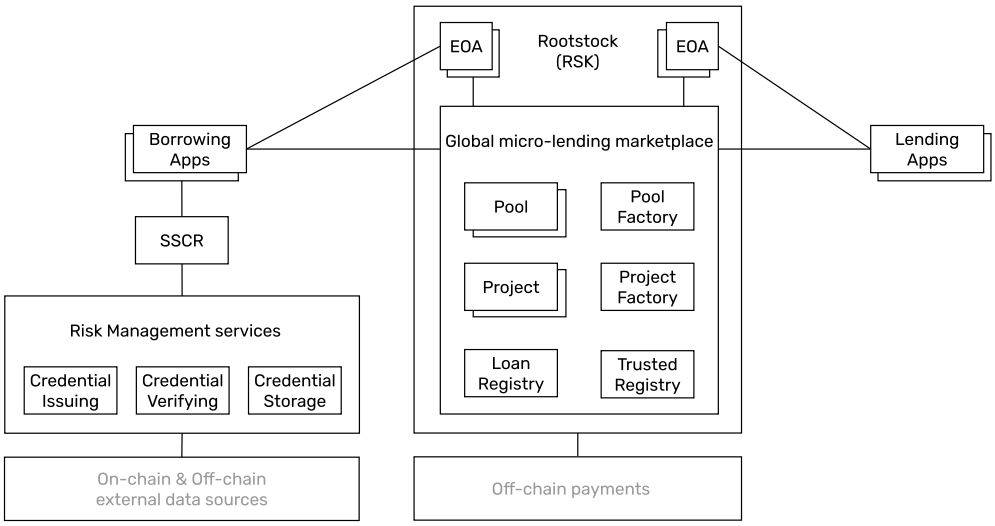
Overview

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



The **Decentralized micro-lending marketplace** is where lenders publish their loan offers with predefined conditions and eligibility criteria, and borrowers apply to get financing based on the automatic matching of these criteria with the credentials in their self-sovereign credit record. It is implemented as a smart contract system on top of Rootstock (RSK) mainnet. More details are available in the *Projects*, *Loans* and *Pools* sections.

The **Self-sovereign credit record (SSCR)** is a unique global decentralized identity containing general-purpose and protocol-specific verifiable credentials. More details are available in the *Self-sovereign Credit Record* section.

The **Protocol apps** are custodial or non-custodial web and mobile applications, integrated with the protocol. **Borrowing (distribution)** apps are end-user web or mobile applications for the borrowers to onboard, collect credentials and apply for loans to the protocol. Such applications can be provided either by an independent financial service provider in a regulated custodial scenario, by local communities, or as a completely decentralized app providing the necessary access to the protocol. **Lending** apps are web applications for the management of lending pools and lending projects.

The marketplace and the applications are integrated with various internal or third-party services, covering mainly risk management functions, such as **credential issuing services**, **credential verification services** (more details are available in the *Credential Verification* section), **credential storage services** and **payment verification services** (more details are available in the *Payment Verification* section).

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 channel
   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 *Verification registry smart contract*...

TBD - do we need this registry? We have an alternative - the signed result can be returned to the borrower, stored in his SSCR and passed to the protocol by the borrower himself...

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 implements a *Payment registry smart contract*...

TBD - do we need this registry? We have an alternative - the signed result can be returned to the initiator of the payment verification and then passed to the protocol by himself...