Magic Square

Whitepaper

Table of Contents

Project Introduction

Magic Square is a unified, integrated, interoperable space for users to discover and experience community-vetted crypto apps, and for crypto app developers to market and monetize their best applications.

Magic Square is a multifaceted system that covers all the UX bases: The Magic Store guides users to discover the highest quality crypto apps with built-in integrated DeFi. Users sign in to all of their apps with one SSI-secured Magic Connect leading to their personalized Magic Spaces: where crypto-app experiences begin and end. Users and **developers** discuss and collaborate in a dedicated Magic Community space. The whitepaper you are about to read is a technical product description. Please see our Lite Paper for a comprehensive overview of the project.

Product Architecture

The Magic Store is the fundamental element of the Magic Square project. Within the Magic Store, there are multiple working elements that dictate the communication between many moving parts. The chief elements in our case are app users, app developers (**developers**), and the validators, who vote on the entrance of new apps into the Store.

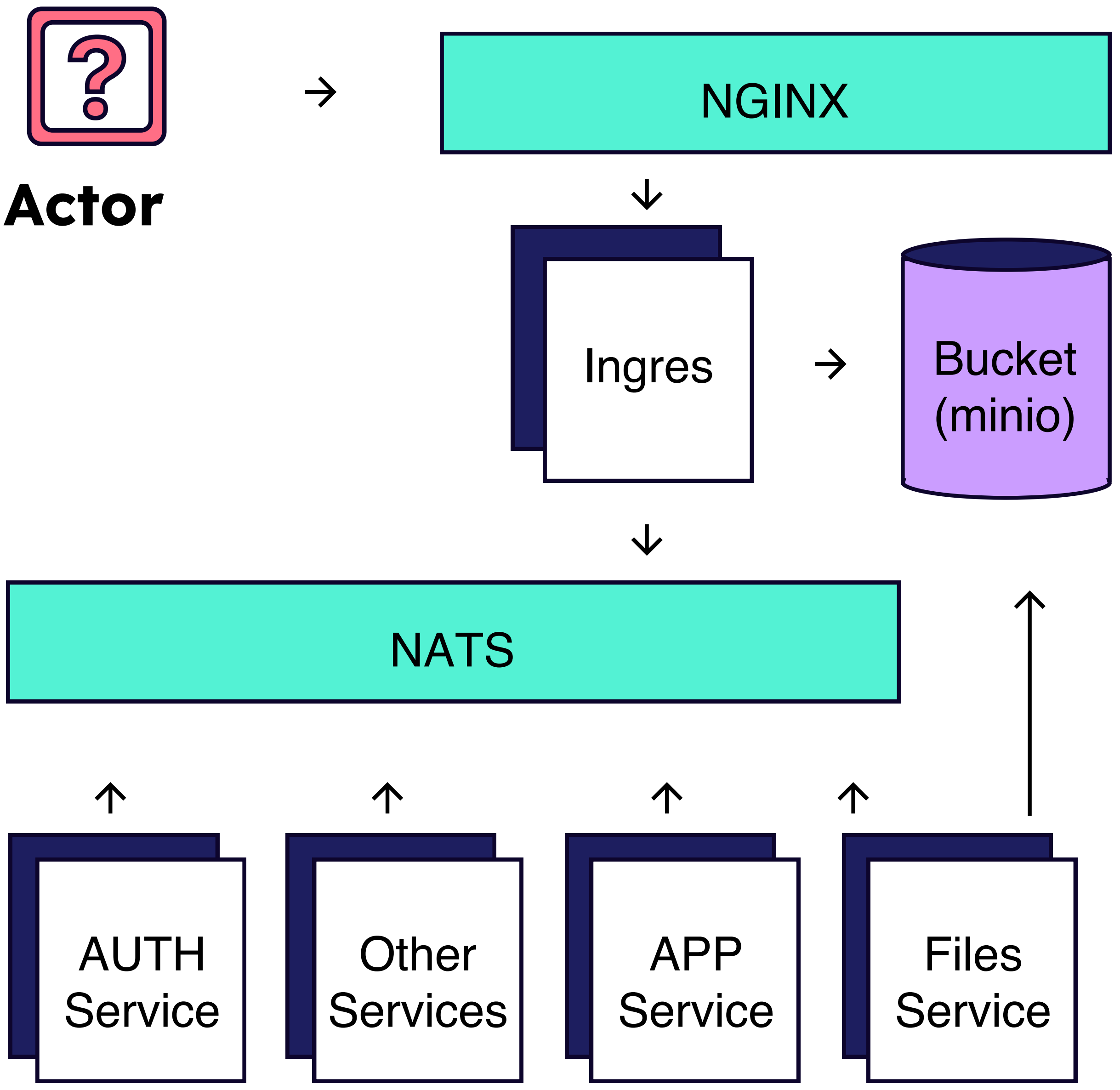


Figure 1. Product Architecture

1. NGINX - Load balancer for ingress, single entry point for the outside world
2. Bucket - a server with minio for storing all user applications
3. Ingres - a service that is used as an API gateway.

   - Also checks if the user has the right role for the endpoint.
   - Uploads all files from the user to the bucket, passing on only information about the file to the services.
4. Auth service - registration, identification, authentication, authorization, and any other user-related operations
5. App service - works with applications, checks user access to apps
6. File service - application validation and deployment

1.1. Service Scheme

From an architectural point of view, an app store can be represented as two independent services: an application download service and a gateway service. The first is responsible for storing and managing applications, while the second simply receives requests from the user and passes them on to the application service.

