Introduction

Wallet and Bank

Truly decentralized finance requires decentralized banks and decentralized wallets.

KeepERC20

To provide security and convenience without compromising decentralization, we propose KeepERC20, a distributed bank-and-wallet based on smart contracts and Chainlink...

Wallet and Bank

Truly decentralized finance requires decentralized banks and decentralized wallets. We are already familiar with decentralized, non-custodial wallets such as Metamask.

But what about **decentralized banks**?

FIG 1. CENTRALIZED

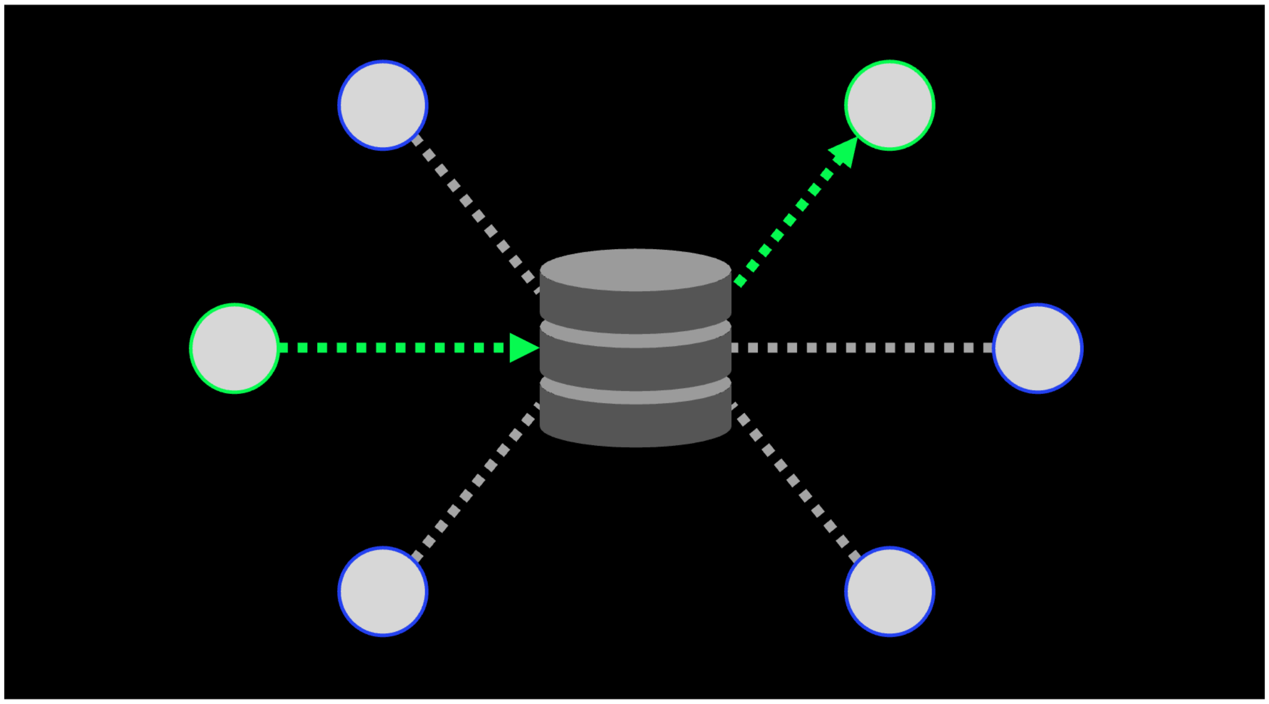
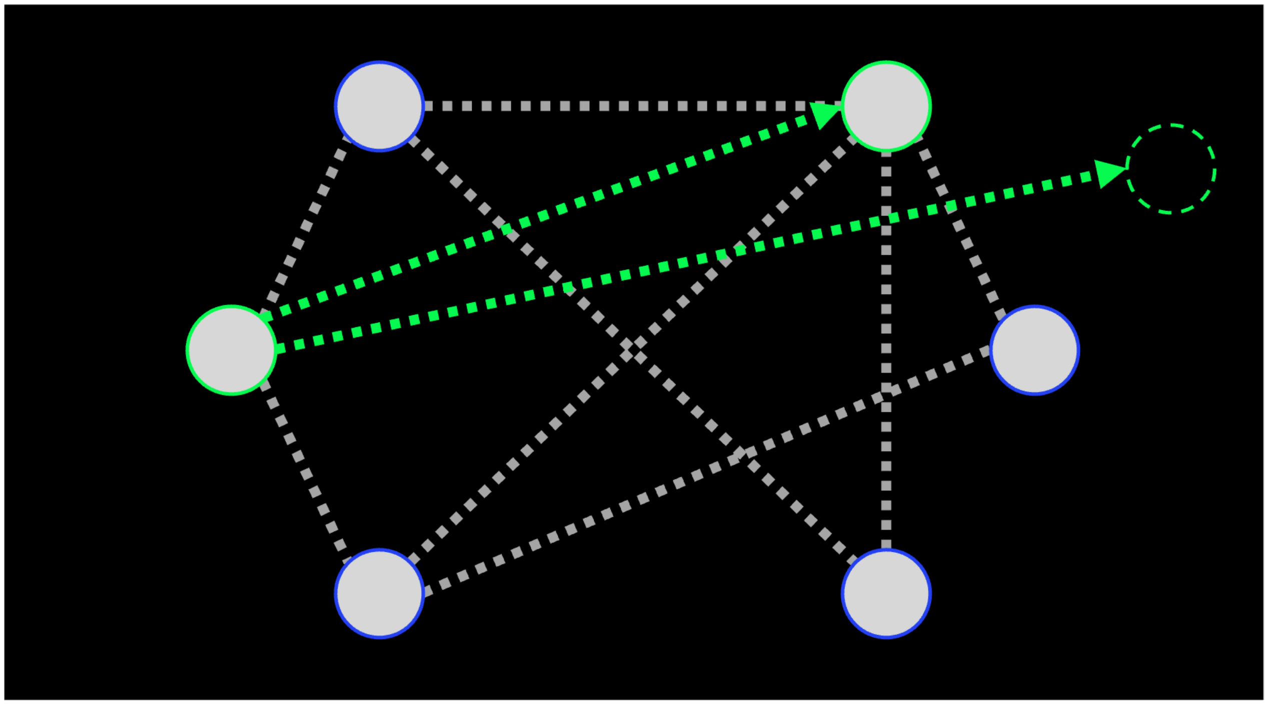


