Trusted financial data provider

## Business activity credential

This credential contains data about the business activity such as income statement, cash flow, and/or balance sheet. It can be used to prove past and future revenue.

- Type: Hard information
- Used to prove: Creditworthiness
- Issued by: Distributor / Trusted supplier or buyer / Trusted financial data provider

## Savings history credential

This credential proves that the user is making regular micro-payments to a savings account and thus shows the financial discipline of the user.

- Type: Hard information
- Used to prove: Creditworthiness
- Issued by: Distributor / Trusted financial institution / Savings account provider

## Credit history credential

This credential represents the borrower's past loans from the protocol (or from external trusted sources).

- Type: Hard information
- Used to prove: Creditworthiness & Trustworthiness
- Issued by: Growr protocol

## Credit score credential

This is a credential that summarizes other atomic credentials and represents the overall creditworthiness of the borrower.

- Type: Hard information
- Used to prove: Creditworthiness
- Issued by: Risk assessor

## Financial health credential

This is a special credential issued for successfully passed "financial health treatment" through education and/or mentoring in embedded financial health incentivization tools.

- Type: Soft information
- Used to prove: Trustworthiness
- Issued by: Financial health provider / Growr protocol

## Community membership credential

This credential asserts the membership of the user in a given organization and the trust of the community.

- Type: Soft information
- Used to prove: Reputation
- Issued by: Local organization (cooperative, union, chamber, employer)

## Social endorsement credential

This credential is received by endorsement from other protocol participants, who have a certain reputation level and/or are trusted by the protocol.

- Type: Soft information
- Used to prove: Reputation
- Issued by: Any protocol participant

# Projects

## Description

*Projects* are a key component of the decentralized marketplace. Each project represents a loan offer with predefined conditions and eligibility criteria. Every loan in the protocol is approved and disbursed from a project. Projects with similar risk parameters and goals are combined into lending pools.

## Characteristics

The project has the following main characteristics:

- *Initiated by:* Lender / Borrower / Distributor (on behalf of borrowers)
- *Owned by:* The initiator
- *Funded by:* Lender (direct first-loss capital) + Investors (through a lending pool)
- *Funding to:* Borrowers having the required set of verifiable credentials
- *Time period:* Fixed upon creation
- *Available amount:* Variable, depending on the funding operations
- *Limit amount:* Defined upon creation, could only be increased
- *Interest rate:* Fixed upon creation
- *Project-based collateral:* Yes, optional (as a reserved amount)

## Implementation

The protocol implements a *Project Factory smart contract* that enables the creation of new projects with varying credit line parameters and eligibility criteria. This smart contract is controlled by the Growr governance board (initially - Growr core developers) through a multi-sig account.
Project Factory contract functions:

- Create a project
- Destroy a project

Each project is created through the project factory as a separate *Project smart contract*. It enables users to apply for micro-loans from its funds. This smart contract is controlled by a lender.
Project contract functions:

- Deposit funds
- Withdraw funds
- Extend limit
- Activate lending
- Deactivate lending
- Close

## Project types

Projects may vary depending on the following specifics:

1. Initiation
   Each project can be initiated by (1) a lender, (2) a borrower or (3) a distributor on behalf of borrowers. The initiator becomes a project owner.
2. Funding
   Each project can be funded in one of the two possible payment approaches - (1) on-chain or (2) off-chain.
   In the *on-chain* payment approach, all transactions for depositing and withdrawing project funds are executed on-chain and processed by the protocol. In this scenario, the project might apply for additional liquidity from an automated lending pool. In the *off-chain* payment approach, project funds are only declared by the lender.
3. Reserves / Collateral
   Optionally, each project can have safety reserves - collateral used as a safety fund against defaulted loans. Those reserves are collected in 2 ways: (1) First-loss capital provided by the project owner upon creating the project or when extending its amount, or (2) Shared payments from the borrowers of the project (see next point). The reserved amount is released when the project is closed and after covering potential losses from defaulted loans.
4. Debt sharing
   In terms of loan repayment, the projects can operate in two models.
   In the first model (*individual*), each borrower repays only his/her loan(s) from the project and unpaid loans remain in default. In the second model (*community*), unpaid loans are compensated by overpaid amounts of the repaid loans.

# Loans

## Description

*Loans* represent the approved debts of each borrower. Each loan is released from a given project.

## Characteristics

The loan has the following main characteristics:

- *Requested by:* Borrower
- *Provided by:* Lender (through a project)
- *Approved by:* Credential verification service, matching project requirements with borrower's presented credentials
- *Time period:* Depends on the borrower's loan request and on project time limits
- *Amount:* Fixed upon loan creation based on the borrower's loan request and presented credentials
- *Interest rate:* Inherited by the project
- *Loan collateral:* Not required

## Implementation

Loans are implemented as records in a loan book in each project smart contract.

## Loan types

Loans may vary depending on the following specifics:

1. Payments model
   Each loan can be disbursed and then repaid in one of the two possible payment approaches - (1) on-chain or (2) off-chain.
   In the *on-chain* payment approach, all transactions for loan disbursement and repayment are executed on-chain and processed by the protocol. In the *off-chain* payment approach, the payment operations are settled by traditional payment providers, who issue and provide the protocol with "proofs-of-pay" to attest the payments are settled.
2. Repayment schedule
   Growr protocol supports 2 models for loan repayment. The first model is a *fixed repayment plan with monthly installments* and a predefined interest amount. The second model is with flexible *ad-hoc repayments*, and the interest amount is a variable calculated based on the period between disbursement and repayment(s). With the next protocol versions, we envision supporting other types of loans.