To make sure our system is crash-tolerant, we'll run the app download and gateway on separate servers (node-1 and node-2).

Now, provided that our services are running, the store can process user requests.
When trying to list all available applications, the following happens:

1. HTTP server receives Request (GET / app) running on node-1.
2. The incoming request is forwarded from the HTTP server to the gateway service, which does all the processing and matching.
3. The user's request is translated into the listApps action of the apps service.
4. The request will go to NATS and the request will be successfully delivered to the node-2 node in the apps service.
5. The apps service will execute the listApps action and return a list of all available applications.
6. A response will be sent to the end-user.

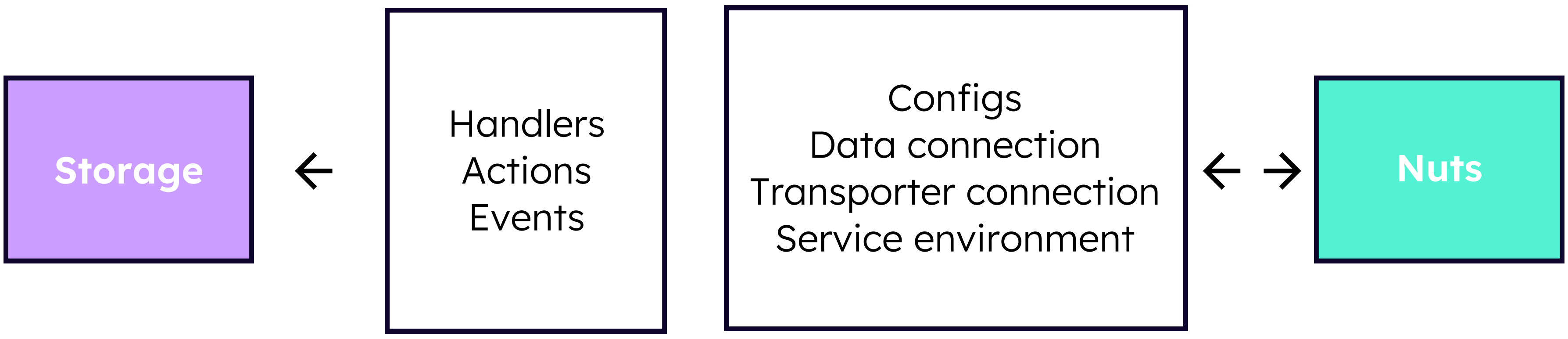


Figure 2. Service Scheme

Actions

- Publicly referred to as “service method”
- Working with actions is based on the principle of remote procedure call (RPC).
- The action is similar to a normal HTTP request - accepts parameters as input, and returns the result.
- If multiple service instances are running, requests will be balanced across instances.

Events - used to handle events of other services

*Each service uses its own databases.

1.2. Security

From a security point of view, we apply the following rules:

- All services of this project are connected to the password-protected message queue (NATS)
- Each service encrypts data using its own asynchronous key
- The service only processes messages that have been encrypted

The public keys necessary for the service to communicate with others are included from the start, without being transmitted over the network.

Due to this, the likelihood of theft or falsification of data by an intruder is minimal.
If an attacker has the message queue login data (NATS), he will see the data only in its encrypted form, and he will not be able to falsify data.

If the attacker obtains both the message queue login data and the public key of one of the services, he will be able to read the data of the specific service of the key he managed to obtain, but falsifying the data will not work without the private key.

If an attacker obtains the data to enter the message queue as well as a pair of keys, he will be able to falsify the data of one service. But since the keys are stored directly in the service, he will only reach them if he obtains direct access to the machine. Also, communication with the data message queue takes place in the local network, which makes it possible to connect

General Product Description

The Magic Square interface includes three main components: The Magic Store, Magic Spaces, and Magic Connect. Each of these serves a different function for the three groups who use the platform: crypto application users (Users), crypto application developers (developers), and new community validators (Validators). Each of these groups plays a unique role in the Magic Store interface, with a unique set of actions and responsibilities. Users must enter the system and obtain a secure persona with their secure Magic Connect login credentials. developers must successfully deploy apps to be validated by the Validators, who vote on the new apps entering the ecosystem.