FIG 2. DECENTRALIZED - X



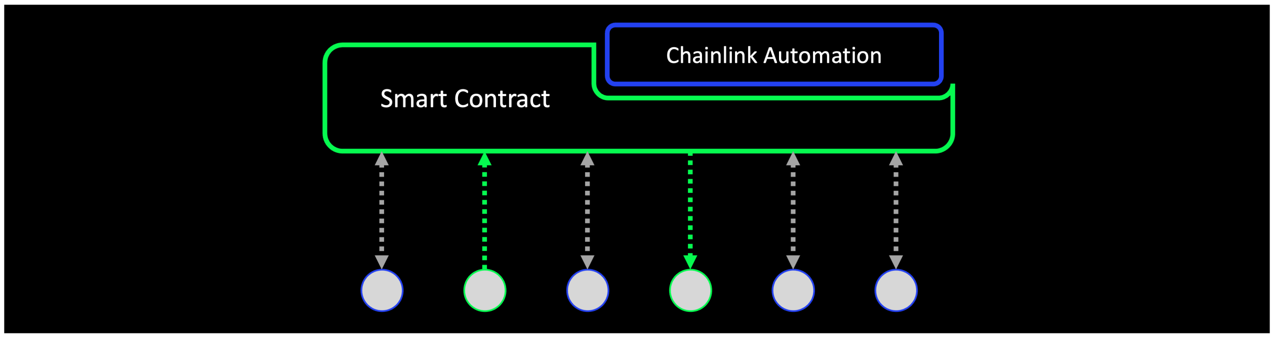
Unfortunately, the blockchain does not have and should not have a centralized player.

Therefore, blockchain users cannot get a convenience from the centralized institution, such as scheduled transfer and mis-transferred asset recovery.

KeepERC20

To provide security and convenience without compromising decentralization, we propose KeepERC20, a distributed bank-and-wallet based on smart contracts and Chainlink Automation.

FIG 3. KEEPERC20



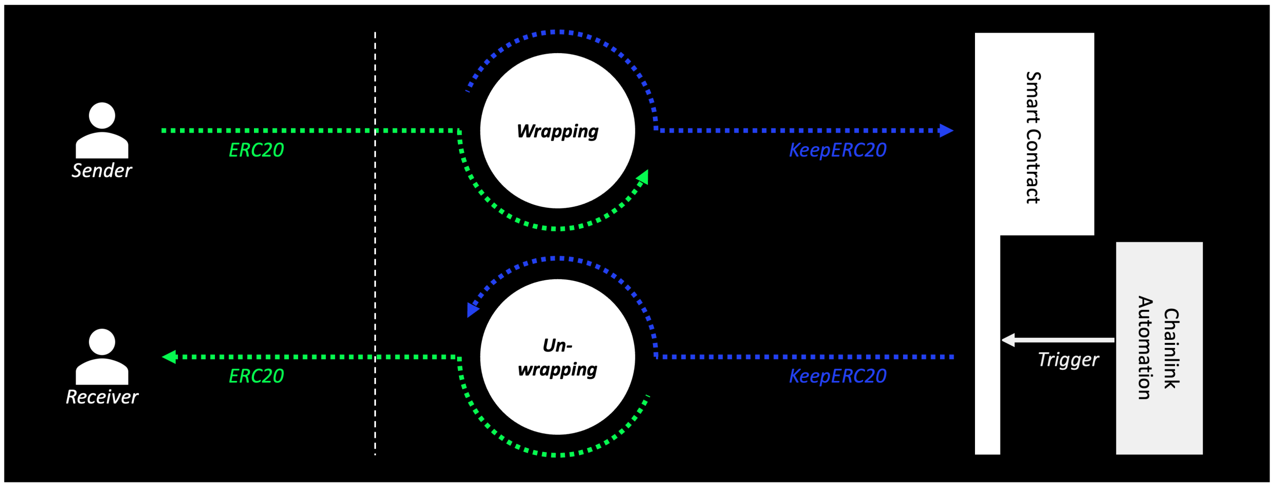
Currently, KeepERC20 offers three functions:

- Scheduled Transfer
- Recoverable Transfer
- Expirable Approve

We will continuously add more convenient and secure functions.

