



Configuratie en administratie.

Ethereum Exchange Extension (EEE).

Gebruikershandleiding

versie 1.0.0 Beta

November 2020.

Blockoin.org is een geregistreerd handelsmerk van Bankoin Inc, onder een gratis gebruiksen commerciële licentie. Gebruiksvoorwaarden op: www.CoinSolidation.org

Inhoud

1.	Inleiding.....	3
2.	Wat is Blockly-programmering?	5
3.	Wat is een uitbreiding?	5
4.	Wat is BlockoinEthereum en BlockoinINFURA?	5
5.	Basisconcepten toegepast in het Ethereum-platform.....	6
6.	Installatie en configuratie van uitbreidings- en etherische testomgeving.....	13
7.	Definitie en gebruik van blokken (generieke functie).....	20
8.	Exchange Ethereum Extension (EEE) functies en gebeurtenissen.....	21
9.	Stappen om een CryptoToken of Cryptomoney Token te maken.	32
10.	Hoe zet je een nieuw goed of je cryptiemunt te koop (Token ERC20).	33
11.	Berekening van de standaard transactiekosten en Slimme contracttransactie	56
12.	CoinSolidation.org Vergoedingen.....	60
13.	Maak uw Android App (Exchange) aan in 15 minuten.....	61
14.	Token CoinSolidation.	64
15.	Licentie en gebruik van software.	65

1. Inleiding.

Ethereum is een open source platform, gedecentraliseerd in tegenstelling tot andere blokketens. Ethereum kan veel meer. Het is programmeerbaar, wat betekent dat ontwikkelaars het kunnen gebruiken om nieuwe soorten applicaties te maken. Zijn digitale munt heet "Ether".

Deze gedecentraliseerde toepassingen (of "dapps") krijgen de voordelen van de cryptomontage- en blokketentechnologie. Ze zijn betrouwbaar en voorspelbaar, wat betekent dat als ze eenmaal "geladen" zijn in het Ethereum, ze altijd op schema zullen lopen. Zij kunnen digitale middelen beheren om nieuwe soorten financiële toepassingen te creëren. Ze kunnen gedecentraliseerd worden, wat betekent dat geen enkele entiteit of persoon er zeggenschap over heeft.

Op dit moment zijn duizenden ontwikkelaars over de hele wereld bezig met het maken van toepassingen op Ethereum en het bedenken van nieuwe soorten toepassingen, waarvan u er veel vandaag de dag kunt gebruiken:

- Crypto-valutaportefeuilles die u in staat stellen om goedkope, directe betalingen te doen met ETH of andere activa
- Financiële toepassingen waarmee u uw digitale activa kunt lenen, uitlenen of investeren
- Gedecentraliseerde markten, waardoor u digitale middelen kunt uitwisselen, of zelfs "voorspellingen" over gebeurtenissen in de echte wereld kunt uitwisselen.
- Games waarbij je activa in het spel hebt en zelfs echt geld kan winnen.

Hoe werkt de ether?

Ether maakt, net als andere crypto-currencies, gebruik van een gedeeld digitaal boek waarin alle transacties worden vastgelegd. Het is openbaar toegankelijk, volledig transparant en zeer moeilijk achteraf aan te passen.

Dit wordt een **blokken** genoemd, en het wordt opgebouwd door het **mijnproces**.

Mijnwerkers zijn verantwoordelijk voor het controleren van groepen ethertransacties om "blokken" te vormen en deze te coderen door complexe algoritmen op te lossen. Deze algoritmen kunnen min of meer moeilijk zijn als een manier om enige consistentie in de blokverwerkingstijd (ongeveer één om de 14 seconden) te behouden.

De nieuwe blokken worden dan gekoppeld aan de vorige blokken en de betreffende mijnwerker krijgt een **beloning, namelijk** een vast aantal ethermunten. Normaal gesproken is dit 5 ethereenheden, hoewel dit aantal als variabel kan worden gezien als de cryptevaluta blijft stijgen.

Hoe werkt Ethereum?

De Ethereum-blokketten lijkt erg op de bitcoin-blokketten, maar dankzij de programmeertaal kunnen ontwikkelaars software maken waarmee ze transacties kunnen beheren en bepaalde resultaten kunnen automatiseren. Deze software staat bekend als een **slim contract**.

Als een traditioneel contract de voorwaarden van een relatie beschrijft, zorgt een slim contract ervoor dat aan die voorwaarden wordt voldaan door ze in code te schrijven. Dit zijn programma's die het contract automatisch uitvoeren zodra aan de vooraf gedefinieerde voorwaarden is voldaan, waardoor de vertraging en de kosten die ontstaan bij het handmatig uitvoeren van een overeenkomst worden geëlimineerd.

Om een eenvoudig voorbeeld te geven: een Ethereum-gebruiker zou een slim contract kunnen afsluiten om een bepaalde hoeveelheid ether naar een vriend te sturen op een bepaalde datum. Zij zouden deze code in de blokreeks schrijven en wanneer het contract is voltooid (d.w.z. wanneer de overeengekomen datum is bereikt) zal de ether automatisch worden verzonden.

Dit basisidee kan worden toegepast op complexere configuraties, en het potentieel ervan is waarschijnlijk onbeperkt, met projecten die al opmerkelijke vooruitgang hebben geboekt in sectoren als verzekeringen, onroerend goed, financiële diensten, juridische diensten en microfinanciering.

Slimme contracten hebben ook een aantal extra voordelen:

1. Ze elimineren het cijfer van de tussenpersoon, bieden de gebruiker totale controle en minimaliseren de extra kosten.
2. Ze worden geregistreerd, gecodeerd en geduplicateerd in de openbare blokketen, waar alle gebruikers de marktactiviteit kunnen zien.
3. Elimineer de tijd en moeite die nodig zijn voor handmatige processen

Natuurlijk zijn slimme contracten nog een heel nieuw systeem met veel details om te polijsten. De code wordt letterlijk vertaald, dus eventuele fouten tijdens het opstellen van het contract kunnen leiden tot ongewenste resultaten die niet kunnen worden gewijzigd.

DApps vs. slimme contracten

Intelligente contracten vertonen overeenkomsten met **DApps** (decentrale toepassingen), maar scheiden deze ook van enkele belangrijke verschillen.

Net als slimme contracten is een DApp een interface die een gebruiker via een decentraal peer netwerk verbindt met een dienst van een provider. Maar, terwijl slimme contracten een vast aantal deelnemers nodig hebben om te worden gecreëerd, hebben DApp's geen limiet aan het aantal gebruikers. Bovendien zijn ze niet alleen beperkt tot financiële toepassingen zoals slimme contracten: een DApp kan elk denkbaar doel dienen.

2. Wat is Blockly-programmering?

Blockly is een visuele programmeermethodologie gebaseerd op de programmeertaal JavaScript. Het bestaat uit een eenvoudige set commando's die we kunnen combineren alsof het de stukjes van een puzzel zijn. Het is een zeer nuttig hulpmiddel voor degenen die op een intuïtieve en eenvoudige manier willen **leren programmeren** of voor degenen die al weten hoe ze moeten programmeren en het potentieel van dit soort programmering willen zien.

Blockly is een vorm van programmeren waarbij je geen enkele achtergrond in welke computertaal dan ook nodig hebt, dit is omdat het gewoon het aansluiten van grafische blokken is alsof we lego of een puzzel spelen, je moet gewoon wat logica hebben en dat is het!

Iedereen kan programma's voor mobiele telefoons (smartphones) maken zonder te knoeien met die programmeertalen die moeilijk te begrijpen zijn, gewoon blokken op een grafische manier in elkaar zetten op een eenvoudige, gemakkelijke en snelle manier om te maken.

3. Wat is een uitbreiding?

Een extensie is een module die gemaakt is in een Java-programmeertaal die in een bestand met de extensie .AIX wordt gegeven.

In het Blockoin.org project hebben we ons gebaseerd op het creëren van gebruiksvriendelijke extensies voor het financiële gebied waarmee ze in een paar minuten mobiele applicaties kunnen maken zonder dat ze een enorme kennis van programmeren hebben.

4. Wat is BlockoinEthereum en BlockoinINFURA?

BlockoinEthereum en BlockoinINFURA zijn software voor vrij gebruik (extensies) die de volgende technologische oplossingen (algoritmen) bevat om te kunnen creëren die in het netwerk van het Ethereum worden gebruikt:

- HET CREËREN VAN NIEUWE ACCOUNTS OP HET ETHERISCHE PLATFORM.
- BALANSOVERLEG, OVERDRACHT VAN ACTIVA (CRYPTOMONEDA-ETHER EN PENNINGEN)
- CREATIE VAN NIEUWE ERC20 LOPERS EN CRYPTOMONEDA-PENNING.
- COMPILATIE, CREATIE, RAADPLEGING EN PUBLICATIE VAN EEN SLIM CONTRACT
- HET CREËREN VAN OFFLINE EN ONLINE TRANSACTIES.
- HET LADEN VAN DE PRIMAIRE EN PUBLIEKE SLEUTELS.
- WISSELKOERSEN & TANKSTATIONOVERLEG.
- ONTWIKKELING, TESTEN EN ETHERISCHE KERNNETWERKOMGEVINGEN (NETWERKEN)
- BEVAT DE 39 INFURA KENMERKEN.

5. Basisconcepten toegepast in het Ethereum-platform.

Wat is een blokken?

De blokken wordt over het algemeen geassocieerd met Bitcoin en andere crypto valuta, maar deze zijn slechts het topje van de ijsberg, omdat het niet alleen wordt gebruikt voor digitaal geld, maar kan worden gebruikt voor alle informatie die een waarde kan hebben voor gebruikers en/of bedrijven. Deze technologie, die haar oorsprong vindt in 1991, toen Stuart Haber en W. Scott Stornetta het eerste werk aan een keten van cryptografisch beveiligde blokken beschreef, werd pas in 2008 opgemerkt, toen het populair werd met de komst van de bitcoin. Maar op dit moment wordt het gebruik ervan in andere commerciële toepassingen gevraagd en de verwachting is dat het op middellange termijn zal groeien in verschillende markten, zoals financiële instellingen of het internet van de dingen aan het internet van de dingen, naast andere sectoren.

De blokken, beter bekend onder de term blokken, is een enkel, overeengekomen record dat verdeeld is over verschillende knooppunten (elektronische apparaten zoals pc's, smartphones, tablets, etc.) in een netwerk. In het geval van crypto-valuta's kunnen we het zien als het boekje waarin elk van de transacties wordt geregistreerd.

De werking ervan kan complex zijn om te begrijpen als we ingaan op de interne details van de uitvoering ervan, maar het basisidee is eenvoudig te volgen.

Het wordt opgeslagen in elk blok:

- 1.- een aantal geldige records of transacties,
- 2.- informatie over dat blok,
- 3.- het verband met het vorige blok en het volgende blok door de hash van elk blok – een unieke code die zou zijn als de vingerafdruk van het blok.

Daarom heeft **elk blok een specifieke en onbeweeglijke plaats binnen de keten**, aangezien elk blok informatie bevat uit de hasj van het vorige blok. De hele keten wordt opgeslagen op elk netwerkknoppen dat de blokken vormt, dus **een exacte kopie van de keten wordt op alle deelnemers aan het netwerk opgeslagen**.

Wat is een adres of account binnen het blokken-ethereumplatform?

Het is een reeks van 42 karakters in het Ethereum-platform die een getal in hexadecimale basis vertegenwoordigt, waar de in het Ethereum gedefinieerde activa zullen worden gedeponerden of verzonden. In andere blokkenplatforms kan het aantal karakters van de account of het adres bijvoorbeeld verschillen:

0x5d2Acdb34c279Aa6d1e94a77F7b18aB938BFb2bB

Wat is een kryptomoney?

Het is een digitale of virtuele valuta die is ontworpen om te functioneren als een medium voor uitwisseling. Het maakt gebruik van cryptografie (digitale beveiliging) om transacties te beveiligen en te verifiëren, en om de creatie van nieuwe eenheden van een bepaalde cryptomoney te controleren.

Wat is Ether?

Ether is de inheemse digitale valuta van het Ethereum blokketenplatform, een decentraal platform dat in 2013 is ontwikkeld. Een Ether is een eenheid die kan worden onderverdeeld in kleinere eenheden die WEI worden genoemd en heeft de volgende equivalentie.

1 Ether = 1,000,000,000,000,000,000 Wei. (In wiskundige termen is het 10^{18}).

Welke eenheden worden behandeld bij gebruik van een Ether?

1	ether
10^3	finney
10^6	szabo
10^9	Shannon of GWEI dit wordt gebruikt voor de GasPrice.
10^{12}	babysit
10^{15}	lovelace
10^{18}	wei

Wat is een penning?

Munten zijn digitale middelen die binnen een bepaald projectecosysteem kunnen worden gebruikt.

Het belangrijkste onderscheid tussen penningen en crypto-currencies is dat de eerste een ander blokketenplatform (niet hun eigen) nodig hebben om te kunnen werken. Ethereum is het meest voorkomende platform voor het maken van penningen, voornamelijk vanwege de

slimme contractfunctie. De penningen die op de Ethereum-blokketten worden gemaakt zijn over het algemeen bekend als ERC-20 penningen, hoewel er andere, meer gespecialiseerde soorten penningen zijn, zoals de ERC-721 penningen die voornamelijk worden gebruikt voor verzamelobjecten (kaarten, gebruik in videospelletjes, artwork, enz.).

Wat is Gas?

Gas is de kosten voor het uitvoeren van een operatie of een reeks operaties op het Ethereum-netwerk. Deze operaties kunnen verschillende zijn: van het maken van een transactie tot het uitvoeren van een slim contract of het creëren van een decentrale applicatie.

Met andere woorden, eenvoudiger gezegd, het gas is de eenheid voor het meten van het werk dat in het etherische systeem wordt gedaan.

Net als in de fysieke wereld zijn er ook in Ethereum banen die meer kosten dan andere: als de operatie die we willen uitvoeren een groter gebruik van middelen door de knooppunten die het platform vormen vereist, zal dit het Gas ook doen toenemen en vice versa.

Het gas dient in de eerste plaats om drie functies te vervullen op het Ethereum-platform:

1.- Wijst een kostprijs toe aan de uitvoering van taken; In Ethereum heeft het uitvoeren van taken ook een kostprijs. **Afhankelijk van de moeilijkheidsgraad van de taak of de snelheid waarmee we die taak willen laten verwerken, zullen de rekenkosten van die operatie dienovereenkomstig hoger of lager zijn** en zal het aantal gassen proportioneel toenemen of afnemen.

2.- Beveilt het systeem; Het ethereumsysteem is een veilig systeem en dit is grotendeels mogelijk dankzij het gas.

Door te eisen dat voor elke uitgevoerde transactie een commissie wordt betaald, zorgt het platform ervoor dat er geen onbruikbare transacties op het netwerk worden verwerkt. Dit helpt om de blokketten lichter te maken, omdat het niet veel nutteloze Megabytes aan informatie zal toevoegen aan de blokketen.

Daarnaast is het systeem met het Gas ook beschermd tegen 'spam' en het oneindige gebruik van lussen: instructies om repeterende taken per code uit te voeren.

Als het gas bijvoorbeeld niet bestond, zou niets voorkomen dat een taak zich oneindig zou herhalen, waardoor het systeem in elkaar zou storten en onbruikbaar zou worden.

3.- Beloning van de mijnwerkers; De mijnwerkers zijn onafhankelijke externe systemen die over de hele wereld verspreid zijn en die belast zijn met de uitvoering van de transacties binnen het Ethereum-platform. Wanneer we een transactie maken of een slim contract uitvoeren, "betalen" we een bepaalde hoeveelheid Gas.

Dit gas dient om de mijnwerkers te "betalen" voor de grondstoffen die ze hebben gebruikt (hardware, elektriciteit en tijd) en voegt ook een beloning toe voor hun werk. Daarom zouden we kunnen zeggen dat het gas ook bijdraagt aan het behoud van het evenwicht van het platform.

We hebben het over het "betalen" van gas -in aanhalingstekens- want dat is wat operaties kosten. Met andere woorden, je "betaalt" voor de kosten die het Ethereum kost om de transacties te verwerken.

Wat is de gasprijs?

Een Gasunit komt overeen met de uitvoering van een instructie, zoals een computerstap.

Dus hoe "verzilveren" de mijnwerkers het gas dat ze als beloning krijgen?

Het gas zelf is niets waard en kan dus niet worden opgeladen. Om dit gebruikte gas "in rekening te brengen", is het noodzakelijk om waarde te geven aan deze verbruikte hulpbronnen, dat wil zeggen om een monetaire waarde te geven aan het werk dat door de mijnwerkers wordt gedaan. De hoeveelheid gas die in een transactie of een slim contract wordt gebruikt, heeft een gelijkwaardige prijs in Ether. Deze prijs wordt de 'GasPrijs' genoemd, deze wordt normaal gesproken toegewezen door de mijnwerkers en is variabel afhankelijk van hoe druk de Ethereum-blokketen is. Het wordt normaal gesproken behandeld in GWEI-eenheden.

Wat is Gas Limit?

Deze gegevens geven de maximale waarde van Gas aan die een transactie kan verbruiken om geldig te zijn.

Normaal gesproken berekent de software die gebruikt wordt om transacties op het Ethereum-netwerk uit te voeren automatisch een schatting van de hoeveelheid Gas die nodig is om de transactie uit te voeren en toont deze ons direct.

Wat is een tankstation?

Het is de plaats waar de waarden die door de mijnwerkers worden behandeld, worden doorverwezen om in consensus te beslissen welke GasPrijs nodig is om de verschillende soorten transacties in de Ethereum-blokketen uit te voeren.

Afhankelijk van hoeveel van de GasPrijs wordt geselecteerd, zal de prioriteitstijd die wordt gegeven aan de uitvoering van de transactie worden gewogen, normaal gesproken worden drie soorten GasPrijs afgehandeld: HANDELAAR: Z.s.m., SNELHEID < 2 minuten, STANDAARD < 5 minuten. Dit zijn gemiddelde waarden en kunnen variëren in de tijd, afhankelijk van de werklast (tracties) van het hoofdnet van het Ethereum.

<https://ethgasstation.info/>

Wat is een beurs?

Een Kryptomaniac Exchange is het ontmoetingspunt waar de uitwisseling van Kryptomaniacs plaatsvindt in ruil voor fiatgeld of andere Kryptomaniacs. In deze online beurshuizen wordt de marktprijs gegenereerd die de waarde van de cryptomoniën aangeeft op basis van vraag en aanbod.

Wat is Exchange Rates?

Dit zijn de koersen voor de waarde van een Ether in de valuta's van elk land. Zo heeft een Ether op de dag van de totstandkoming van dit handboek een waarde in Amerikaanse dollars van \$430,94

Wat is een Smart Contract?

In ethereum een Smart contract is een programma in programmeertaal genaamd Solidity dat binnen de blockchain Ethereum, die instructies heeft om automatisch te worden uitgevoerd op basis van vooraf vastgestelde regels, deze eigenschap maakt een evolutie in alle bestaande blockchain als u kunt veel Smart contract aangepast aan verschillende particuliere en openbare sectoren waardoor efficiëntere activiteiten van bedrijven of waar nodig aan te passen aan systemen of programma's van alle soorten.

Wat is "nonce"?

Het is een incrementele "hexadecimaal getal" teller die de transacties in elke richting of rekening bijhoudt die in Ethereum is aangemaakt.

Wat is een transactie?

Het is de uitvoering of overdracht van een bepaald type niet-materieel actief dat binnen het ethereumsysteem een vooraf vastgestelde waarde kan krijgen en dat later kan worden omgezet in een materiële waarde voor een bedrijf of persoon.

Wat is txHash?

Het is een hexadecimaal getal dat helpt om het resultaat van elke transactie in detail te volgen.

Welke soorten transacties zijn er?