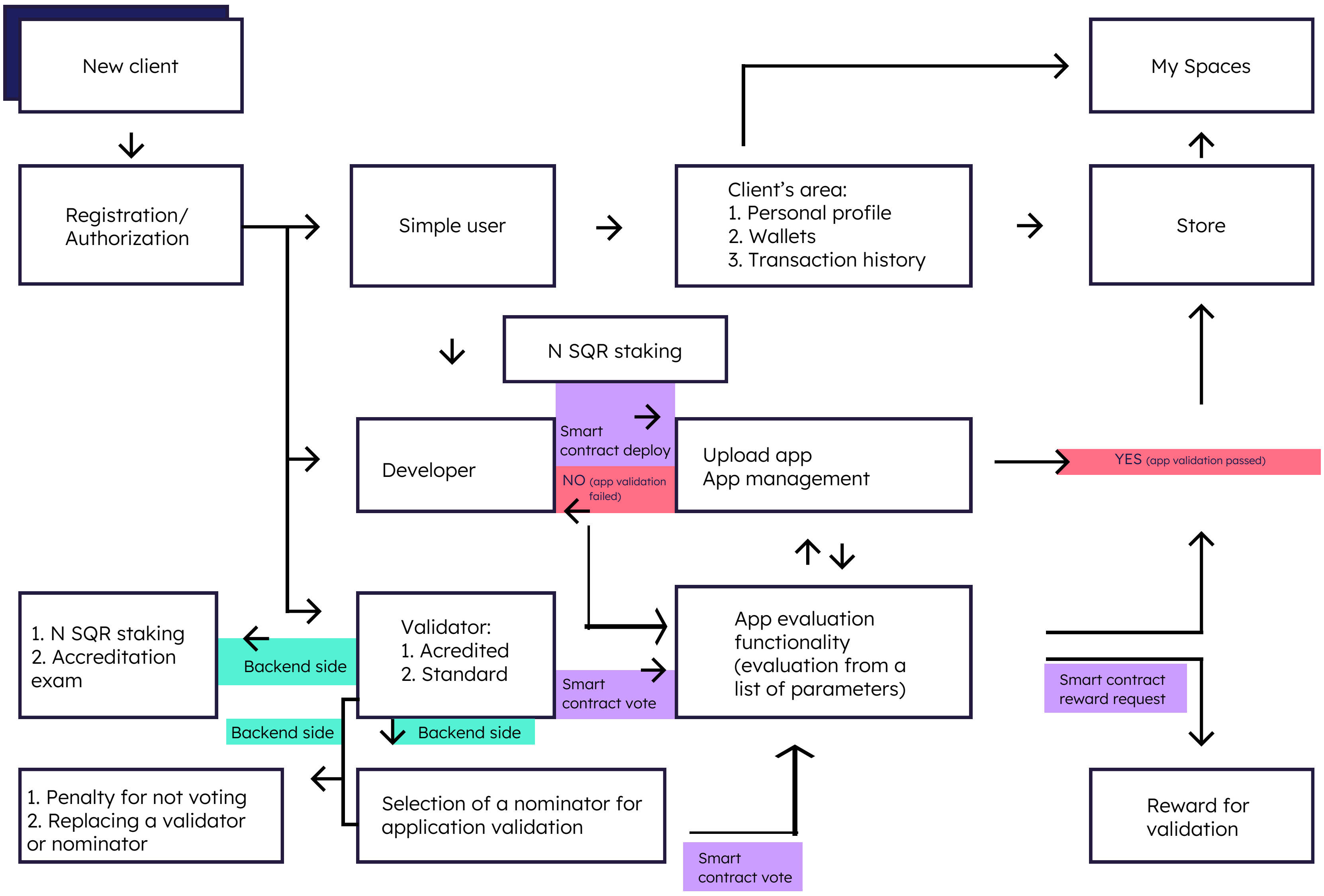


Figure 3. General Product Scheme (MVP version)

2.1. User Flow

Magic Square Users discover apps on the Magic Store, download them to their personalized Magic Spaces, where they can access and open their apps via a single, secure Magic Connect login.

I. Magic Store

The Magic Store allows Users to discover all of their crypto apps and NFT collections in one place, and access them with the click of a button. Within each app featured on the Magic Store, there are integrated DeFi "bundles" that support native app currencies which can be accessed with one click.

The Magic Store operates under a policy of complete transparency, listing the validator voting results, user reviews, and live app statistics, such as the number of total downloads to Spaces, on each app page on the Store.

II. Magic Spaces

After Users discover an app on the Magic Store, they add the app to their customized dashboard of crypto apps and widgets known as "Magic Spaces". Magic Spaces allows Users to store, open, and use all of their apps in one location, via their single Magic Connect login. There are an unlimited number of Spaces available to the user to organize their apps and widgets, and Magic Spaces can be used and accessed across devices.

III. Magic Connect

In order to access their Magic Spaces and use their apps, Users must enter the Magic Square system via Magic Connect. Magic Connect allows Users to establish a connection to all of their apps via one Self-Sovereign Identity (SSI) secured login.

Magic Connect is an HTTP-based service with smart contracts on the blockchain. It serves as a holder of the Verifiable Data Registry which stores Decentralized Identifiers (DIDs) and maintains an interaction between all the participants.

The Magic Connect system includes the following components Users, Verifiable Data Registry, Issuers, and Verifiers. Issuers are defined as the institutions that issue Verifiable Credentials (VCs). Verifiers are the institutions that are requesting the User's VCs.

Magic Connect User Flow

The Magic Connect User interface can be described in three stages: obtaining a Decentralized Identifier (DID), obtaining a Verifiable Credential (VC), and Submitting the Verifiable Credential

Stage 1: Obtaining a Decentralized Identifier

When a user runs the Magic Connect mobile application and website. The connection between the user's mobile wallet and the website goes through the Wallet Connect protocol.
The user launches a special registration form on the website that only requests a connection to the Solana wallet.

The user connects their Solana wallet and signs the transaction with their private key. By signing this transaction, the User allows Magic Connect to create a DID by issuing the smart contract within the Solana blockchain. Magic Connect ascertains DID creation[10] and generates the QR-code. The user then scans this QR-code with the mobile wallet, and information about their DID uploads to the application. The application then notifies the user that the process has been completed successfully.

Now the User holds their DID in their wallet.

Stage 2: Obtaining a Verifiable Credential

When a user wants to obtain a Verifiable Credential, the Verifier will ask Magic Connect for the user's DID and some additional data (i.e. KYC). If the user has already passed the KYC with a Magic Connect-registered Issuer, the user can activate the existing KYC. Otherwise, the user will need to enter Magic Square’s registered services marketplace and search for available Issuers. In the case of KYC, Issuers will be country-specific.

After the user selects an Issuer, they will ask Magic Connect for verification by this Issuer. If verification is paid, Magic Connect handles the necessary payments. Magic Connect will redirect the user to the Issuer through an HTML form, HTML web application, or special link that can be used to communicate with the provider. The Issuer will receive a user request and DID from Magic Connect.