Overview

FIG 4. OVERVIEW OF KEEPERC20



The user can participate in KeepERC20 system with any ERC20 token.

- The sender's ERC20 tokens are internally wrapped into KeepERC20 tokens to benefit from various functions through smart contract and Chainlink Automation.
- KeepERC20 tokens are internally unwrapped into ERC20 tokens, so the receiver can easily use received tokens for other Dapps.

Some tasks may be impossible with just simple token wrapping, such as ownership-related function calls. In that case KeepERC20 system creates a safe contract wallet internally and provides those functions seamlessly.

Functions

Scheduled Transfer

The scheduled transfer is a function that automatically transfers ERC20 tokens after a few blocks. A series of bytes can be transferred together, so the contract call is also p...

Recoverable Transfer

The recoverable transfer can be treated as a kind of insurance because users can use it to prepare for mispayment.

Expirable Approve

ERC20's Approve is often the target of attack. The expirable-approve reduces the possibility of the hack by automatically canceling approval.

Scheduled Transfer

The scheduled transfer is a function that automatically transfers ERC20 tokens after a few blocks. A series of bytes can be transferred together, so the contract call is also possible. It is monitored and managed through the Upkeep of Chainlink Automation.

A predefined fee is collected as an ERC20 token when requesting a scheduled transfer.

Recoverable Transfer

The recoverable transfer can be treated as a kind of insurance because users can use it to prepare for mispayment. When the tokens were sent to an address where the private key didn’t exist or sent to the wrong contract, there was no way to recover them before.

However, if you use KeepERC20's recoverable transfer, you can get them back completely. Except for some fees.

The asset transfer through this function is finalized only when the receiver publishes unwrap transaction. If tokens are sent to the wrong address so no one can access them, the tokens will automatically return to the sender through Chainlink Automation after the expiration.

Expirable Approve

ERC20's *Approve* is often the target of attack. The expirable-approve reduces the possibility of the hack by automatically canceling approval.

Since the *increase* and *decrease* in ERC20's *Allowance* is a function that only the token owner can call, a contract wallet is created internally to control it with the KeepERC20 contract and Chainlink automation.

Register Upkeeps

Mumbai testnet is used for a concrete example.

Mumbai Testnet MATIC and LINK

You need sufficient MATICs and LINKs for registering Upkeep.

Mumbai Faucet

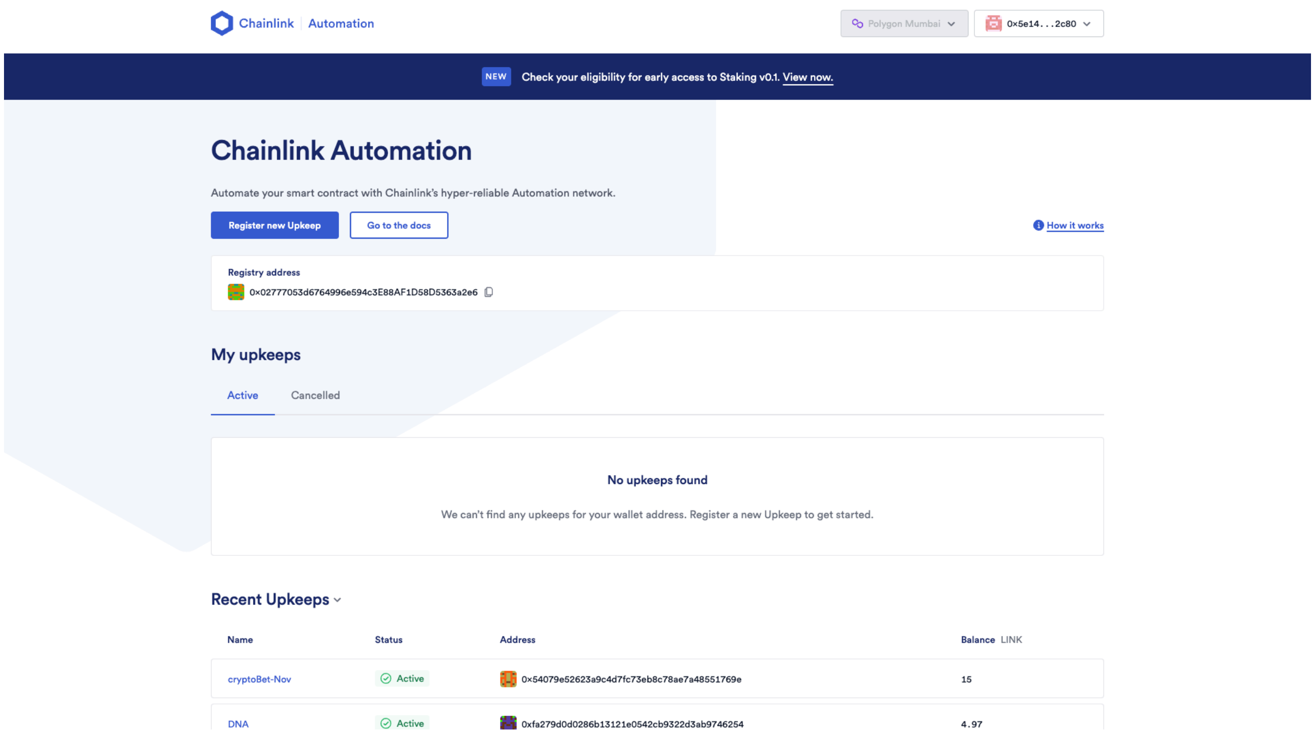
- Polygon
- Alchemy

LINK Faucet

- Chainlink

Register Upkeep

Goto Chainlink Automation page.



