RoC V1 to V2 migration plan overview

Date : April 4th, 2024

INTRO

The current ROC (RIF on Chain) running implementation, which uses MOC V1 protocol, will be migrated into a new MOC V2 protocol standard. The new MOC V2 protocol incorporates a new tech stack, new features as well as simplifications to current features to ease any other future development and ecosystem growth.

This document will give a technical overview on how this migration will be conducted, how it will be articulated, and how it impacts integrators and final users.

GOAL

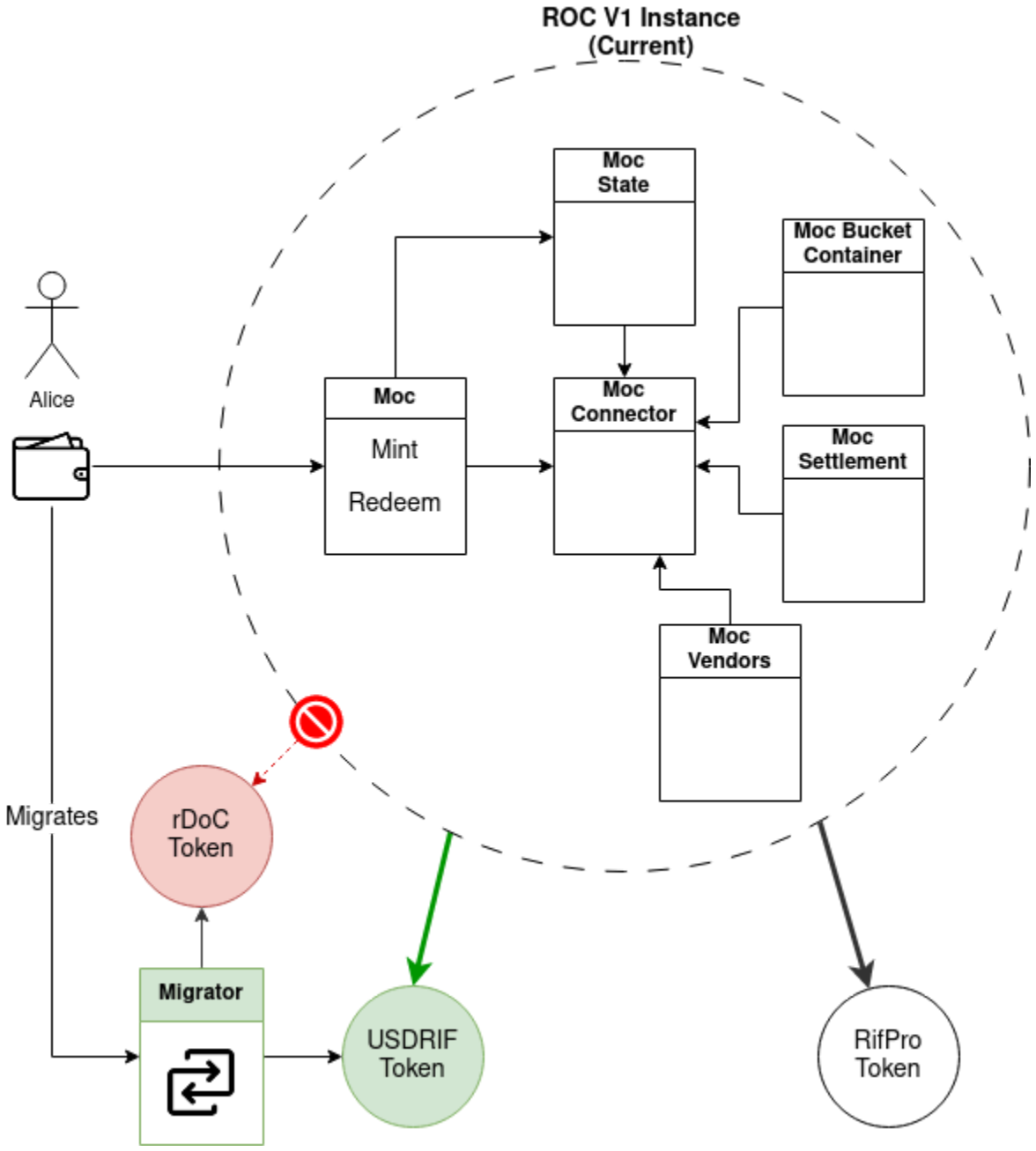
MOC 0.X (also referred to as V1 or Legacy) model, while successful in its own context, limits the evolution of ROC both technologically and functionally. That is why the Team proposed a new model (colloquially known as MOC 1.0 or V2), which is conceptually similar but improves certain aspects of its predecessor. I In this migration, we'll focus on ROC, which uses the RRC20 branch of the protocol, being the Rif Token, its collateral.