# Pools

## Description

Projects can be funded in 2 ways – (1) directly by the project owner (usually - the lender) or (2) by applying for funds from a lending pool. The latter approach requires the creation of global lending pools that group projects with similar risk levels and financing conditions. The lending pools play the role of investment funds for the projects.

## Characteristics

- *Owned by:* Protocol
- *Funded by:* Investors / DeFi protocols
- *Funding to:* Projects with similar risk levels and financing conditions
- *Time period:* Unlimited
- *Available amount:* Variable, depending on the funding operations
- *Utilized amount:* Variable, depending on the project utilization
- *Interest rate:* Fixed

## Implementation

The protocol has a *Pool Factory smart contract* that enables the creation of new pools combining projects with certain risk levels and financing conditions. This smart contract is controlled by Growr governance board (initially - Growr core developers) through a multi-sig account.
Pool Factory contract functions:

- Create a pool
- Destroy a pool

Each pool is created through the pool factory as a separate *Pool smart contract*. It enables investors to provide liquidity to the projects through those pools. The pool smart contracts are controlled by ... TBD.
Pool contract functions:

- Deposit funds
- Withdraw funds
- Activate funding
- Deactivate funding
- Close

## Pool types

Pools may vary depending on the following specifics:

1. Funding
   Each pool can attract liquidity in 2 ways - (1) direct deposits from investors or (2) integration with DeFi protocols.

# Credential Verification

When Borrowers apply for a loan from the marketplace, loan approval in the protocol is performed in an innovative decentralized manner. The goal of the approval process is to validate the credentials of the borrower and to assert his/her eligibility to receive a loan from a given project.

Smart contracts are usually not technically capable and economically practical at executing credit risk assessment operations themselves, and they cannot call upon external risk assessment services beyond the constraints of their chain. Therefore, the credit risk assessment through exchange and verification of credentials is executed off-chain and then confirmed on-chain in a way that no personal data is stored on-chain. That's why Growr protocol relies on two components to accomplish this task: *Credential Verification Service* and *Loan Assessment Registry*.

## Credential Verification Service

### Description

The Credential Verification Service checks whether the borrower matches the eligibility criteria of the project. It executes the following actions:

1. Check project requirements and verify that the borrower has all the necessary credentials
2. Verify the validity of each presented borrower's credential:

- Verify that the credential presentation is signed with the borrower's identity
- Verify that the credential is signed by a trusted credential issuer
- Verify that the credential is not expired
- Verify that the credential is not revoked

3. Create a lightweight and time-limited privacy-preserving *Verification Result* asserting that the given borrower matches the eligibility criteria of the given project.

### Implementation

This component is implemented as third-party services, trusted by the protocol. They are available through a discovery service and using a decentralized communication protocol.

### Functions

- Verify loan application

## Loan Assessment Registry

### Description

The Loan Assessment Registry is used to store the hashed signed verification result, generated by the Credential Verification Service. The registry does not contain any personal data or any credentials of the borrower to prevent leakage of sensitive personal information on-chain.

## Implementation

The protocol has a *Loan Assessment registry smart contract* that validates and stores verification result records.

We envision an alternative implementation scenario without this registry. In this credential verification flow, the signed verification result is returned to the borrower app, stored in his SSCR and then passed to the marketplace for approval of the loan.

# Payment Verification

Payment operations within a given project can be executed off-chain via traditional payment providers, who issue and provide the protocol with "proofs-of-pay" to attest the payments are settled. Each proof-of-pay contains information about the status of a given external financial transaction (eg. invoice payment).

Similar to credential verification, smart contracts are not capable of verifying payment operations outside their chain. Therefore, the payment settlement is verified off-chain and then confirmed on-chain. That's why Growr protocol relies on two components to accomplish this task: *Payment Verification Service* and *Payment Assessment Registry*.

## Payment Verification Service

### Description

The Payment Verification Service performs the verification of an external financial transaction. Depending on the payment channel, the verification may include one of the following:

- Confirming the existence of a blockchain transaction in another protocol
- Integration with trusted invoice validation services
- Integration with trusted payment processors (card providers or banks)

### Implementation

This component is implemented as third-party services, trusted by the protocol. They are available through a discovery service and using a decentralized communication protocol.

### Functions

- Confirm payment
- Reject payment

## Payment Registry

### Description

The Payment Registry is used to store a hashed signed payment proof, generated by the Payment Verification Service. The registry does not store on-chain any personal or financial data.

### Implementation

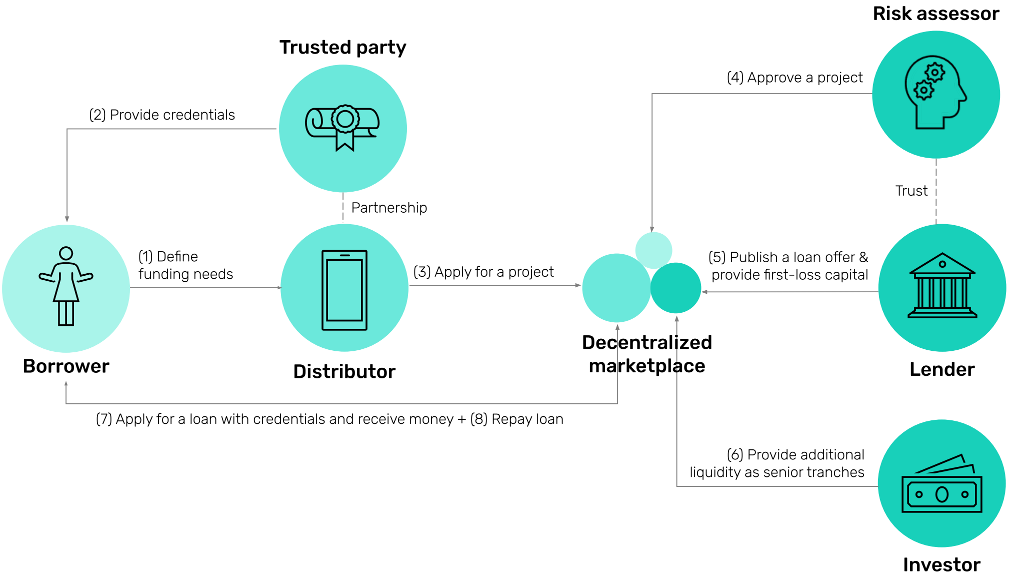
The protocol has a *Payment registry smart contract* that validates and stores payment proof records.