Check your eligibility for early access to Staking v0.1. View now.

Chainlink Automation

Automate your smart contract with Chainlink's hyper-reliable Automation network.

Registry address
0x02777053d6764996e594c3E88AF1D58D5363a2e6

My upkeeps

No upkeeps found

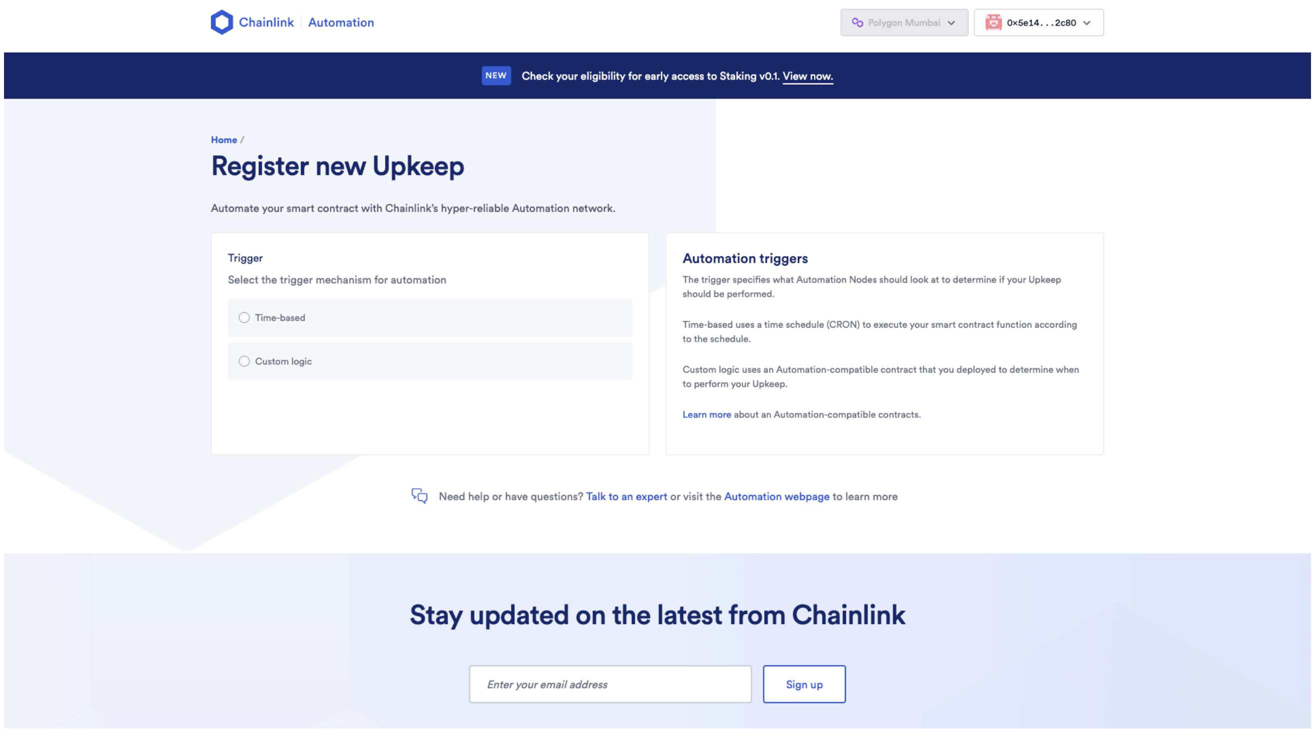
We can't find any upkeeps for your wallet address. Register a new Upkeep to get started.

Recent Upkeeps

Name	Status	Address	Balance LINK
cryptoBet-Nov	Active	0x54079e52623a9c4d7fc73eb8c78ae7a48551769e	15
DNA	Active	0xfa279d0d0286b13121e0542cb9322d3ab9746254	4.97

Register new upkeep through:

- Select `Custom logic`
- Input address of KeepERC20 contact to perform Upkeep on


Chainlink | Automation
Polygon Mumbai
0x5e14...2c80
NEW
Check your eligibility for early access to Staking v0.1. View now.
Home /

Register new Upkeep

Automate your smart contract with Chainlink's hyper-reliable Automation network.

Trigger

Select the trigger mechanism for automation

- Time-based
- Custom logic

Automation triggers

The trigger specifies what Automation Nodes should look at to determine if your Upkeep should be performed.