Je hebt twee soorten, een is de transactie "offline" dit creëert zonder de noodzaak om verbinding te hebben met het hoofdnetwerk van Ethereum kan worden opgeslagen totdat u ervoor kiest om verbinding te maken met het netwerk van Ethereum en laat de transactie, hebben het voordeel van de veiligheid, omdat de hele transactie wordt verwerkt offline die

voorkomt dat elke anomalie die zou kunnen worden in de netwerkverbinding. De andere transactie is de "online" transactie die altijd verbonden moet zijn met het internet met de veiligheidsvoordelen en -nadelen die de verbinding met zich meebrengt.

Wat is INFURA.io?

Infura.io is een platform dat een set van tools en infrastructuren biedt die ontwikkelaars in staat stellen om hun applicatieblokketten gemakkelijk te nemen, van testen tot schalen, met eenvoudige en betrouwbare toegang tot Ethereum en IPFS.

Wat is een adres op het Ethereum-netwerk?

Een adres of account bestaat uit drie delen, het adres, de openbare sleutel en de privésleutel, deze twee sleutels zijn een reeks getallen en karakters in hexadecimaal formaat die worden gebruikt om te verzenden en te ontvangen (actief) of ether (digitale valuta).

De primaire sleutel mag nooit met iemand worden gedeeld, omdat het de sleutel is die de vrijgave van het saldo (de transacties) op de rekening toestaat.

De publieke sleutel is bekend bij het hele publiek en wordt met iedereen gedeeld, omdat het de referentie is om te bevestigen dat de transactie correct is, zowel in termen van waarde als naar wie hij wordt gestuurd.

Voorbeelden:

```
{  
  "private": "429a043ea6393b358d3542ff2aab9338b9c0ed928e35ec0aed630b93adb14a1c",  
  "public":  
    "049b4b7e72701a09d3ee09165bba460f2549494a9d9fd7a95aaac57c2827eac162fd9e105b  
    2461cd6594ca8ca6a8daf10fe982f918be1b0060c87db9cfbcd289a8",  
  "address": "88ab6dcecc3603c7042f4334fc06db8e8d7062d5"  
}
```

Wat is Coinsolidation.org?

Het is een platform in het consolideren van crypto-currencies, we hebben de eerste hybride adressen in de gecentraliseerde en gedecentraliseerde financiële wereld gecreëerd, met behulp van Blockly technologie die een vorm van visuele programmering is zonder de

noodzaak van geavanceerde kennis van programmeren op basis van functionele uitbreidingen. Zie onze WhitePaper op www.coinsolidation.org

Welke soorten netwerken voor het testen en het hoofdnetwerk in het blokketen-Ethereum?

<https://mainnet.infura.io>

Hoofdnetwerk, productienetwerk waar alle transacties in real time worden uitgevoerd.

<https://ropsten.infura.io>

Ropsten testnetwerk laat blokketenontwikkelingen toe om hun werk te testen in een live omgeving, maar zonder de noodzaak van echte ETH-tokens, met een algoritme genaamd (*Proof-of-Work*).

<https://kovan.infura.io>

Kovan testnetwerk, dat in tegenstelling tot zijn voorganger ropsten gebruik maakt van een algoritme genaamd Proof of Authority.

<https://rinkeby.infura.io>

Test Netwerk, maakt gebruik van een algoritme genaamd Proof of Authority. Een van de meest gebruikte voor testen.

<https://goerli.infura.io>

Goerli is een openbaar testnetwerk voor Ethereum dat de POA (Proof of Authority) consensusmotor ondersteunt bij verschillende klanten. Spambestendig.

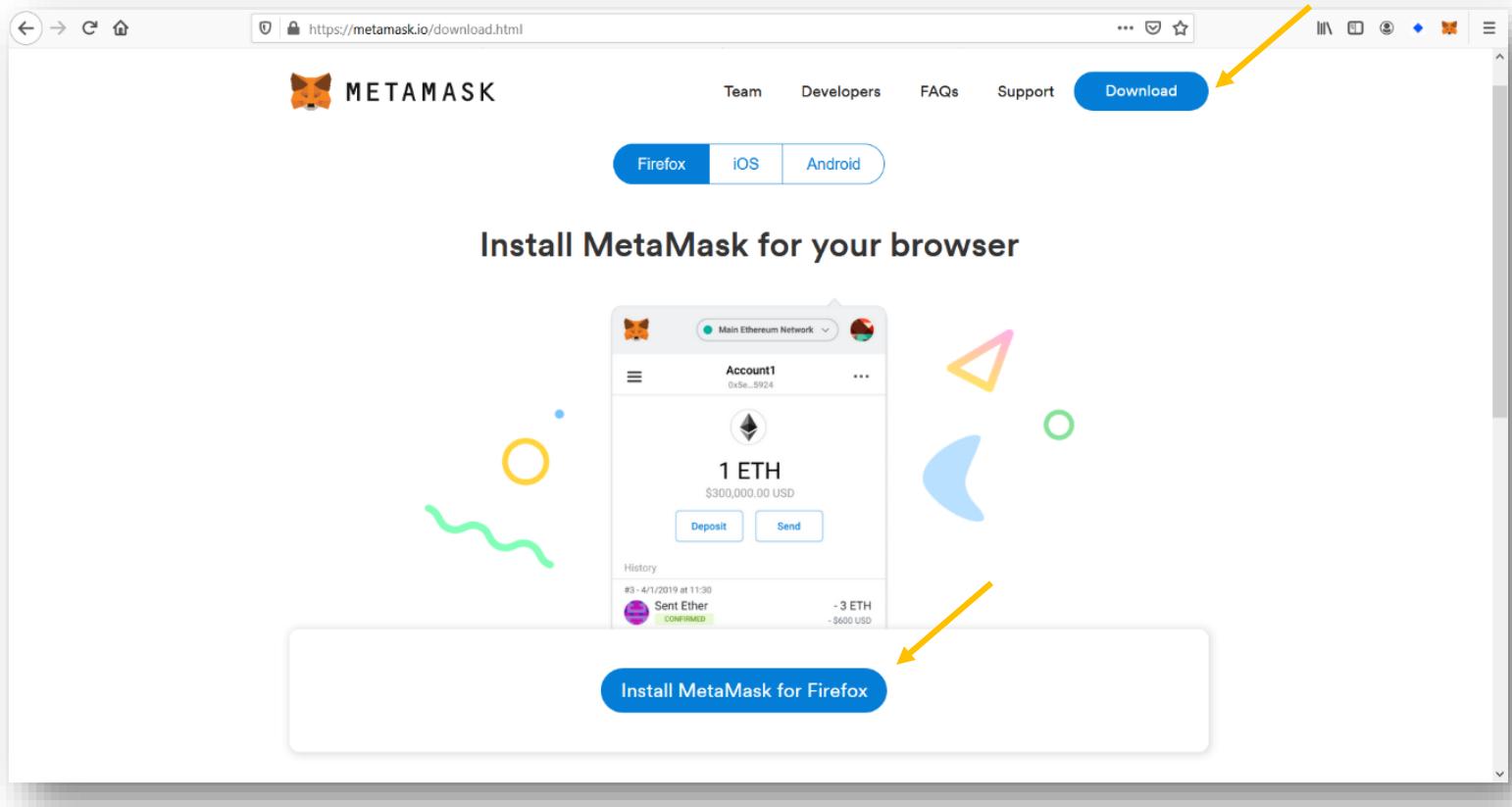
In totaal zijn er 5 netwerken op basis van de blockchain Ethereum, één voor de productie of het hoofd en vier voor het testen, dan zullen we het netwerk van Rinkeby gebruiken om onze testomgeving te installeren.

6. Installatie en configuratie van uitbreidings- en etherische testomgeving.

We hebben eerst een testomgeving nodig. Het leren gebruiken van de verschillende functionaliteiten van de twee extensies die gebruikt zullen worden voor het creëren, verzenden, publiceren, beoordelen en verkrijgen van gegevens van alle transacties die we op het Ethereum-platform maken.

Het eerste wat we moeten doen is onze testomgeving inrichten. We zullen de **METAMASK** applicatie moeten installeren. Dit is een portemonnee om Ethereum accounts aan te maken, te importeren en transacties te beheren vanuit de browser in onze internetbrowser die we zullen gebruiken is Mozilla en kan ook gebruikt worden in Chrome.

Ga naar de officiële website [www.metamask.io](https://metamask.io/download.html) en klik op de knop "Download".



Klik vervolgens op de knop "MetaMask voor Firefox installeren" en accepteer de standaardopties. Na de installatie zien we rechtsboven een icoontje waar we de Metamask software al geïnstalleerd zien.

Klik op het Metamask-pictogram en de optie om een bestaande account te importeren of een nieuwe aan te maken verschijnt. In ons geval importeren we een account dat we al hebben lopen met behulp van de methode van "herstellen door middel van zaad". Als je er

geen hebt, maak dan gewoon een nieuw account aan. In dit geval wordt ons in beide gevallen gevraagd een wachtwoord toe te kennen.

Later voeren we met het "wachtwoord" in en kunnen we controleren welk saldo we hebben, logischerwijs is het saldo als het nieuw is, dan is het nihil.

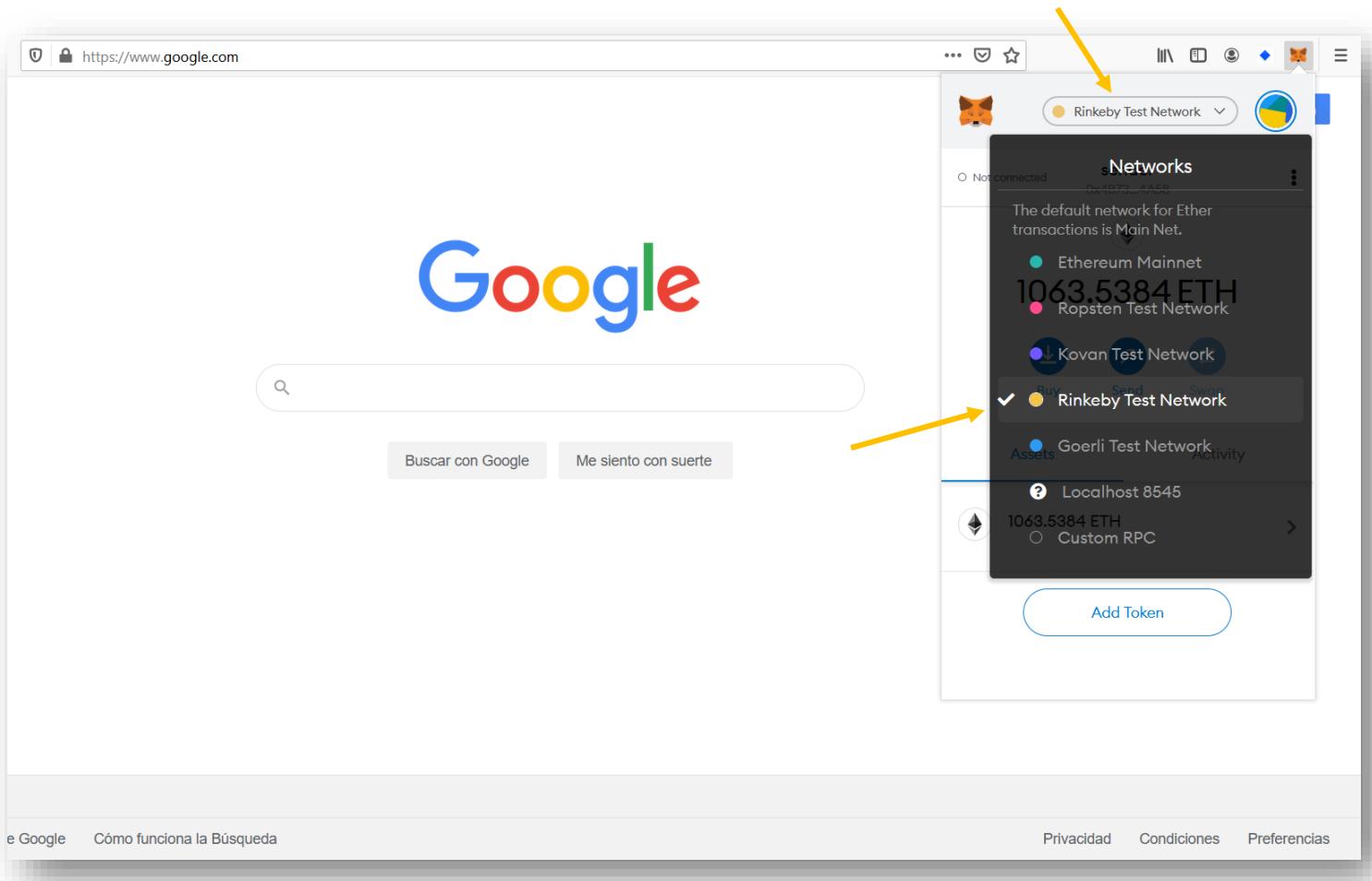
The screenshot shows a web browser window with the Google homepage loaded at <https://www.google.com>. Overlaid on the page is a wallet interface for Ethereum. The wallet interface displays the following information:

- Ethereum Mainnet account selected (indicated by a yellow arrow).
- Address: 0x4B73...4A58
- Balance: 0.0719 ETH (\$31.29 USD)
- Buttons: Buy, Send, Swap
- Sections: Assets (selected), Activity
- Link: Add Token

At the bottom of the browser window, there are links for Google search help and privacy settings.

We gaan nu bekijken hoe we wat ethers (digitale munten) gaan storten op onze Rinkeby testaccount.

Bovenaan hebben we de optie om het type netwerk te kiezen dat we zullen gebruiken, standaard wanneer je het invoert plaatst het je in het "Ethereum Mainnet", echter, wanneer je klikt kunnen we alle optionele netwerken bekijken die we kunnen kiezen, in ons geval hebben we het Rinkeby netwerk geselecteerd.



De volgende stap is het storten van ethers op ons Rinkery netwerk waarnaar wordt verwezen als testaccount.

BELANGRIJKE OPMERKING: De ethers (digitale valuta) die we op onze rekening storten en die verwijzen naar het Rinkery testnetwerk hebben geen waarde in de crypto-markt, ze worden door ons alleen gebruikt voor het testen van software. Controleer altijd op welk netwerk we werken om fouten in transacties te voorkomen.

Om ether te krijgen voor het testen moeten we de volgende procedure uitvoeren.

In elk twitteraccount dat je hebt zullen we twitter moeten invoeren en een opmerking maken die alleen het account en/of het adres dat we hebben in Metamask bevat dan zullen we de link van het commentaar moeten kopiëren aangezien we dit hebben zullen we naar de volgende link gaan om ons account te verankeren.

<https://faucet.rinkeby.io/>

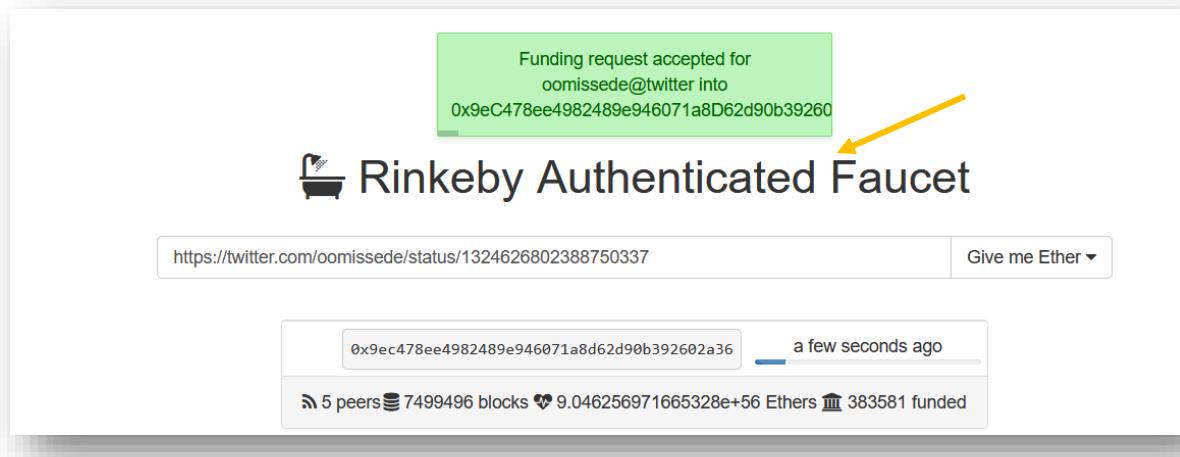
We hebben een twittercommentaar gemaakt en de link uit het commentaar gekopieerd.

A screenshot of a Twitter tweet from user AGAVE (@oomissede). The tweet content is a long Ethereum address: 0x88ac6dcecc3603c7042f4334fc06db8e8d7062d5. Below the tweet, it says "Traducir Tweet" and "2:22 a. m. · 6 nov. 2020 · Twitter Web App". At the bottom, there are icons for reply, retweet, like, and share. A yellow arrow points to the URL in the browser's address bar: https://twitter.com/oomissede/status/1324628185468854272.

Op de site <https://faucet.rinkeby.io/> kopiëren we de twitterlink en op de knop "Geef me Ether" kiezen we het gewenste bedrag.

A screenshot of the Rinkeby Authenticated Faucet website. At the top, it says "Rinkeby Authenticated Faucet". Below that is a form with an input field containing the Twitter URL: https://twitter.com/oomissede/status/132462802388750337. To the right of the input field is a button labeled "Give me Ether ▾". A dropdown menu is open, showing three options: "3 Ethers / 8 hours", "7.5 Ethers / 1 day", and "18.75 Ethers / 3 days". Two yellow arrows point to the "Give me Ether" button and the dropdown menu. Below the form, there is a section titled "How does this work?" with instructions for requesting funds via Twitter or Facebook. At the bottom, there is a note about reCaptcha protection.

Als alles correct is, geven we ten slotte een aankondiging dat het depot werd aanvaard en afhankelijk van de werklast van het Rinkeby-netwerksysteem zal het depot in een paar minuten of misschien langer duren.



Nu we Ethers op onze rekening hebben staan, kunnen we beginnen met testen op het Ethereum-platform.

We hebben twee extensies voor de interactie met de Ethereum-blokketten.