The migration must follow the project's principles of decentralization, transparency, security, and openness, while also ensuring the least possible downtime for operations, easing the transitions for both users and integrators.

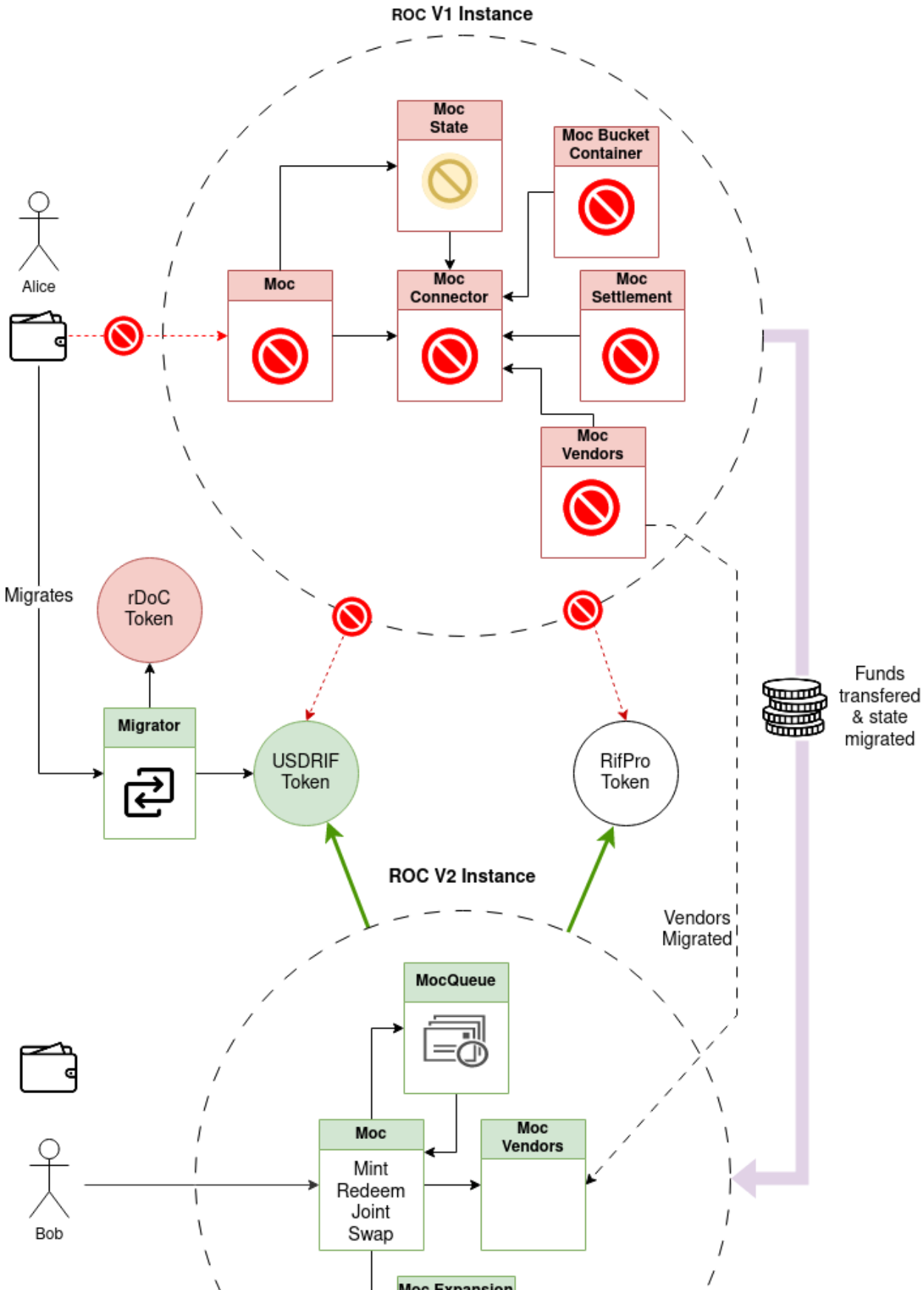
PROPOSAL

In general terms, the migration involves moving away from the legacy model while maintaining a similar interface of current moderating contracts. The state variables and balances are used as the seed for initializing the new model. This process is automated through a single transaction, which guarantees atomicity.