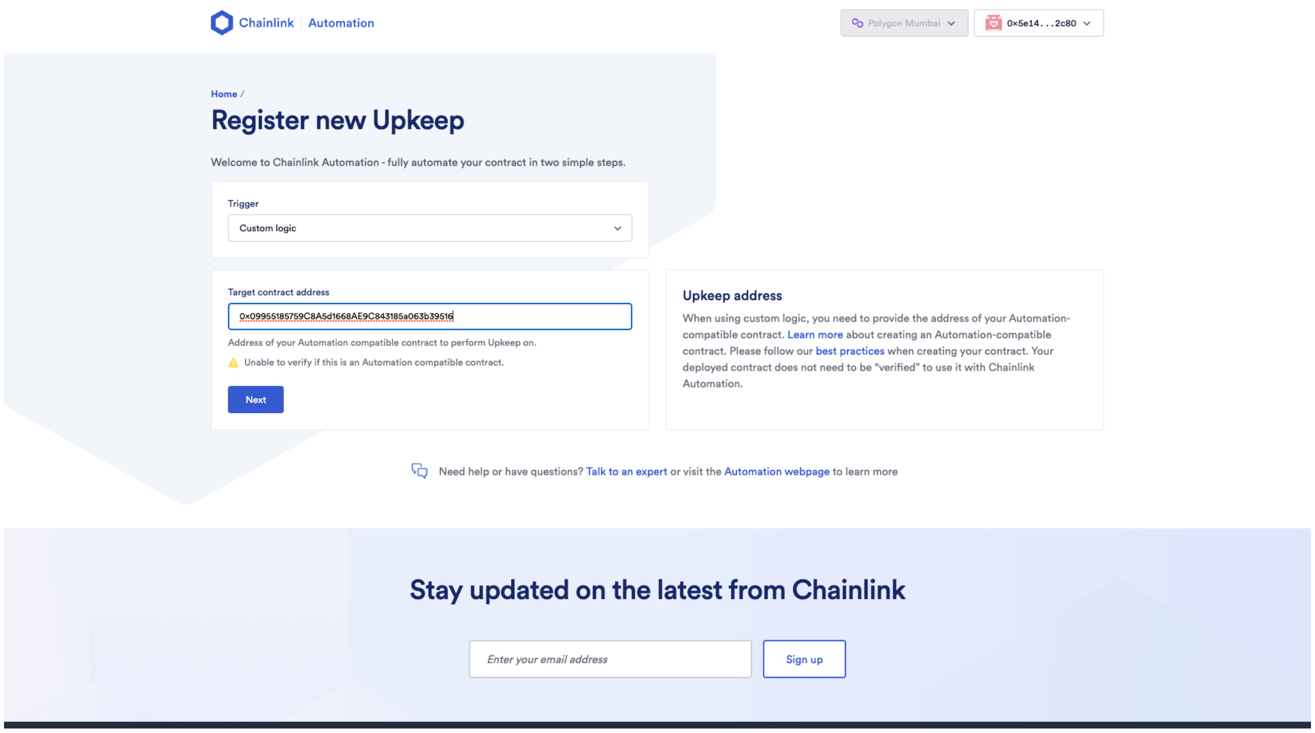
Time-based uses a time schedule (CRON) to execute your smart contract function according to the schedule.

Custom logic uses an Automation-compatible contract that you deployed to determine when to perform your Upkeep.

Learn more about an Automation-compatible contracts.

Need help or have questions? Talk to an expert or visit the Automation webpage to learn more

Stay updated on the latest from Chainlink



Register new Upkeep

Welcome to Chainlink Automation - fully automate your contract in two simple steps.

Trigger: Custom logic

Target contract address: 0x09955185759C8A5d1668AE9C843185a063b39516

Address of your Automation compatible contract to perform Upkeep on.

Unable to verify if this is an Automation compatible contract.

Upkeep address

When using custom logic, you need to provide the address of your Automation-compatible contract. Learn more about creating an Automation-compatible contract. Please follow our best practices when creating your contract. Your deployed contract does not need to be "verified" to use it with Chainlink Automation.

Need help or have questions? Talk to an expert or visit the Automation webpage to learn more

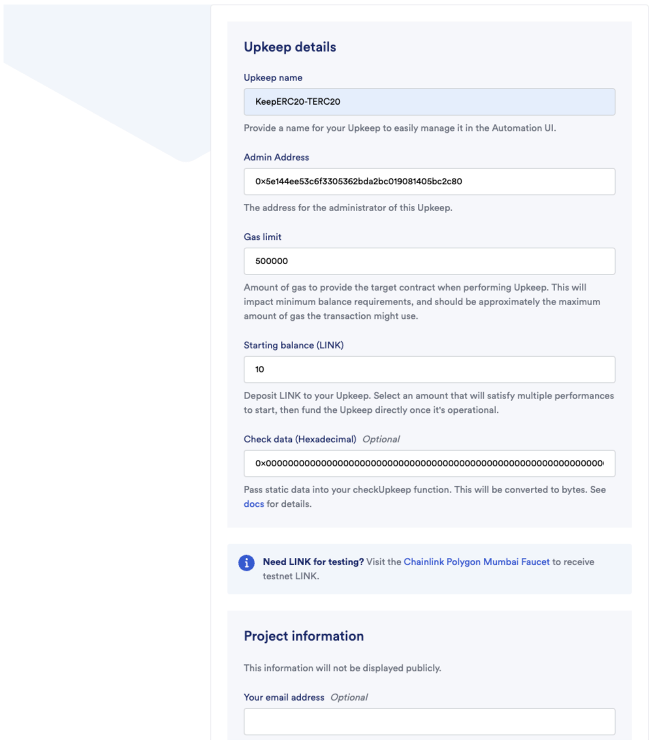
Stay updated on the latest from Chainlink

Set `check data` for pagination (lowerBound, upperBound) For example, `0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000a` for 0-to-10.