We envision an alternative implementation scenario without this registry. In this payment verification flow, the signed payment proof is returned to the borrower app, stored in his SSCR and then passed to the marketplace for repayment of the loan.

# Mechanics Overview

## Protocol Processes

The following diagram provides a high-level overview of the Growr protocol mechanics.


Risk assessor
Trusted party
(4) Approve a project
(2) Provide credentials
Trust
Partnership
(1) Define funding needs
(3) Apply for a project
(5) Publish a loan offer & provide first-loss capital
Borrower
Distributor
Decentralized marketplace
Lender
(7) Apply for a loan with credentials and receive money + (8) Repay loan
(6) Provide additional liquidity as senior tranches
Investor

Below is a summary of the protocol processes depicted on the diagram:

1. The *borrower* defines their funding needs to or in collaboration with a *distributor*, such as a cooperative, guild, wholesale buyer, digital wallet, or another provider. See the Borrower Onboarding section.
2. One or more *trusted parties* provide credentials to the *borrower* to start building their self-sovereign credit record. See the Borrower Credit Record section.
3. The *distributor* applies for a project with details about the local activities and the financing needs of the borrowers, to receive a credit line from the *decentralized marketplace*.
4. A *risk assessor*, working for a *lender*, or a third party trusted by them, reviews each application, to assess the risk and determine the loan price and conditions.
5. A *lender* (a bank, non-banking lender, or a fintech) approves the credit line if it fulfills the risk policy. Upon approval, the *lender* creates a loan offer in the form of a project on the *decentralized marketplace* with predefined eligibility criteria. See the Project Setup section.
6. *Investors* can provide additional funding to the project as a senior tranche via the *decentralized marketplace*, fully delegating the actual lending activity to the lender and the protocol, according to their risk appetite. See the Project Funding section.
7. *Borrowers* go through a simple application process to receive a loan from the *decentralized marketplace* after asserting their eligibility with their verifiable credentials. See the Loan Application section. The disbursed amount is received by the *borrowers* in the borrowing application, where they can make a direct payment to a merchant or use it in another way to achieve their goals. See the Loan Disbursement section.
8. *Borrowers* can view and manage every aspect of their loans. They can easily pay off the loan, which helps them build their credit record and receive future loans with better conditions. See

the Loan Repayment section.

9. Upon project closure, the collected amount is repaid with the following priority: 1) senior tranches (principal and yield) to *investors*, 2) junior tranches (principal and yield) to *lenders*, 3) rewards to *borrowers*. See the Project Closure section.

The Growr protocol aims at standardization of the protocol mechanics. However, depending on the protocol participants and the tools and services they use, implementation details might vary. In general, we can distinguish the following use case specifics.

- *Distribution*: access to the protocol is provided in a custodial model by an independent distributor, or in a non-custodial model directly using a decentralized borrowing application.
- *Risk assessment*: performed by a lender with an internal risk assessment function, or by an independent trusted risk assessor.
- *Funding*: provided partially by a lender with additional capital from global investors, or fully by the lender.
- *Loan payments*: they are processed on-chain as part of the protocol mechanics or off-chain.

## Protocol participants

### Borrowers

*Borrowers*, represented by self-employed, micro-businesses, and smallholder farmers, apply for productive loans from the marketplace, most often with the help of a local distributor, and then repay the loan plus its price.

### Distributors

*Distributors* facilitate access to the protocol by grouping several borrowers with similar needs and presenting project applications to the marketplace on their behalf. They can be:

- *Local cooperatives, guilds or other community organizations* that are formed by borrowers to gain better access to loans and to standardize their relationship with the rest of the participants in the ecosystem.
- *Federated mints* [21], enabling access to micro-lending to their users.
- *Telcos, retailers and gig-economy platforms* that onboard and vet the users into their services and then facilitate their access to the protocol as embedded financial services.
- *Digital wallets and fintech providers* that already offer financial services and can expand to unsecured decentralized lending.

### Trusted parties

*Trusted parties* assert facts about the borrowers in the form of verifiable credentials. They can be:

- The *distributor*, issuing credentials for its members or users.
- *Merchants, buyers, unions, chambers or other local organizations* that serve the community or have knowledge of their members.

- *Independent third-party data providers* that can issue credentials related to the activity of the borrower and relevant to the risk assessment process, such as KYC/AML, account data, and on-chain activity.
- *Financial health providers* that publish educational materials and tools to help borrowers develop good financial habits and issue credentials that assert knowledge, skills, and accomplishments.

## Lenders

*Lenders* publish offers to the loan marketplace by creating and funding *projects*. The lenders provide first-loss capital to finance the whole project or part of it.

## Risk Assessors

*Risk assessors* review project applications, assess the risk and determine the loan price and parameters. In most cases, this role is internally covered by the lender.