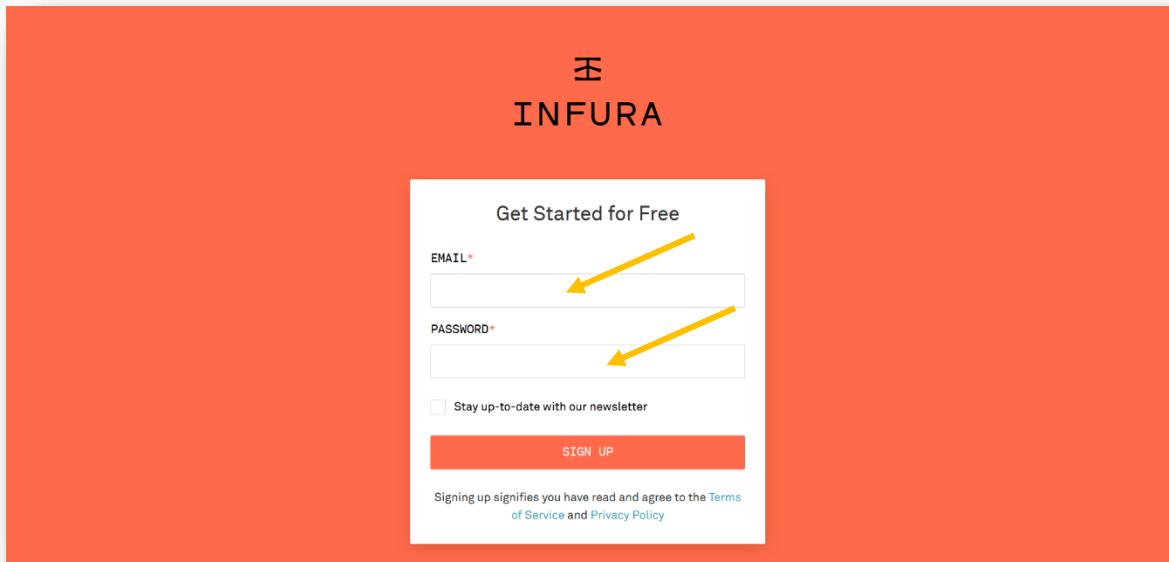
De **co-solidatie** uitbreiding. **BlockoinETHEREUM.aix** bevat de functies om de volgende functies uit te voeren:

- Aanmaken van nieuwe accounts (adressen) in blockchain Ethereum (privateKey, publicKey, Address)
- Opslag van accountgegevens (adressen) in binaire bestanden.
- Importeren van binaire bestandsrekeningen (adressen)
- Het verkrijgen van een account (adres) via privateKey.
- Aanmaken en versturen van transacties tussen rekeningen (adressen) "online".
- Aanmaken, ondertekenen en versturen van offline PushRaw-transacties.
- Raadpleging van transactiegegevens Tx.
- Balanscontrole op adressen en slimme contracten.
- Compiler voor Slimme contracten.
- Creatie, publicatie en uitvoering van slimme contracten.
- Creatie, publicatie en uitvoering van Token ERC20 (cryptomoney token).
- Het verkrijgen van ABI-code uit een slim contract.
- Verificatie van de netwerkverbinding.
- Vraag naar de waarde van de ether in de cryptomarkt in elk land van de wereld (inheemse valuta van het land) - Wisselkoersen.
- GasPrijs-overleg.
- Raadpleging van "nonce" van een specifieke rekening.

De **muntsöïdatie-uitbreiding**. **BlockoinINFURA.aix** biedt ons de 40 functionaliteiten van het Infura.io-platform. Voor meer informatie kunt u de documentatie van json-rpc rechtstreeks raadplegen op <https://infura.io/docs>.

Om de INFURA extensie te kunnen gebruiken moet u een account aanmaken op de infura.io site omdat we een KEY API nodig hebben om vragen te kunnen stellen aan het Ethereum netwerk, en om gebruik te kunnen maken van de Ethereum test netwerken.

Hoe we een rekening openen is eenvoudig, zoals hieronder weergegeven.



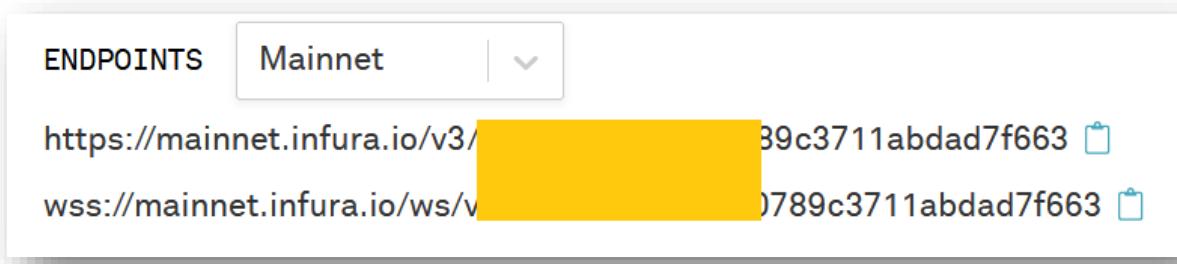
Zodra het account is aangemaakt, gaan we naar binnen en kunnen we de KEY API van de verschillende netwerken hebben. We gaan naar linksboven en klikken op Ethereum, dan zien we de projecten, creëren we een nieuw project en stellen we ons voor aan dat project.

A screenshot of the Infura dashboard for the Ethereum network. The top navigation bar shows the Infura logo and the text 'Ethereum'. On the left, there's a sidebar with icons for 'ETHEREUM' (selected), 'FILECOIN', 'DOCS', 'COMMUNITY', and 'SUPPORT'. The main content area shows a list of projects under 'PROJECTS'. One project is listed: 'COINSOLIDATION' (Created September 12, 2020), which is 'Active'. There are also sections for 'REQUESTS PREV 7 DAYS' (line chart showing activity) and 'PLAN' (Core plan at 100,000 Requests/Day). A red 'CREATE NEW PROJECT' button is located in the top right corner of the main content area. Three yellow arrows point to the 'ETHEREUM' icon in the sidebar, the 'CREATE NEW PROJECT' button, and the 'COINSOLIDATION' project entry.

Binnen het project hebben we de gegevens van de KEY API en de verschillende netwerken die we kunnen gebruiken.

Binnen het project kunnen we de API KEY (PROJECT ID) bekijken en selecteren op welk netwerk we gaan werken of de volledige links van de ENDPOINTS kiezen.

Om bijvoorbeeld het hoofdnetwerk van het Ethereum te gebruiken, zouden we de volgende competitie nemen:

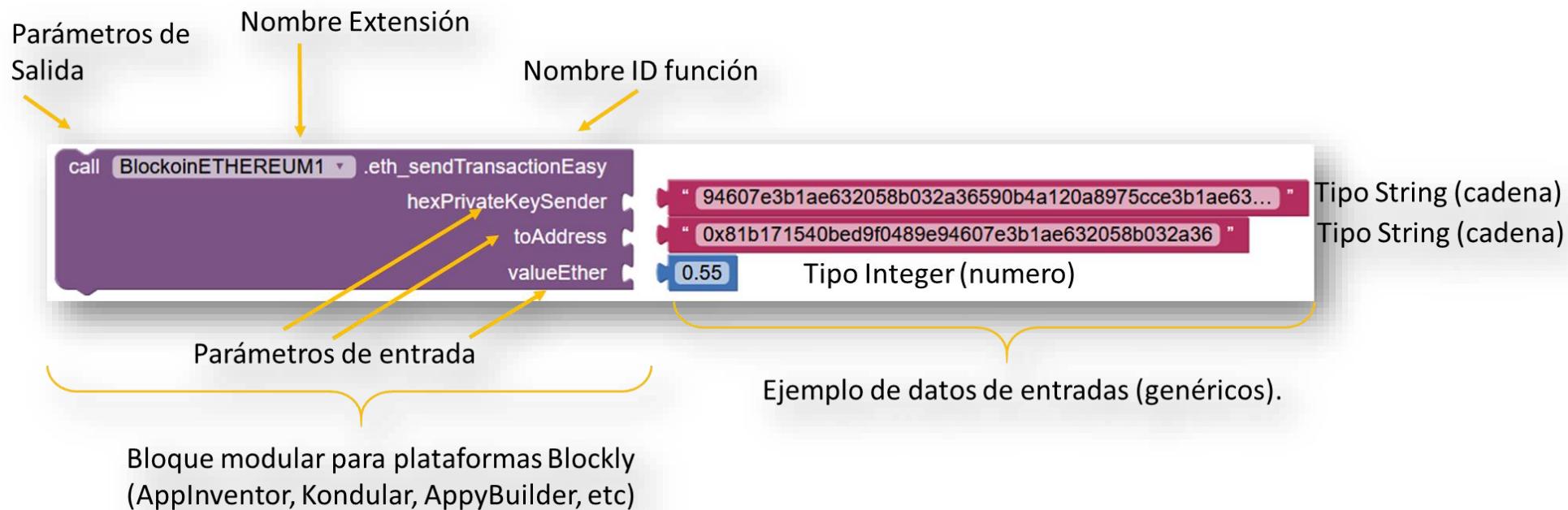


OPMERKING: De extensies zijn getest op AppInventor, Kondular, Thunkable en AppyBuilder systemen.

7. Definitie en gebruik van blokken (generieke functie)

We beginnen met het uitleggen van de verdeling van de gegevens die alle blokken zullen hebben, hun syntaxis van gebruik en configuratie.

In het volgende voorbeeld zien we een modulair blok met zijn invoer- en uitvoerparameters en de soorten invoergegevens, deze gegevens kunnen van het type String (tekenreeks) of Integer (geheel getal of decimaal) zijn. We laten zien hoe het wordt gebruikt en configureren het voor de goede werking ervan.



Elk moduleblok zal zijn beschrijving hebben en zal worden benoemd in het geval dat het een verplichte of optionele afhankelijkheid heeft van andere blokken die worden gebruikt als invoerparameters, het integratieproces zal worden aangekondigd.

8. Exchange Ethereum Extension (EEE) functies en gebeurtenissen.

***Testing netwerk dat we **Rinkeby** zullen gebruiken, als urlNetwork, wanneer u echte transacties wilt maken hoeft u alleen het urlNetwork netwerk te veranderen voor het **mainnet**.

Blokkeren om de internetverbinding te controleren - (**CheckInternetConnection**).

call **BlockoinETHEREUM1** .**CheckInternetConnection**

Invoerparameters: Niet van toepassing.

Uitgangsparameters: Geeft "Waar" als u verbinding heeft of geeft "Vals" terug als er geen verbinding is.

Beschrijving: Blokkeren om de internetverbinding te controleren en gegevens (transacties) te verzenden.

Blokkeren om een **nieuw** "Offline" adres te genereren - (**GenereerNewAddressEthereum**)

call **BlockoinETHEREUM1** .**GenerateNewAddressEthereum**

phraseHex

"exchange ethereum extension for systems blockly"

Invoerparameters: **phraseHex <String>**.

Uitgangsparameters: Gebeurtenis (**OutputGenerateNewAddressEthereum**)

Uitgangen: **PrivateKey<String>**, **PublicKey<String>** , **adresEthereum<String>**.

when **BlockoinETHEREUM1** .**OutputGenerateNewAddressEthereum**

privKeyEther **pubKeyEther** **addressEthereum**

do

Omschrijving: Maak een nieuw etherisch adres (account) aan op basis van een zin of een reeks getallen. Het nieuwe adres kan worden aangemaakt zonder netwerk- of internetverbinding - "Offline".

Blokkeren om een nieuw "Online" adres te genereren - (**GenereerNewAddressEthereum**)

call **BlockoinETHEREUM1** .GenerateNewAddressEthereumAPI

Invoerparameters: Niet van toepassing.

Uitgangsparameters: Terugkeer in JSON dataformaat; privateKey, publicKey, adres.

Voorbeeld van een uitvoer:

```
{  
  "private": "9ab4b2fba728a55643414f26adc04eea080740860cbe39b13fe4acb43dbb9f83",  
  "public":  
    "0421d559aa98d6fc77688ae9f19b3dc502c05f0b7f70bbe924cb712d6243a489695b1c1ee03993b3b6d97277  
    97e780677a5469800b4d98374bdb910ed99fa2b5c8",  
  "address": "92a2f157d5aec3fa79f92995fea148616d82c5ef"  
}
```

Omschrijving: Maak een nieuw etherische adres (account) aan. Het is noodzakelijk om toegang tot of verbinding met het internet te hebben, aangezien de generatie via de REST API-service - "Online" - verloopt.

Blokkeren om een nieuw "Offline" adres te genereren en de publieke en private sleutels op te slaan in binaire bestanden - (**GenereerNewAddressEthereumStoreKeys**)

call **BlockoinETHEREUM1** .GenerateNewAddressEthereumStoreKeys
pathFilePrivateKey **" /mnt/sdcard/privateKey.bin "**
pathFilePublicKey **" /mnt/sdcard/publicKey.bin "**

Invoerparameters: **pathFilePrivateKey<String>** , **pathFilePublicKey<String>**.

Uitgangsparameters: Gebeurtenis (**OutputGenerateNewAddressEthereumStoreKeys**)

Uitvoer: **adresEthereum<String>** , **privateKeyECC<String>** , **publicKeyECC<String>** ,
privateKeyHex<String> , **publicKeyHex<String>**.

when **BlockoinETHEREUM1** .OutputGenerateNewAddressEthereumStoreKeys
 addressEthereum **privateKeyECC** **publicKeyECC** **privatekeyHex** **publicKeyHex**
do **[empty]**

Omschrijving: Creëert een nieuw willekeurig etherisch adres (account) en slaat de publieke en private sleutels op in binaire bestanden die gebruikt zullen worden voor het importeren en exporteren van accountgegevens. Het nieuwe adres kan worden aangemaakt zonder netwerk- of internetverbinding - "Offline".

Blok om publieke sleutel te genereren - (`GeneratePublicKeyHexFromPrivateKeyHex`).

```
call BlockoinETHEREUM1 .GeneratePublicKeyHexFromPrivateKeyHex
    hexPrivateKey "429a043ea6333b358d3542ff2aab9338b9c0ed928e35ec0a..."
```

Invoerparameters: `hexPrivateKey <String>`.

Uitgangsparameters: Gebeurtenis (`OutputGeneratePublicKeyHexFromPrivateKeyHex`)

Uitgangen: `adres<String>`, `publicKeyHex<String>`.

```
when BlockoinETHEREUM1 .OutputGetAddressEthereumFromPrivateKey
    address publicKeyHex
do
```

Omschrijving: Creëert de openbare sleutel op basis van de invoer van een privésleutel.

Blokkeren om het saldo van een adres **te krijgen** - (`GetBalanceAddrEthereum`).

```
call BlockoinETHEREUM1 .GetBalanceAddrEthereum
    addressEthereum "0x92a2f157d5aec3fa79f92995fea148616d82c5ef"
```

Invoerparameters: `hexPrivateKey <String>`.

Uitgangsparameters: Gebeurtenis (`OutputGeneratePublicKeyHexFromPrivateKeyHex`)

Uitgangen: `balans<String>`, `totaal_ontvangen<String>`, `totaal_verzonden<String>`, `onbevestigd_evenwicht <String>`, `definitief_evenwicht<String>`, `n_tx<String>`, `onbevestigd_n_tx<String>`, `definitief_n_tx<String>`.

```
when BlockoinETHEREUM1 .OutputDataBalanceAddrEthereumWEI
    balance total_received total_sent unconfirmed_balance final_balance n_tx unconfirmed_n_tx final_n_tx
do
```

Omschrijving: Geeft de balans en gedetailleerde rekening (adres) gegevens weer.

Blokkeren om te controleren of de netwerkinterface van uw mobiel is geactiveerd - (GetDataNetworkConnection).

call BlockinETHEREUM1 .GetDataNetworkConnection

Invoerparameters: Niet van toepassing.

Uitgangsparameters: Gebeurtenis (OutputGeneratePublicKeyHexFromPrivateKeyHex)

Uitgangen: **interfasienaam<String>**, is aangesloten<String>.

```
when BlockinETHEREUM1 .OutputGetDataNetworkConnection
    interfacename isconnected
do
```

Omschrijving: Geeft de naam van de mobiele interface weer en geeft aan of de interface is geactiveerd of niet.

Blok om transactie "Offline" te ondertekenen - (SignerGenericPushRawTransactionOffline)

call BlockinETHEREUM1 .SignerGenericPushRawTransactionOffline

urlNetwork	" [https://rinkeby.infura.io/v3/] 440789c3... "
hexPrivateKeySender	" 9eC478ee49824890b4a120a8975cce3b1ae632058b0301f8... "
nonceNumber	get global nonce
gasPrice	" 250000000000 "
gasLimit	21000
toAddress	" 0x92a2f157d5aec3fa79f92995fea148616d82c5ef "
valueWei	" 10000000000000000000 " × " 0.01 "

Vereiste afhankelijkheid(en): Blok (eth_getTransactionCount), Blok (eth_SendRawTransactionInfura)

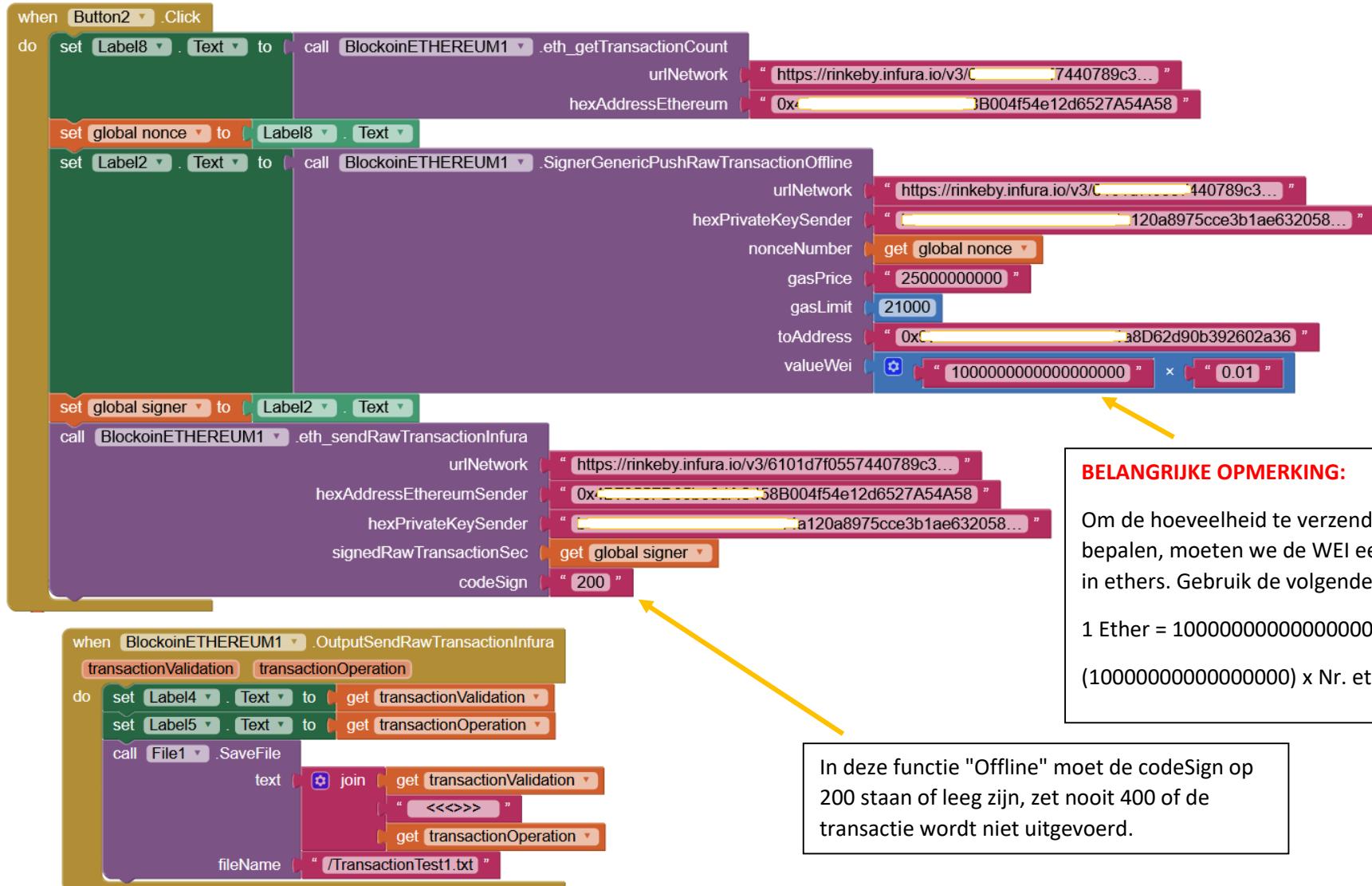
Parámetros de entrada: urlNetwork <String>, hexPrivateKeySender <String>, nonceNumber <String>, gasPrice <String>, gasLimit <Integer>, toAddress <String>, waardeWEI <Integer>.

Uitgangsparameters:

Uitvoer: Ondertekende transactie die moet worden verzonden. <String>.

Beschrijving: Bereidt een nieuwe transactie voor die moet worden verzonden (gecodeerd en ondertekend). Dit kan worden verwerkt zonder netwerk- of internetverbinding - "Offline".

Voorbeeld van volledig gebruik met blokafhankelijkheid (`SignerGenericPushRawTransactionOffline`).