> **TIP**
>
> You can submit multiple Upkeeps for one KeepERC20 contact with different pagination.

CAUTION

Registering the same contract's Upkeeps with the same `check data` is possible but useless and inefficient.



Upkeep details

Upkeep name

KeepERC20-TERC20

Provide a name for your Upkeep to easily manage it in the Automation UI.

Admin Address

0x5e144ee53c6f3305362bda2bc019081405bc2c80

The address for the administrator of this Upkeep.

Gas limit

500000

Amount of gas to provide the target contract when performing Upkeep. This will impact minimum balance requirements, and should be approximately the maximum amount of gas the transaction might use.

Starting balance (LINK)

10

Deposit LINK to your Upkeep. Select an amount that will satisfy multiple performances to start, then fund the Upkeep directly once it's operational.

Check data (Hexadecimal) *Optional*

0x000000000000000000000000000000000000000000000000000000000000000

Pass static data into your checkUpkeep function. This will be converted to bytes. See docs for details.

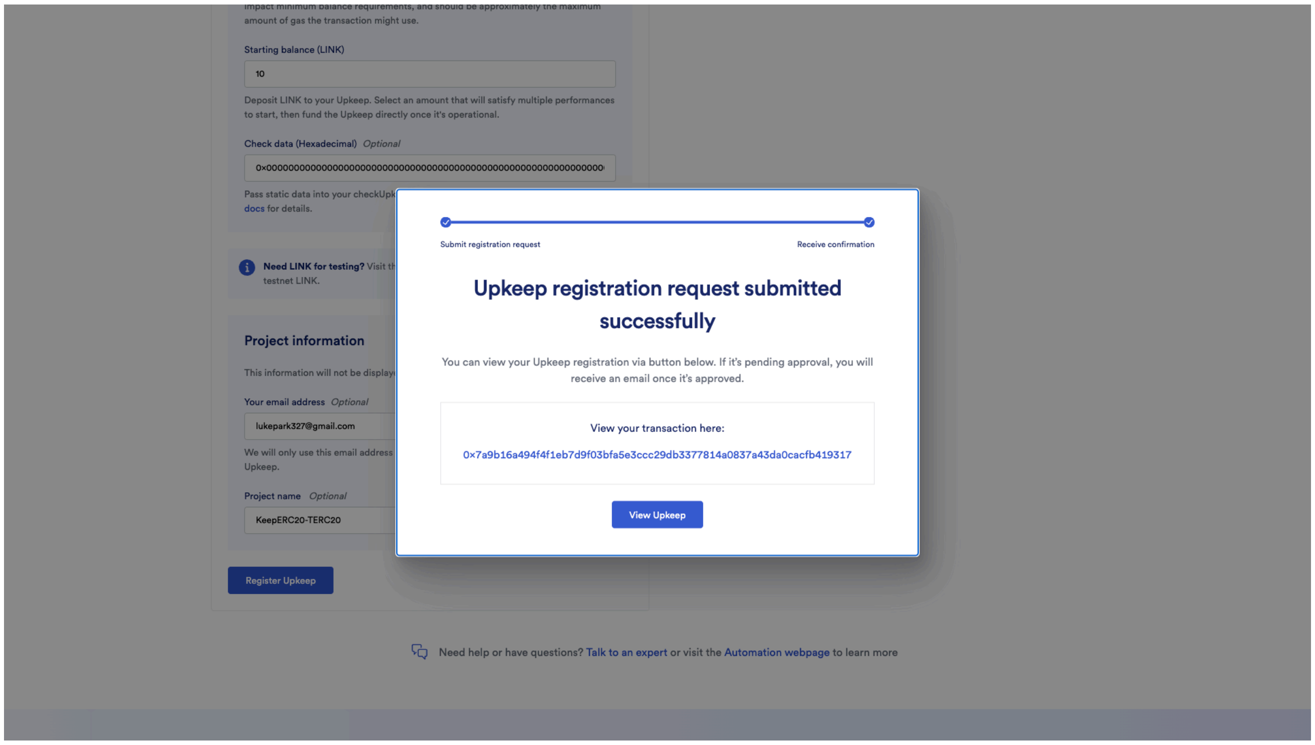
Need LINK for testing? Visit the Chainlink Polygon Mumbai Faucet to receive testnet LINK.

Project information

This information will not be displayed publicly.

Your email address *Optional*

Finally, register Upkeep.



Upkeep registration request submitted successfully

You can view your Upkeep registration via button below. If it's pending approval, you will receive an email once it's approved.