## Investors

*Investors* can be decentralized finance protocols or large institutional or individual liquidity providers who allocate capital to the loan marketplace through liquidity pools and delegate the actual lending activity to the lenders in the form of senior tranches for the financed projects.

## Decentralized marketplace

The *global decentralized micro-lending marketplace* is the open-source smart contract infrastructure and the tools for decentralized lending.
# Borrower Onboarding

The Growr protocol does not enforce any specific requirements upon wallet implementation. Wallet developers may design features and functionalities that correspond to the desired user experience.

In a custodial model, the *borrowers* onboard using a mobile or web application provided by a *distributor*, such as telcos, retailers, digital wallets and fintech providers. The distributor onboards the users into their application and then facilitates their access to the protocol. The private keys of the user wallets are managed by the distribution app.

In a non-custodial model, the *borrowers* would use a borrowing dApp to create and manage their self-custody wallet, then claim credentials and apply for a loan from the protocol without involving a third party.

During onboarding, the key step is to assign a unique DID address to every *borrower*. The address is generated using an SSI framework.

Below is a diagram that depicts the process:

```
sequenceDiagram
    participant Borrower
    participant Borrowing App
    participant SSCR
    Note over Borrowing App: Custodial distribution app or Self-custody wallet
    Borrower->>Borrowing App: Sign-up
    activate Borrowing App
    alt custodial model?
        Borrowing App->>Borrower: Request verification (eg. OTP)
        Borrower->>Borrowing App: Provide verification
    end
    Borrowing App->>SSCR: Initialize DID address
    alt non-custodial model?
        Borrowing App->>Borrower: Provide private keys
    end
    deactivate Borrowing App
    Note over Borrower: Start building his credit record
```

# Borrower Credit Record

## Credit Record and Verifiable Credentials

The Growr protocol relies on a new type of decentralized identity that we call the **Self-sovereign Credit Record (SSCR)**. The *SSCR* is intended to represent a borrower's unique global identity and financial record, storing various general-purpose and protocol-specific verifiable credentials.

The *SSCR* contains both hard information (facts such as credit score and history, debt-to-income ratio, bank account verification, and business financial indicators) and soft information (such as endorsement, community membership, and self-declared business plans) that are used in the credit risk assessment.

The verifiable credentials, stored in *SSCR*, are issued by trusted third parties called *credential issuers*. The credentials are issued and owned by the *borrower*, who is the subject of the credential. The *borrower* presents a given set of his credentials when applying for a loan from the marketplace.

Below is a high-level diagram that depicts the lifecycle of the verifiable credentials:

```
sequenceDiagram
    participant Borrowing App
    participant CIS as Credential Issuing Service
    participant SSCR
    participant CVS as Credential Verification Service
    Note over CIS: Issuing verifiable credentials
    Borrowing App->>CIS: Claim credential
    CIS->>CIS: Issue credential
    CIS->>Borrowing App: Provide credential
    Borrowing App->>SSCR: Store credential
    Note over CVS: Verifying credentials
    SSCR->>Borrowing App: Provide credential
    Borrowing App->>CVS: Present credential
    CVS->>CVS: Verify credential
    CVS->>Borrowing App: Provide result
```

## Credential Issuing

Below is a standard process for issuing verifiable credentials:

```
sequenceDiagram
    participant Borrower
    participant Borrowing App
    participant CIS as Credential Issuing Service
    participant SSCR
    Note over Borrowing App: Custodial distribution app or Self-custody wallet
    Borrower->>Borrowing App: Provide data
    activate Borrowing App
    Borrowing App->>CIS: Request credential requirements & fee
    activate CIS
    CIS-->>Borrowing App: Provide credential offer
    Borrowing App->>CIS: Claim credential
    CIS->>Borrowing App: Request proof
    Borrowing App->>CIS: Provide signed proof
    CIS->>CIS: Issue credential
    CIS->>Borrowing App: Return credential
    deactivate CIS
    Borrowing App->>SSCR: Store credential
    Borrowing App-->>Borrower: "Issued credential"
    deactivate Borrowing App
```

## Credential Revocation

*Credential issuers* might need to revoke a certain credential before it expires, eg. when the subject is no longer eligible for the credential or when they need to correct or update the information in the credential. To achieve this, they must implement the W3C's architecture design for Status List. According to this pattern, during credential issuing, the issuers would embed in the credential a URL to fetch a revocation list and the index in this list that corresponds to the given credential.

## Credential Verification

Below is a standard process for verifying credentials:

```
sequenceDiagram
    participant Borrower
    participant Borrowing App
    participant CVS as Credential Verification Service
    participant SSCR
    participant TSR as Trusted Service Registry
    participant CSL as Credential Status List
    Note over Borrowing App: Custodial distribution app or Self-custody wallet
    Borrower->>Borrowing App: Request credentials verification
    activate Borrowing App
    Borrowing App->>SSCR: Get credentials
    Borrowing App->>Borrowing App: Build and sign credential presentation
    Borrowing App->>CVS: Send credential presentation
    deactivate Borrowing App
    activate CVS
    CVS->>CVS: Verify borrower's signature
    CVS->>CVS: Verify issuer's signature
    CVS->>TSR: Check issuer
    CVS->>CVS: Verify credential validity
    opt
        CVS->>CSL: Check credential status
    end
    alt If OK?
        CVS->>Borrowing App: Verification result
        activate Borrowing App
        Borrowing App-->>Borrower: "Verified credential"
        deactivate Borrowing App
    else If not OK
        CVS->>Borrowing App: Verification failed
        activate Borrowing App
        Borrowing App-->>Borrower: "Invalid credential"
        deactivate Borrowing App
    end
    deactivate CVS
```

# Loan Application

## Loan Offering

The *borrowers* request a loan offer from the decentralized marketplace using a custodial or non-custodial borrowing application.