After receiving the user's request, the Issuer will verify the data and issue cryptographically signed credentials, known as "Verifiable Credentials" (VC). The VC, in the form of the JSON file, contains verified user data, as well as metadata such as the date of issue or expiration, the proof object, and the user's DID from the Magic Connect registry. This VC returns to the user's application or account and thereafter can be downloaded, presented, or deleted by the user.

Issuers send VC documents directly to Users, not through Magic Connect, the registry or any other blockchain. This way, the interaction is totally private. Magic Connect does not share any information about the Issuer, the VC, or the type of information it verifies. Only the user has access to the data. The user is responsible for storing the data. The Issuer keeps a record of issued VCs.

Stage 3: Submitting the Verifiable Credential

In order to submit their VC to the Verifier and access the desired services, Users must upload their Verifiable Credential (VC) to the Magic Connect system. Users can register on any service connected to the Magic Connect via the following process:

1. In the login window on the service's website, Users select "login via Magic Connect".
2. The service (connected to Magic Connect via the SDK and API) provides the user with a QR-code generated by the Magic Connect system.
3. The user scans this QR-code with the Magic Connect application and sends their DID to the Verifier.
4. After registration, the Verifier may request additional verified data, for example, KYC, verified email, or mobile phone number.
5. Depending on the country and conditions, the service can accept information only from known and confirmed Issuers.
6. Users send their VC received from the Issuer via a Wallet Connect to the Verifier's website (VCs are transferred via a secure connection from the mobile wallet to the service's website).

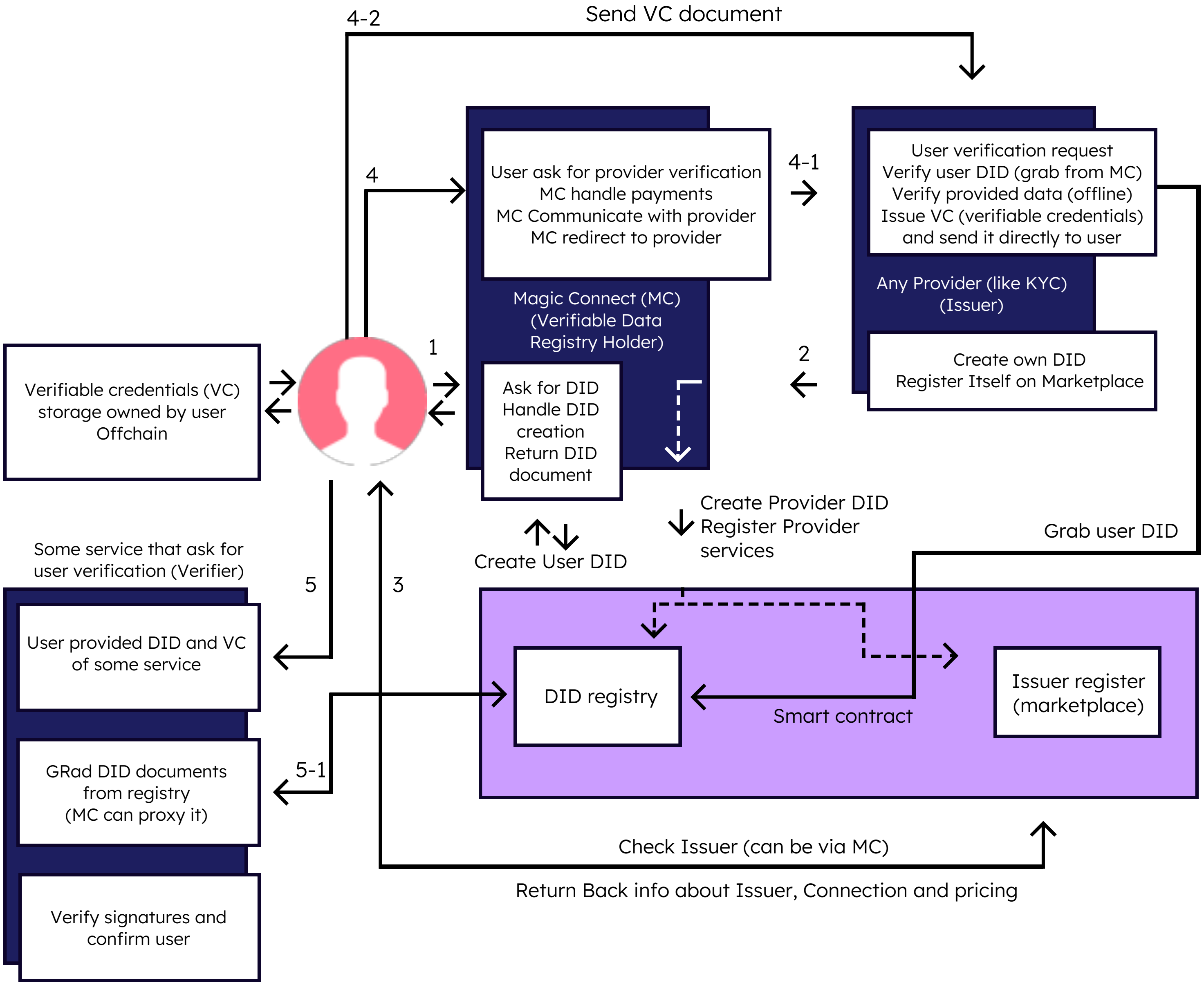


Figure 4. Magic Connect structure

The Verifier can check the authenticity of the DID (contains matched public keys, verification methods, etc.) and the reputation of the Issuer at the DID registry. After this verification, the user is confirmed for the service.