View your transaction here:

0x7a9b16a494f4f1eb7d9f03bfa5e3ccc29db3377814a0837a43da0cacfb419317

Your email address *Optional*

lukepark327@gmail.com

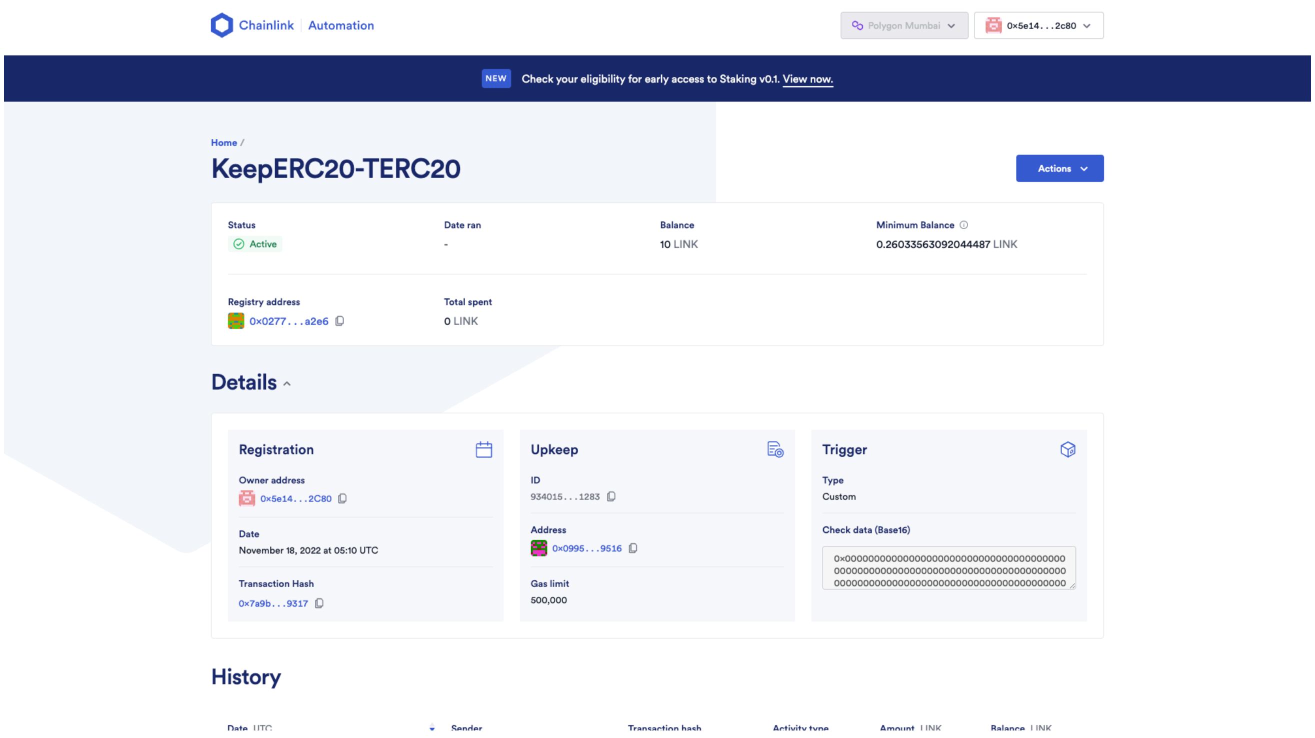
Project name *Optional*

KeepERC20-TERC20

Need help or have questions? Talk to an expert or visit the Automation webpage to learn more

You can manage your Upkeeps through Chainlink Automation page.

This is the sample Upkeep service named **KeepERC20-TERC20**.



Chainlink | Automation
Polygon Mumbai
0x5e14...2c80
NEW Check your eligibility for early access to Staking v0.1. View now.
Home /
Actions

KeepERC20-TERC20

Status: Active
Date ran: -
Balance: 10 LINK
Minimum Balance: 0.26033563092044487 LINK
Registry address: 0x0277...a2e6
Total spent: 0 LINK

Details

Registration
Owner address: 0x5e14...2C80
Date: November 18, 2022 at 05:10 UTC
Transaction Hash: 0x7a9b...9317