Before:


ROC V1 Instance
(Current)
Moc
State
Moc Bucket
Container
Alice
Moc
Mint
Redeem
Moc
Connector
Moc
Settlement
Moc
Vendors
Migrates
rDoC
Token
Migrator
USDRIF
Token
RifPro
Token

After:


ROC V1 Instance
Moc State
Moc Bucket Container
Alice
Moc
Moc Connector
Moc Settlement
Moc Vendors
Migrates
rDoC Token
Migrator
USDRIF Token
RifPro Token
Funds transfered & state migrated
ROC V2 Instance
Vendors Migrated
MocQueue
Bob
Moc
Mint
Redeem
Joint
Swap
Moc Vendors
Moc Expansion

As illustrated above, after the upgrade/migrate proposal is executed:

- A user (Alice) that holds rDoc, will still be able to migrate them to the new USDRIF Token, and use them on V2
- A user (Bob), that starts interacting with the platform, will operate directly with RoC V2. He won't notice any protocol "history", and he will have new features as joint & swaps operations.
- Both old and new users will interact with existing legacy token RifPro.
- A blockchain expert user, trying to interact with legacy unsupported contracts, will get an error message.
- A blockchain expert user, would be able to audit and track all the balance movements and state mapping, verifying that every user kept their value holdings.

Technical Migration Procedure

Even though all V1 contracts are OZ upgradable, the changes on storage and technologies from V2 are so radical, that we opted for a migration approach instead. Anyway, to enable the migration, some upgrades were indeed performed on V1, ultimately resulting in all functional contracts being "disabled" to avoid confusion.

The migration essentially happens in two major steps:

1. **RoC v2 initialization**

In the first stage, all contracts related to the new RoC V2 solution will be deployed, and all "static" variables and system relationships will be initialized.
At this point, the system is not operative, its basic functions cannot be used as the queue and the bucket have not yet been linked, which only happens during migration itself. At a later stage, the changer will verify anyway that V2 is "clean", meaning no one had operated on it to show transparency to the Voting members.

Details:

- V2 Contracts deploy
    - Moc Core itself
    - Moc Queue
    - Moc Vendors

    - Flux Capacitor settings providers (bridge to v1 Stopper model)
- Initialize contracts:
    - Delegation of functions in the current Governance system
    - Legacy RoC riskPro Token “RifPro” is linked to RoC v2 (but “ownership” still remains in MOC V1)
    - Governance Token link (FeeToken in V2)
    - Oracles link (PriceProviders will be the same)
    - Stopper account assignment, same as V1
    - Configuration of all "static" system variables (e.g., time between settlement)
- Vendor registration and markups (through VendorGuardian account)
- Flux Capacitor state setup
- Changer contract deployed and configured with both legacy V1 and new V2 addresses.

2. Changer execution

Changer should articulate the following changes and migration on a single transaction:

a. Upgrade V1 to make it “migrable”

This upgrade adds a new method to the required V1 contracts, allowing them to migrate special relevant states to V1.

b. Assets Transfer

In this process, all assets from RoC legacy are transferred to their corresponding locations in RoC V2. This includes:

- Split CommissionSplitter balance
- Transfer RIF Token balance
- Initialize AC (RIF), TC (RIFPRO) and TP (USDRIF) state variables with analog V1 values.

The address of RoC V2 must be fixed by the upgrade, in such a way that it can be audited preventively to ensure that it is indeed the correct one.

c. RIFUSD addition as Pegged Token

RIFUSD Token, needs to be added to the Rif V2 Bucket as TP with the corresponding PriceProvider and settings.

d. Role Transfering

rifPRO Token from RoC V1 has exclusive mint, burn and pausing capabilities from the primary contract, *MoCExchange.sol*. This capacity must be transferred to the new RoC V2.

Similar, but not identical as this is handled by Open Zeppelin AccessControl rather than Ownable, USDRIF Token minting and burning roles need to be given to V2.

e. RoC V2 model internal state mapping

In this process, the new RoC V2 contract(s) take all relevant state from Moc, MocState, and MoCBProxManager legacy, and, using public "view" functions, obtain the necessary values to initialize their internal state.

For example:

- `nACcb <- mocState.riskProTotalSupply()`
- `Peg_cont[0].nTP <- MoCBucketContainer.getBucketNStableToken("B0")`
- `coll_bag.nTCcb <- MoCBucketContainer.getBucketNRiskPro("B0")`

f. Executors and bucket registration

As part of the initialization of the queue, allowed executors need to be whitelisted (they will be V1 Oracles accounts) and the Riff bucket needs to be registered to the queue. Even if MocQueue and MocCore were already deployed at this stage, as they have cross dependencies, they cannot be assigned on initialization. The bucket (Rif MocCore) registration process is done on the changer itself as we have governance approval.

g. RoC v2 Verification

After verifying that all the previous steps were successful, the operation in its entirety of functions is effectively ready in MOC V2. If any of the required conditions of the Changer are not met, the execution will revert.

3. OPTIONAL: Upgrade RoC V2 to clean up migration methods

It is recommended to wait for the next functional upgrade to implement this change. It has no functional sense other than long-term code maintainability.

4. OPTIONAL: Vendors to un-stake

New Vendor's logic does not require any staking, so they can withdraw both their accumulated rewards and staking from the previous one. But this has no direct effect over the migration and can happen at any time.