There are different scenarios for getting the loan offers, depending on the protocol implementation, and in particular - how the project details are stored (cached). In the presented process below, the *Project Factory smart contract* holds only a list of addresses of the *Project smart contracts* and the *borrowing application* must "ask" each *Project smart contract* for its details.

```
sequenceDiagram
    participant Borrower
    participant Borrowing App
    participant PRFC as Project Factory contract
    participant PRC as Project contract
    Note over Borrower: Borrower has credentials
    Borrower->>Borrowing App: Apply for loan
    activate Borrowing App
    Borrowing App->>PRFC: Request project list
    activate PRFC
    PRFC-->>Borrowing App: Provide project list
    deactivate PRFC
    loop Get offers from projects
        Borrowing App->>PRC: Request project details
        activate PRC
        PRC->>PRC: Check eligibility
        PRC-->>Borrowing App: Provide parameters for potential loan
        deactivate PRC
    end
    Borrowing App->>Borrowing App: Compare offers
    Borrowing App->>Borrower: Provide loan parameters
    deactivate Borrowing App
```

## Loan Approval

When *borrowers* apply for a loan from the decentralized marketplace, the loan approval process in the protocol is performed in an innovative decentralized manner. The goal is to validate the credentials of the borrower and to assert his/her eligibility to receive a loan from a given project.

Smart contracts are usually not technically capable and economically practical at executing verification operations themselves, and they cannot call upon external services beyond the constraints of their chain. Therefore, the loan approval through exchange and verification of credentials is executed off-chain and then confirmed on-chain in a way that no personal data is stored on-chain.

In the loan approval process, the responsibility is split as follows:

- *Risk Assessors*, owned or trusted by the *Project Owner* determine project risk parameters, required credentials and whitelisted verification services.

- *Credential Issuers* assert facts about the borrowers in the form of credentials.
- *Credential Verification Service* validates the credentials and asserts the borrower's eligibility to receive a loan from a given project.
- *Project smart contracts* validate the signatures of the borrower and verification service and ultimately approve the loan.

Below is a diagram that describes the process:

```
sequenceDiagram
    participant Borrower
    participant Borrowing App
    participant CVS as Credential Verification Service
    participant PRC as Project contract
    Note over Borrowing App: Custodial distribution app or Self-custody wallet
    Note over Borrower: ...Borrower has credentials and selected loan offer
    Borrower->>Borrowing App: Accept loan offer
    activate Borrowing App
    Borrowing App->>Borrowing App: Create credential presentation
    Borrowing App->>CVS: Present credentials
    activate CVS
    CVS->>PRC: Get project requirements
    CVS->>CVS: Verify credentials
    Note right of CVS: See: Credential Verification process
    alt credentials not OK?
        CVS-->>Borrowing App: Failed verification
        Borrowing App-->>Borrower: "Not valid credentials"
    end
    CVS->>CVS: Validate eligibility
    alt requirements not met?
        CVS-->>Borrowing App: Failed eligibility check
        Borrowing App-->>Borrower: "Not eligible for the loan"
    end
    CVS->>CVS: Issue & sign verification result
    CVS-->>Borrowing App: Return verification result
    deactivate CVS
    Borrowing App->>PRC: Provide verification result and apply for loan
    activate PRC
    PRC->>PRC: Validate signatures
    PRC->>PRC: Check loan amount and duration
    PRC->>PRC: Check for available funds in the project
    alt result not OK?
        PRC-->>Borrowing App: Loan rejected
        Borrowing App-->>Borrower: "Loan rejected"
    else result OK?
        PRC->>PRC: Approve loan and add to loan book
        alt off-chain payment model?
            PRC-->>Borrowing App: Loan approved
            Borrowing App-->>Borrower: "Loan approved"
            Note over PRC: See: Loan Disbursement process
        else on-chain payment model?
            PRC->>Borrowing App: Disburse loan amount
            Borrowing App-->>Borrower: "Loan approved and disbursed"
        end
    end
    deactivate PRC
    deactivate Borrowing App
```

## Loan Parameters

Once the loan is approved, the *Project smart contract* registers it in its loan book and returns a response with the following loan parameters:

- *Owner:* Address of the borrower
- *Approved loan amount:* requested by the borrower but if outside of the predefined project range, then it is the nearest acceptable
- *Approved duration (in months):* requested by the borrower but if outside of the predefined project range, then it is the nearest acceptable
- *Interest rate (APR) %:* Depending on project parameters and borrower's credentials
- *Interest amount:* Formula = Amount x APR x Duration in months / 12
- *Total amount to be repaid:* Formula = Amount + Interest amount
- *Installment amount:* Formula = Total amount / Duration in months
- *Document Ref Id:* Id of a payment document that will assert the loan disbursement (in case of an off-chain payment model)

Loan parameters cannot be changed after the loan is created.

## Privacy

Upon successful completion of all verification checks, the *credential verification service* creates a lightweight privacy-preserving **verification result** asserting that a given *borrower* matches the eligibility criteria of a given project. The *credential verification services* hashes and signs the result such as it does not contain any information from the borrower's credentials, thus preventing the leakage of any sensitive personal information on-chain.

In addition, the **loan book** stored on-chain on the protocol does not contain any personal data, except the address of the borrower's wallet. In the future protocol implementation, we also envision the usage of the zero-knowledge proof (ZKP) model, in which even the SSCR address of the user is not revealed.