Upkeep
ID: 934015...1283
Address: 0x0995...9516
Gas limit: 500,000

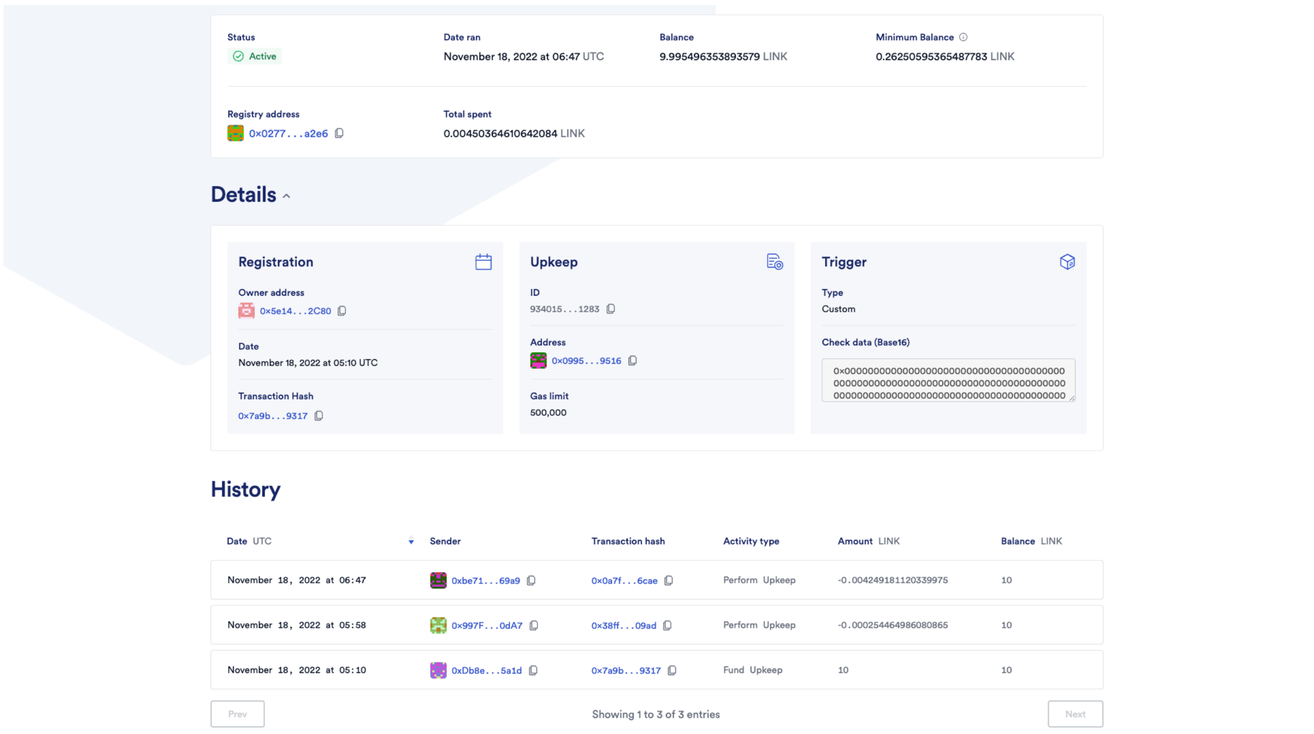
Trigger
Type: Custom
Check data (Base16):

```
0x000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
```

History

Date UTC | Sender | Transaction hash | Activity type | Amount LINK | Balance LINK

You can see the Upkeep's transaction history:



Status: Active
Date ran: November 18, 2022 at 06:47 UTC
Balance: 9.995496353893579 LINK
Minimum Balance: 0.262505953654877783 LINK
Registry address: 0x0277...a2e6
Total spent: 0.00450364610642084 LINK

Details

Registration
Owner address: 0x5e14...2C80
Date: November 18, 2022 at 05:10 UTC
Transaction Hash: 0x7a9b...9317

Upkeep
ID: 934015...1283
Address: 0x0995...9516
Gas limit: 500,000

Trigger
Type: Custom
Check data (Base16):

```
0x000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000000000000000000000000000000
```

History

Date UTC	Sender	Transaction hash	Activity type	Amount LINK	Balance LINK
November 18, 2022 at 06:47	0xbe71...69a9	0x0a7f...6cae	Perform Upkeep	-0.004249181120339975	10
November 18, 2022 at 05:58	0x997F...0dA7	0x38ff...09ad	Perform Upkeep	-0.000254464986080865	10
November 18, 2022 at 05:10	0xDb8e...5a1d	0x7a9b...9317	Fund Upkeep	10	10

For Developers

Intefaces

TBD

How to Use

See KeepERC20-wrapper for more details.

Intefaces

TBD

Please refer to GitHub.

How to Use

See KeepERC20-wrapper for more details.

Requirements

```
$ npm install
```

Set `.env`

and/or `.env.test` for test environment.

Now we can use pre-defined values as environment variable, with a prefix `dotenv -e <ENV> --`. For example, `dotenv -e .env.test --`.

Run Node

```
$ dotenv -e .env.test -- npx hardhat node --network hardhat
```

Deploy

```
$ dotenv -e .env -- npx hardhat run scripts/deploy.js --network
localhost
```

Mumbai testnet is used for a concrete example.

```
$ dotenv -e .env.test -- npx hardhat run scripts/deploy.js --
network mumbai

Compiled 2 Solidity files successfully

<Set>
Owner:   0x1ccE14942bD77f5c8EdFe408f7116595E18ccaF4
(0.19886533549475788 ETH)
User1:   0x0E723d5710E79907b0E6D3661F3fed0D3452C04c
(0.8923594874453868 ETH)
User2:   0xdBf13a0374E70f01DB7d1a570Be84e067B9E1Be1 (0 ETH)
Fee:     0x21De12f081958D5590AB70C172703345286bcDc9 (0 ETH)

<Deploy>
Deploy Token:
0x6f7ebA5Ccf6c1e524df9F0f353843B233f82e48F
Deploy Factory:
0x516e99AccB8Ebd6FC04C5FE4C516b8fF172a37e7
Deploy KeepToken:
0x09955185759C8A5d1668AE9C843185a063b39516
```

Usecase

There can be many use-cases of KeepERC20.

For example, it is possible to invest in **Dollar Cost Averaging** that purchases a certain amount of cryptocurrency periodically through DEX.

KeepERC20 allows users to use blockchain more conveniently and safely. We expect that KeepERC20 will open a new horizon for using blockchain and DApp.

Community

External Links

- WebApp

Treasury

TBD

External Links

- WebApp
- Docs
- GitHub

Contact

- lukepark327@gmail.com

Treasury

TBD