BELANGRIJKE OPMERKING:

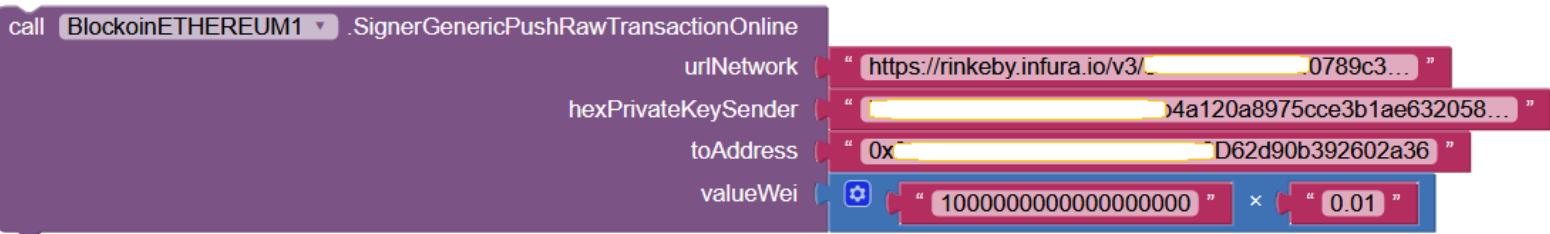
Om de hoeveelheid te verzenden Ether te bepalen, moeten we de WEI eerst omzetten in eters. Gebruik de volgende conversie:

$$1 \text{ Ether} = 1000000000000000000 \text{ WEI} = 10^{18}$$

$$(1000000000000000000) \times \text{Nr. ether}$$

In deze functie "Offline" moet de codeSign op 200 staan of leeg zijn, zet nooit 400 of de transactie wordt niet uitgevoerd.

Blok om "Online" transactie te ondertekenen - (**SignerGenericPushRawTransactionOnline**).

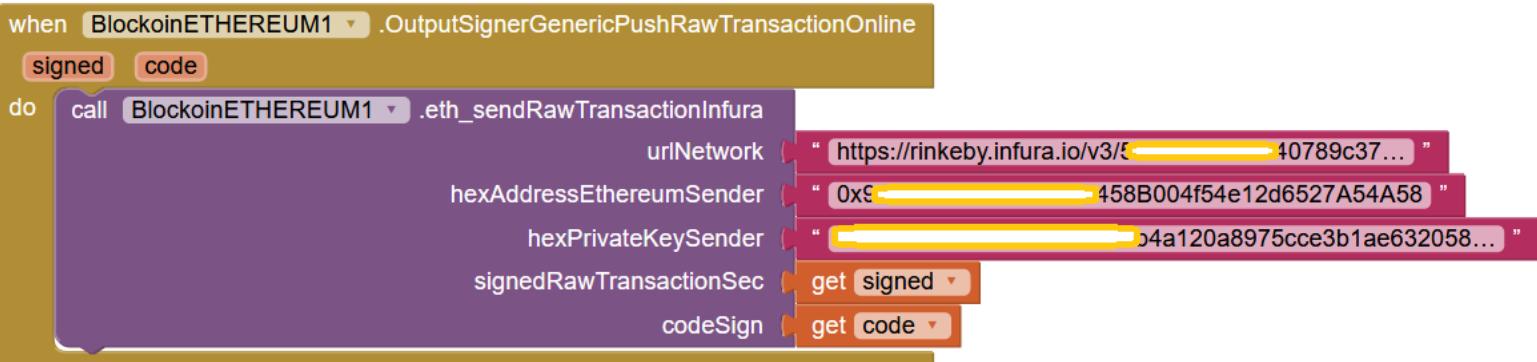


Verplichte eenheid (eenheden): Blok (**eth_SendRawTransactionInfura**).

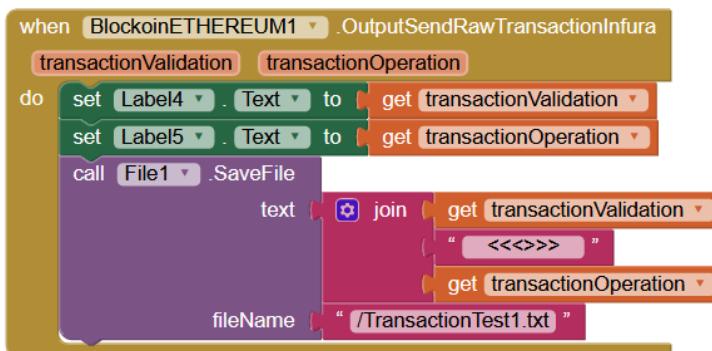
Invoerparameters: **urlNetwork <String>**, **hexPrivateKeySender <String>**, **toAddress <String>**, **waardeWEI <Integer>**.

Uitgangsparameters: Gebeurtenissen in de volgende volgorde (**OutputSignerGenericPushRawTransactionOnline**) en (**OutputSendRawTransactionInfura**).

Uitvoer: **ondertekend<String>**, **code<String>**.

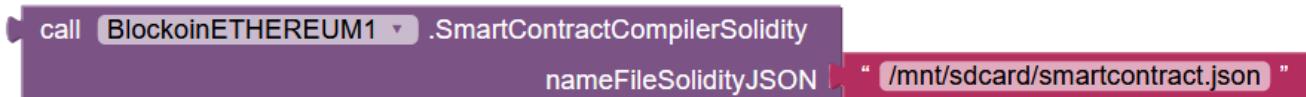


Uitgangen blok **eth_sendRawTransactionInfura**: **transactionValidation<String>**, **transactionOperation<String>**.



Beschrijving: Bereidt een nieuwe transactie voor die moet worden verzonden (gecodeerd en ondertekend). Een netwerk- of internetverbinding is vereist - "Online".

Blok voor het opstellen van SmartContractCompilerSolidity (SmartContractCompilerSolidity).



Invoerparameters: **nameFileSolidityJSON <String>**.

Output parameters: Toont het samengestelde smart contract, deze functie helpt ons om te controleren of het goed geschreven is alvorens een smart contract in het Ethereum netwerk te publiceren.

Uitgangen: **Gecompileerde code**.

Het Smart contract moet in een JSON formaat bestand zijn.

Voorbeeld van een basis Smart contract in Solidity taal.

```
pragma soliditeit ^0,5,0;

dodelijk contract {
adreseigenaar;
functie mortal() { eigenaar = msg.afzender; }
functie kill() { als (msg.afzender == eigenaar) zelfmoord(eigenaar); }
contractgroeter is dodelijk {
Seriegroet;
functie groet (string _groet) openbaar { groet = _groet; }
functie groet() constante retournen (string) {retourgroet;}
```

Voorbeeld van een eerder Smart contract in JSON-formaat met commentaar.

```
# Controleer de soliditeitscompilatie via een niet-gepubliceerde test
# Met behulp van "groeter" contract soliditeit voorbeeld, de "hallo
wereld" van Ethereum.
```

Bestand: smartcontract.json

```
{
"soliditeit": "contract sterfelijk": definieer de variabele eigenaar van
het type adres ; deze functie wordt uitgevoerd bij de initialisatie en
stelt de eigenaar van het contract in */n     functie sterfelijk() {
eigenaar = msg.afzender; }n      /* Functie om het geld van het
contract te recupereren */n     functie doden() { als (msg.afzender ==
eigenaar) zelfmoord; {\n}contractgroeter is sterfelijk {\n      /* definieer
variabele begroeting van het type tekenreeks */{\n        tekenreeks
begroeting};\n        dit loopt wanneer het contract wordt uitgevoerd */n
```

```

functie groet (string _greeting) publiek {\n            begroeting =\n_groet;\n            /* hoofdfunctie */\n            functie groet() constante\nretouren (string) {\n            retouren groet;\n}      ",\n"params." "Hallo BlockCypher Test.\n}

```

BELANGRIJKE OPMERKING: Het JSON-formaat moet altijd een regeleinde hebben aan het einde van elke regel.

Voorbeeld van samengestelde Smartcontract output.

```

[
{
    "naam": "u003cstdin\u003e:greeter",
    "soliditeit": "contract sterfelijk": definieer de variabele eigenaar
    van het type adres ; deze functie wordt uitgevoerd bij de
    initialisatie en stelt de eigenaar van het contract in /*/n      functie
    sterfelijk() { eigenaar = msg.afzender; }n          /* Functie om het geld
    van het contract te recupereren */n      functie doden() { als
    (msg.afzender == eigenaar) zelfmoord; {\n}contractgroet is sterfelijk {\n
    /* definieer variabele begroeting van het type tekenreeks */{\n
    tekenreeks begroeting};\ndit loopt wanneer het contract wordt uitgevoerd
    */n      functie groet (string _greeting) publiek {\n            begroeting =\n_groet;\n            /* hoofdfunctie */\n            functie groet() constante\nretouren (string) {\n            retouren groet;\n}      ",
    "Bin":
"606060405260405161023e38038061023e8339810160405280510160008054600160a060
020a031916331790558060016000509080519060200190828054600181600116156101000
203166002900490600052602060002090601f016020900481019282601f10609f57805160
ff19168380011785555b50608e9291505b8082111560cc57600081558301607d565b50505
061016e806100d06000396000f35b828001600101855582156076579182015b8281111560
7657825182600050559160200191906001019060b0565b509056606060405260e060020a6
00035046341c0e1b58114610026578063cfaf321714610068575b005b6100246000543373
ffffffffffffffffff908116911614156101375760005473fff
ffffffffffff908116911614156101375760005473fff
a06020601f600260001961010086881615020190941693909304928301819004028101604
0526080828152929190828280156101645780601f10610139576101008083540402835291
60200191610164565b6040518080602001828103825283818151815260200191508051906
0200190808383829060006004602084601f0104600f02600301f150905090810190601f16
80156101295780820380516001836020036101000a031916815260200191505b509250505
06
    "abi": [
        {
            "constant": vals,
            "inputs": [],
            "naam": "doden",
            "uitgangen": [],
            "type": "functie"
        },
        {
            "constant": waar,
            "inputs": [],
            "naam": "groet",
            "outputs": [

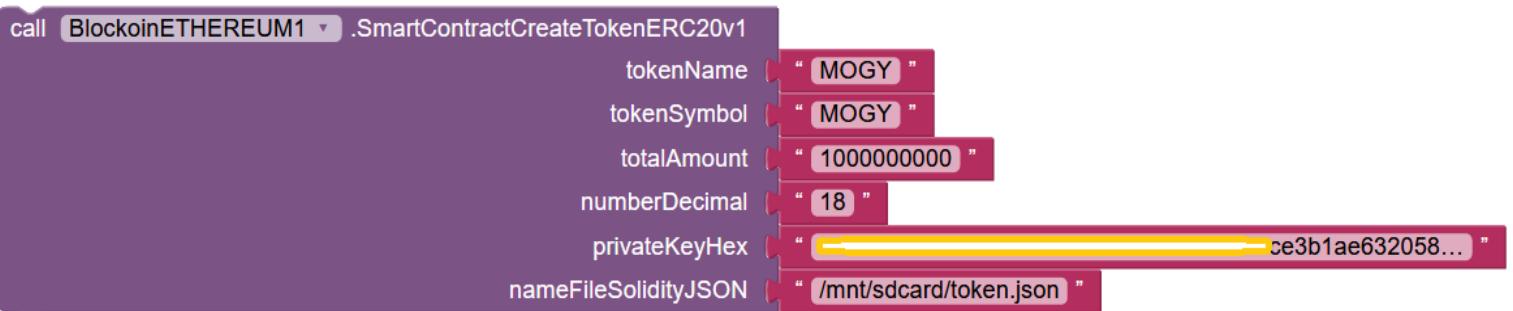
```

```

{
  "naam": "",
  "type": "string"
},
],
"type": "functie"
},
{
  "ingangen": [
    {
      "naam": "_groet",
      "type": "string"
    }
  ],
  "type": "bouwer"
}
],
"params": [
  "Hallo BlockCypher Test"
]
},
{
  "naam": "sterfelijk",
  "soliditeit": "contract sterfelijk": definieer de variabele eigenaar
van het type adres ; deze functie wordt uitgevoerd bij de
initialisatie en stelt de eigenaar van het contract in */n   functie
sterfelijk() { eigenaar = msg.afzender; }n           /* Functie om het geld
van het contract te recupereren */n   functie doden() { als
(msg.afzender == eigenaar) zelfmoord; {\n}contractgroet is sterfelijk {\n
/* definieer variabele begroeting van het type tekenreeks */{\n
tekenreeks begroeting};\ldit loopt wanneer het contract wordt uitgevoerd
*/n   functie groet (string _greeting) publiek {\n           begroeting =
_groet;\n           /* hoofdfunctie */n   functie groet() constante
retouren (string) {\n           retouren groet;\n           ",
"Bin":
"606060405260008054600160a060020a03191633179055605c8060226000396000f36060
60405260e060020a600035046341c0e1b58114601a575b005b60186000543373ffffffffffff
fffffffffffffffffffffffffffff90811691161415605a5760005473fffffffffffff
fffffffffffffffffffff16ff5b56",
"abi": [
  {
    "constant": vals,
    "inputs": [],
    "naam": "doden",
    "uitgangen": [],
    "type": "functie"
  },
  {
    "inputs": [],
    "type": "bouwer"
  }
],
"params": [
  "Hallo BlockCypher Test"
]
}
]
}

```

Blokkeren om ERC20 Token te compileren, te creëren en te publiceren -

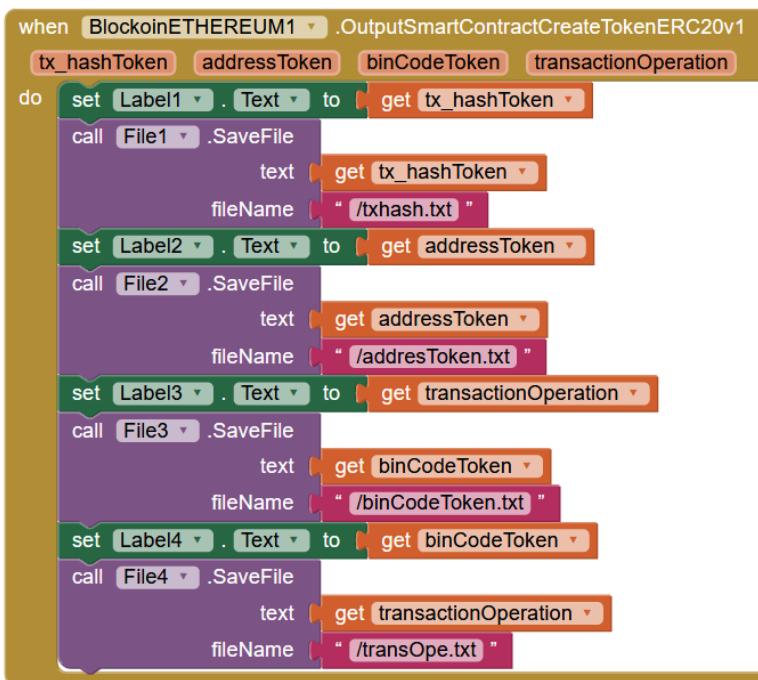


Verplichte eenheid (eenheden): Blok (**CreateTestingFie**). **BELANGRIJK**: Eerst moet u dit blok gebruiken om er zeker van te zijn dat het pad correct is, dit is omdat als u geen geldig pad opgeeft in het blok (**SmartContractCreateTokenERC20v1**) de tokenaanmaak niet zal worden uitgevoerd omdat de invoerparameter "nameFileSolidityJSON" wordt gebruikt om een tijdelijk bestand aan te maken.

Invoerparameters: **tokenName <String>**, **tokenSymbol <String>**, **totalAmount <String>**, **numberDecimal <String>**, **privateKeyHex <Integer>**, **nameFileSolidityJSON <String>**. Dit **bestand is het** geldige pad om **een tijdelijk** bestand **aan te maken**, je moet er zeker van zijn dat het pad geldig is om te testen of het bestand is aangemaakt je kunt het Block (**CreateTestingFile**) gebruiken na het gebruik ervan controleren of het succesvol is aangemaakt.

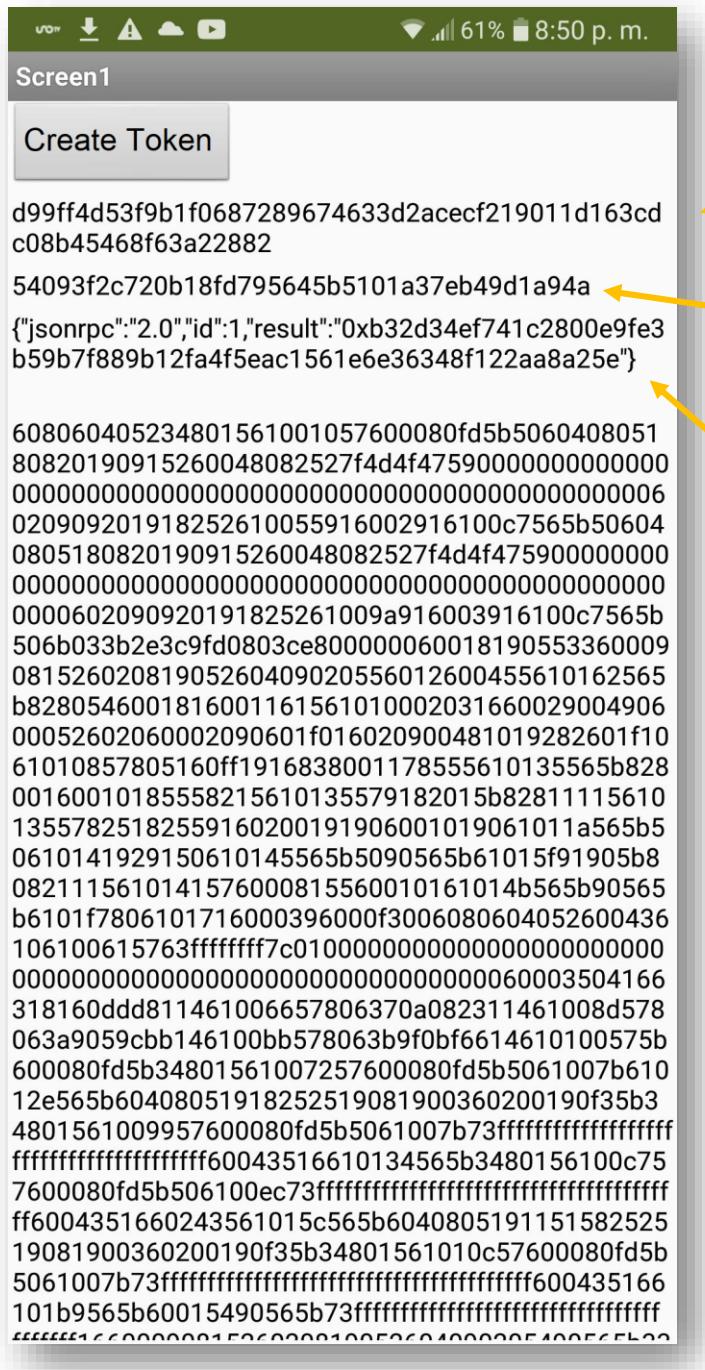
Uitgangsparameters: Gebeurtenis (**OutputSmartContractTokenERC20v1**).

Uitgangen: **tx_hashToken<String>** , **addressToken<String>** , **binCodeToken<String>** , **trasactionOperatie<String>**.



Omschrijving: Slim contract "Token ERC20" - actief gepubliceerd op het Ethereum netwerk.
In versie 1 (v1) is de gaslimietparameter al ingesteld op 500.000 WEI.

Voorbeeld van de uitvoer van de vorige functie **SmartContractCreateTokenERC20v1**.