Magic Connect Issuer Flow

The following describes the interaction between the Issuers and Magic Connect in order for an Issuer to be listed in the Magic Connect system.

1. The Issuer asks Magic Connect to create its DID
2. Magic Connect returns the requested materials to the Issuer
3. The Issuer requests to be listed on Magic Connect's service registry "marketplace", where the Issuer can be discovered by Users
4. Thereafter, the Issuer is registered in the Magic Connect system.

Of course, it is possible for Users and Issuers to communicate directly without Magic Connect, but we expect that Users will use this special service registry as a marketplace of Issuers.

IV. Client Area

Users enjoy a "Personal Profile" area where they can upload an avatar icon, manage personal data (password, email, etc.), set up Two Factor Authentication, and manage their package settings and upgrade. There are three user packages available to Users based on staking tier:

Tier 1 - Base Package

- Users are allotted up to three apps in their Spaces
- without staking
- Users stake 150 SQR to unlock unlimited app downloads to Spaces

Tier 2 - Pro Package

- Users stake 500 SQR to unlock:
  - Use to Earn - daily rewards based on in-store activity
  - Referral Rewards - earn according to their active referrals
  - Karma Score - receive higher daily rewards by accumulating Karma for on-chain activity

Tier 3 - Influencer Package

- In addition to Tier 2 perks, Users with Karma above 5000 can stake 1000 SQR tokens and receive a 25% boost to their daily rating, which translates to higher rewards.
- Influencers receive early access to all events, exclusive contests, etc.

2.2 Developers

Developers, in the context of Magic Square, are crypto app developers. Developers who wish to list their apps on the Magic Store must first submit their apps for validation by the Magic Square community validators.

Developer Payment and App Listing

In order for an application to be registered for validation, the developer will send to smart contract with 10,000 SQR Tokens:

35% is paid to Validators

15% is paid to Magic Square

50% is staked for the duration of the app's listing on the Magic Store

If the application does not meet the minimum listing score, 65% of the tokens will be returned to the developer (35% will be sent to the validators as a reward for their work).

Application Deployment Flow

Application Deployment Flow below describes the Developer’s submission process of a new application for validation.

- **The developer uploads** the application's statics to our servers and receives a unique subdomain name (before approval it is a hash), where the files will be unpacked, checked to ensure that the file structure meets our requirements.
  - If the file meets the requirements, the application receives the status “new”. In this state, the app is available for use and is visible only to the owner. The owner can check if everything works as expected, re-upload the original files.
- **Registration of application for validation**
  - Provide necessary information about the application for the info page: name, description, categories, icon, and screenshots
  - Complete payment for the deployment of the Smart Contracts.
  - The application goes into the "preparing" status
- **Data about the application is entered into the system.**
  - A smart contract is deployed
  - The application is tested on third-party services
  - The app is ready for validation
- **Validation**
  - The system sends invitations to the validators and the application receives the “to_validate” or “on_validation” status
  - In this stage, the application becomes available for review by a randomly generated list of validators, selected from a database of those to whom validation rights have been granted/transferred
  - After several weekly voting rounds, the application will either collect the required number of votes and receive a rating, or not. The app’s status will change to either 'approved' or 'declined'

If "declined" - 65% of tokens will be returned to the owner, the application becomes available to the developer again (lowered to stage 1) until he decides to remove it or fix it and repeat the validation process

If "approved" - the app becomes public and available to Users on the Magic Store.

The following diagram demonstrates the process of uploading a new application to the Magic Store server. (Figure 5. Initialization Flow)

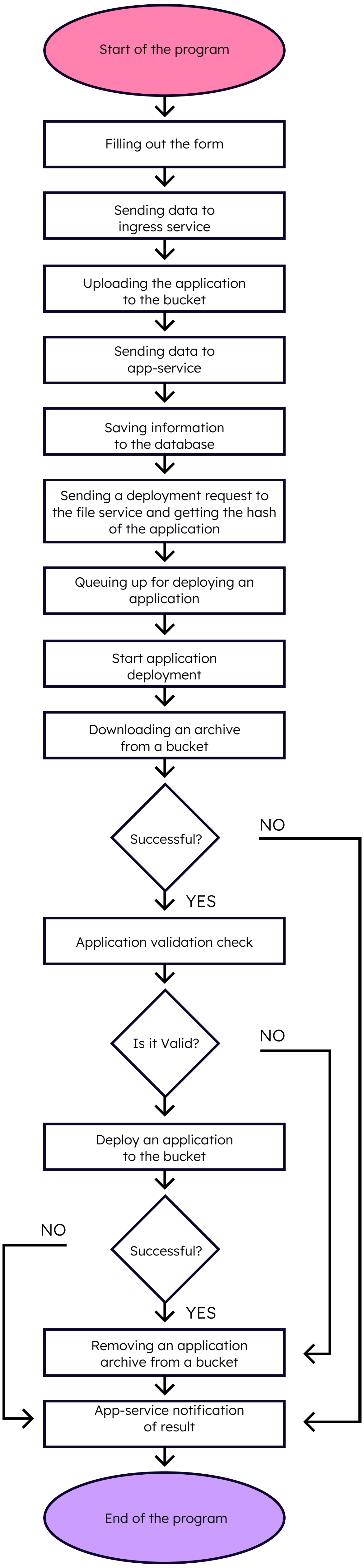

Start of the program
Filling out the form
Sending data to ingress service
Uploading the application to the bucket
Sending data to app-service
Saving information to the database
Sending a deployment request to the file service and getting the hash of the application
Queuing up for deploying an application
Start application deployment
Downloading an archive from a bucket
Successful?
NO
YES
Application validation check
Is it Valid?
NO
Deploy an application to the bucket
NO
Successful?
YES
Removing an application archive from a bucket
App-service notification of result
End of the program

Figure 5. Initialization Flow

Smart Contract Service (SCS) Flow

The following SCS flow describes the interaction of the backend of the application with a smart contract.