# Loan Disbursement

Loans can be disbursed in two ways depending on the *payment model* parameter of the project.

## On-chain Disbursement

In the *on-chain* payment model, the disbursement transaction is executed on-chain immediately after loan creation (as part of the Loan Approval process).

## Off-chain Disbursement

In the *off-chain* payment model, the disbursement transaction is settled by a traditional payment provider (payment processor) and it is recorded in the protocol after the *payment verification service* issue a "proof-of-pay" credential.

Below is a diagram that describes the process:

```
sequenceDiagram
    participant Lending App
        participant PVS as Payment Verification service
    participant PRC as Project contract
    participant PP as Payment Processor
    Note over Lending App: Could be any protocol participant
    Note over PRC: ...Loan with off-chain payments is approved
    Lending App->>PVS: Check payment document status
    activate PVS
        PVS->>PP: Verify external transaction
        PVS->>PVS: Issue and sign a verification result
        alt payment is cancelled?
            PVS->>PRC: Provide result for cancelled payment
            activate PRC
            PRC->>PRC: Validate signature
            PRC->>PRC: Cancel a loan
            PRC-->>PVS: "Loan cancelled"
            deactivate PRC
            PVS-->>Lending App: "Loan cancelled"
        else payment is executed?
            PVS->>PRC: Provide result for executed payment
            activate PRC
            PRC->>PRC: Validate signature
            PRC->>PRC: Mark loan as disbursed
            PRC-->>PVS: "Loan disbursed"
            deactivate PRC
            PVS-->>Lending App: "Loan disbursed"
        end
    deactivate PVS
```

The *payment verification service* performs the verification of an external financial transaction. Depending on the payment channel, the verification may include one of the following:

- Confirming the existence of a blockchain transaction in another protocol
- Check payment document status through an integration with an invoice validation service

- Check transaction status through integration with a payment processor (a card provider or a bank)

# Loan Repayment
The loan repayment process depends on several project parameters:

- *Payment Model:* On-chain/Off-chain
- *Community Model:* Yes/No
- *Repayment model:* Fixed monthly schedule / Ad-hoc

In the *on-chain* payment model, the repayment transaction is executed on-chain through a transfer between the borrower and the project smart contract. In the *off-chain* payment model, the repayment transaction is settled by a traditional payment provider (payment processor) and it is recorded in the protocol after the *payment verification service* issue a "proof-of-pay" credential.

If the loan is *individual* (community model = no), each borrower repays only his/her loan(s) and any unpaid loans in the project remain in default. If the loan follows the *community model*, the unpaid loans are compensated by additional risk deposits, collected on top of the repaid loans.

If the loan has a *fixed repayment plan* the interest amount is predefined and the repayment amount (principal + interest) is split into several monthly installments. If the loan has a flexible *ad-hoc repayment* model, then the interest amount is a variable calculated based on the period between disbursement and repayment.

## On-chain Repayment

Below is a diagram that describes the process:

```
sequenceDiagram
        participant Borrower
    participant Borrowing App
    participant PRC as Project contract
    Note over Borrowing App: Custodial distribution app or Self-custody wallet
        Borrower->>+Borrowing App: Initiate repayment
    Borrowing App->>+PRC: Transfer repayment amount
        Deactivate Borrowing App
        PRC->>PRC: Process repayment
    alt Ad-hoc repayments model?
        PRC->>PRC: Calculate interest amount
    end
    alt Community loan?
        PRC->>PRC: Calculate risk deposit amount
    end
    PRC->>PRC: Register repayment transaction
    alt Final repayment?
        PRC->>PRC: Determine return amount
    end
    alt Has return amount?
        PRC->>Borrowing App: Return amount
    end
    PRC-->>+Borrowing App: "Successful repayment"
    Borrowing App-->>-Borrower: "Successful repayment"
    deactivate PRC
```

For loans with a *fixed repayment plan*, the *Project smart contract* performs the following processing:

- If repaid amount = installment amount -> normal installment repayment
- If repaid amount < installment amount -> partial installment repayment
- If repaid amount > installment amount -> normal + partial installment repayment
- If repaid amount = remaining loan amount -> final loan repayment
- If repaid amount > remaining loan amount -> final loan repayment + overpaid amount (amount for return)

For loans with a flexible *ad-hoc repayment* model, the *Project smart contract* first calculates the accrued interest amount before performing the processing steps above:

- Interest amount = Principal (disbursed amount) * APR % * Duration in days / 360
- Installment amount = Principal + Interest amount

For *community loans*, the *Project smart contract* calculates and holds a deposit risk amount:

- Deposit risk amount = Principal * Risk deposit rate %

For loans with a *cash-back* incentive that are fully repaid on-time, the *Project smart contract* calculates the cash-back reward:

- Cash-back reward = Principal * Cash-back rate %

In case the Borrower qualifies for a cash-back payment, the *Project smart contract* repays the amount to the *borrower*. Alternatively, to promote better financial discipline, this cash-back amount can be time-locked and used as a verifiable credential in further loan applications.

## Off-chain Repayment

In the *off-chain* payment model, the repayment transaction is settled by a traditional payment provider (payment processor) and it is recorded in the protocol after the *payment verification service* issue a "proof-of-pay" credential.