Transactie (Tx_hash) van de oprichting van het Ethereum-netwerk. Deze kan worden geraadpleegd op:

Adres van het nieuwe contract dat verwijst naar het nieuwe ERC20-muntstuk dat is gecreëerd.

Transactie (Tx_hash) van de gevalideerde operatie in het CoinSolidation.org systeem

Deze is te vinden op:
www.etherscan.io

Binaire code (BIN) van het nieuwe tokencontract dat in de EVM (Ethereum Virtual Machine) werd verwerkt.

9. Stappen om een CryptoToken of Cryptomoney Token te maken.

Stap 1.

Controleer of het tijdelijke pad op het mobiele apparaat waar het Smart contract is aangemaakt, geldig is en of een bestand met succes kan worden aangemaakt. Dit wordt gedaan met behulp van de Block (**CreateTestingFile**). Controleer of het testbestand in de invoer "pathTestFile" is aangemaakt, in het algemeen wordt het pad gegeven door: **/mnt/sdcard/name_file.txt**

Stap 2 (optioneel).

In deze stap controleren we of de rekening (adres) waarvan de transactie wordt aangerekend voldoende saldo heeft om de transactie van het aanmaken en publiceren van een Smartcontract uit te voeren. Dit kan worden geverifieerd met behulp van het blok (**eth_VerifiBalanceForTransaccionSmartContract**). Dit gebruik van dit blok is optioneel, aangezien de blokken die de **SmartContractCreateTokenERC20v1** of **SmartContractCreateTokenERC20v2** genereren, deze verificatie al intern bevatten.

Stap 3.

Selecteer welk blok u wilt gebruiken om het ERC20-token aan te maken, u heeft twee opties:

a.- Blok **SmartContractCreateTokenERC20v1** heeft al de impliciete waarde van Gas Limit toegewezen gekregen met een waarde van 500.000 WEi.

b.- Blok **SmartContractCreateTokenERC20v2** heeft de mogelijkheid om de Gas Limiet te configureren naar de behoefte van de eindgebruiker of ontwikkelaar. Opgemerkt moet worden dat als een zeer lage gasgrens van minder dan 350.000 Wei wordt gegeven, het zeer goed mogelijk is dat de Smart contr.

Stap vier.

Gebruik ofwel **SmartContractCreateTokenERC20v1** of **SmartContractCreateTokenERC20v2-blokken** om ervoor te zorgen dat bij gebruik van de invoervariabele "nameFileSolidityJSON" deze gelijk is aan de ingangsvariabele "pathTestFile" van het blok (**CreateTestingFile**), die al in stap 1 is gecontroleerd.

Stap vijf.

Voordat u een ERC20-token aanmaakt met een van de **SmartContractCreateTokenERC20v1-** of **SmartContractCreateTokenERC20v2-blokken**, wordt aanbevolen om de Eventwaarden (resultaten) op te slaan om de resultaten op te slaan (tx_hashToken, addressToken, binCodeToken, transactionOperation). Zie voorbeeld van de uitvoer van de vorige functie **OutPutSmartContractCreateTokenERC20v1**.

Stap zes.

Voer de creatie van de ERC20-munt uit en publiceer deze vervolgens voor de verkoop. Zie paragraaf 10.

10. Hoe zet je een nieuw goed of je cryptiemunt te koop (Token ERC20).

Aangezien we een ERC20 - Cryptomoney Token (Zie **SmartContractCreateTokenERC20v1 Block** of met het **SmartContractCreateTokenERC20v2 Block**) hebben gemaakt, moeten we het uploaden naar een bepaalde Exchange zodat iedereen in de wereld het kan kopen. Een Exchange is een site op het internet waar nieuwe muntjes worden uitgegeven.

De uitwisselingen worden in twee soorten onderverdeeld, namelijk gecentraliseerd en gedecentraliseerd. Het belangrijkste verschil is dat men wordt bestuurd en gecontroleerd door een of ander internationaal orgaan (gecentraliseerd) en de gedecentraliseerde organen hebben niemand bij de controleurs. Hoewel dit misschien meer vertrouwen geeft, is de realiteit dat de gedecentraliseerde systemen de laatste tijd meer kracht hebben gekregen en meestal zonder grote problemen worden gebruikt.

Een van de beste praktijken bij de behandeling van activa van welke aard dan ook is dat ze niet allemaal op één rekening staan, maar dat ze in meerdere rekeningen worden verdeeld voor de beveiliging van zowel particuliere als openbare sleutels.

In ons geval zullen we gebruik maken van een Decentrale Uitwisseling, maar met een niet slechte geschiedenis, zullen we gebruik maken van www.forkdelta.app.

Op dit moment moeten we de applicatie voor de browser (Mozilla of Chrome) **METAMASK** www.metamask.io al geïnstalleerd hebben, dit is omdat de Exchange om uw pagina www.forkdelta.app te bezoeken verbinding moet maken met de account die we hebben Ethereum.

Een belangrijk punt is dat onze Ethereum-account die we al in METAMASK hebben een saldo van min of meer 10 USD moet hebben, want als we onze nieuwe ERC20 Token publiceren die we met het **SmartContractCreateTokenERC20v1 Block** of met het **SmartContractCreateTokenERC20v2 Block** hebben gecreëerd, zullen we de transactie moeten betalen om het op de beurs te publiceren.

Om gebruik te kunnen maken van de Exchange www.forkdelta.app moeten we de volgende gegevens hebben die we willen publiceren voor de verkoop op de Exchange.

Adres van de nieuwe ERC20-token die we eerder hebben gemaakt.

0x54093F2C720b18Fd795645b5101A37EB49d1A94a

Aantal decimalen die onze touch gebruikt.

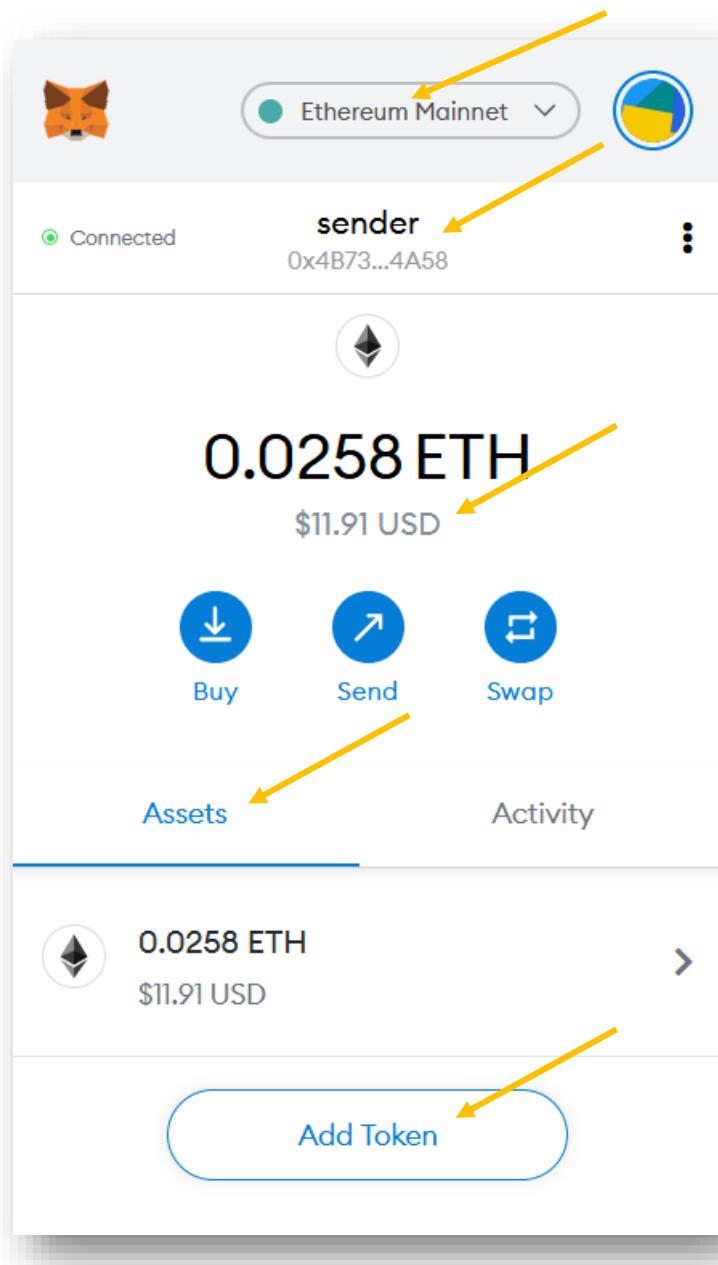
18

De naam die de penning identificeert.

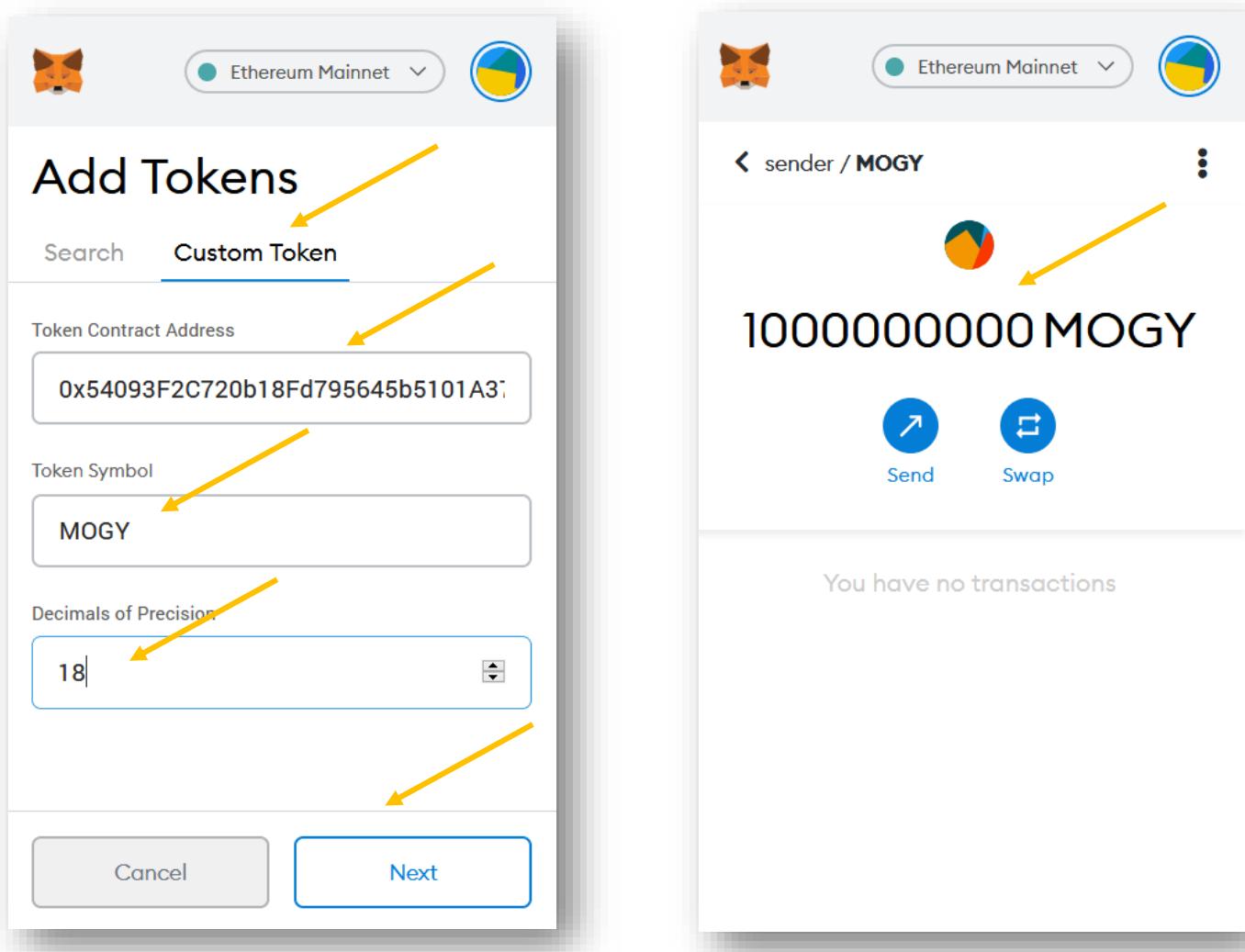
MOGY

Laten we beginnen met te controleren of de token die we hebben gemaakt de hierboven genoemde parameters heeft en dat het de parameters waren waarmee het in eerste instantie is gemaakt.

Laten we naar **METAMASK** gaan en er eerst voor zorgen dat we op de rekening staan waarmee we de penning hebben aangemaakt. Ga dan naar beneden en klik op de knop "Token toevoegen".

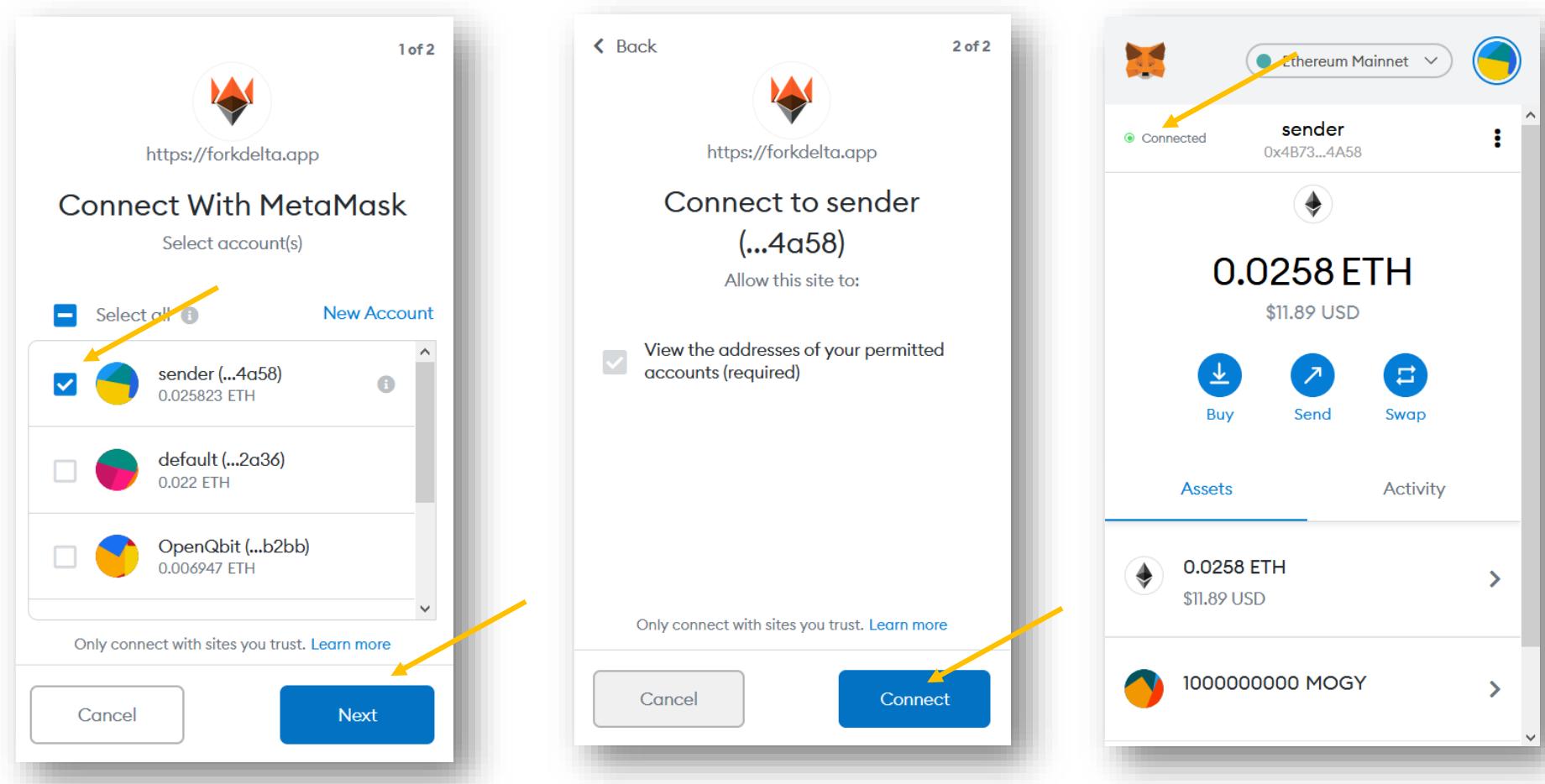


We zullen nu onze nieuwe penning toevoegen om uw huidige balans te zien. Nadat u op de knop "Custom token" bovenaan de pagina hebt geklikt, voert u de gevraagde informatie in de volgende velden in en klikt u op de knop "Next". Daarna verschijnt onze nieuwe penning met het saldo dat u heeft. Als je geen saldo hebt, controleer dan of je op de rekening staat waarmee je het speelfiguur hebt aangemaakt. Als je niet op een rekening staat waarmee je het speelfiguur hebt aangemaakt, heb je een nulsaldo omdat je nog geen speelfiguur hebt gekocht.



Zodra we onze nieuwe token te koop aanbieden op de beurs [www.forkdelta.app](https://forkdelta.app), zullen we deze uploaden.

Wanneer je op de Exchange site bent, wordt je gevraagd om verbinding te maken met de site <https://forkdelta.app>. Klik op de "Next" knop hieronder, dan "Connect" en tot slot kun je controleren of je bent verbonden met de site van forkdelta.app.



Het is tijd om de nieuwe token uit te geven op de Exchange <https://forkdelta.app>.

Klik op het bovenste menu "DAI" en ga naar het einde van de scroll en kies de optie "Overig".

The screenshot shows the ForkDelta exchange interface. At the top left, there is a balance summary for DAI and ETH. The main area features a 'Order Book' and a 'Price Chart'. On the left, a sidebar lists various tokens like ZAP, ZCG, ZDR, ZIL, ZIP, ZRX, ZSC, ZXBT, cV, eGO, and ePRX. A prominent yellow arrow points from the text above to the 'DAI' dropdown menu in the top navigation bar. Another yellow arrow points to the 'Other' option in the dropdown menu. The 'Order Book' displays several orders for DAI/ETH, DAI, and ETH. The 'Price Chart' shows price levels from 0.00 to 1.00 over a 24-hour period. To the right, a 'Trades' section lists recent trade volumes for DAI/ETH, DAI, and ETH. Below the chart, a 'My Transactions' section includes tabs for 'Important', 'Trades', 'Orders', and 'Funds'. A 'chat on Telegram' button is also present. At the bottom, there's a 'URL & Status' section with a bookmark link and a 'How to Start Trading' guide. A 'Tweets' section on the right shows a tweet from @ForkDelta.

Dan registreren we de nieuwe penning met de gegevens die we al kennen.

The screenshot shows the ForkDelta app interface. A modal window titled "Other token" is open, prompting for token details. The "Address" field contains the value `0x54093F2C720b18Fd795645b5101A37EB49d1A94a`. The "Name" field contains `MOGY`. The "Decimals" field contains the value `18`. The background features a dark-themed dashboard with sections for "Balance", "Order Book", "Trades", "Volume", and "New Order". The "Trades" section displays a table of recent trades for DAI/ETH, DAI, and ETH. The "New Order" section allows users to place buy or sell orders for DAI, DAI/ETH, or ETH. The "My Transactions" section shows a list of recent activity, including a bookmark to the project's URL and a note about the improved order processing system.

Op dit moment zal de nieuwe token in de linkerbovenhoek van de Exchange verschijnen, we hebben het alleen geupload naar de Exchange, we moeten het publiceren zodat iedereen het kan kopen en zien. Nu moeten we nog wat Ether (0,015) storten van onze rekening naar de Exchange.

The image consists of two side-by-side screenshots of the ForkDelta website, both titled "ForkDelta MOGY".