1. CreateVotingMethod - creates a Voting contract with empty votes, token balance, and rules
2. TransferMethod - deposits tokens into a Voting
3. CheckPaymentsMethod - for checking the number of tokens on a Voting to Start Voting
4. CreateStartVotersListMethod - creates a list of voters' wallets for the first voting round
5. UpdateVotersListMethod - replaces voters who did not vote in this round with new voters in accordance with the round of voting and the results; Also returns a list of Users for penalties
6. GetVotingResultMethod - this method is run by CRON and checks the result at the end of each voting round. Will return the result, the number of votes we are currently receiving, the round of voting, and the rating of the review.
7. CloseVotingMethod - this method will be called when the vote ends: the end of the last round or the completed vote. According to the final result, this method will close the vote and send tokens to the APP wallet and Magic Store wallet, or allow the APP holders to request withdrawal of tokens if the vote fails. Requests from voters for rewards will also be allowed.

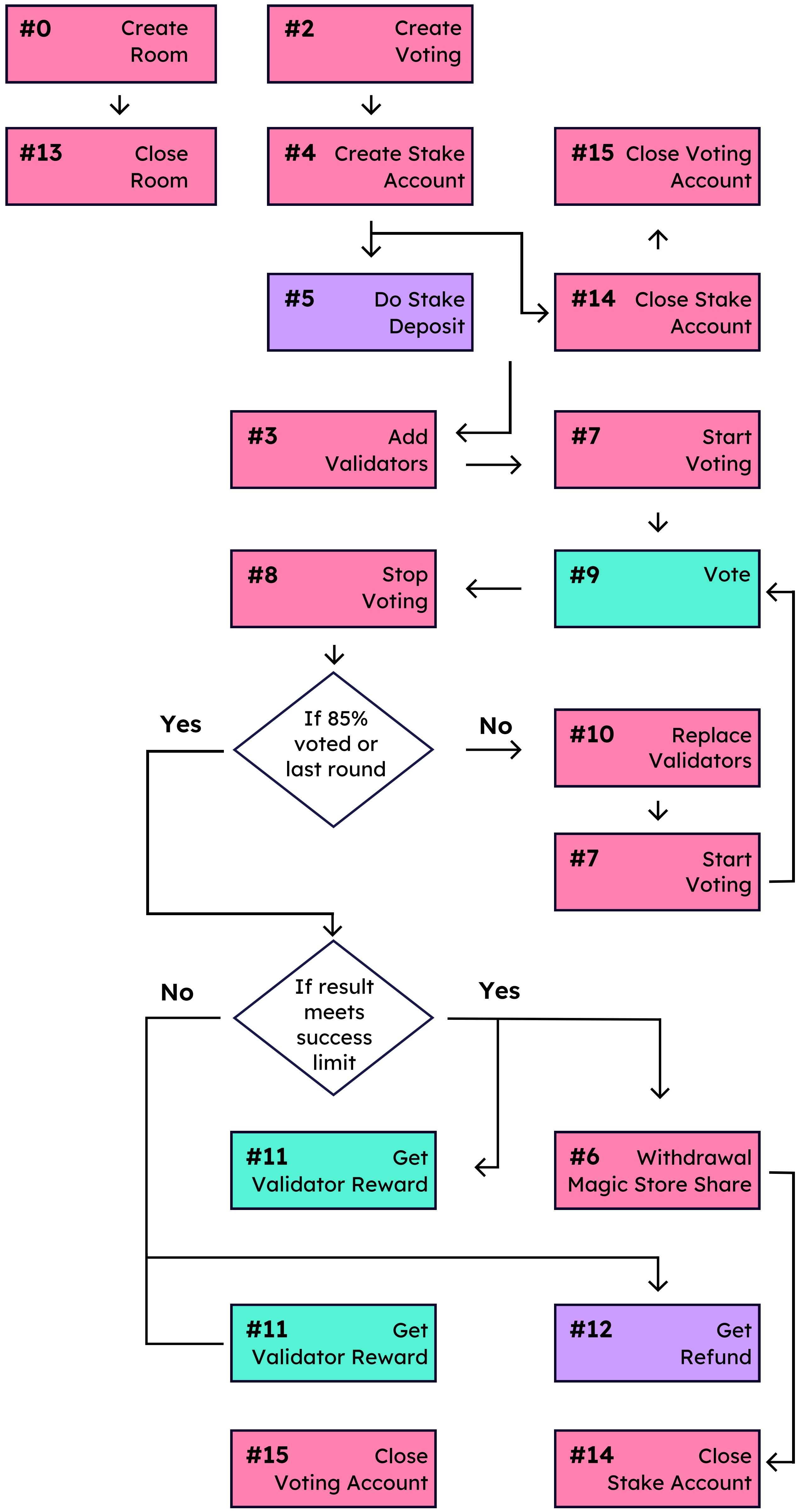
On-Chain Voting Flow

The following diagram describes the on-chain voting process of new apps

1. Through the backend, the Application developer initiates a request to the General to create a Voting.
2. The Voting is created and a deposit can be made to its address.
3. After the deposit, the Application developer starts the voting procedure
4. The Voting smart contract receives the initial list of voters and establishes:
    - a time for voting to begin,
    - the duration of the round
    - the rules for distributing rewards, refunds
    - the address for reservation in case of a successful vote