Below is a diagram that describes the process:

```
sequenceDiagram
    participant Borrowing App
        participant PVS as Payment Verification service
    participant PRC as Project contract
    participant PP as Payment Processor
    Note over Borrowing App: Could be any protocol participant
    Borrowing App->>+PP: Request payment
    activate Borrowing App
    PP->>PP: Execute payment
    PP-->>-Borrowing App: "Payment status"
    Borrowing App->>PVS: Check payment document status
    deactivate Borrowing App
    activate PVS
        PVS->>PP: Verify external transaction
        PVS->>PVS: Issue and sign a verification result
        alt payment is cancelled?
            PVS-->>Borrowing App: "Payment cancelled"
        else payment is executed?
```

```
            PVS->>PRC: Provide result for executed payment
            activate PRC
            PRC->>PRC: Validate signature
            Note over PRC: Loan Repayment process
            PRC-->>PVS: "Loan repaid"
            deactivate PRC
            PVS-->>Borrowing App: "Loan repaid"
        end
    deactivate PVS
```

## Loan Book

*Project smart contracts* are responsible to keep track of the entire loan history. The contracts hold a chain of encrypted "repayment commitments" registered with each disbursement and repayment transaction.

Each record in the loan book contains information about the borrower's SSCR address, Loan Id, accrued commitment and due date. When a *borrower* repays part of the loan, the record is marked as expired/invalid and a new record with updated information is registered in the book. When a *borrower* fully repays the loan, the last record is invalidated and no new commitment record is registered.

Expired commitment records in the book will be indicators of positive repayment history. Valid (active) commitment records with a due date in the past will be indicators of overdue loans. Since all *Project smart contracts* use the same loan book structure, the latter can be used to determine the entire credit history of a given *borrower*.

The data stored in the loan book does not contain any personal data and is encrypted in a way that only the key owner of the SSCR address (the *borrower*) can decrypt his/her credit history.

# Project Setup

## Process

A key component in Growr protocol is the lending *project*. The project represents a loan offer with predefined conditions and eligibility criteria and every loan in the protocol is approved and disbursed through a project.

Each project can be initiated by a *lender*, a *borrower* or by a *distributor* on behalf of borrowers. The initiator becomes a project owner.

Each project defines a set of verifiable credentials, required from the *borrowers* to get an uncollateralized loan. Some projects might require partial collateral, collected as risk deposit payments from all project borrowers.

The diagram below illustrates the process of project creation and setup:

```
sequenceDiagram
    participant Project Owner
    participant Lending App
    participant PFC as Project Factory contract
    participant PC as Project contract
    Note over Project Owner: Lender, Borrower or Distributor
    Project Owner->>Lending App: Initiate project creation
    activate Lending App
    Project Owner->>Lending App: Configure project parameters
    Project Owner->>Lending App: Configure project eligibility criteria
    opt optional
        Project Owner->>Lending App: Configure credential verification service address
    end
    opt optional
        Project Owner->>Lending App: Configure payment verification service address
    end
    Lending App->>+PFC: Provide details and request project creation
    PFC->>PFC: Validate parameters
    PFC->>PC: Create project
    opt Pool funding?
        Note over PC: Request funds from lending pool
    end
    PFC-->>-Lending App: "Project address"
    Lending App-->>Project Owner: "Project created"
    deactivate Lending App
    Note over Project Owner: Project ready for funding
```

## Project Parameters

The parameters for project creation include:

- *Owner:* Address of the wallet that controls the project
- *Name:* Descriptive name of the project, used for presentation purposes only
- *Payment Model:* On-chain/Off-chain
- *Currency/Token:* Used when the payment model is "on-chain"

- *Payment Verification Address*: Address of trusted service(s) that provide proof of external financial transactions in case of the "off-chain" payment model
- *Eligibility criteria:* Defines a set of verifiable credentials, requested upon loan application
- *Credential Verification Address:* Address of trusted service(s) that check whether a borrower matches the project's eligibility criteria
- *Limit:* Defines the total project amount available for disbursement
- *Min & max loan amount:* Could be fixed or formula depending on the borrower's credentials
- *Min & max loan duration:* Fixed upon creation
- *Annual interest rate:* Determines the price of the loan
- *Disbursement fee:* Optional fee
- *Community Model:* Yes/No
- *Risk deposit rate:* Applicable for community loans and determines the price of the shared risk
- *Cash-back rate:* Incentive for on-time loan repayment
- *Repayment model:* Fixed monthly schedule / Ad-hoc
- *Pool funding:* Yes/No
- *Pool address:* Address of a lending pool, from which the project will request funding
- *Leverage factor:* Used only for projects funded through lending pools

Once created, project parameters cannot be changed. The only parameter that is subject to a change is the project limit and it could only be increased. In addition, the *Project smart contract* provides a possibility for the owner to stop or pause the lending operations within the project i.e. to permanently or temporarily block future loan approvals.

# Project Funding

Projects are funded by their owners who provide the first-loss capital, as well as by *investors* who provide additional senior tranches.

Projects can be funded in several ways depending on 2 project parameters:

- *Payment Model:* On-chain/Off-chain
- *Pool funding:* Yes/No

In the *on-chain* payment model, all transactions for depositing and withdrawing project funds are executed on-chain and processed by the protocol. In the *off-chain* payment approach, project funds are only declared by the project owner but all transactions are executed off-chain.

If the *pool funding* parameter is activated for a given project, it enables the project to receive additional liquidity (senior tranches) from an automated lending pool.