Left Screenshot (MOGY Deposit):

- Balance:** Shows a balance of 1000000000.000 MOGY in the Wallet column and 0.000 in the ForkDelta column.
- Deposit Form:** An "Amount" input field contains "MOGY" and a "Deposit" button.
- Volume:** A search bar and a list of tokens (ASTRO, REQ, SXDT, SNOV) with their current prices.
- Search Bar:** "Escribe aquí para buscar" (Type here to search).

Right Screenshot (ETH Deposit):

- Balance:** Shows a balance of 0.026 ETH in the Wallet column and 0.000 in the ForkDelta column.
- Deposit Form:** An "Amount" input field contains "0.015" and a "Deposit" button. A tooltip above the input field says: "Use this to deposit from your personal Ethereum wallet ("Wallet" column) to the current smart contract ("ForkDelta" column)."
- Volume:** A search bar and a list of tokens (ASTRO, REQ, SXDT, SNOV) with their current prices.
- Search Bar:** "Escribe aquí para buscar" (Type here to search).

Sinds we de storting hebben gedaan kunnen we zoveel muntpjes storten als we willen op de Exchange www.forkdelta.app.

Om een bepaald bedrag aan muntjes te storten moeten we een inkooporder aanmaken, dit gebeurt onderaan waar "Nieuwe bestelling" staat en we klikken op de optie "Verkopen". Vervolgens voeren we de hoeveelheid penningen in die we beschikbaar willen stellen aan elke koper in de wereld, de prijs in Ether die we elk penningen willen verkopen (eenheidsprijs) en de parameter "Vervalt" is de hoeveelheid tijd die we willen hebben om deze penningen te koop te hebben:

Verlooptijd = 14 seconden X ingevoerde hoeveelheid.

Voorbeeld: 14 seconden X 10000 = 140.000 seconden = 1,62 dagen

The screenshot shows the ForkDelta interface for trading MOGY tokens. On the left, there's a 'Balance' section with tabs for Deposit, Withdraw, and Transfer. Under the Token tab, it shows 1000000000.000 MOGY in the Wallet and 0.000115 ETH in the ForkDelta account. Below this is a note: 'Make sure MOGY is the token you actually want to trade. Multiple tokens can share the same name.' On the right, the 'Order Book' is empty with the message 'There are no orders here.' At the bottom, the 'New Order' form is displayed. It has tabs for Buy and Sell, with 'Sell' selected. The 'Sell' tab contains fields for MOGY (amount: 10000), MOGY/ETH (price: 0.05), and ETH (amount: 500.000). The 'Expires' field is set to 10000. A large orange 'Sell' button is at the bottom of the form. Five yellow arrows point from the text labels 'Sell', 'MOGY', 'MOGY/ETH', 'ETH', and 'Expires' to their respective input fields in the 'Sell' tab of the 'New Order' form.

Daar zijn je nieuwe lopers te koop, iedereen kan binnenkomen en ze kopen. Soms herkent het de nieuwe penning niet, dus moeten ze worden verkocht vanuit de Metamask-applicatie.

Voorbeeld van raadpleging op de site www.etherscan.io van de nieuw gecreëerde penning.

The screenshot shows the Etherscan interface for a transaction. The transaction hash is 0xd99ff4d53f9b1f0687289674633d2acecf219011d163cdc08b45468f63a22882. The status is Success. The block number is 11240201 with 224 block confirmations. The timestamp is 47 mins ago (Nov-12-2020 02:49:56 AM +UTC) | Confirmed within 30 secs. The transaction originated from address 0x4b7355fd05be6dac458b004f54e12d6527a54a58 and went to a contract at address [Contract 0x54093f2c720b18fd795645b5101a37eb49d1a94a Created]. The value was 0 Ether (\$0.00). The transaction fee was 0.011014691 Ether (\$5.06). The gas price was 0.000000041 Ether (41 Gwei) and the gas limit was 500,000. The gas used by the transaction was 268,651 (53.73%). A yellow arrow points from the 'To' field to a callout box containing the text: 'Contractadres nieuw aangemaakt token.' Another yellow arrow points from the 'Gas Price' field to a callout box containing the text: 'Alleen de kosten van het etherische netwerk.'

Etherscan

All Filters Search by Address / Txn Hash / Block / Token / Ens

Eth: \$459.17 (-0.93%) | 24 Gwei

Home Blockchain Tokens Resources More Sign In

Transaction Details

Sponsored: Crypto.com - The Crypto Super App - Buy Crypto at True Cost with 0% fee on credit card purchase. [Get App Now.](#)

Overview State Comments

② Transaction Hash: 0xd99ff4d53f9b1f0687289674633d2acecf219011d163cdc08b45468f63a22882

② Status: Success

② Block: 11240201 224 Block Confirmations

② Timestamp: 47 mins ago (Nov-12-2020 02:49:56 AM +UTC) | Confirmed within 30 secs

② From: 0x4b7355fd05be6dac458b004f54e12d6527a54a58

② To: [Contract 0x54093f2c720b18fd795645b5101a37eb49d1a94a Created] ✓

② Value: 0 Ether (\$0.00)

② Transaction Fee: 0.011014691 Ether (\$5.06)

② Gas Price: 0.000000041 Ether (41 Gwei)

② Gas Limit: 500,000

② Gas Used by Transaction: 268,651 (53.73%)

Contractadres nieuw aangemaakt token.

Alleen de kosten van het etherische netwerk.

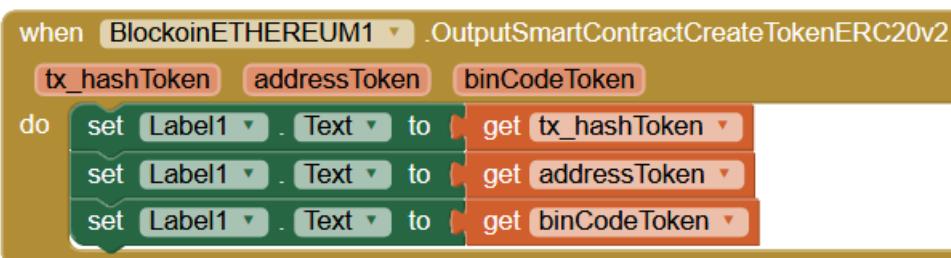
Blokkeren om ERC20 Token te compileren, te creëren en te publiceren -
(SmartContractCreateTokenERC20v2)



Parámetros de entrada: **tokenName <String>**, **tokenSymbol <String>**, **totalAmount <String>**, **numberDecimal <String>**, **gas_limit <Integer>**, **privateKeyHex <Integer>**, **nameFileSolidityJSON <String>**.

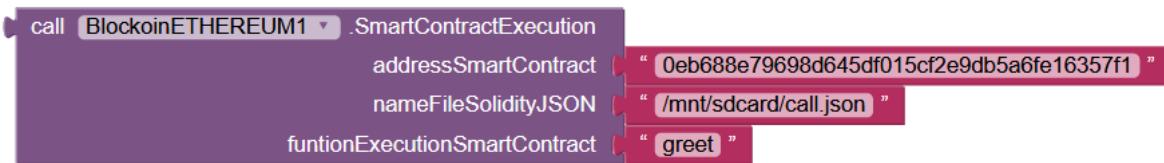
Uitgangsparameters: Gebeurtenis (**OutputSmartContractCreateTokenERC20v2**).

Uitgangen: **tx_hashToken<String>** , **addressToken<String>** , **binCodeToken<String>**.



Omschrijving: Slim contract "Token ERC20" - actief gepubliceerd op het Ethereum netwerk. De versie 2 (v2) kan worden geoptimaliseerd de waarde van de Gas Limiet omdat afhankelijk van het slimme contract hoe snel u de publicatie op het etherische netwerk wilt maken. Aanbevolen wordt om voor de publicatie minimaal een waarde van 350.000 WEI te hanteren, zonder dat er sprake is van een storing of vertraging.

Het verschil tussen de functies van het aanmaken van TokenERC20v1 en het aanmaken van TokenERC20v2 is dat in het geval van versie 1 de parameter van GasLimit al voorgeconfigureerd is en in versie 2 kan worden geconfigureerd naar de behoeften van de gebruiker of de ontwikkelaar.

Blokkeren om ERC20-token op te roepen of uit te voeren - (**SmartContractExecution**)

Invoerparameters: adresSmartContract<String>, naamFileSolidityJSON<String>,
functieeluitvoeringSmartContract<String>.

Uitgangsparameters: Uitvoering van de functie die in het genoemde Smart Contract is gespecificeerd.

Uitvoeringen: **Uitvoering van een slim contract.**

OPMERKING: Om te kunnen worden uitgevoerd, moet een bestand met JSON-formaat worden gemaakt dat de parameters bevat van de primaire sleutel van het adres dat u het Smart contract wilt uitvoeren, u moet de gaslimiet (WEI) invoeren.

Voorbeeld van een JSON-bestand voor het uitvoeren van de functies van het samengestelde Smartcontract in voorbeeld van de hierboven genoemde compilerfunctie. De bestandsnaam kan willekeurig zijn.

Bestand: call.json

```
{
  "privé": "3ca40...",
  "gas_limiet": 20000
}
```

Voorbeeld van de output van de "groet"-functie van het Smart contract:

```
{
  "gas_limiet": 20.000,
  "address": "0eb688e79698d645df015cf2e9db5a6fe16357f1",
  "resultaten": [
    "Hallo BlockCypher Test"
  ]
}
```

ERC20 Token Property Display Block - (**SmartContractGetCreationTx**)

```
call BlockoinETHEREUM1 .SmartContractGetCreationTx
```

```
txSmartContract
```

```
" d99ff4d53f9b1f0687289674633d2acecf219011d163cdc0... "
```

Invoerparameters: **txSmartContract<String>**

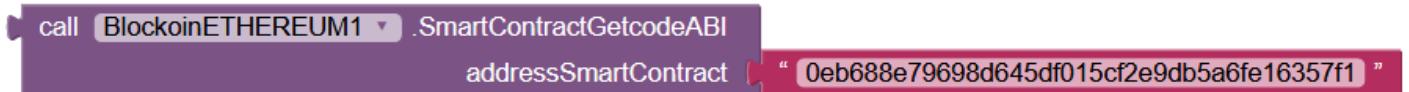
Uitgangsparameters: Toont de eigenschappen van het Smart contract waarnaar wordt verwezen met (**Tx_hash**).

Omschrijving: Toont de belangrijkste kenmerken en ABI-code van het Smart contract waarnaar wordt verwezen.

Voorbeeld eigenschappen van ERC20 token, voorbeeld tokenNaam "MOGY" eerder gemaakt met de functie (**martContractCreateTokenERC20v1**)

```
{
  "block_hash": "cadb8914c43ed28fb370c9e1b6f5d8818ba31a1e944d1f7049441b982902dea8",
  "blokhoogte": 11240201,
  "blok_index": 170,
  "hash": "d99ff4d53f9b1f0687289674633d2acecf219011d163cdc08b45468f63a22882",
  "adressen": [
    "4b7355fd05be6dac458b004f54e12d6527a54a58"
  ],
  "totaal": 0,
  "vergoedingen": 110146910000000,
  "grootte": 958,
  "gas_limiet": 500.000,
  "gas_gebruikt": 268651,
  "gas_prijs": 41000000000,
  "contract_creatie": waar,
  "doorgestuurd": "200.77.24.87",
  "confirmed": "2020-11-12T02:49:56Z",
  "received": "2020-11-12T02:50:13.185Z",
  "zie": 0,
  "double_spend": vals,
```


Blok om de ERC20-tokencompilatiecode weer te geven - (**SmartContractGetcodeABI**)



Invoerparameters: **adresSmartContract<String>**

Uitgangsparameters: toont de Smart contractcode waarnaar wordt verwezen.

Omschrijving: Toont de ABI-code van het Smart-contract waarnaar wordt verwezen.

Voorbeeld van een ABI-code van een generiek Smart-contract:

```
{
  "soliditeit": "contract sterfelijk": definieer de variabele eigenaar van
  het type adres ; deze functie wordt uitgevoerd bij de initialisatie en
  stelt de eigenaar van het contract in */n     functie sterfelijk() {
  eigenaar = msg.afzender; }n           /* Functie om het geld van het
  contract te recupereren */n     functie doden() { als (msg.afzender ==
  eigenaar) zelfmoord; \n}contractgroet is sterfelijk \n      /* definieer
  variabele begroeting van het type tekenreeks */\n      tekenreeks
  begroeting};\ndit loopt wanneer het contract wordt uitgevoerd */n
  functie groet (string _greeting) publiek {\n      begroeting =
  _greet;\n      /* hoofdfunctie */n      functie groet() constante
  retouren (string) {\n      retouren groet;\n}      ",
  "Bin":
"606060405260405161023e38038061023e8339810160405280510160008054600160a060
020a031916331790558060016000509080519060200190828054600181600116156101000
203166002900490600052602060002090601f016020900481019282601f10609f57805160
ff19168380011785555b50608e9291505b8082111560cc57600081558301607d565b50505
061016e806100d06000396000f35b828001600101855582156076579182015b8281111560
7657825182600050559160200191906001019060b0565b509056606060405260e060020a6
00035046341c0e1b58114610026578063cfae321714610068575b005b6100246000543373
ffffffffffffffffff0908116911614156101375760005473fff
ffffffffffff0908116911614156101375760005473fff
a06020601f600260001961010086881615020190941693909304928301819004028101604
0526080828152929190828280156101645780601f10610139576101008083540402835291
60200191610164565b6040518080602001828103825283818151815260200191508051906
0200190808383829060006004602084601f0104600f02600301f150905090810190601f16
80156101295780820380516001836020036101000a031916815260200191505b509250505
06
  "abi":
"[{\\"constant\":false,\\"inputs\":[],\\"name\":\"kill\",\\\"outputs\":[],\\\"ty
pe\\\":\\\"function\\\\"}, {\\"constant\":true,\\"inputs\":[],\\"name\":\"greet\",\\\"
outputs\": [{\\\"name\\\":\\\"\",\\\"type\\\\":\\\"string\\\"}],\\\"type\\\\":\\\"functi
on\\\\"}, {\\"inputs\": [{\\\"name\\\\":\\\"_greeting\\\",\\\"type\\\\":\\\"string\\\"}],\\\"type\\\\":\\\"const
ructor\\\\"}]",
  "creation_tx_hash":
"61474003e56d67aba6bf148c5ec361e3a3c1ceea37fe3ace7d87759b399292f9",
  "created": "2016-07-20T01:54:50Z",
  "address": "0eb688e79698d645df015cf2e9db5a6fe16357f1"}  

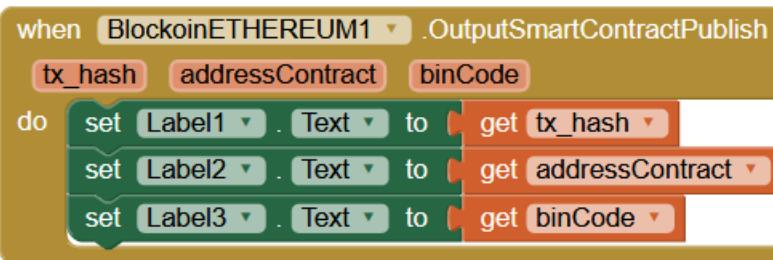
*Voor het gebruik van het volgende blok (SmartContractPublish) dient u het blok
(SmartContractCompilerSolidity) te gebruiken om te controleren of het Smartcontract goed
geschreven is.
```

Blokkeren om de ERC20-token op het Ethereum-netwerk te publiceren - (SmartContractPublish).

```
call BlockoinETHEREUM1 .SmartContractPublish
      nameFileSolidityJSON " /mnt/sdcard/PublishSmartContract.json "
```

Invoerparameters: nameFileSolidityJSON<String>

Uitgangsparameters: Toont de eigenschappen van het Smart contract waarnaar verwezen wordt. In het geval (OutputSmartContractPublish).



Omschrijving: Publiceer in het etherische netwerk het Smart contract waarnaar in het JSON-bestand wordt verwezen.

Voorbeeldbestand: **PublishSmartContract.json**

```
{
  "soliditeit": "contract sterfelijk": definieer de eigenaar van
  het type adres ; deze functie wordt uitgevoerd bij de initialisatie en
  stelt de eigenaar van het contract in */n   functie sterfelijk() {
  eigenaar = msg.afzender; }n           /* Functie om het geld van het
  contract te recupereren */n   functie doden() { als (msg.afzender ==
  eigenaar) zelfmoord; {\n}contractgroet is sterfelijk {\n           /* definieer
  variabele begroeting van het type tekenreeks */\n           tekenreeks
  begroeting};\ndit loopt wanneer het contract wordt uitgevoerd */n
  functie groet (string _greeting) publiek {\n           begroeting =
  _greet;\n           /* hoofdfunctie */n           functie groet() constante
  retouren (string) {\n           retouren groet;\n}       ",
  "params": "Hallo test,
  "publiceren": "Groter,
  "privé": "3ca40...
  "gas_limiet": 500000
}
```

Zoals in bovenstaande code te zien is, is het dezelfde JSON-code die in het voorbeeld van de compileerfunctie wordt gebruikt om de parameters aan het einde van het JSON-bestand toe te voegen:

Params: Impliciete parameters van de Smart contr.

Publiceren: Naam van de wijze waarop het Smart contract zal worden gepubliceerd.

Privé: De primaire sleutel van de rekening die het Smart contract zal uitvoeren moet een saldo hebben.

Gas_limit: Is het saldo in WEI dat u wilt besteden voor de publicatie van het Smart contract.

Voorbeeld van output bij uitvoering (publicatie Smart contract) het Smart contract wordt aangegaan in het etherische netwerk. Het toont de uitgevoerde transactie "**creation_tx_hash**" en het toegewezen adres van het aangemaakte Smart contract "**address**".

```
[
{
    "naam": "groeter",
    "soliditeit": "contract sterfelijk": definieer de variabele eigenaar
    van het type adres ; deze functie wordt uitgevoerd bij de
    initialisatie en stelt de eigenaar van het contract in /*/n      functie
    sterfelijk() { eigenaar = msg.afzender; }/*/n           /* Functie om het geld
    van het contract te recupereren /*/n      functie doden() { als
    (msg.afzender == eigenaar) zelfmoord; {\n}contractgroet is sterfelijk {\n
    /* definieer variabele begroeting van het type tekenreeks */{\n
    tekenreeks begroeting};\n dit loopt wanneer het contract wordt uitgevoerd
    /*/n      functie groet (string _greeting) publiek {\n          begroeting =
    _greet;\n          /* hoofdfunctie *//n      functie groet() constante
    retouren (string) {\n          retouren groet;\n          ",
    "Bin":
"606060405260405161023e38038061023e8339810160405280510160008054600160a060
020a031916331790558060016000509080519060200190828054600181600116156101000
203166002900490600052602060002090601f016020900481019282601f10609f57805160
ff19168380011785555b50608e9291505b8082111560cc57600081558301607d565b50505
061016e806100d06000396000f35b828001600101855582156076579182015b8281111560
7657825182600050559160200191906001019060b0565b509056606060405260e060020a6
00035046341c0e1b58114610026578063cfae321714610068575b005b6100246000543373
ffffffffffffffffff908116911614156101375760005473fff
ffffffffffff908116911614156101375760005473fff
a06020601f600260001961010086881615020190941693909304928301819004028101604
0526080828152929190828280156101645780601f10610139576101008083540402835291
60200191610164565b6040518080602001828103825283818151815260200191508051906
0200190808383829060006004602084601f0104600f02600301f150905090810190601f16
80156101295780820380516001836020036101000a031916815260200191505b509250505
06
    "abi": [
        {
            "constant": vals,
            "inputs": [],
            "naam": "doden",
            "uitgangen": [],
            "type": "functie"
        },
        {
            "constant": waar,
            "inputs": [],
            "naam": "groet",
            "outputs": [
                {
                    "naam": ""
                    "type": "string"
                }
            ],
            "type": "functie"
        }
    ]
}
```

```

        "type": "functie"
    },
    {
        "ingangen": [
            {
                "naam": "_groet",
                "type": "string"
            }
        ],
        "type": "bouwer"
    }
],
"gas_limiet": 500.000,
"creation_tx_hash": "61474003e56d67aba6bf148c5ec361e3a3c1ceea37fe3ace7d87759b399292f9",
"address": "0eb688e79698d645df015cf2e9db5a6fe16357f1",
"params": [
    "Hallo-test"
]
},
{
    "naam": "sterfelijk",
    "soliditeit": "contract sterfelijk": definieer de variabele eigenaar van het type adres ; deze functie wordt uitgevoerd bij de initialisatie en stelt de eigenaar van het contract in /*n functie sterfelijk() { eigenaar = msg.afzender; }n /* Functie om het geld van het contract te recupereren */n functie doden() { als (msg.afzender == eigenaar) zelfmoord; \n}contractgroet is sterfelijk {\n/* definieer variabele begroeting van het type tekenreeks */{\n    tekenreeks begroeting};\ndit loopt wanneer het contract wordt uitgevoerd */n functie groet (string _greeting) publiek {\n    begroeting = _greet;\n    /* hoofdfunctie */n functie groet() constante retouren (string) {\n        retouren groet;\n    }
    "Bin":
"606060405260008054600160a060020a03191633179055605c8060226000396000f36060
60405260e060020a600035046341c0e1b58114601a575b005b60186000543373ffffffffffff
fffffffffffffffffffff90811691161415605a5760005473fffffffffffff
fffffffffffffffffffff16ff5b56",
"abi": [
    {
        "constant": vals,
        "inputs": [],
        "naam": "doden",
        "uitgangen": [],
        "type": "functie"
    },
    {
        "inputs": [],
        "type": "bouwer"
    }
],
"gas_limiet": 500.000,
"params": "Hallo test."
}
]

```

Testblok om een bestand te maken - (`createTestingFile`)

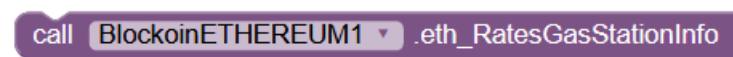


Invoerparameters: `pathTestFile<String>`

Uitgangsparameters: Er wordt een testbestand aangemaakt in het betreffende pad.