Voting Flow

Access by: Admin Developer Validator



#0 Create Room

#13 Close Room

#2 Create Voting

#4 Create Stake Account

#15 Close Voting Account

#5 Do Stake Deposit

#14 Close Stake Account

#3 Add Validators

#7 Start Voting

#8 Stop Voting

#9 Vote

If 85% voted or last round

Yes

No

#10 Replace Validators

#7 Start Voting

If result meets success limit

No

Yes

#11 Get Validator Reward

#6 Withdrawal Magic Store Share

#11 Get Validator Reward

#12 Get Refund

#15 Close Voting Account

#14 Close Stake Account

Voting State

- Uninitialized
- Initial
- Stake Acc Created
- Stake Tokens Recived
- Started
- Round Failed
- Voters Replaced

- Succeeded
- Failed
- Ready To Close

2.3 Validators

Validators serve an essential role in the Magic Square system: they are the gatekeepers to the Magic Store, ensuring the quality and security of the featured apps. Each app must surpass a community validation process before being listed on the Store.

In the Magic Square system we have three types of Validators:

- Accredited Validators
- Nominees
- Standard Validators

Accredited Validators

A validator who stakes 5,000 SQR Tokens and successfully passes the Accreditation Test becomes an Accredited Validator (AV).

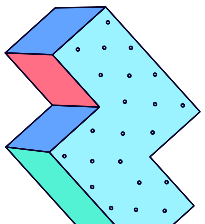
AV seat vacancies will occur if:

1. A current AV steps down from their seat

New AVs will be chosen based on SBO (Second Best Offer) in SQR Tokens via an auction that will last 1 week from the date of posting the open seat.

Nominees

If a validator stakes SQR tokens to become an AV but does not pass the Accreditation Test, either because they failed the test, or because they did not want to take the test, they can nominate one Nominee that will need to pass the Accreditation Test in their place. The Nominee will not stake SQR tokens.



In order to choose a Nominee, a Validator can create a contest and choose the winner. The contest should include relevant knowledge of the revenue share model.

The Nominee receives a portion of the Reward tokens earned by the Validator. Their revenue share and the name of their AV are visible on the Nominee’s profile.

If the AV gives up their seat, the role of their Nominee will be immediately annulled.
If the Nominee skips voting for 30 days, their role will be canceled and the Validator will need to find a replacement Nominee.

Standard Validators

Any Magic Square user can apply to become a Standard Validator (SV). There can be an unlimited number of standard validators, but only one hundred SVs can participate in each validation process.

If an SV does not participate in three consecutive validation processes for which they received an invitation, the SV is automatically removed from the SV list, and cannot apply to become an SV again.

Validation Flow

The following describes the validation process from the time a new app is submitted for validation by a Developer until the app is accepted and listed on the Magic Store or rejected from listing.

- After a developer lists an app and sends the right number of tokens to the Smart Contract, the Validators will get a notification that new Voting is available.
- Validators request the Voting smart contract function with the parameters of their app ratings
- Each challenge calculates the number of voters, the final score and checks whether enough votes have been collected.
- After the expiration of the voting period, the backend calls the smart contract to receive the Voting results or continue the Voting.
    - In case of successful completion of the Voting, a portion of the specified tokens is transferred to ensure further operations, the rest becomes available for withdrawal requests.
    - In the case of an unsuccessful vote, and there are available remaining voting rounds, the validators that did not vote are replaced, and a new round begins.
    - “Ban points” are given to validators who neglect to vote.
    - If there are no remaining Voting attempts - it is noted that the Voting has failed. In this case, all parties can call the function to receive funds back.

App Validation Criteria

Validators rate applications based on the following criteria:

- Utility
- Speed
- Simplicity
- Design
- Experience

After an app is verified by the minimum required number of validators and receives an adequate score, it is listed on the Magic Store.

The diagram below shows a validator voting flow using a smart contract (Figure 7. Voting flow).