## Direct funding (On-chain junior tranches)

Junior tranches are always provided by the owner of the project. Available project funds (non-utilized amount or generated yield) can also be withdrawn only by the project owner. Below is a diagram that describes the process:

```
sequenceDiagram
   participant Project Owner
   participant Lending App
   participant PC as Project contract
   participant PFC as Project Factory contract
   Note over PC: ...Project is created
   Project Owner->>Lending App: Initiate project funding
   activate Lending App
   Lending App->>PFC: Get project address
   Lending App->>+PC: Deposit funds to project address
   PC->>PC: Register deposit transaction
   PC-->>-Lending App: Return deposit receipt
   Lending App-->>Project Owner: "Successful deposit"
   deactivate Lending App
   PC->>PC: Generate yield through lending
   Project Owner->>Lending App: Request withdrawal
   activate Lending App
   Lending App->>+PC: Request withdrawal
   PC->>PC: Check available funds
   opt Pool funding?
       PC->>PC: Check leverage factor
       PC->>PC: Check profitability rate
   end
   PC->>PC: Register withdrawal transaction
   PC->>-Lending App: Send fund + yield
   Lending App-->>Project Owner: "Successful withdrawal"
   deactivate Lending App
```

## Lending pools (On-chain senior tranches)

Projects with activated *pool funding* can apply for funds from a global lending pool resulting in so-called "contract-to-contract (C2C) lending". Each lending pool combines projects with similar risk levels and financing parameters.

Below is a high-level diagram that depicts the lifecycle of the lending pools:

```
sequenceDiagram
    participant Lending App
    participant POFC as Pool Factory contract
    participant POC as Pool contract
    participant PRC as Project contract
    Note over POFC: Pool creation
    Lending App->>POFC: Request pool creation
    POFC->>POC: Create pool
    Note over POC: Pool funds deposit
    loop
        Lending App->>POC: Deposit funds
    end
    Note over PRC: Pool-to-project funding
    loop
        PRC->>POC: Request funds
        POC->>PRC: Approve request and provide funds
    end
    loop
        PRC->>POC: Return funds + yield
    end
    Note over POC: Pool funds withdrawal
    loop
        Lending App->>POC: Request withdrawal
        POC->>Lending App: Approve request and provide funds + yield
    end
```

The process for *pool creation* is very similar to the process for project creation. The parameters for pool creation include:

- *Owner:* Address of the wallet that controls the pool
- *Name:* Descriptive name of the pool, used for presentation purposes only
- *Currency/Token*
- *Funding criteria:* Defines a set of rules, required for approval of a project (can contain a predefined value or a value range restricting any of the project parameters)
- *Limit:* Defines the total pool amount available for project funding
- *Min & max project amount:* Could be fixed or formula depending on the project's parameters
- *Annual interest rate:* Determines the price of the loan

Once created, pool parameters cannot be changed. The only parameter that is subject to change is the pool limit. In addition, the *Pool smart contract* provides a possibility for the owner to stop or pause the lending operations within the pool i.e. to permanently or temporarily block future projects funding.

The processes for *pool deposit and withdrawal* are very similar to the respective processes on the project level.

The process for *pool-to-project funding* is presented in more detail in the next diagram:

```
sequenceDiagram
    participant Lending App (Lender)
    participant PRFC as Project Factory contract
    participant PRC as Project contract
    participant POC as Pool contract
    participant Lending App (LP)
    Note over POC: ...Pool is created and funded
    Note over PRC: ...Project with pool funding is created
    PRC->>POC: Register funding request
    activate POC
    POC->>POC: Verify project eligibility
    POC->>POC: Approve request and whitelist the project
    deactivate POC
    Note over PRC: Utilization above %
    activate PRC
    PRC->>+POC: Request amount
    POC->>POC: Approve request
    POC->>-PRC: Disburse principal amount
    deactivate PRC
    loop Regular interest repayment
        PRC->>POC: Repay interest
    end
    Note over PRC: Utilization below %
    activate PRC
    PRC->>+POC: Repay principal
    POC-->>-PRC: Confirm repayment
    deactivate PRC
```

If a project owner considers using liquidity from a global lending pool, he must activate the pool funding during project setup. He must also configure the pool's address so the newly created *Project smart contract* will automatically register a funding request on the *Pool smart contract*.

The *Pool smart contract* verifies the project parameters and approves the request by whitelisting the project address.

Based on the project utilization, the *Project smart contract* can automatically apply for funds to the *Pool smart contract*. The approved funds have 2 limits - the first one is defined as a parameter of the pool and the second one is a formula (Project Balance * Leverage Factor).

The *project* is required to regularly repay the interest amount to the *pool*. Depending on the configured utilization threshold, the *project* can also partially or fully repay the principal to the *pool*.

This process ensures automatic liquidity management between the *Pool smart contract* and the whitelisted *Project smart contracts*.

# Project Closure

Projects can be closed either by their owner (only when there are no active loans) or automatically by the *Project Factory smart contract* when the following conditions are met:

- The project has expired (no more loans are approved)
- All approved loans are either repaid or expired
- The overdue repayment period (if configured) has ended

Upon project closure and in the case of an *on-chain payment model*, the *Project smart contract* calculates amounts and repays the project balance as described in the diagram below:

```
sequenceDiagram
   participant PRFC as Project Factory contract
   participant PRC as Project contract
   participant POC as Pool contract
   participant Project Owner
   participant Borrower
   PRFC->>PRC: Request closure
   activate PRC
   PRC->>PRC: Check closing conditions are met
   PRC->>PRC: Calculate repayment amounts
   opt Pool funding?
       PRC->>POC: Transfer outstanding senior tranche principal and interest
   end
   opt Project repayment ratio > threshold?
       loop For each borrower
           PRC->>Borrower: Transfer outstanding risk deposits proportionally
           opt Individual loan repaid on-time?
               PRC->>Borrower: Transfer "cash-back" reward
           end
       end
   end
   PRC->>Project Owner: Transfer outstanding junior tranche principal and interest
   PRC-->>PRFC: "Ready for closure"
   deactivate PRC
   PRFC->>PRC: Destroy the project
```