Beschrijving: Dit dient om het geldige pad voor het aanmaken van tijdelijke bestanden te controleren om er zeker van te zijn dat het correct is bij gebruik van het blok (`SmartContractCreateTokenERC20v1`) of het blok (`SmartContractCreateTokenERC20v2`).

Blokkeren om de tarieven van de Gasprijs te krijgen - (`eth_RatesGasStationInfo`).



Invoerparameters: `nameFileSolidityJSON<String>`

Uitgangsparameters: Toont de eigenschappen van het Smart contract waarnaar verwezen wordt. In het geval (`OutputEth_GasStationInfo`) worden de geleverde waarden gegeven in GWEI.

De Gasprijs is de waarde die wordt betaald aan de systemen die de transacties in het Ethereum uitvoeren. Deze systemen worden algemeen bekend als "mijnwerkers" en de waarden van de Gasprijs is een functie van hoe snel (tijd en prioriteit) de transactie in het Ethereum-netwerk wordt uitgevoerd.

De geleverde waarden zijn een functie van de volgende looptijden. Deze tijden zijn bij benadering en kunnen variëren afhankelijk van hoe je het doet met de eisen (transacties) op het Ethereum netwerk.

Snel < 2 minuten.

Snelste < 30 seconden.

SafeLow < 30 minuten.

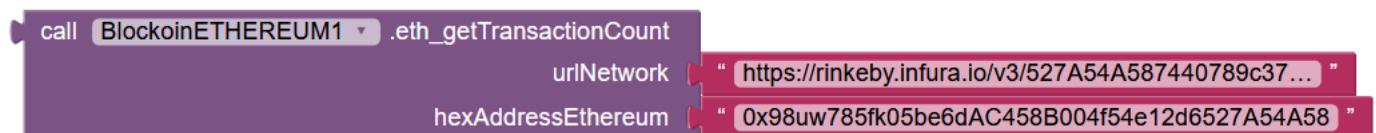
Gemiddeld < 15 minuten.

Bij transacties met de Exchange Ethereum Extension (EEE) wordt altijd de Gasprijs = Gemiddeld gebruikt.



Beschrijving: Krijgt de bijgewerkte Gasprijs op het moment van de vraag om een nieuwe transactie aan te maken.

Blok om het nummer "nonce" te verkrijgen - (**eth_getTransactionCount**).



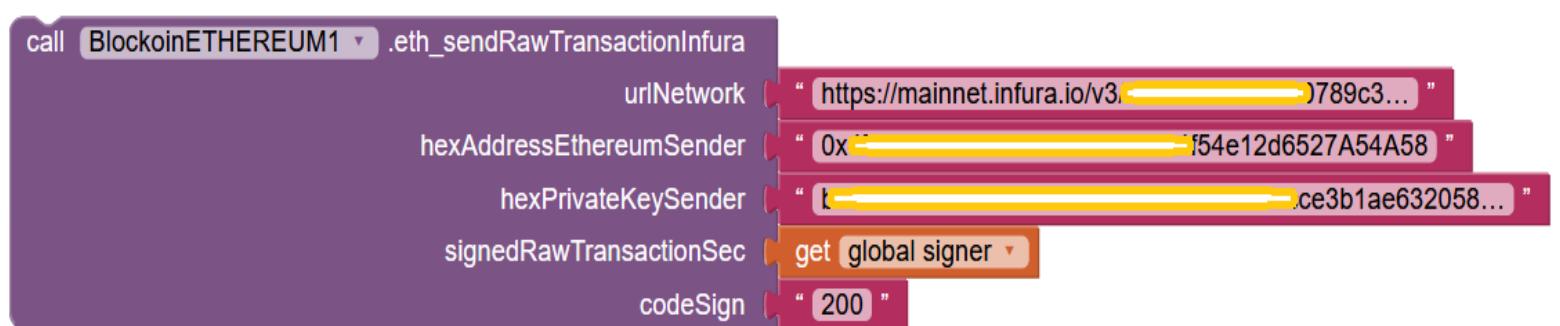
Invoerparameters: **urlNetwork<String>**, **hexAddressEthereum<String>**.

Uitgangsparameters: Het toont in hexadecimaal formaat het opeenvolgende nummer "nonce" van het genoemde adres.

Het "nonce"-nummer is een oplopend nummer dat het aantal transacties bijhoudt dat vanaf een bepaald adres is verricht.

Omschrijving: Verkrijgt het "nonce" nummer van het genoemde adres.

Blokkeren om een ondertekende transactie te verzenden - (**eth_SendRawTransactionInfura**).

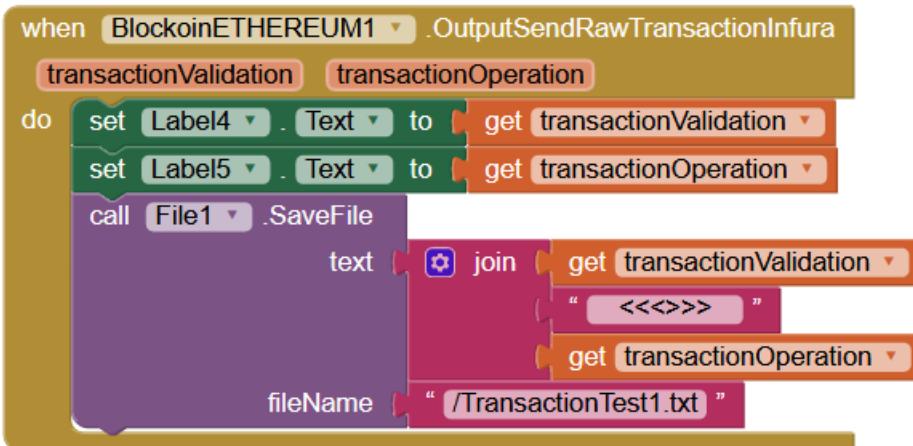


Vereiste afhankelijkheid(en): Blok (**eth_getTransactionCount**), Blok (**SignerGenericPushRawTransactionOffline**). Bekijk het voorbeeld van het blok (**SignerGenericPushRawTransactionOffline**).

Invoerparameters: urlNetwork <String>, hexAddressEthereumSender <String>, hexPrivateKeySender <String>, SignedRawTrasactionSec <String>, codeSign <String>.

Uitgangsparameters: Gebeurtenis (OutputSendRawTransactionInfura)

Uitvoer: transactieValidatie<String> , transactieOperatie<String>.



Beschrijving: Het levert twee hexadecimale waarden op als gevolg van de transacties. De transactieValidatiewaarde is de transactie die op het Ethereum-netwerk wordt verricht en die de impliciete kosten van het Ethereum bevat. De waarde van transactionOperation is de transactie die wordt gemaakt in het Coinconsolidation netwerk met de kosten van \$0,5 cent USD voor elke transactie op basis van de waarde van de ether op het moment van de transactie. Deze kosten zijn voor de betaling van de diensten van het Coinconsolidation.org netwerk en wordt geïnvesteerd in het onderhoud, de ondersteuning en de creatie van uitbreidingen voor de crypto- en activabranche wereldwijd.

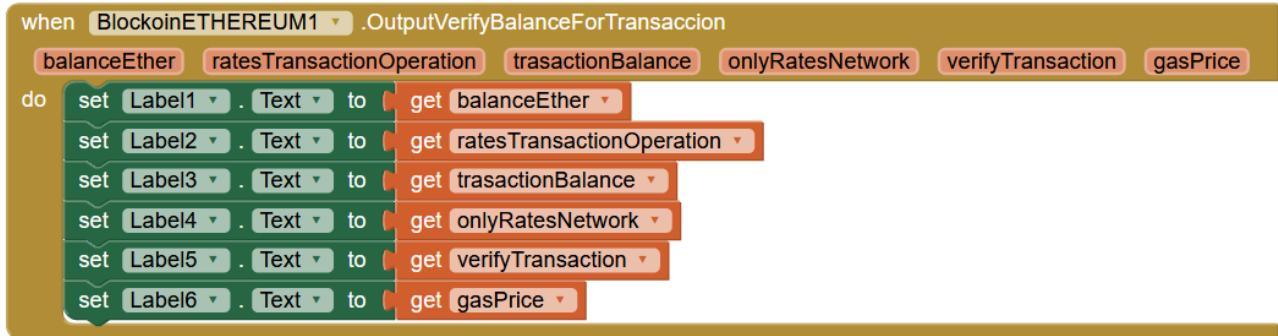
Blok om de kosten van een standaard transactie te berekenen -
(eth VerifyBalanceForTransaction)



Invoerparameters: adresEthereumSender<String>, waardeEthertoSend<String>.

Uitgangsparameters: Event (OutputVerifyBalanceForTransaction).

Uitgangen: balansEther<String> , koersenTransactieOperatie<String> , trasactionBalance<String> , OnlyRatesNetwork<String> , verificatieTransactie<String> , gasPrijs<String>.



Beschrijving: Geeft details over wat de kosten van een standaardtransactie zullen zijn met betrekking tot het adres van de boeking. De uitvoerparameter "verifyTransaction" vertelt ons of de transactie "Waar" kan worden gemaakt of dat het adres waarnaar wordt verwezen niet genoeg saldo heeft, zal ons een "False" geven.

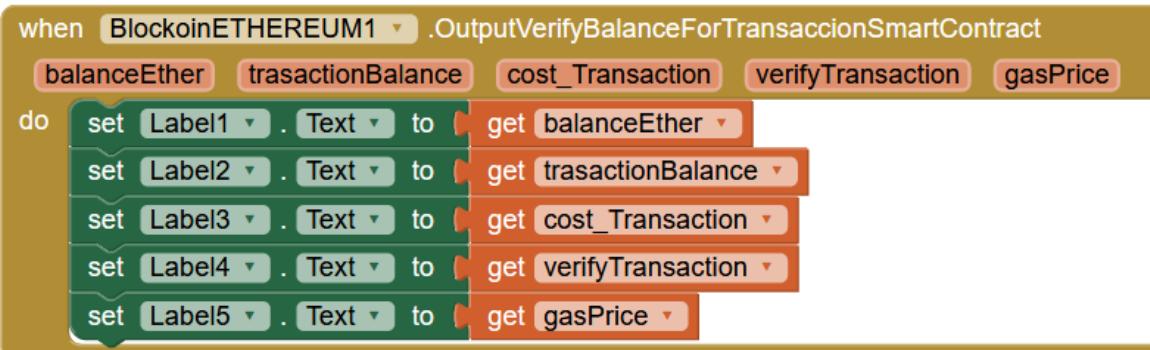
Blok om de kosten van een standaardtransactie te berekenen -
(eth_VerifiBalanceForTransaccionSmartContract)



Invoerparameters: adresEthereum<String>, gasLimit<String>.

Uitgangsparameters: Event (OutputVerifyBalanceForTransaccionSmartContract)

Uitgangen: balansEther<String> , transactionBalance<String> , cost_Transaction<String> , controleerTransaction<String> , gasPrice<String>.



Beschrijving: Geeft details over wat de geschatte kosten van een standaardtransactie zouden zijn voor het plaatsen van een **Smart contract**, met verwijzing naar het adres van

binnenkomst. De uitvoerparameter "verifyTransaction" vertelt ons of de transactie "Waar" kan worden gemaakt of dat het adres waarnaar wordt verwezen niet genoeg saldo heeft, zal ons een "Valse" geven.

balansEther: De balans van het genoemde adres wordt geleverd in ethers.

trasactionBalans: Balans na de transactie.

cost_Transaction: Is de kosten van de transactie om het smart contrat te publiceren.

verificatieTransactie: (saldoEther minus cost_Transactie).

gasPrijs: Huidige waarde van de GasPrijs die door de "mijnwerkers" wordt gebruikt, deze kan elke minuut variëren.

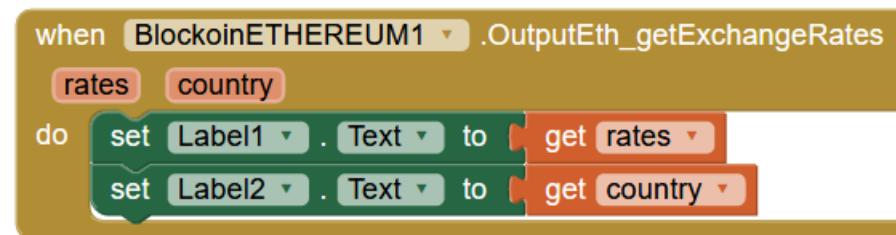
Blok om de Etherprijs in de valuta van het opgegeven land te krijgen - (eth_getExchangeRates).



Invoerparameters: **countryRates<String>**. Controleer de uitvoerparameter "land" waar deze alle landenvalutatypes bevat om de gewenste te kiezen.

Uitgangsparameters: Event (**OutputEth_getExchangeRates**).

Uitvoer: **tarieven<String>, landen<String>** uitvoer in JSON-formaat alle tarieven van de landen van de wereld.



Beschrijving: Het levert de huidige prijs van een Ether op tegen de wisselkoers van de valuta van het betreffende land.

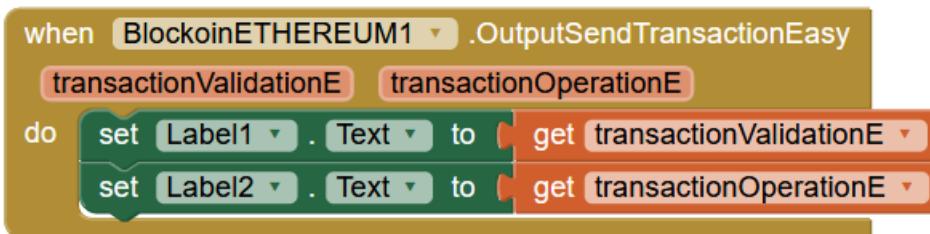
Blokken om een standaardtransactie uit te voeren met de minimale invoerparameters - (eth_sendTransactionEasy).



Invoerparameters: hexPrivateKeySender<String>, toAddress<String>, waardeEther<String>.

Uitgangsparameters: Event (OutputSendTransactionEasy)

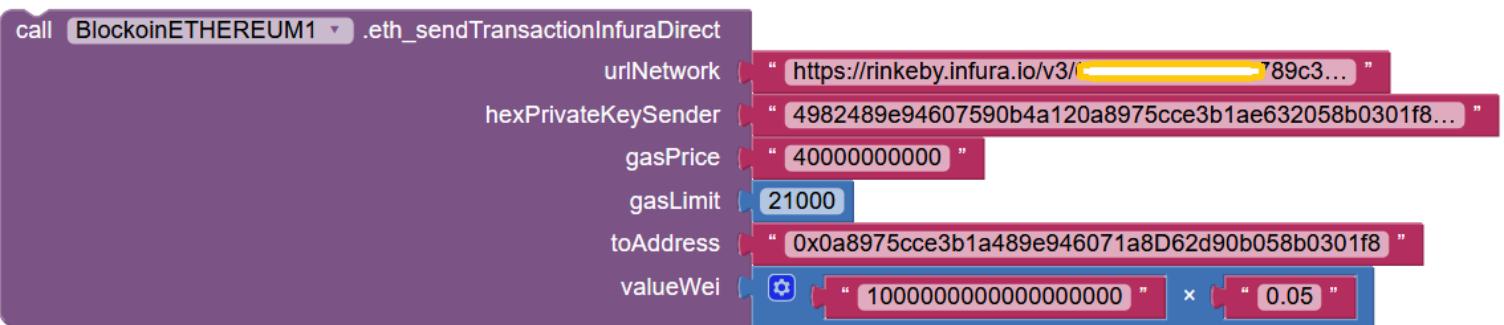
Uitvoer: transactieValidatie<String> , transactieOperatie<String>.



Omschrijving: Functie om een standaard transactie te maken in het netwerk van het Ethereum, deze functie is van direct nut je hoeft geen rekening te hebben in INFURA en je hoeft maar 3 invoerparameters te hebben, je hoeft maar genoeg saldo te hebben om de gewenste transactie te maken.

Transacties worden direct op het Ethereum netwerk geplaatst met behulp van de officiële Ethereum Web3j bibliotheek en ons netwerk Coinconsolidation.org.

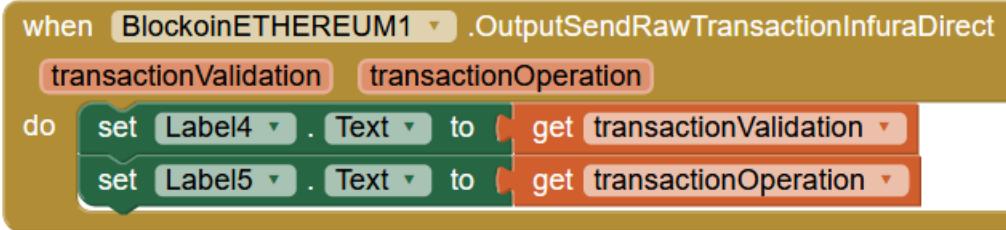
Blokkeren om een standaardtransactie uit te voeren met de minimale invoerparameters - (eth_sendTransactionInfuraDirect).



Invoerparameters: urlNetwork<String>, hexPrivateKeySender<String>, gasPrice<String> , gasLimit<String>, toAddress<String>, waardeWEI<String>.

Uitgangsparameters: Gebeurtenis (OutputSendTransactionInfuraDirect)

Uitvoer: transactieValidatie<String> , transactieOperatie<String>.



Omschrijving: Functie om een standaard transactie te versturen die al de impliciete digitale handtekening bevat, dit is nuttig voor mensen die al voorkennis hebben van de componenten van een transactie en deze parameters willen optimaliseren volgens hun behoeften.