Voting Flow

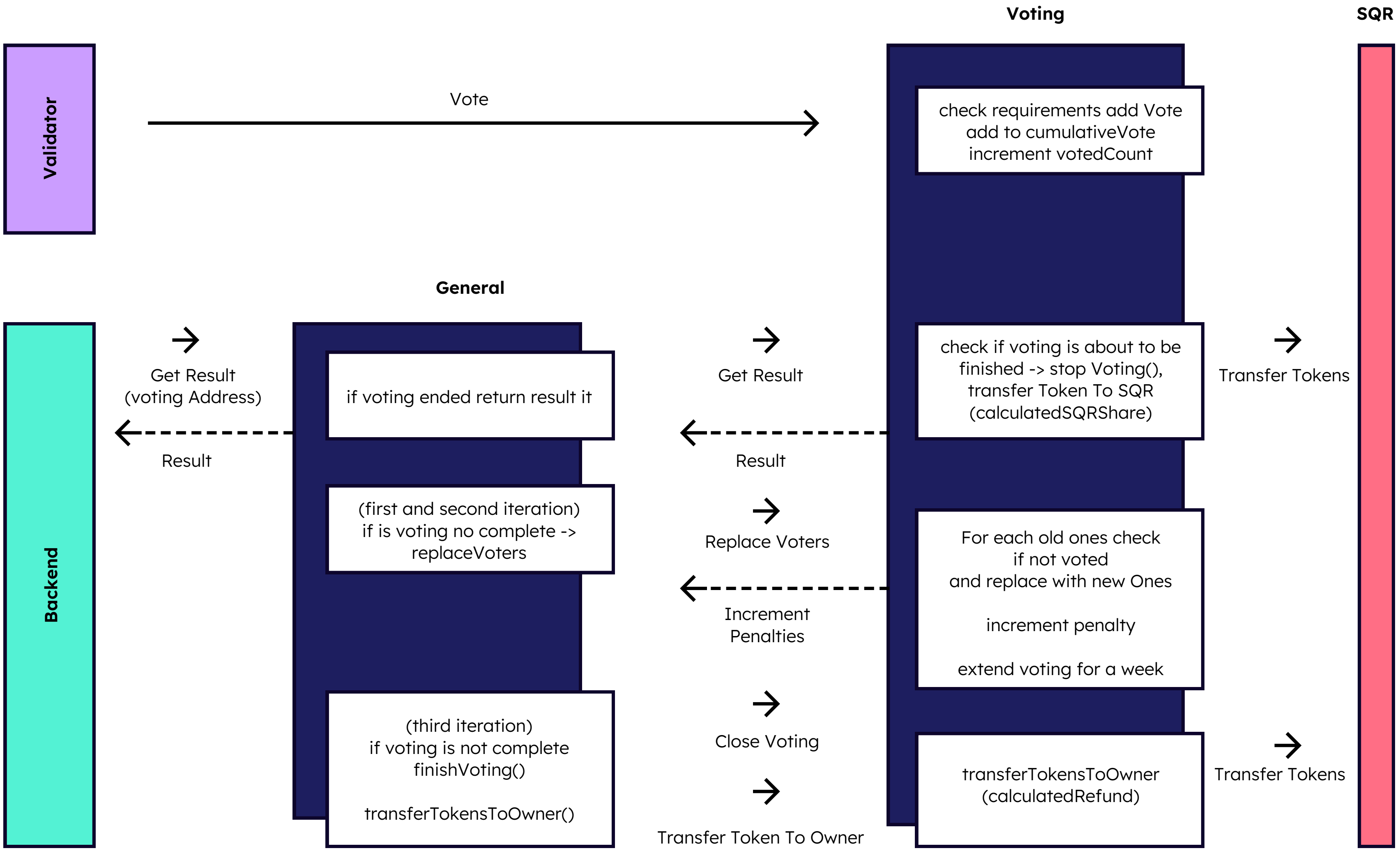

Voting
SQR
Validator
Vote
check requirements add Vote add to cumulativeVote increment votedCount
General
Backend
Get Result (voting Address)
Result
if voting ended return result it
(first and second iteration) if is voting no complete -> replaceVoters
(third iteration) if voting is not complete finishVoting()
transferTokensToOwner()
Get Result
Result
Replace Voters
Increment Penalties
Close Voting
Transfer Token To Owner
check if voting is about to be finished -> stop Voting(), transfer Token To SQR (calculatedSQRShare)
For each old ones check if not voted and replace with new Ones
increment penalty
extend voting for a week
transferTokensToOwner (calculatedRefund)
Transfer Tokens
Transfer Tokens

Figure 7. Voting flow

Validation Rewards

For each validation process, the validators receive a reward equal to 3500 SQR tokens, divided into the following shares -

80% of the reward goes to the accredited validators (2800 SQR tokens)
20% of the reward goes to the standard validators (700 SQR tokens)

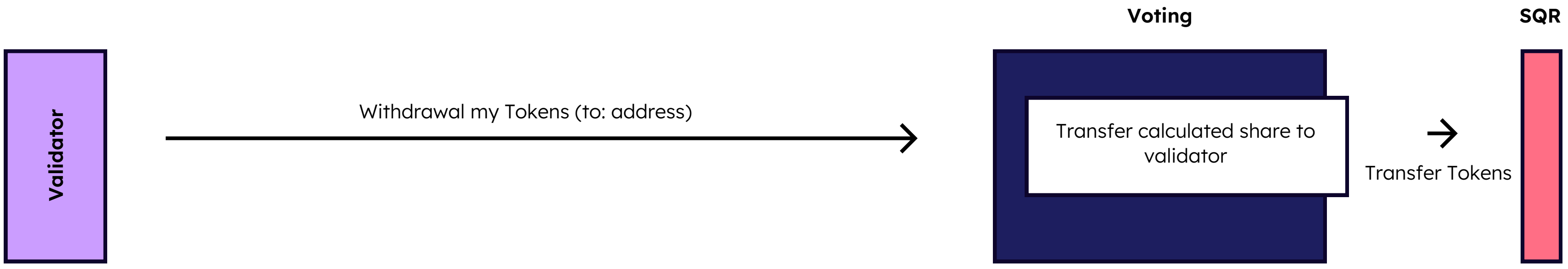
Reward Withdrawal Scheme

When a validator calls the smart contract voting function, specifying the token withdrawal address, the Voting smart contract requests the smart contract of SQR tokens and transfers the specified amount, recording that the funds have been received.

This scheme is fundamental for any withdrawal of tokens from Users, including rewards for voting, refunds, use-to-earn rewards, contest prizes, etc.

The diagram below shows the flow of receiving voting rewards (Figure 8. Validator withdrawal).

Validator Withdrawal


Voting
SQR
Validator
Withdrawal my Tokens (to: address)
Transfer calculated share to validator
Transfer Tokens