11. Berekening van de standaard transactiekosten en Slimme contracttransactie

Voor de berekening van een standaard transactie zijn 3 parameters nodig in het Ethereum netwerk.

- 1.- **Gasprijs**.
- 2.- **Gasgrens**.
- 3.- **Huidige waarde van een Ether** (digitale ether).

GasPrijs: Dit wordt normaal gesproken gegeven in eenheden van GWEI (GigaWEI). Dit komt overeen met, als 1 ether 1.000.000.000.000.000 wei heeft dan is 1 GWE gelijk aan 1.000.000.000.000. Deze eenheid dient als betaling voor de systemen die alle transacties van het Ethereum-netwerk uitvoeren en wordt "mijnwerkers" genoemd, het wordt over de hele wereld verspreid. De GasPrijs is geen vaste waarde en is variabel en kan van minuut tot minuut of seconde veranderen. Degenen die de waarde van GasPrice bepalen zijn de "mijnwerkers" en het hangt af van hoe verzadigd het Ethereum-netwerk is.

GasLimit: Deze waarde wordt normaal gesproken gegeven in eenheden van WEI en in een standaard transactie is een gemiddelde standaardwaarde 21.000 WEI, hoewel het hoger of lager kan zijn, afhankelijk van welk type transactie u wilt maken, in een standaard transactie gebruiken we de waarde voor 40.000 WEI om ervoor te zorgen dat we geen afwijzing hebben voor het hebben van onder de Gas Limit.

Ether's waarde: Deze waarde is ook variabel en is te wijten aan verschillende parameters van een wereldwijde financiële markt, kan deze waarde worden verkregen van entiteiten die altijd hebben bijgewerkt de waarde van Ether wereldwijd genoemd gecentraliseerde en gedecentraliseerde uitwisselingen.

BELANGRIJKE OPMERKING: In de Exchange Ethereum Extension (EEE) gebruiken we een hogere Gas Limit (40.000) dit is te wijten aan het feit dat in het geval van het niet bereiken van het minimum quotum vastgesteld door de "mijnwerkers" de TRANSACTIE NIET GESLOTEN is, echter het Ethereum netwerk, als het de kosten die het heeft gemaakt in de berekening van de transactie in rekening brengt zonder het maken van de transactie, Om deze reden leggen sommige gebruikers niet uit waarom hun transactie niet wordt uitgevoerd, maar ze zullen wel een bedrag of alle GasLimit die werd aangeboden toen de transactieaanvraag werd gelanceerd in rekening worden gebracht, om deze reden moeten we altijd duidelijk zijn wat de waarden van GasLimit en de Gasprijs zullen zijn.

De GasPrijs kan worden geraadpleegd met de functie `eth_GetRatesGasStation` en de GasLimit voor standaard transacties moet hoger zijn dan 21.000 WEI en transacties voor het publiceren en/of uitvoeren van Smart contract moeten minimaal 500.000 WEI bedragen.

Standaard transactiekosten.

Het wordt gedefinieerd door de volgende formule:

Kosten = (GasLimit x (GasPrijs / (1.000.000.000.000.000)) x Etherwaarde.

Voorbeeld:

Ga uit van de volgende waarden:

GasLimit = 40.000, GasPrice = 45 GWEI, Ether Value = \$406.

Kosten = (40.000 x (45.000.000.000 / (1.000.000.000.000)) x 406

Standaard transactiekosten = 0,0018 ether x 406 USD = \$0,73 USD

In het geval van een Slimme contracttransactie is het noodzakelijk om te weten wat voor soort Slimme contract zal worden gepubliceerd, aangezien de kosten recht evenredig zijn met de werklast die de "mijnwerkers" zullen moeten uitvoeren om het Slimme contract te verwerken, met andere woorden, de hoeveelheid verwerking in de computersystemen die de "mijnwerkers" afhandelen is eenvoudiger.

In het geval van het Smart contract zou een standaardwaarde zijn om de GasLimit te starten met een waarde van 500.000 WEI.

Een belangrijk punt om rekening mee te houden is dat wanneer men een GasLimit voorstelt, de "mijnwerkers" niet noodzakelijkerwijs al het voorgestelde bedrag zullen nemen, dat wil zeggen, wanneer men een transactie stuurt, berekenen de "mijnwerkers" de berekeningsinspanning en nemen wat uit de GasLimit wordt gehaald, die in sommige gevallen minder of gelijk kan zijn aan de standaard aangeboden GasLimit van 21.000 WEI.

Alle transacties zullen kunnen worden geraadpleegd op de site www.etherscan.io waar we de details van elke transactie kunnen raadplegen.

Voorbeeld van een standaardtransactie, waarbij de transactie werd verstuurd met een gaslimiet van 40.000 WEI's, maar de "mijnwerkers" bij de berekening van de verwerkingskosten slechts 52,5% namen, d.w.z. de standaardwaarde die 21.000 WEI's bedraagt.

The screenshot shows a transaction details page from Etherscan. The transaction hash is 0x7dae251f21c616fb04a94112633c97e04d11c91f4d06dd9b85ed11112f02703a. It was successful and has 2 block confirmations. The transaction was sent from 0x4b7355fd05be6dac458b004f54e12d6527a54a58 to 0x5d2acdb34c279aa6d1e94a77f7b18ab938fb2bb at 0.52 seconds ago (Nov-11-2020 06:17:24 AM +UTC). The value was 0.001084598698481562 Ether (\$0.50). The transaction fee was 0.000672 Ether (\$0.31). The gas price was 0.000000032 Ether (32 Gwei) and the gas limit was 40,000. The gas used was 21,000 (52.5%).

Parameter	Value	Note
Transaction Hash	0x7dae251f21c616fb04a94112633c97e04d11c91f4d06dd9b85ed11112f02703a	TransactieOperatie of Transactie Validatie
Status	Success	
Block	11234630	2 Block Confirmations
Timestamp	52 secs ago (Nov-11-2020 06:17:24 AM +UTC)	Confirmed within 38 secs
From	0x4b7355fd05be6dac458b004f54e12d6527a54a58	
To	0x5d2acdb34c279aa6d1e94a77f7b18ab938fb2bb	Transactiekosten in Ether: $21.000 \times 0,000000032 = 0,000672$ Ether (0,31 USD)
Value	0.001084598698481562 Ether (\$0.50)	
Transaction Fee	0.000672 Ether (\$0.31)	
Gas Price	0.000000032 Ether (32 Gwei)	Voorgestelde gasgrens: 40.000
Gas Limit	40,000	
Gas Used by Transaction	21,000 (52.5%)	Werkelijke gaslimiet gebruikt in de transactie: 21.000 WEI
Nonce	36	
Position	79	

Door terug te keren naar de Smart contracttransactie en de 500.000 WEI GasLimit te nemen, krijgen we de volgende kosten als we alles gebruiken.

Slimme contracttransactiekosten.

Het wordt gedefinieerd door de volgende formule:

Kosten = (GasLimit x (GasPrijs / (1.000.000.000.000.000)) x Etherwaarde.

Voorbeeld:

Ga uit van de volgende waarden:

GasLimit = 500.000, **GasPrice** = 45 GWEI, **Ether Value** = \$406.

Kosten = (500.000 x (45.000.000.000 / (1.000.000.000.000))) x 406

Transactiekosten publiceren Slimme contract = 0,0225 ether x 406 USD = \$9.135 USD

Alle transacties kunnen worden geraadpleegd op de site www.etherscan.io

OPMERKING: Om de totale kosten van de transactie te krijgen moet u ze optellen:

Totale transactiekosten = *Ethereum-netwerk kosten + Coinsolidation.org-vergoeding*

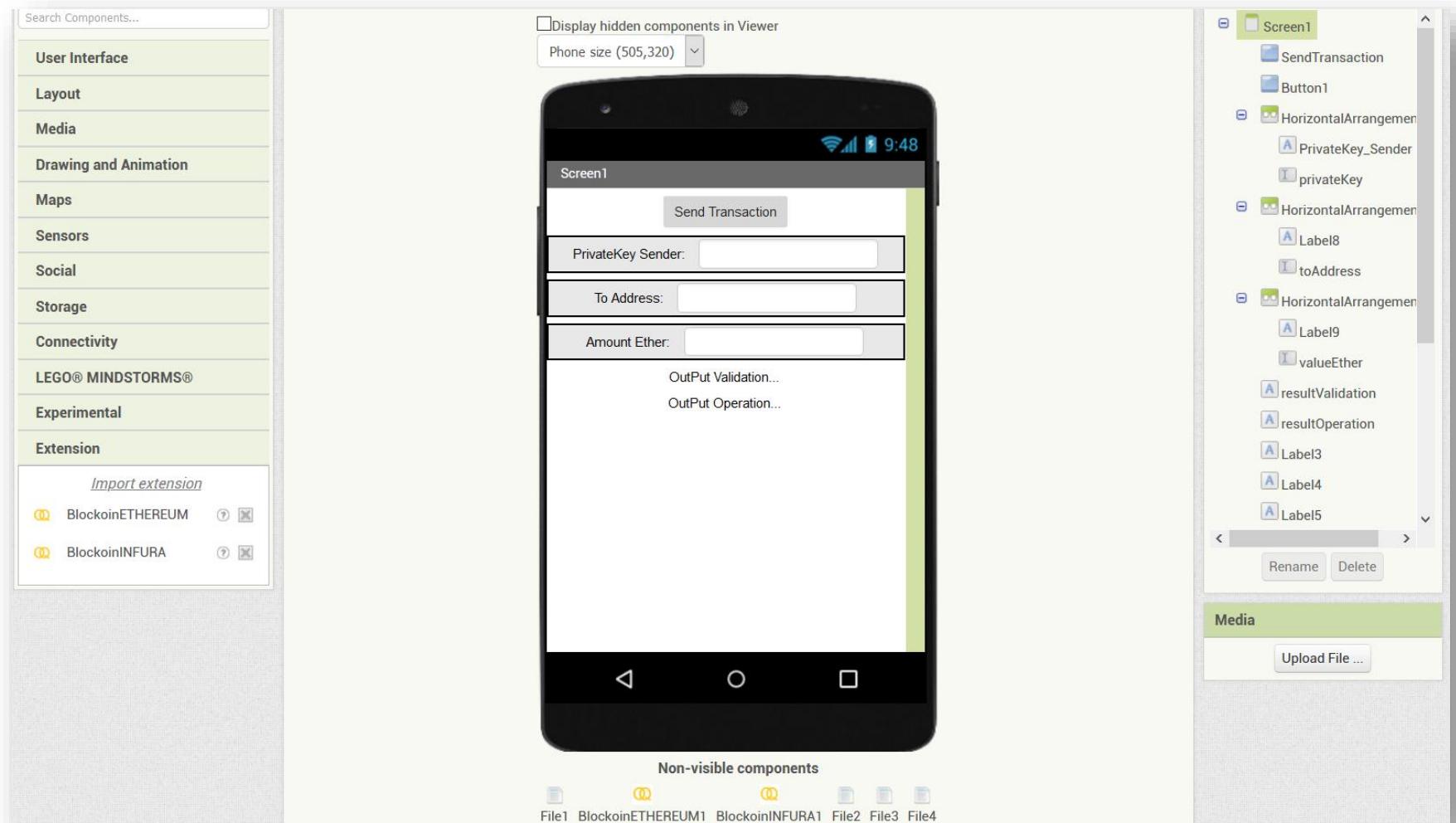
12. Coinsolidation.org Vergoedingen

Standaard transactie: \$ 0,5 cent USD + kosten van het Ethereum-netwerk.

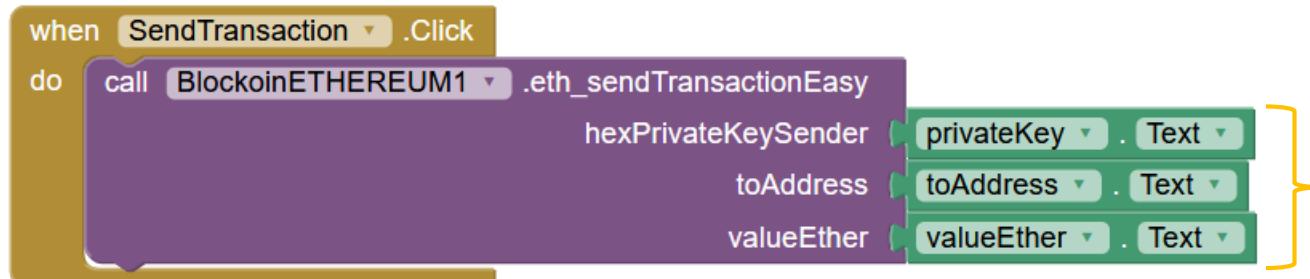
Transactie publiceren en/of uitvoeren van een Smart contract: \$15 USD + kosten van het Ethereum netwerk.

13. Maak uw Android App (Exchange) aan in 15 minuten.

Ontwerp in App Inventor (Scherm). - 5 minuten.



Functieblokken (`eth_SendTransactionEasy`) en event (`OutPutSendTransactionEasy`) - 5 minuten

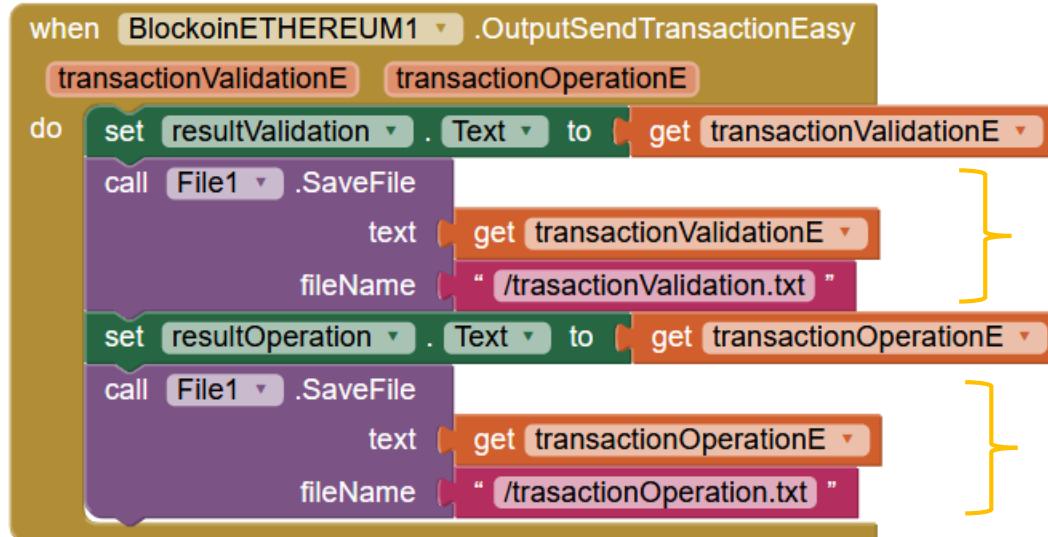


Voer de gegevens in:

PrivateKey: Primaire sleutel tot het adres van de afzender.

toAddress: Hexadecimaal adres van de ontvanger.

waardeEther: Geef de hoeveelheid Ether die wordt verzonden.



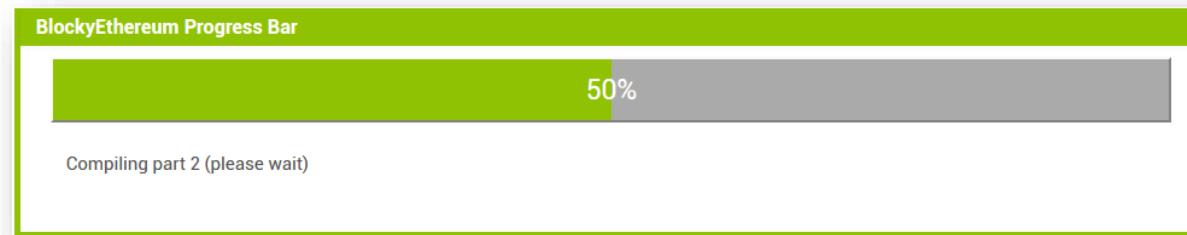
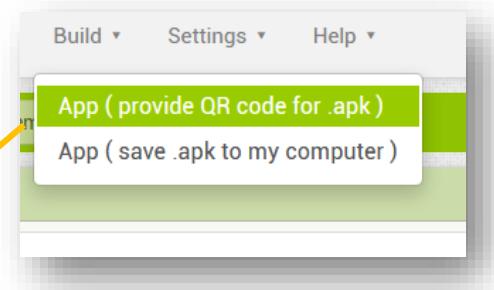
Sla de resultaten op in tekstbestanden:

Functie File1: File **trasactionValidation.txt**

Sla de resultaten op in tekstbestanden:

File2-functie: Dossier **trasactionValidation.txt**

We compileren, genereren APK-bestand om het te installeren op het Android-toestel. - 5 minuten



OPMERKING: Wanneer de transactie wordt uitgevoerd, duurt het ongeveer 6 tot 8 seconden om de knop "Transactie verzenden" los te laten. Vanwege de verbindingstijd met het Ethereum-netwerk.

14. Token CoinSolidation.

Token CoinSolidation is een project met drie belangrijke richtlijnen.

Stel je voor dat je je eigen Token-cryptocurrency hebt om je bedrijf te laten groeien. Met CoinSolidation kunt u het gemakkelijk en eenvoudig hebben.

De eerste is het creëren van het eerste netwerk van uitbreidingen op basis van de visuele programmeermethodologie Blockly, waarbij door het eenvoudige en intuïtieve gebruik door iedereen zonder voorafgaande kennis van programmeren, de uitbreidingen die kunnen worden geraadpleegd in onze Roadmap (Witboek) zijn gericht op twee fundamentele sectoren in de wereldconomie, de sector van de crypto-valuta's en/of penningen en de sector van de valuta's (fiat) of gemeenschappelijk gebruik wereldwijd, zoals de dollar, euro's EU, Ponden of elke andere valuta die in gebruik zijn.

De tweede richtlijn in het CoinSolidation project is het creëren van een nieuw algoritme om adressen van huidige en toekomstige cryptomontages te consolideren. We ontwikkelen een nieuw modelalgoritme om een adres te creëren dat verschillende adressen uit verschillende blokkades samenvoegt tot een universeel adres zie onze (White Paper) op www.CoinSolidation.org of <https://github.com/coinsolidation/whitepaper>.

De derde richtlijn is het toepassen van Quantum Computing technologie in de beveiliging van de CoinSolidation omgeving, dit zal worden toegepast met de reeds ontwikkelde uitbreidingen van QRNG (Quantum Random Number Generator) en PQC (Post-Quantum Computing) beveiligingsalgoritmes. Deze zijn te vinden in de officiële uitbreidingsrepository op de Github site, <https://github.com/coinsolidation>.

Algemene kenmerken van de CryptoToken COINSolidation

Naam: COINSolidation

Symbol: CUAG

Type: NFT

Lanceringsland: Estland

Officiële website: www.Coinsolidation.org

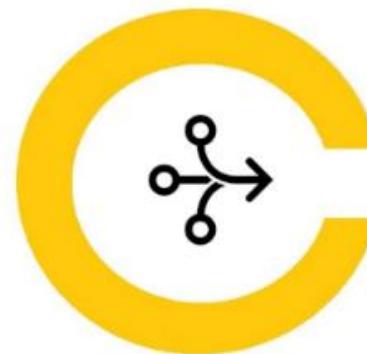
Bedrijf: Coinsolidation International.

Lanceringsdatum: 30 december 2020

Gemaakt door: Lugu Samaya.

Consensusalgoritme: bewijs van Quantum.

Adresalgoritme: Geconsolideerd universeel adres.



15. Licentie en gebruik van software.

Licenties, gebruiksvoorwaarden zie www.coinsolidation.org of schrijf naar info@coinsolidation.org.